ABSTRACT

Title of dissertation: AGE OF INFORMATION IN LARGE NETWORKS, DISTRIBUTED COMPUTATION AND LEARNING

Baturalp Buyukates, Doctor of Philosophy, 2021

Dissertation directed by: Professor Sennur Ulukus
Department of Electrical and Computer Engineering

We consider timely data delivery in real-time communication networks that have gained significant importance with recent promising applications including augmented reality/virtual reality (AR/VR), autonomous vehicular networks, and smart factories. Age of information is a network performance metric introduced to guarantee access to fresh data in such systems. Considering increasing connectivity in communication networks and ever growing prominence of distributed computation and learning systems, in this dissertation, we study age of information in large networks with a particular focus on its scalability with growing network size as well as the design of distributed computation and learning systems that handle time-critical data using potentially large number of worker nodes with as little age as possible.

First, we consider a multihop multicast network with a single source node sending time-sensitive updates to n^L end nodes where L denotes the number of hops. Updates from the source go through relay nodes in each hop to reach the end nodes. We show that by utilizing appropriate transmission stopping thresholds in each

hop, the age of information at the end nodes can be made a constant independent of n. We then find the optimum stopping value for each hop for arbitrary shifted exponential link delays.

Next, we focus on a single hop multicast network where two update streams share the same network, called type I and type II updates. We show that utilizing an earliest k_1 and k_2 transmission scheme for type I and type II updates, respectively, prevents information staleness for both update streams. We find the optimum k_1 and k_2 stopping thresholds for arbitrary shifted exponential link delays to individually and jointly minimize the average age of both update streams.

Then, we consider the age scaling in a large peer-to-peer network consisting of n randomly paired source-destination pairs. We first propose a three-phase transmission scheme which utilizes *local cooperation* among the nodes along with *mega update packets* and show that it achieves an average age scaling of $O(n^{\frac{1}{4}} \log n)$ per-user as n grows. Next, we show that, under a hierarchical implementation of the proposed scheme, an average age scaling of $O(n^{\alpha(h)} \log n)$ per-user is achievable, where h denotes the number of hierarchy levels and $\alpha(h) = \frac{1}{3 \cdot 2^h + 1}$. The proposed hierarchical scheme asymptotically achieves a per-user average scaling of $O(\log n)$.

Next, we consider the version age of information scaling in gossip networks, where a total of n nodes are clustered into distinct communities and they are allowed to share their versions of the source information with their neighbors within each cluster. By assuming different topologies for the clusters, we show that per node average version age scalings of $O(\sqrt{n})$, $O(n^{\frac{1}{3}})$, and $O(\log n)$ are achievable in disconnected, ring, and fully connected cluster models, respectively. We then in-

crease connectivity across clusters by implementing a hierarchical gossip mechanism to further improve the version age scaling results, and find the version age-optimum cluster sizes for various settings.

Then, we consider a status updating system in which the update packets are data rich and need to be processed further to extract the embedded information. This processing is handled in a distributed manner at a computation unit which consists of a master node and n worker nodes. We investigate the age performance of uncoded and coded (repetition coded, MDS coded, and multi-message MDS (MM-MDS) coded) schemes in the presence of stragglers. We show that, asymptotically, MM-MDS coded scheme outperforms the other schemes, and characterize the age-optimal codes.

Next, we study age of information in federated learning and propose a novel timely communication scheme specifically designed for learning applications that use highly temporal rapidly changing client datasets such as recommendation systems and next place forecasting tasks. The proposed timely communication scheme aims to incorporate time-critical client data into the global model updates with as little age as possible by also considering the limited client availability and communication resources. We show that, in addition to ensuring timeliness, the proposed policy significantly improves the average iteration time of training without hurting the convergence performance of the algorithm.

Then, we consider utilizing the age of information metric to improve convergence performance in distributed learning with coded computation and partial recovery for straggler mitigation. We propose a novel age-based encoding framework

that regulates the recovery frequency of the partial computations received from the workers. We show through several experiments on a linear regression problem that the proposed age-based encoding strategy significantly improves the convergence performance compared to conventional static encoding schemes.

Next, we propose a novel gradient coding (GC) scheme with dynamic clustering, called GC-DC, to improve the average iteration time of the existing static gradient coding techniques. The proposed GC-DC scheme clusters the workers and applies the GC scheme in each cluster separately. Under a time correlated straggling behavior for the workers, GC-DC dynamically forms the clusters based on the past straggling behavior to distribute straggling workers to clusters as uniformly as possible. We show through extensive simulations on both homogeneous and heterogeneous worker profiles that the GC-DC scheme significantly improves the average iteration time compared to the existing static GC schemes without any increases in the communication load.

Finally, we study a status updating system with a single sampler which takes samples from an observed phenomenon and sends them to a monitor node through a single server that implements a blocking policy. We consider two scenarios with partial non-i.i.d. components: Gilbert-Elliot service times and i.i.d. interarrival times; and i.i.d. service times and Gilbert-Elliot interarrival times. We characterize the average age experienced by the monitor node and determine the age-optimal state transition matrix for both of these scenarios.

AGE OF INFORMATION IN LARGE NETWORKS, DISTRIBUTED COMPUTATION AND LEARNING

by

Baturalp Buyukates

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Professor Sennur Ulukus, Chair/Advisor
Professor Behtash Babadi
Professor Richard La
Professor Nuno Martins
Professor Radu Balan

Dedication

To my parents Ayse and Kenan for always believing in me.

Acknowledgments

I would like to express my deepest gratitude to my advisor Professor Sennur Ulukus for her unwavering support and continued guidance during my Ph.D. studies. Her unparalleled dedication to research, extensive experience, and invaluable insights have always inspired me through the many ups and downs of this long journey. I would like to thank her for giving me the opportunity to work with her. It has been and continues to be a true pleasure and a rewarding experience for me to be guided by her in my academic life. My work in this dissertation would not have been possible without her endless encouragement and relentless help.

I would like to extend my sincere thanks to Professors Behtash Babadi, Richard La, Nuno Martins, and Radu Balan for being in my dissertation committee and offering their valuable feedback. I must also thank Professor Anthony Ephremides for his helpful suggestions on my research proposal. I am thankful to all the professors that I interacted with at the University of Maryland. In particular, I would like to express my gratitude to Professor Prakash Narayan. I had the great pleasure of working with him as his teaching assistant for two semesters.

I am grateful to Professor Deniz Gunduz, of Imperial College London, and Professor Alkan Soysal, of Virginia Tech, for our collaborations and their insightful advice on my work. Special thanks to Emre Ozfatura and Melih Bastopcu for all of our fruitful technical discussions, some of which have turned into joint works that are part of this dissertation. I am also thankful to Professor Emine Ulku Saritas, of Bilkent University, for the opportunity to work with her during my undergraduate

studies and the support that I have received for my doctoral applications.

I am thankful to all my colleagues at the University of Maryland for making our lab a friendly place to share and discuss ideas. Many thanks to my lab mates Batuhan Arasli, Brian Kim, Priyanka Kaswan, Matin Mortaheb, Cemil Vahapoglu, Sajani Vithana, Zhusheng Wang, Purbesh Mitra, Mustafa Doger, Shreya Meel, Melih Bastopcu, Yi-Peng Wei, Karim Banawan, Abdulrahman Baknina, Ahmed Arafa, Pritam Mukherjee, Berk Gurakan, Ajaykrishnan Nageswaran, Sagnik Bhattacharya, and Vinay Praneeth Boda.

I would like to sincerely thank my roommate and dearest friend at the University of Maryland, Melih Bastopcu. We started this journey together and he has been present in every moment of my Ph.D. studies. It has been an absolute pleasure to share this time with him. I would like to thank him for his exceptional friendship and perspective. I thank my other roommate, Batuhan Arasli, for being a great companion and for his never disappointing stories. I am also grateful to Hulya Biler for her constant support and profound belief in me. I will miss our talks on Ph.D. and life in general.

Many thanks to my other friends at the University of Maryland for making this journey enjoyable. I thank Ozde Ozkaya, Semih Kara, Ece Yegane, Gamze Yavuzer, Deepayan Bhadra, and Siddharth Tyagi. I am grateful to Yagmur Bozturk, Cagri Aydinkarahaliloglu, Sevgi Kafali, Cem Ataman, Tugce Akin, Muge Kurtipek, Can Colak, Goksu Karadag, and Zeynep Dincer for their generous friendship that exceeds cities, countries, and even continents. It is because of the joy they bring in my life that I find myself motivated to work everyday. I will cherish our memories forever.

I am forever indebted to my family for their unconditional support and care not just during my Ph.D. studies but in life in general. My deepest gratitude goes to my parents, Ayse Buyukates and Kenan Buyukates, and my sister Irem Buyukates for always believing in me and supporting all my dreams. Special thanks to Yagmur Bozturk for being there for me no matter what. I am extremely lucky to have her in my life. This dissertation is dedicated to all of them.

Table of Contents

List of Figures x

List of Tables xv

1 Introduction 1

2 Age of Information in Multihop Multicast Networks 12
- 2.1 Introduction . . . 12
- 2.2 System Model and Age Metric . . . 14
- 2.3 Building Block: Single-Hop Network with Exogenous Arrivals . . . 16
- 2.4 Two-Hop Network . . . 28
- 2.5 Extension to L Hops . . . 33
- 2.6 Numerical Results . . . 38
- 2.7 Appendix: Proof of Lemma 2.1 . . . 42
- 2.8 Conclusion . . . 44

3 Age of Information in Multicast Networks with Multiple Update Streams 46
- 3.1 Introduction . . . 46
- 3.2 System Model and Age Metric . . . 48
- 3.3 Age Analysis . . . 50
  - 3.3.1 At-will Update Generation . . . 50
  - 3.3.2 Exogenous Update Arrivals . . . 58
- 3.4 Conclusion . . . 61

4 Scaling Laws for Age of Information in Wireless Networks 63
- 4.1 Introduction . . . 63
- 4.2 System Model and Age Metric . . . 66
- 4.3 Age Analysis of a Single S-D Pair . . . 70
- 4.4 Three-Phase Transmission Scheme . . . 71
- 4.5 Three-Phase Transmission Scheme with Hierarchy . . . 81
  - 4.5.1 Motivation and Outline of the Scheme . . . 81
  - 4.5.2 Detailed Description of the Scheme for $h = 1$ . . . 84
- 4.6 Note on Phases I and III . . . 94
- 4.7 Numerical Results . . . 96

4.8 Discussion . . . 99
4.9 Conclusion . . . 104

5 Scaling Laws for Version Age of Information in Gossip Networks 106
5.1 Introduction . . . 106
5.2 System Model and the Age Metric . . . 112
5.3 Version Age with Community Structure . . . 114
5.3.1 Version Age in Clustered Disconnected Networks . . . 117
5.3.2 Version Age in Clustered Ring Networks . . . 119
5.3.3 Version Age in Clustered Fully Connected Networks . . . 122
5.4 Version Age with Community Structure Under Connected Cluster Heads . . . 125
5.4.1 Version Age in Clustered Disconnected Networks with Connected Cluster Heads . . . 128
5.4.2 Version Age in Clustered Ring Networks with Connected Cluster Heads . . . 129
5.4.3 Version Age in Clustered Fully Connected Networks with Connected Cluster Heads . . . 132
5.5 Version Age in Hierarchical Clustered Gossip Networks . . . 133
5.5.1 Version Age for $h = 3$ Hierarchy Levels . . . 136
5.5.2 Version Age for $h > 3$ Hierarchy Levels . . . 138
5.6 Numerical Results . . . 138
5.7 Conclusion . . . 143

6 Timely Distributed Computation with Stragglers 145
6.1 Introduction . . . 145
6.2 System Model and Age Metric . . . 148
6.3 Age of Uncoded and Coded Task Distribution Algorithms . . . 151
6.3.1 Uncoded Scheme . . . 154
6.3.2 Repetition Coded Scheme . . . 156
6.3.3 MDS Coded Scheme . . . 158
6.3.4 Multi-message MDS (MM-MDS) Coded Scheme . . . 159
6.4 Optimizing Age by Parameter Selection . . . 166
6.5 Numerical Results . . . 172
6.6 Conclusion . . . 173

7 Timely Communication in Federated Learning 175
7.1 Introduction . . . 175
7.2 System Model . . . 178
7.3 Average Age Analysis . . . 180
7.4 Numerical Results . . . 186
7.5 Conclusion . . . 193

8 Age-Aware Distributed Computation for Bias Reduction 194
8.1 Introduction . . . 194
8.2 System Model and Problem Formulation . . . 196
8.2.1 DGD with Coded Computation . . . 197
8.2.2 Coded Computation with Partial Recovery (CCPR) . . . 198
8.2.3 Distributed Dynamic Coded Submatrix Generation . . . 199
8.2.4 Problem Definition . . . 203
8.3 Solution Approach: Timely Dynamic Ordering . . . 205
8.3.1 Fixed Vertical Shifts . . . 205
8.3.2 Age-Based Vertical Shifts . . . 206
8.4 Numerical Results . . . 207
8.5 Conclusion . . . 213

9 Gradient Coding with Dynamic Clustering for Straggler-Tolerant Distributed Learning 215
9.1 Introduction . . . 215
9.2 Preliminaries: Gradient Coding (GC) and Clustering . . . 217
9.2.1 Gradient Coding (GC) . . . 218
9.2.2 Gradient Coding with Static Clustering (GC-SC) . . . 219
9.3 GC with Dynamic Codeword Assignment . . . 223
9.4 GC with Dynamic Clustering (GC-DC) . . . 226
9.4.1 Codeword Construction . . . 226
9.4.2 Codeword Distribution . . . 227
9.4.3 Dynamic Clustering . . . 229
9.5 Greedy Dynamic Clustering Strategy . . . 232
9.5.1 Phase I - Worker Placement . . . 235
9.5.2 Phase II - Conflict Resolution . . . 236
9.6 Numerical Results . . . 241
9.6.1 Gilbert-Elliot Model with Homogeneous Workers . . . 242
9.6.2 Heterogeneous Worker Model . . . 244
9.6.2.1 Gilbert-Elliot Model with Heterogeneous Workers . . 245
9.6.2.2 Heterogeneous Workers with Time-Varying Rates . . 247
9.6.3 Simulations for Shared Access Scenario . . . 249
9.7 Conclusion . . . 252

10 Age of Information with Gilbert-Elliot Servers and Samplers 255
10.1 Introduction . . . 255
10.2 System Model and Age Metric . . . 258
10.3 Average Age Analysis . . . 260
10.3.1 Gilbert-Elliot Service Times and I.i.d. Interarrival Times . . . 262
10.3.2 Gilbert-Elliot Interarrival Times and I.i.d. Service Times . . . 264
10.4 Age-Optimal Transition Matrix P . . . 266
10.4.1 Gilbert-Elliot Service Times and I.i.d. Interarrival Times . . . 266
10.4.2 Gilbert-Elliot Interarrival Times and I.i.d. Service Times . . . 267
10.5 Age-Optimal Transition Matrix P under Average Cost Constraint . . . 268

10.5.1 Gilbert-Elliot Service Times and I.i.d. Interarrival Times . . . 271
10.5.2 Gilbert-Elliot Interarrival Times and I.i.d. Service Times . . . 271
10.6 Numerical Results 272
10.7 Conclusion 274

11 Conclusions 275

Bibliography 279

List of Figures

1.1 Sample AoI evolution curves. Average age evaluation through calculating areas. 3

2.1 Two-hop multicast network with a single source node sending updates through n middle nodes each of which is tied to further n end nodes. 13
2.2 Multicast network model: (a) two-hop operation, (b) details of hops $\ell - 1$ and ℓ. 14
2.3 The arrival and update structure of a building block. 17
2.4 Update process for an end node in the building block setting. Filled circles indicate the arrivals successfully received by this particular end node. Crosses indicate updates that arrive at the source node when it is busy transmitting another update. Empty circles show the updates that are transmitted by the source node but are not received by this particular end node, i.e., they are preempted from the perspective of this particular end node. 19
2.5 Sample age evolution $\Delta_{(k,\mu)}(t)$ of an end node. Updates that find the system idle arrive at times T_i at the source. Here, update j arrives at time T_{j-1} and immediately goes into service. Successful update deliveries are marked with $\bullet$ and in this figure, updates $j-1$, j and $j+2$ are delivered successfully whereas update $j+1$ is terminated. . 24
2.6 $\Delta_{(k,\mu)}$ as a function of stopping threshold k for several μ values with $\lambda = 1$. $\circ$ marks the minimized $\Delta_{(k,\mu)}$: (a) when $c = 1$, (b) when $c = 0$. 28
2.7 $\Delta'_{(k_1,k_2)}$ as a function of k_2 for $n = 500$. $\circ$ marks the minimized average age $\Delta'_{(k_1,k_2)}$: (a) $c = 1$ and $(\tilde{\lambda}, \tilde{c}) = (1, 1)$ for varying λ, (b) $\tilde{c} = 1$ and $(\lambda, c) = (1, 1)$ for varying $\tilde{\lambda}$. 40
2.8 $\Delta'_{(k_1,k_2)}$ as a function of k_1 for $n = 500$. $\circ$ marks the minimized average age $\Delta'_{(k_1,k_2)}$: (a) $c = 1$ and $(\tilde{\lambda}, \tilde{c}) = (1, 1)$ for varying λ, (b) $\tilde{c} = 1$ and $(\lambda, c) = (1, 1)$ for varying $\tilde{\lambda}$. 41
2.9 $\Delta'_{(k_1,k_2)}$ as a function of k_2 for $n = 500$ when link delays are exponential. $\circ$ marks the minimized average age $\Delta'_{(k_1,k_2)}$: (a) $c = 0$ and $(\tilde{\lambda}, \tilde{c}) = (1, 0)$ for varying λ, (b) $\tilde{c} = 0$ and $(\lambda, c) = (1, 0)$ for varying $\tilde{\lambda}$. 43

2.10 $\Delta'_{(k_1,k_2)}$ as a function of link delay parameters λ and $\tilde{\lambda}$ for $n = 500$: (a) $\Delta'_{(k_1,k_2)}$ as a function of λ, (b) $\Delta'_{(k_1,k_2)}$ as a function of $\tilde{\lambda}$. 44
2.11 $\Delta'_{(k_1,k_2,k_3)}$ as a function of k_3 for $n = 100$. $\circ$ marks the minimized average age $\Delta'_{(k_1,k_2,k_3)}$: (a) $c = 1$ and $(\tilde{\lambda}, \tilde{c}) = (\tilde{\tilde{\lambda}}, \tilde{\tilde{c}}) = (1, 1)$ for varying λ, (b) $\tilde{c} = 1$ and $(\lambda, c) = (\tilde{\tilde{\lambda}}, \tilde{\tilde{c}}) = (1, 1)$ for varying $\tilde{\lambda}$. 45

3.1 Multicast network with a single source node sending two types of updates to n nodes. 47
3.2 Sample age evolution $\Delta_I(t)$ of a node. Update cycle j starts at time T_{j-1} and lasts until T_j. Successful type I update deliveries are marked with $\bullet$ and in this figure, in cycles $j-1$, j and $j+2$ a type I update is delivered successfully whereas in cycle $j+1$ no type I delivery occurred. 52
3.3 Pareto optimal curve for jointly minimizing Δ_I and Δ_{II} with $0 < \beta < 1$, $p_1 = p_2 = 0.5$ and $(\lambda, c) = (\tilde{\lambda}, \tilde{c}) = (1, 1)$ when (a) updates are generated at-will (b) updates arrive exogenously with a total rate of $\mu = 1$. 58

4.1 Sample age $\Delta(t)$ evolution for a single S-D pair. Update deliveries are shown with symbol $\bullet$. Session j starts at time T_{j-1} and lasts until $T_j = Y_j + T_{j-1}$. 69
4.2 Cell formation for $M = 4$ and $n = 100$. Simultaneous intra-cell transmissions are depicted for three S-D pairs from cells P, Q, R and S. 73
4.3 In Phase II cells take turns to perform inter-cell transmissions. These inter-cell transmissions are shown for the same three S-D pairs depicted in Fig. 4.2. 75
4.4 Proposed three-phase hierarchical transmission scheme. 82
4.5 Average age scaling under the proposed three-phase hierarchical transmission scheme for $h = 0$, $b = \frac{1}{2}$, $\lambda = 5$ and $\tilde{\lambda} = 2$ for varying n when (a) nodes are on a grid, (b) nodes are randomly uniformly and independently distributed. 97
4.6 (a) Average age scaling under the proposed three-phase hierarchical transmission scheme for $h = 0$, $b = \frac{1}{2}$ when nodes have non i.i.d. transmission rates for varying n. (b) Comparison of round robin scheme and the proposed three-phase hierarchical transmission scheme for $h = 0$, $b = \frac{1}{4}$, $\lambda = 5$ and $\tilde{\lambda} = 2$ for varying n. 98

5.1 Tiered network model where blue node at the center represents the source, yellow nodes represent the cluster heads, and green nodes represent the end users. Here, nodes in each cluster form a bi-directional ring network. Other possible network topologies within a cluster are shown in Fig. 5.2. 109

5.2 Different network topologies that can be used within each cluster: (a) disconnected, (b) uni-directional ring, (c) bi-directional ring, and (d) fully connected. Fig. 5.1 uses the one in (c). In this figure, cluster size is $k = 6$. 110
5.3 Tiered network model where blue node represents the source, yellow nodes represent the cluster heads $c_1, \ldots, c_6$, and green nodes represent the end users. Here, cluster heads form a bi-directional ring network with $m = 6$. Each cluster is associated with a cluster of $k = 6$ nodes. Here, only one such cluster is shown. Nodes in each cluster form a bi-directional ring network. Other possible network topologies within a cluster are shown in Fig. 5.2. 127
5.4 Two-level hierarchical network model where blue node represents the source and green nodes represent the end users. Here, nodes in each cluster form a bi-directional ring network of $k_1 = k_2 = 6$ nodes. We have a single cluster in the first level and $k_1 = 6$ clusters in the second level. Here, only one such second level cluster is shown. Other possible network topologies within a cluster are shown in Fig. 5.2. . . 134
5.5 Version age of a node with fully connected, ring, and disconnected cluster models with $n = 120$, (a) $\lambda_e = 1$, $\lambda_s = 1$, $\lambda_c = 1$, and $\lambda = 1$, (b) $\lambda_e = 1$, $\lambda_s = 10$, $\lambda_c = 1$, and $\lambda = 1$, (c) $\lambda_e = 1$, $\lambda_s = 10$, $\lambda_c = 10$, and $\lambda = 1$, (d) $\lambda_e = 1$, $\lambda_s = 10$, $\lambda_c = 1$, and $\lambda = 2$. 139
5.6 Version age of a node with fully connected, ring, and disconnected cluster models with $n = 120$, $\lambda_e = 1$, $\lambda_s = 10$, $\lambda_{ca} = 4$, $\lambda_{cb} = 6$, and $\lambda = 1$ when cluster heads form a ring network among themselves. . . 141

6.1 System model with a single source node and a computation unit (CU) that consists of a master node and n identical worker nodes. 146
6.2 Sample age evolution $\Delta(t)$ at the destination (source) node. Successful updates are indexed by j. The jth successful update leaves the source node at T_{j-1} and arrives at the CU at T'_{j-1}. Update cycle at the CU is the time in between two successive arrivals and is equal to $Y_j = S_j + Z_j = T'_j - T'_{j-1}$. 152
6.3 The earliest k computed tasks for $n = 10$, $\ell = 3$, and $k = 7$. 160
6.4 Δ_{unc}, Δ_{rep} and Δ_{mds} for varying k with $n = 100$, and $(\lambda, c) = 1$: (a) When $\mu = 1$, (b) when $\mu = 0.5$. Symbol $\circ$ marks the optimal k values. 171
6.5 (a) k^*, k_1^*, and k_2^* values as a function of n for $\ell = 2$ with $\mu = 1$, and $(\lambda, c) = 1$. Note that $k^* = k_1^* + k_2^*$. (b) Δ_{mm-mds}, as a function of load ℓ for $(n\ell, k^*)$-MDS code with $n = 100$, $\mu = 0.01$, and $(\lambda, c) = 1$. 172

7.1 Federated learning model where a parameter server (PS) trains a learning model using n clients without actually storing clients' data centrally. 177

7.2 Sample age evolution at client j. Iteration t starts at time T_{t-1}. Filled circles and crosses show the time instances at which client j generates its local update and the PS receives that local update, respectively. Here, client j successfully delivers its local update to the PS in iterations t and $t+2$. 183
7.3 Average age experienced by a client as a function of k with $n = 100$, $\lambda = 1$, and $c = 1$ for varying $\tilde{\mu}$. In each curve we use the age-optimal m. The age-optimal k values are shown with a circle. 187
7.4 Average age experienced by a client as a function of k with $n = 100$, $\tilde{\mu} = 1$, and $c = 1$ for varying λ. In each curve we use the age-optimal m. The age-optimal k values are shown with a circle. 188
7.5 Average age experienced by a client as a function of k with $n = 100$, $\lambda = 1$, and $\tilde{\mu} = 1$ for varying c. In each curve we use the age-optimal m. The age-optimal k values are shown with a circle. 189
7.6 Average age experienced by a client as a function of k with $n = 100$, $\lambda = 1$, $c = 1$, and $\tilde{\mu} = 1$ for varying m. The age-optimal k values are shown with a circle. 190
7.7 Average iteration time under different schemes when $k = 10$, $m = 20$, $n = 100$ for $c = 1$, $\tilde{\mu} = 1$, $\lambda = 1$ averaged over 50000 iterations. 191
7.8 Convergence performance of the proposed scheme for varying k with $m = 40$, $n = 100$ $c = 1$, $\lambda = 1$, and $\tilde{\mu} = 1$ for a linear regression task. 192

8.1 Illustration of partial recovery in a naive distributed computation scenario with 6 workers, 2 of which are stragglers. 200
8.2 Sample age evolution of $\mathbf{W}_k\boldsymbol{\theta}_{\bar{t}}$. Time t marks the beginning of iteration $t+1$. $\mathbf{W}_k\boldsymbol{\theta}_{\bar{t}}$ is recovered at iterations $\bar{t} = t$ and $\bar{t} = t+4$. 202
8.3 Test accuracy (log-scale) vs. number of iterations with $q = 0.3$ and 15 persistent stragglers. 209
8.4 Average ages of the partial computations with $q = 0.3$ and 15 persistent stragglers. 210
8.5 Test accuracy (log-scale) vs. number of iterations with $q = 0.3$ and straggling behavior based on a 2-state Markov chain with a state transition probability of $p = 0.05$. 212
8.6 Test accuracy (log-scale) vs. number of iterations for the uncoded scheme with $q = 0.2$ and 15 persistent stragglers. 213

9.1 Two possible straggler realizations where red and green circles represent the straggling and non-straggling workers, respectively. 223
9.2 The proposed worker placement strategy. 239
9.3 Average per-iteration completion time under the Gilbert-Elliot model with homogeneous workers for $K = 20$, $P = 5$, $r = 3$, $n = 3$, and $p = 0.05$ (a) under imperfect SSI, (b) under perfect SSI. 244

9.4 Average per-iteration completion time under the Gilbert-Elliot model with homogeneous workers for $K = 20$, $P = 5$, $r = 3$, $n = 3$, and $p = 0.2$ (a) under imperfect SSI, (b) under perfect SSI. 245
9.5 Average per-iteration completion time under the Gilbert-Elliot model with heterogeneous workers for $K = 20$, $P = 5$, $r = 3$, $n = 3$, $p = 0.05$, and $\tau = 0.5$ (a) under imperfect SSI, (b) under perfect SSI. 247
9.6 Average per-iteration completion time under the Gilbert-Elliot model with heterogeneous workers for $K = 20$, $P = 5$, $r = 3$, $n = 3$, $p = 0.2$, and $\tau = 0.5$ (a) under imperfect SSI, (b) under perfect SSI. 248
9.7 Average per-iteration completion time under the heterogeneous worker model with time-varying rates for $K = 20$, $P = 5$, $r = 3$, $n = 3$, $p = 0.05$, and $\tau = 1$ (a) under imperfect SSI, (b) under perfect SSI. . 250
9.8 Average per-iteration completion time under the heterogeneous worker model with time-varying rates for $K = 20$, $P = 5$, $r = 3$, $n = 3$, $p = 0.05$, and $\tau = 0.1$ (a) under imperfect SSI, (b) under perfect SSI. 251
9.9 Average per-iteration completion time under the heterogeneous worker model with time-varying rates for $K = 20$, $P = 5$, $r = 3$, $n = 3$, $p = 0.2$, and $\tau = 0.1$ (a) under imperfect SSI, (b) under perfect SSI. . 252
9.10 Average per-iteration completion time under the shared access model for $K = 20$, $P = 5$, $r = 3$, $n = 3$, $\lambda = 0.1$, $\gamma = 7$, $C = 10$ (a) under imperfect SSI, (b) under perfect SSI. 253

10.1 A single sampler sends time-sensitive status updates to a single monitor node through a server node. We consider Gilbert-Elliot server and Gilbert-Elliot sampler settings. 256
10.2 Sample age evolution $\Delta(t)$ at the monitor node. Successful updates are indexed by j. The jth successful update arrives at the server node at T_{j-1}. Update cycle at the server node is the time in between two successive arrivals and is equal to $Y_j = S_j + Z_j = T_j - T_{j-1}$. 261
10.3 Feasible (p, q) pairs for the problem in (10.23) when (a) $\frac{c-c_b}{c_g-c} > 1$ and (b) $\frac{c-c_b}{c_g-c} < 1$ where the line is $q(c_b - c) + p(c_g - c) = 0$ in both cases. . 270
10.4 Age of information as a function of probability p for Gilbert-Elliot service times. Symbol $\circ$ marks the simulation results and curves indicate the values obtained from (10.7). 272
10.5 Age of information as a function of probability q for Gilbert-Elliot service times. Symbol $\circ$ marks the simulation results and curves indicate the values obtained from (10.7). 273

List of Tables

2.1 Optimal α_i values for L-hop network when all link delays are shifted exponentials with $(1, 1)$. 39

4.1 Comparison of the expected durations of the phases with $h = 0$ as in Section 4.4 and $h = 1$ hierarchy level with $0 < a < b \leq 1$. 83

5.1 The summary of the scaling of version age in gossip networks. 112
5.2 Comparison of the $(\Delta^*_{S_1}, k^*)$ pairs with (as in Section 5.4) and without (as in Section 5.3) gossip at the cluster heads. 142
5.3 Average version age of a single node at the third hierarchy level, $\Delta^3_{S_1}$, with $h = 3$, $n = 120$, $\lambda_e = 1$, $\lambda_s = 1$, and $\lambda_a + \lambda_b = 5$. For given (λ_a, λ_b) pairs, we find the optimum k_1, k_2, and k_3 values that minimize $\Delta^3_{S_1}$. 143

8.1 The value of the objective function in (8.14) when $a_{th} = 2$ for varying tolerance levels q. 211

CHAPTER 1

Introduction

With the proliferation of data-driven time-critical applications that involve real-time status information such as augmented reality/virtual reality (AR/VR), autonomous vehicular networks, and smart factories, freshness of the received messages has become a critical and desirable feature. In autonomous vehicular networks, for example, vehicles continuously sense the surrounding environment and generate time-stamped status information regarding traffic, road conditions, and so on, in the form of status update packets. Timely delivery of these updates to nearby vehicles and roadside units is essential for safe and reliable operation. For a realistic sense of presence and heightened user experience in AR/VR applications, timely delivery of images, video and sound data is critical. In a rapidly changing factory environment where workers, surrounding sensors, and robots need to exchange real-time information, timeliness in the communications is required to ensure reliable operation and increase efficiency. In all these applications, information loses its value as it becomes stale, which in turn brings up the need for timely delivery of status information to the interested parties since more recent measurements better capture the source pro-

cesses. Introduced in [1], the age of information metric has been proposed to assess timeliness of information in such systems and is the main focus of this dissertation.

A typical model to study age of information includes a source node which acquires time-stamped status updates from a physical phenomenon. These time-sensitive update packets are transmitted over a network to the receiver(s) and the *age of information* in this network or simply the *age* is the time elapsed since the most recent update at the receiver was generated at the transmitter. In other words, at time t, age $\Delta(t)$ of a packet which was generated at time $u(t)$, is $\Delta(t) = t - u(t)$.

The age at the receiver increases linearly in time in the absence of any update deliveries and is reset to a smaller value when an update is received. A small age at the receiver indicates that the receiver has fresh information. Fig. 1.1(a) shows a sample evolution of age $\Delta(t)$, as a function of time t, at the receiver. In this example, an update generated at t_i completes service, i.e., is received at the receiver, at t'_i and updates generated at t_3, t_5, and t_6 do not start service, i.e., they are dropped, as the previous update was in service when these updates are generated. When the update generated at t_1 arrives at the receiver at time t'_1 with $d_1 = t'_1 - t_1$ being the service time, the age drops exactly to d_1 as it is the age of the packet at that time. The time average age of the status updates is the area under the sawtooth curve in Fig. 1.1(a) normalized by the time interval of observation; Fig. 1.1(b) shows the shaded area of one of many trapezoids (plus a triangle between t_4 and t'_4) that make the overall area.

Studying delay or throughput alone is not sufficient to guarantee access to fresh data in real-time communication networks as the age captures not only the

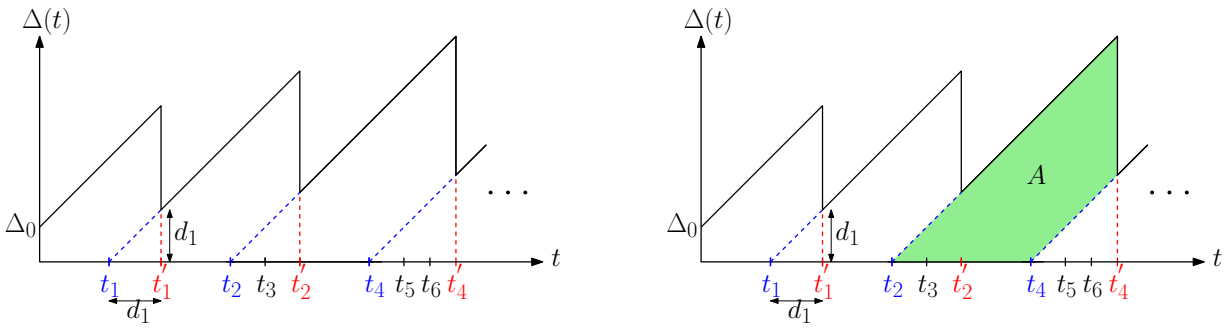


Figure 1.1: Sample AoI evolution curves. Average age evaluation through calculating areas.

packet delay in a communication setting, but also the effects of inter-generation time of update packets. That is, good age performance corresponds to neither high throughput nor low delay. As shown in references [2] and [3], for the optimized age performance, we need regular packet delivery with low delay. The way to achieve the maximum throughput is to send as many updates as possible from the source. However, this may cause congestion in the system resulting in stale packet deliveries at the receiver. Likewise, packet delay in the network can be reduced by decreasing the update frequency which in turn yields outdated information at the receiver since the update delivery rate is low. Thus, age of information studies aim to strike a balance between these two opposing trends to design systems in which the information is received in a timely manner by the receiver nodes.

In the following, we briefly review the rapidly developing literature on age of information; detailed surveys can be found in [4–6]. Age of information has been extensively studied in the context of queueing theory under various interarrival and service profiles and queueing model assumptions [1, 7–28]. References [29] and [30] investigate packet management strategies including blocking and preemption for

M/G/1/1 and G/G/1/1 queues, respectively. References [31–37] study multihop networks in which update packets are relayed from one node to another. References [38] and [39] study the stationary distribution and moments of age in queueing systems. In [40] and [41], the benefits of waiting in status update systems are investigated. Age of information has found further applications in the literature in the context of social networks [42], remote estimation [43–51], scheduling in networks [2, 52–74], energy harvesting systems [75–86], caching and coding [87–102, 102, 103], vehicular and UAV systems [104–107], reinforcement learning [108–112], and so on.

Common to all these works is the fact that they study the analysis and optimization of age of information in systems with small number of source-destination pairs. With increasing connectivity in communication networks and rapidly growing number of information sources supplying time-sensitive information, the issue of scalability of age of information has emerged. This motivates to study how the age performance of the network changes with growing network size.

In this dissertation, our goal is to analyze age of information in a large network setting with many source-destination pairs with a focus on its scalability as a function of the network size and to design timely distributed computation and learning systems that handle highly temporal time-critical data by utilizing potentially large number of worker nodes. Motivated by the ever growing prominence of distributed computation and learning systems, another goal of this dissertation is to utilize the age metric as a tool to improve the performance of the distributed computation and learning systems. In the remainder of this introduction, we summarize the chapters of this dissertation briefly.

In Chapter 2, we study the scalability of age of information in a multihop multicast network with a single source node and a large number nodes at each hop. We first analyze the single-hop problem with exogenous arrivals where the source node directly communicates with n end users but cannot generate the updates itself. We then characterize the average age of information for the two-hop case using our single-hop with exogenous arrivals result as a building block. In the two-hop multicast network, each of the n nodes in the first hop are connected to n further nodes in the second hop such that there are n^2 end users. We show that the average age of the end users is limited by a constant under i.i.d. shifted exponential link delays and stopping thresholds k_1 and k_2 in hops 1 and 2, respectively. That is, the source node transmits each update packet until the earliest k_1 of the n first hop nodes receive that update packet. Each such first hop node relays that particular update packet to the earliest k_2 of the n second hop nodes that are connected to it. We then extend the result of two-hop network to L-hop multicast networks for general L. We determine the optimal stopping threshold for each hop ℓ, k_ℓ, that minimizes the average age for arbitrary shifted exponential link delays.

In Chapter 3, building on Chapter 2, we consider an extended scenario in which two update streams, type I and type II streams, share the same multicast network. That is, we study age of information in a large multicast network where there is a single source node that sends time-sensitive updates to n receiver nodes, where each update packet is either type I or type II. This induces a trade-off among the ages of different update streams. We characterize this trade-off and determine the average age at the receiver nodes for both of the update streams. We analyze

two cases: update streams are generated by the source node at-will and update streams arrive exogenously to the source node. We show that, in both cases, the average age of either of the update streams at an individual node can be made a constant independent of n using stopping thresholds k_1 and k_2 for type I and type II updates, respectively. In particular, the source node transmits each type I update packet to the earliest k_1 and each type II update packet to the earliest k_2 of n receiver nodes. We determine the optimum k_1 and k_2 stopping thresholds for arbitrary shifted exponential link delays to individually and jointly minimize the average age of both update streams and characterize the pareto optimal curve for the two ages.

In Chapter 4, we study age of information in a multiple source-multiple destination setting with a particular focus on its scaling in large wireless networks. Unlike Chapters 2 and 3, where we study a multicast network structure, in Chapter 4, we consider a fixed area network of n randomly located source-destination (S-D) pairs that want to send time-sensitive update packets to each other. Each source node wants to keep its destination node as up-to-date as possible. We divide the network into cells of M users each and propose a three-phase transmission scheme to serve all n S-D pairs such that the time average age of each node is small. The proposed scheme utilizes *local cooperation* among the nodes along with *mega update packets* each of which containing the updates of all nodes from a particular cell. We show that under the proposed scheme average age of an S-D pair scales as $O(n^{\frac{1}{4}} \log n)$ as the number of users, n, in the network grows. Next, we introduce hierarchy to improve the age scaling result. That is, we further divide cells into smaller subcells and

apply the proposed three-phase transmission scheme again. Using this hierarchical scheme, we show that an average age scaling of $O\left(n^{\alpha(h)}\log n\right)$ per-user is achievable where $\alpha(h) = \frac{1}{3\cdot 2^h+1}$ and h denotes the number of hierarchy levels. We note that $\alpha(h)$ tends to 0 as h increases, and asymptotically, the average age scaling of the proposed hierarchical scheme is $O(\log n)$.

In Chapter 5, we adopt the *version age* metric as our timeliness measure, which is a discrete freshness metric that counts how many versions out-of-date a particular receiver is compared to the information at the source node. We investigate version age scaling in *clustered gossip networks*, where we have a single source node and n receiver nodes that are grouped into equal-sized clusters. Each cluster corresponds to a distinct community such that nodes that belong to different communities cannot exchange information. Unlike our scaling analyses in Chapters 2-4, in gossip networks, nodes in the system exchange their current stored version of the source update with their neighboring nodes by *local gossiping*. We use dedicated cluster heads in each cluster to facilitate communication between the source and the nodes within that cluster akin to base stations in a cellular network. Inside clusters, nodes are connected to each other according to a given network topology. We consider disconnected, ring, and fully connected network topologies for each cluster. For each of these network topologies, we characterize the average version age at each node utilizing a stochastic hybrid systems (SHS) approach and find the average version age scaling as a function of the network size n. Our results indicate that per node average version age scalings of $O(\sqrt{n})$, $O(n^{\frac{1}{3}})$, and $O(\log n)$ are achievable in disconnected, ring, and fully connected cluster models, respectively.

Next, we increase connectivity in the network and allow gossiping among the cluster heads to improve version age at the receiver nodes. With that, we show that when the cluster heads form a ring network among themselves, we obtain per node average version age scalings of $O(n^{\frac{1}{3}})$, $O(n^{\frac{1}{4}})$, and $O(\log n)$ in disconnected, ring, and fully connected cluster models, respectively. Then, we introduce hierarchy to the considered clustered gossip network model and show that when we employ h levels of hierarchy, per user average version age scaling of $O(n^{\frac{1}{2h}})$ is achievable in the case of a ring network in each cluster across all hierarchy levels. Finally, we find the version age-optimum cluster sizes as a function of the source, cluster head, and node update rates through numerical evaluations.

In Chapter 6, we consider the problem of timely distributed computation in a system with a single source node and a computation unit (CU) which consists of a single master node and n worker nodes. The source node collects time-sensitive computation-intensive data that need additional processing to extract the useful information and sends them to the CU for processing over a channel with random transmission delays. The master node distributes the received computation task to the worker nodes. Upon computation, the master node aggregates the results and sends them back to the source node to keep it *updated*. We investigate the timeliness in these systems and design computation distribution algorithms that can combat stragglers as well as achieve a minimum average age of information. We derive the average age for uncoded and coded (repetition coded, MDS coded, and multi-message MDS (MM-MDS) coded) schemes in the presence of stragglers, and show that asymptotically, i.e., for large n, MDS coded scheme achieves a smaller

average age than uncoded and repetition coded schemes. Next, we observe that when worker nodes have multiple computations to perform (MM-MDS coded scheme), age performance of the MDS coded scheme is further improved. Finally, we find the optimal codes such that the average age is minimized.

In Chapter 7, we study timeliness in a *federated learning* framework. We have a parameter server (PS) that trains a global model by using n clients without actually storing the client data centrally at a cloud server. Focusing on a setting where the client datasets are fast changing and highly temporal in nature, we investigate the timeliness of model updates and propose a novel timely communication scheme to make sure that locally generated user data are reflected in the learning model with as little age as possible without slowing down the convergence of the global model. Under the proposed scheme, at each iteration, the PS waits for m available clients and sends them the current model. Then, the PS uses the local updates of the earliest k out of m clients to update the global model at each iteration. We find the average age of information experienced by each client and numerically characterize the age-optimal m and k values for a given n. Our results indicate that, in addition to ensuring timeliness, the proposed communication scheme results in significantly smaller average iteration times compared to random client selection without hurting the convergence of the global learning task.

In Chapter 8, we introduce an age-based coded computation framework in a distributed learning system in the presence of *stragglers*. These straggling workers constitute a significant performance bottleneck for the per-iteration completion time in distributed synchronous gradient descent (GD). Coded distributed computation

techniques have been introduced recently to mitigate stragglers and to speed up GD iterations by assigning redundant computations to workers. Partial recovery of the gradient vector can further reduce the computation time at each iteration; however, this can result in biased estimators, which may slow down convergence, or even cause divergence. Estimator bias is particularly prevalent when the straggling behavior is correlated over time, which results in the gradient estimators being dominated by a few fast servers. To mitigate biased estimators, in this chapter, we design a *timely* dynamic encoding framework for partial recovery that includes an ordering operator that changes the codewords and computation orders at workers over time. To regulate the recovery frequencies, we adopt an *age* metric in the design of the dynamic encoding scheme. The proposed age-based scheme prioritizes the recovery of computations with relatively large age. We show through numerical results that the proposed dynamic encoding strategy increases the timeliness of the recovered computations, which, as a result, reduces the bias in model updates, and accelerates the convergence compared to conventional static partial recovery schemes.

In Chapter 9, we focus on the straggler mitigation problem in distributed learning. In this chapter, we introduce a novel paradigm of dynamic coded computation, which assigns more data to the workers than the actual computation load to give the PS certain flexibility to dynamically choose from a set of possible codes for each worker depending on the past straggling behavior. In particular, we propose gradient coding (GC) with dynamic clustering, called GC-DC, and regulate the number of stragglers in each cluster by dynamically forming the clusters at each iteration by utilizing the extra degree-of-freedom offered by the additional data at the work-

ers. Under a time-correlated straggling behavior, GC-DC adapts to the straggling behavior over time; in particular, at each iteration, GC-DC aims to distribute the stragglers across clusters as uniformly as possible based on the past straggler behavior. For both homogeneous and heterogeneous worker models, we numerically show that GC-DC provides significant improvements in the average per-iteration completion time without an increase in the communication load compared to the original GC scheme.

In Chapter 10, we study a system with non i.i.d. service and interarrival times. In particular, we consider a system with a single sampler and a single server and study Gilbert-Elliot servers and samplers. The sampler, i.e., source node, sends time-sensitive status updates to a single monitor node through the server node. First, we consider a Gilbert-Elliot service profile at the server node. In this model, service times at the server node follow a finite state Markov chain with two states: *bad* state b and *good* state g where the server is faster in state g. We determine the time average age at the monitor node and characterize the age-optimal state transition matrix P with and without an average cost constraint on the service operation. Next, we consider a Gilbert-Elliot sampling profile at the source. In this model, the interarrival times follow a finite state Markov chain with two states: *bad* state b and *good* state g where samples are more frequent in state g. We find the time average age experienced by the monitor node and characterize the age-optimal state transition matrix P.

In Chapter 11, we present conclusions of this dissertation.

CHAPTER 2

Age of Information in Multihop Multicast Networks

2.1 Introduction

In this chapter, we study age of information and its scalability in large multicast networks. Reference [42] studies a mobile social network with a single service provider and n communicating users, and shows that under Poisson contact processes among users and uniform rate allocation from the service provider, the average age of the content at the users grows logarithmically in n. In contrast, [113] shows that appropriate stopping threshold k prevents information staleness at single-hop multicast networks with n receiver nodes. Motivated by this observation, in this chapter, we study the scalability of age of information in multihop multicast networks using similar threshold ideas.

We consider a multihop system where in the first hop a single source node broadcasts time-stamped updates to n first-hop receiver nodes using n links with i.i.d. random delays, and in the second hop, each first-hop receiver node relays the update packets it has received to n further nodes that are connected to it. This network architecture continues in further hops such that each receiver node in hop

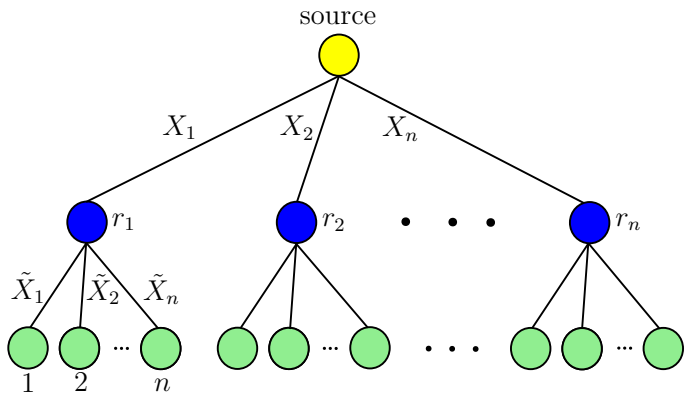


Figure 2.1: Two-hop multicast network with a single source node sending updates through n middle nodes each of which is tied to further n end nodes.

$\ell - 1$ acts as a transmitter in hop ℓ and it is connected to n further receiver nodes in hop ℓ. Fig. 2.1 shows the described model for two hops. We study the age of information experienced by the end nodes, and in particular, its scaling as a function of n.

We first consider the age of information in a two-hop ($L = 2$) multicast network and then extend our results to general multihop multicast networks with L hops, where L is arbitrary and show that, under an earliest k_ℓ scheme at each hop ℓ, a constant average age at the end nodes that is independent of n is achievable. In particular, the source node transmits each update packet to the earliest k_1 of the n first-hop nodes, and each first-hop node that receives the update relays it to the earliest k_2 out of n second-hop nodes that are connected to it and so on. We find age-optimal k_ℓ stopping thresholds at each hop ℓ for arbitrary shifted exponential link delays.

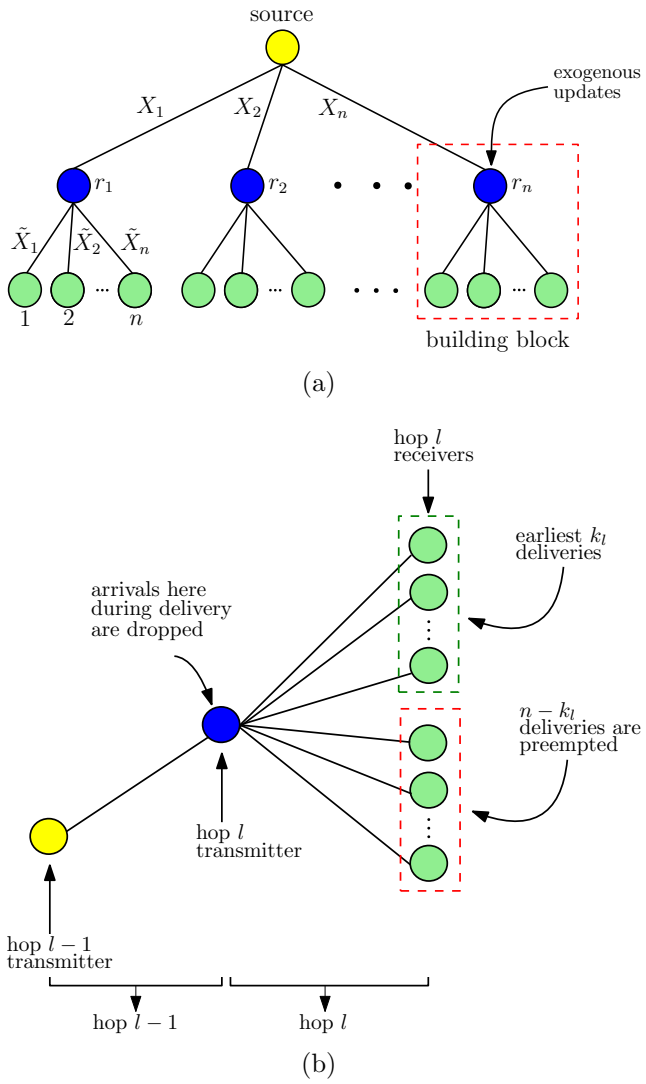


Figure 2.2: Multicast network model: (a) two-hop operation, (b) details of hops $\ell - 1$ and ℓ.

2.2 System Model and Age Metric

We start describing our system model within the simpler two-hop setting. In the two-hop multicast network, an update takes X time to reach from the source node to a particular mid-level node and $\tilde{X}$ time to reach from that mid-level node to a particular end node where X and $\tilde{X}$ are shifted exponential random variables with parameters (λ, c) and $(\tilde{\lambda}, \tilde{c})$, respectively, where c and $\tilde{c}$ are positive constants. Note

that the constant shift parameters capture additional delay that may be experienced along with the exponential link delay and when they are zero random variables corresponding to service times become exponentially distributed. In each hop, the service times of individual links are i.i.d. realizations of random variables X and $\tilde{X}$, e.g., in the first hop service times of individual links are i.i.d. X_i and in the second hop service times of individual links are i.i.d. $\tilde{X}_i$.

In the two-hop scenario, age is measured for each of the n^2 end nodes and for node i at time t age is the random process $\Delta_i(t) = t - u_i(t)$ where $u_i(t)$ is the time-stamp of the most recent update at that node. When the source node sends out update j, it waits for the acknowledgment from the earliest k_1 of n middle nodes. After it receives all k_1 acknowledgment signals, we say that update j has been completed and the source node generates update $j+1$. At this time, transmissions of the remaining $n-k_1$ packets are terminated. Thus, if a node in the first hop is not in the earliest k_1 nodes to receive update j then service of this update is preempted. In the second hop, these earliest k_1 nodes that have received update j start transmitting this update to their end nodes and they stop whenever k_2 of their end nodes have received the current update. Middle nodes implement a blocking scheme when they are busy transmitting to the end nodes, i.e., they discard arriving packets when they are not idle. When the middle nodes finish transmitting the current update to k_2 of their children nodes, they wait for the arrival of the next update from the source node.

For general L, principles that are explained above are repeated at every hop, e.g., the transmitters in hop ℓ wait for k_ℓ of their children nodes to receive the

current update before they declare that the current update has been completed; they drop all other incoming updates from transmitters in hop $\ell - 1$ as they are transmitting the current update; and when their update is received by k_ℓ receivers they preempt the remaining $n - k_\ell$ updates. When this is over, transmitters in hop ℓ wait for the next update from their parent nodes in hop $\ell - 1$; see Fig. 2.2(b).

The metric we use, time averaged age, is given by

$$\Delta = \lim_{\tau \to \infty} \frac{1}{\tau} \int_0^\tau \Delta(t) dt, \tag{2.1}$$

where $\Delta(t)$ is the instantaneous age of the last successfully received update as defined above. We will use a graphical argument similar to [113] to derive the average age at an individual end node. Since all link delays are i.i.d. for all nodes and packets, each end node i experiences statistically identical age processes and will have the same average age. Therefore, it suffices to focus on a single end user for the age analysis.

2.3 Building Block: Single-Hop Network with Exogenous Arrivals

We first note that at the second hop what we essentially have is n parent nodes each tied to n children nodes. Therefore, each second hop transmitter and its children nodes correspond to the single-hop network analyzed in [113] with one important difference: second hop transmitters cannot generate update packets. They can only relay packets sent from the source node. Thus, in this section, we first analyze a single-hop network in which update packets arrive exogenously with a given expected

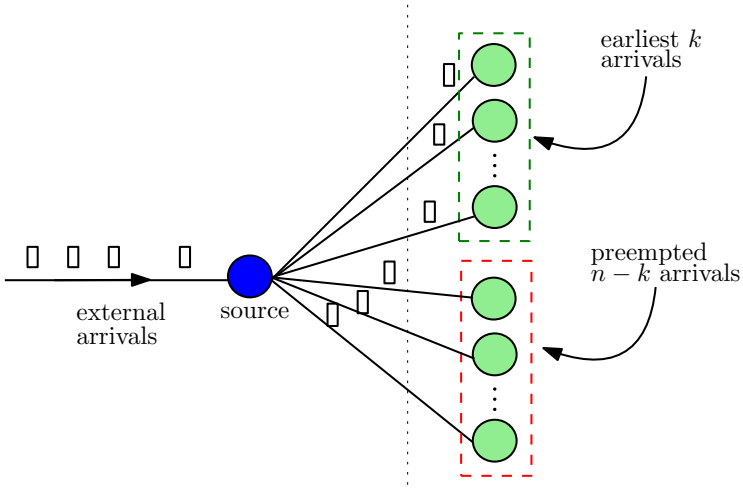


Figure 2.3: The arrival and update structure of a building block.

interarrival time $\frac{1}{\mu}$. Then, using this network as a building block we analyze the L-hop network described in Section 2.2, first for $L = 2$, i.e., a two-hop network, then for general L. We have i.i.d. shifted exponential service times between the source and each of its n children nodes as in [113]. Similarly, transmission of the current update stops when k out of n nodes receive the update. Thus, in this section, we extend the results of [113] to the case of exogenous arrivals, and determine a k threshold which depends on λ, c and μ. Fig. 2.3 shows the arrival and update structure of a building block. Fig. 2.2(a) shows this building block as part of a two-hop network; and Fig. 2.2(b) shows it as a part of an ℓth hop in an L-hop network.

Under this model with i.i.d. link delay X, an update takes $X_{k:n}$ units of time to reach k out of n nodes where we denote the kth order statistic of random variables $X_1, \ldots, X_n$ as $X_{k:n}$. Here $X_{k:n}$ is the kth smallest of $X_1, \ldots, X_n$, e.g., $X_{1:n} = \min\{X_i\}$ and $X_{n:n} = \max\{X_i\}$. For shifted exponential random variable X,

we have

$$\mathbb{E}[X_{k:n}] = c + \frac{1}{\lambda}(H_n - H_{n-k}), \tag{2.2}$$

$$\mathrm{Var}[X_{k:n}] = \frac{1}{\lambda^2}(G_n - G_{n-k}), \tag{2.3}$$

where $H_n = \sum_{j=1}^{n} \frac{1}{j}$ and $G_n = \sum_{j=1}^{n} \frac{1}{j^2}$. Using these,

$$\mathbb{E}[X_{k:n}^2] = c^2 + \frac{2c}{\lambda}(H_n - H_{n-k}) + \frac{1}{\lambda^2}\left((H_n - H_{n-k})^2 + G_n - G_{n-k}\right). \tag{2.4}$$

We say that the system is busy when a transmitted update has not been received by k out of n nodes yet. Once k end nodes receive an update, transmitter stops the service to the remaining $n-k$ nodes. Then, the system is idle until the next update arrives. Fig. 2.4 shows a realization of the update process between the source node and a particular end node. Note that realizations of the update process for different end nodes might be different depending on when and for which updates these nodes have been one of the earliest k nodes. On the other hand, although realizations are different, each end node experiences the same random process. Updates arrive at the source node with an interarrival time, R, where $\mathbb{E}[R] = \frac{1}{\mu}$. In the most general setting, R is an arbitrary i.i.d. random variable. In Fig. 2.4, arrows that are above and below the source node line indicate the received and transmitted updates at the source node, respectively.

An important aspect of our model is that updates at the source node are divided into three groups, namely successful updates, dropped updates, and pre-

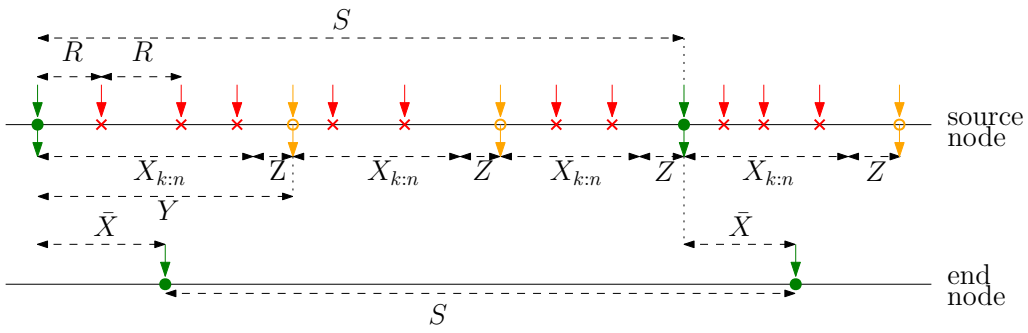


Figure 2.4: Update process for an end node in the building block setting. Filled circles indicate the arrivals successfully received by this particular end node. Crosses indicate updates that arrive at the source node when it is busy transmitting another update. Empty circles show the updates that are transmitted by the source node but are not received by this particular end node, i.e., they are preempted from the perspective of this particular end node.

empted updates. In Fig. 2.4, filled circles correspond to successfully received updates. They indicate that the update received at the source node started transmission to n end nodes and this particular end node received the update. We denote the time between the transmission and the reception of successful updates (filled circles) as the service time, $\bar{X}$. The order of this particular node might be smaller than k, thus $\bar{X} \leq X_{k:n}$. The source continues the service of the first filled circled update until k nodes receive the update. During this time if new updates arrive at the source, they are dropped; the dropped updates (crosses in Fig. 2.4) never go into service, they are lost. Once k nodes receive this update, system becomes idle and the source waits for the arrival of the next update.

We denote the waiting time until the next arrival with Z. The next arrival which is shown with an empty circle in Fig. 2.4 starts the service. However, for this particular realization of the update process, this particular end node is not one of the earliest k end nodes during the transmission of empty circled update. Since the

source stops service once k end nodes receive the update, this transmitted update never arrives at this particular end node. We denote those updates that start a service but do not arrive at this particular end node as preempted updates. Even though preemption usually means stopping a current service in order to start a new one immediately, here we use the word preemption to mean that the current service is stopped, and a new service will start after an idle period when a new update arrives at the source. However, similar to regular meaning of preemption, in our model as well, current preempted update leaves service and a fresh update takes over.

We denote the time between two consecutive departures from the source as $Y = X_{k:n} + Z$. Note that there are a random number of dropped updates during each realization of Y. In Fig. 2.4, between the first successful (filled circle) update and the next update that starts the service (empty circle), there are three dropped updates (crossed). In other words, the fourth received update is able to start a new service. In addition, there are multiple realizations of Y before the next successful (filled circle) update, since this particular end node is not able to receive empty circled updates. We denote the time between two successful updates, i.e., two consecutive updates that depart from the source and successfully arrive at the end node, with S. Remember that Y is the time between two updates that depart from the source and start the service but are not guaranteed to arrive at this particular end node. We can relate the interarrival times of departing updates and arriving updates using $S = \sum_{i=1}^{M} Y_i$. In Fig. 2.4, this particular end node receives the first and the fourth updates that depart the source and enter the service. Therefore, in this example

we have $M = 3$. In addition, remember that R is the interarrival time between the updates that arrive at the source node. For a general R, there is no closed-form expression with known variables that relates interarrival times at the end node, S, to interarrival times at the source node, R. However, when R is exponential Z is exponential as well due to the memoryless property of the exponential distribution. Thus, for exponential R we have $S = \sum_{i=1}^{M}(X_{k:n} + R)_i$.

When the current update reaches k earliest nodes, the source node terminates the remaining $n - k$ transmissions and begins to wait for the next arrival and then repeats the process upon arrival of the next update packet. Since the link delays are i.i.d., the end users receive the packet in service with probability $p = \frac{k}{n}$. If an end user receives update j and the next one it receives is update $j + M$, then M is geometrically distributed with p with moments

$$\mathbb{E}[M] = \frac{1}{p} = \frac{n}{k}, \qquad \mathbb{E}[M^2] = \frac{2-p}{p^2} = \frac{2n^2}{k^2} - \frac{n}{k}. \tag{2.5}$$

Similar to [113], the average age for the earliest k stopping scheme with exogenous packet arrivals with rate μ is

$$\Delta_{(k,\mu)} = \frac{\mathbb{E}[A]}{\mathbb{E}[S]}, \tag{2.6}$$

where A denotes the shaded area in Fig. 2.5 and S is its length. Remember from Fig. 2.4 that the random variable S is the interarrival time at the end nodes. Inspecting Fig. 2.5 to calculate A, we find $\mathbb{E}[A] = \mathbb{E}[S^2]/2 + \mathbb{E}[S]\,\mathbb{E}[\bar{X}]$. Here, $\bar{X}$ denotes the

service time of a successful update such that $\mathbb{E}[\bar{X}] = \mathbb{E}[X_i | i \in \mathcal{K}]$ where $\mathcal{K}$ is the set of earliest k nodes that receive the update. Now, (2.6) becomes[1]

$$\Delta_{(k,\mu)} = \mathbb{E}[\bar{X}] + \frac{\mathbb{E}[S^2]}{2\,\mathbb{E}[S]}. \tag{2.7}$$

We can write the first two moments of S in terms of Y as

$$\mathbb{E}[S] = \mathbb{E}[M]\,\mathbb{E}[Y], \tag{2.8}$$

$$\mathbb{E}[S^2] = \mathbb{E}[M]\,\mathbb{E}[Y^2] + \mathbb{E}[Y]^2\,\mathbb{E}[M^2 - M]. \tag{2.9}$$

Inserting these into (2.7) we obtain

$$\Delta_{(k,\mu)} = \mathbb{E}[\bar{X}] + \frac{\mathbb{E}[M^2]}{2\,\mathbb{E}[M]}\,\mathbb{E}[Y] + \frac{\mathrm{Var}[Y]}{2\,\mathbb{E}[Y]}. \tag{2.10}$$

In the following theorem, we determine the age of an update at an end node for a single-hop building block model using (2.10).

Theorem 2.1 *For a single-hop building block model with exogenous arrivals that have expected interarrival time $\frac{1}{\mu}$, for the earliest k stopping scheme, the average*

[1]Since the building block studied in this section will eventually be used in the upcoming sections in the analysis of two-hop and L-hop networks, we should emphasize that the definitions of $\bar{X}$ and S do not assume any network structure. As far as (2.7) is considered, it is not important where an update that is received by an end node is generated. $\bar{X}$ is the time that the update spends in the system from the time it is generated (possibly by a node other than the one that relays it to the end node) until the time it is received at the end node. S is the time between two consecutive successfully received updates, regardless of whether the updates are generated at the node that transmits them or not. This reasoning is important for our derivations in multihop multicast networks.

age of an update at an individual end node is

$$\Delta_{(k,\mu)} = \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}[X_{i:n}] + \frac{2n-k}{2k}(\mathbb{E}[X_{k:n}] + \mathbb{E}[Z]) + \frac{\mathrm{Var}[X_{k:n}+Z]}{2(\mathbb{E}[X_{k:n}] + \mathbb{E}[Z])}. \tag{2.11}$$

Proof: The first term comes from $\mathbb{E}[\bar{X}]$ as

$$\mathbb{E}[\bar{X}] = \mathbb{E}[X_j | j \in \mathcal{K}] = \sum_{i=1}^{k}\mathbb{E}[X_{i:n}]Pr[j=i|j\in\mathcal{K}] = \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}[X_{i:n}], \tag{2.12}$$

where we used the fact that, since we have k out of n nodes selected independently and identically in $\mathcal{K}$, we have $Pr[j=i|j\in\mathcal{K}] = \frac{1}{k}$. The second and third terms are obtained upon substitution of $Y = X_{k:n} + Z$ and $\mathbb{E}[M]$ and $\mathbb{E}[M^2]$ expressions given in (2.5) into (2.10). ■

When we have general interarrival times as we have in this problem, $X_{k:n}$ and Z may be dependent. However, with exponential interarrivals we can show their independence using the memoryless property, and simplify the age expression given in (2.11) as follows.

Corollary 2.1 *When the arrival process is Poisson with rate μ, the age of an end node is*

$$\begin{aligned}\Delta_{(k,\mu)} =& \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}[X_{i:n}] + \frac{2n-k}{2k\mu}(\mu\,\mathbb{E}[X_{k:n}] + 1) \\ &+ \frac{\mu\mathrm{Var}[X_{k:n}]}{2(\mu\,\mathbb{E}[X_{k:n}]+1)} + \frac{1}{2(\mu^2\,\mathbb{E}[X_{k:n}]+\mu)}.\end{aligned} \tag{2.13}$$

Proof: When the arrival process is Poisson with μ, in other words, R is exponential

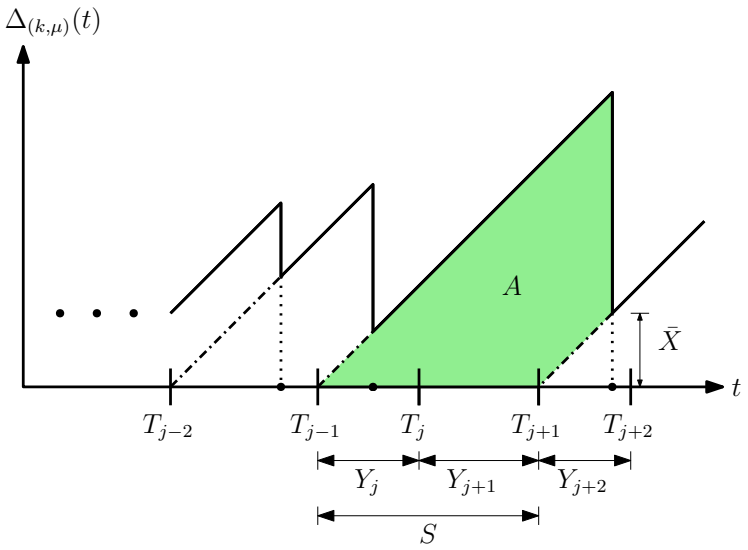


Figure 2.5: Sample age evolution $\Delta_{(k,\mu)}(t)$ of an end node. Updates that find the system idle arrive at times T_i at the source. Here, update j arrives at time T_{j-1} and immediately goes into service. Successful update deliveries are marked with $\bullet$ and in this figure, updates $j-1$, j and $j+2$ are delivered successfully whereas update $j+1$ is terminated.

with expected interarrival time $\frac{1}{\mu}$, the random variable Z which corresponds to the residual interarrival time is exponentially distributed with the same parameter, i.e., $Z = R$. In addition, Z is independent of X due to the memoryless property of the exponential distribution. Then,

$$\text{Var}[Y] = \text{Var}[X_{k:n} + Z] = \text{Var}[X_{k:n}] + \text{Var}[Z], \tag{2.14}$$

and we plug in $\mathbb{E}[Z] = \frac{1}{\mu}$ and $\text{Var}[Z] = \frac{1}{\mu^2}$. ■

When the service times X are i.i.d. shifted exponential random variables and when n is large, we can further simplify the age expression in (2.13) as follows.

Corollary 2.2 *For large n and $n > k$, set $k = \alpha n$. For shifted exponential (λ, c) service times X, the average age for the earliest k scheme with exogenous Poisson*

arrivals with rate μ can be approximated as

$$\begin{aligned}\Delta_{(k,\mu)} \approx & \frac{c}{\alpha} + \frac{c}{2} + \frac{1}{\lambda} - \frac{1}{2\lambda}\log(1-\alpha) + \frac{1}{\alpha\mu} - \frac{1}{2\mu} \\ & + \frac{1}{2}\left(\mu^2 c - \frac{\mu^2 \log(1-\alpha)}{\lambda} + \mu\right)^{-1}. \end{aligned} \tag{2.15}$$

Proof: Using the order statistics above,

$$\delta_1 = \frac{1}{k}\sum_{i=1}^{k} \mathbb{E}[X_{i:n}] = c + \frac{H_n}{\lambda} - \frac{1}{k\lambda}\sum_{i=1}^{k} H_{n-i}. \tag{2.16}$$

As in [113], we have $\sum_{i=1}^{k} H_{n-i} = \sum_{i=1}^{n-1} H_i - \sum_{i=1}^{n-k-1} H_i$ and the series identity $\sum_{i=1}^{k} H_i = (k+1)(H_{k+1} - 1)$. Using these we get

$$\delta_1 = c + \frac{1}{\lambda} - \frac{n-k}{k\lambda}(H_n - H_{n-k}) \approx c + \frac{1}{\lambda} + \frac{1-\alpha}{\alpha\lambda}\log(1-\alpha), \tag{2.17}$$

since for large n, we have $H_i \approx \log(i) + \gamma$. Also,

$$\delta_2 = \frac{2n-k}{2\mu k}(\mu\,\mathbb{E}[X_{k:n}] + 1) = \frac{2n-k}{2\mu k}\left(\mu\left(c + \frac{H_n - H_{n-k}}{\lambda}\right) + 1\right) \tag{2.18}$$

$$\approx \frac{(2-\alpha)c}{2\alpha} + \frac{\alpha-2}{2\alpha\lambda}\log(1-\alpha) + \frac{2-\alpha}{2\alpha\mu}. \tag{2.19}$$

Next, we note that we have

$$\lim_{n\to\infty} \frac{\mu \mathrm{Var}[X_{k:n}]}{2(\mu\,\mathbb{E}[X_{k:n}] + 1)} = 0. \tag{2.20}$$

We see this from the expected values of order statistics,

$$\frac{\mu \mathrm{Var}[X_{k:n}]}{2(\mu\,\mathbb{E}[X_{k:n}]+1)} = \frac{\mu(G_n - G_{n-k})}{2(\mu\lambda^2 c + \mu\lambda(H_n - H_{n-k}) + \lambda^2)}. \tag{2.21}$$

We know that the sequence G_n converges to $\frac{\pi^2}{6}$. As n increases $G_{n-k} = G_{(1-\alpha)n}$ also goes to the same value making the numerator 0. Thus, as n tends to ∞ (2.20) is achieved. Similarly,

$$\delta_3 = \frac{1}{2(\mu^2\,\mathbb{E}[X_{k:n}]+\mu)} \approx \frac{1}{2}\left(\mu^2 c - \frac{\mu^2 \log(1-\alpha)}{\lambda} + \mu\right)^{-1}. \tag{2.22}$$

Summing δ_1, δ_2, and δ_3 yields the expression in (2.15). ■

Note that the age expression in (2.15) when n is large is a function of the ratio $\alpha = \frac{k}{n}$ only implying that the age converges to a constant even when the packets arrive exogenously, similar to [113] where the packets are generated at will at the source.

Although there is no explicit closed form solution for the optimal α, denoted as α^*, which minimizes (2.15), we can calculate it numerically. For instance, when $(\lambda, c) = (1, 1)$ and Poisson arrival rate is $\mu = 1$, age minimizing α is $\alpha^* = 0.837$. This optimal value is higher than that of the original case in which the source itself generates the packets at will, which is $\alpha^* = 0.732$ found in [113]. This is because with exogenous arrivals at the source node, update interarrival time at the end nodes is larger. In this regard, we see that when the Poisson arrival rate is increased α^* decreases. This is expected because when the arrival rate is high (i.e.,

update packets arrive frequently), the source node prefers to wait for the *freshest* one instead of sending the current update to more and more end users. Similarly, when the arrival rate is low (i.e., update packets arrive infrequently), α^* is higher because in this case source knows that interarrival time is higher so that before it waits for the next packet it wants to update as many end nodes as it can (see Fig. 2.6(a)). We also note that as we take $\mu \to \infty$ in (2.15), we get

$$\Delta_{(k)} \approx \frac{c}{\alpha} + \frac{c}{2} + \frac{1}{\lambda} - \frac{1}{2\lambda}\log(1-\alpha), \tag{2.23}$$

which is the age expression in [113]. Thus, age expression under exogenous Poisson arrivals with rate μ converges to the case in which source generates the packets itself as μ tends to ∞.

Finally, we note an interesting aspect of the problem with exogenous arrivals: It is shown in [113] that when the service time random variable X is exponential (i.e., shift variable is zero), the average age is minimized when $k = 1$. However, this is not the case when updates arrive exogenously. We observe in Fig. 2.6(b) that the age minimizing k value can be greater than 1, and it depends on the update arrival rate, μ. As μ increases, the optimal k decreases and approaches 1. The reason for this is that the random waiting time with exogenous arrivals introduces a random shift to the exponential distribution of the service time.

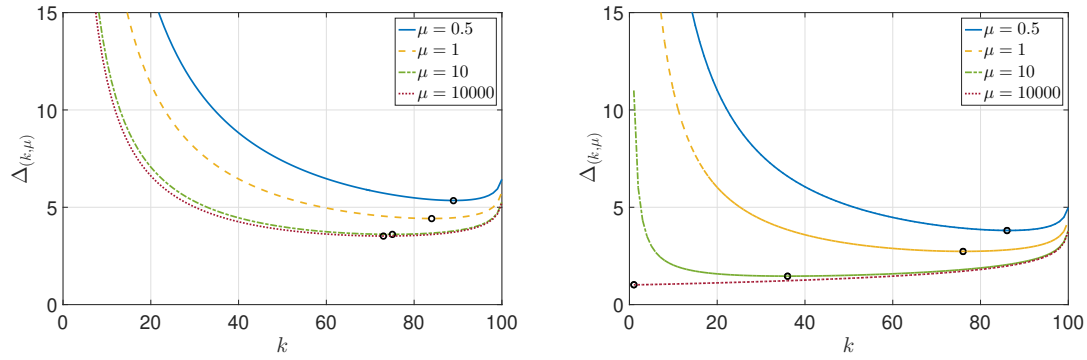


Figure 2.6: $\Delta_{(k,\mu)}$ as a function of stopping threshold k for several μ values with $\lambda = 1$. $\circ$ marks the minimized $\Delta_{(k,\mu)}$: (a) when $c = 1$, (b) when $c = 0$.

2.4 Two-Hop Network

Using the building block problem solved in the previous section, we are now ready to solve the two-hop problem ($L = 2$) described in Section 2.2 as a preliminary step towards solving the most general case for arbitrary L. In the two-hop system, middle nodes cannot generate updates rather they receive them from the source node. Thus, each middle node and its n children nodes in the second hop can be modeled as in Section 2.3. Since the source node sends updates to the first k_1 of its nodes, a middle node receives a certain update packet with probability $p_1 = \frac{k_1}{n}$. If a middle node receives update j and the next one it receives is update $j + M_1$, as in the building block problem, M_1 is a geometrically distributed random variable with parameter p_1.

Since the source generates a new update at will once the current update is delivered to k_1 middle nodes, the interarrival time between updates that start service in the first hop is $Y_1 = X_{k_1:n}$. Let random variable S_1 denote the interarrival time at

the receiver nodes in the first hop. In other words, let S_1 denote the time between updates that leave service in the first hop. Since each update cycle takes $X_{k_1:n}$ units of time, successful interarrival time at the middle nodes can be written as

$$S_1 = \sum_{i=1}^{M_1} (Y_1)_i = \sum_{i=1}^{M_1} (X_{k_1:n})_i, \tag{2.24}$$

where the mean interarrival time is $\mathbb{E}[S_1] = \mathbb{E}[M_1]\,\mathbb{E}[X_{k_1:n}]$. Note that we have $Z_1 = 0$ for the source node since it generates the updates as soon as the previous one is completed (transmitted to k_1 nodes). The receiver nodes in the first hop immediately relay the update that they have received. Therefore, the interarrival time between the successfully received updates at the receiver nodes in the first hop, S_1, is equal to update interarrival time at the transmitter nodes in the second hop, R_2.

After receiving the updates with interarrival time $R_2 = S_1$, middle nodes transmit each update until it is delivered to k_2 of their children nodes. Similar to the first hop, when a middle node transmits an update, an end node receives the update with probability $p_2 = \frac{k_2}{n}$. Geometrically distributed M_2 with parameter p_2 denotes the number of cycles between successive updates to an end node. The interarrival time between updates that depart from a middle node and start service in the second hop is $Y_2 = \tilde{X}_{k_2:n} + Z_2$. Let random variable S_2 denote the interarrival time between updates that successfully arrive at the receiver nodes in the second

hop. Then,

$$S_2 = \sum_{i=1}^{M_2} (Y_2)_i = \sum_{i=1}^{M_2} (\tilde{X}_{k_2:n} + Z_2)_i. \tag{2.25}$$

Note that in this model a successful update reaches an end node without being preempted in each of the hops. Thus, the service time of a successful update denoted by $\bar{X}$ is the sum of link delays in each hop and corresponds to the total time spent in the system by that update. Then, the total service time of a successful update delivered to some node i through middle node j is $\mathbb{E}[\bar{X}] = \mathbb{E}[X_j | j \in \mathcal{K}] + \mathbb{E}[\tilde{X}_i \ | i \in \mathcal{K}_j]$. Here the set $\mathcal{K}$ is the set of first k_1 middle nodes that receive the update and the set $\mathcal{K}_j$ defined for each j in $\mathcal{K}$ is the set of first k_2 end nodes that receive the update. Thus, for an update to reach an end node that end node has to be among the earliest k_2 children nodes of its middle node and the corresponding middle node has to be one of the earliest k_1 middle nodes. Now, by using (2.10), the average age of an end node for a two-hop system is given in the following theorem.

Theorem 2.2 *For a two-hop system with the earliest k_1, k_2 stopping scheme, the average age at an individual end node is*

$$\begin{aligned}\Delta_{(k_1,k_2)} =& \frac{1}{k_1} \sum_{i=1}^{k_1} \mathbb{E}[X_{i:n}] + \frac{1}{k_2} \sum_{i=1}^{k_2} \mathbb{E}[\tilde{X}_{i:n}] + \frac{2n - k_2}{2k_2} (\mathbb{E}[\tilde{X}_{k_2:n}] + \mathbb{E}[Z_2]) \\ &+ \frac{\mathrm{Var}[\tilde{X}_{k_2:n} + Z_2]}{2(\mathbb{E}[\tilde{X}_{k_2:n}] + \mathbb{E}[Z_2])}.\end{aligned} \tag{2.26}$$

Proof: This theorem follows from Theorem 2.1 upon observing that the second

hop is the same as the building block problem. However, successful updates in the two-hop setting spend time in both hops to reach the end nodes. Noting also that successful updates do not wait in the system till they reach the end nodes, we find

$$\mathbb{E}[\bar{X}] = \frac{1}{k_1}\sum_{i=1}^{k_1}\mathbb{E}[X_{i:n}] + \frac{1}{k_2}\sum_{i=1}^{k_2}\mathbb{E}[\tilde{X}_{i:n}]. \tag{2.27}$$

Using (2.27) in (2.10) yields the theorem. ■

This theorem is valid for any distribution for X and $\tilde{X}$. Random variable Z_2 denotes the residual interarrival time before the next update arrives to the middle node. When we no longer have exponential interarrival times, it is not easy to determine the first and second order statistics of Z_2. However, we can upper bound the average age of our model $\Delta_{(k_1,k_2)}$ with the average age under exponential interarrivals to the middle nodes $\Delta'_{(k_1,k_2)}$ using the following lemma.

Lemma 2.1 *For a two-hop system, let $\Delta'_{(k_1,k_2)}$ denote the average age of an end node under exponential interarrivals to middle nodes with mean $\mathbb{E}[R_2]$. Then, $\Delta_{(k_1,k_2)} \leq \Delta'_{(k_1,k_2)}$.*

The proof of Lemma 2.1 follows from the DMRL (decreasing mean residual life) [114] property of interarrival times and NBUE (new better than used in expectation) [114] property of service times and is provided in Section 2.7. Note that as long as the aforementioned conditions on the interarrival and service time distributions are met, this lemma also applies to the case with $L > 2$ hops. This generalization is made in Section 2.7. Since $\Delta_{(k_1,k_2)} \leq \Delta'_{(k_1,k_2)}$, in order to prove that the average age

in the two-hop system is bounded by a constant, all we need to show is that $\Delta'_{(k_1,k_2)}$ is upper bounded by a constant as the network grows, i.e., n increases.

Corollary 2.3 *For a two-hop system, assuming exponential interarrivals to the middle nodes with $\mathbb{E}[R_2]$, the average age at an end node under the earliest k_1, k_2 stopping scheme is*

$$\begin{aligned}\Delta'_{(k_1,k_2)} =&\frac{1}{k_1}\sum_{i=1}^{k_1}\mathbb{E}[X_{i:n}] + \frac{1}{k_2}\sum_{i=1}^{k_2}\mathbb{E}[\tilde{X}_{i:n}] + \frac{2n-k_2}{2k_2}\,\mathbb{E}[\tilde{X}_{k_2:n}] + \frac{2n^2-nk_2}{2k_1k_2}\,\mathbb{E}[X_{k_1:n}] \\ &+ \frac{k_1 Var[\tilde{X}_{k_2:n}]}{2(k_1\,\mathbb{E}[\tilde{X}_{k_2:n}] + n\,\mathbb{E}[X_{k_1:n}])} + \frac{n^2\,\mathbb{E}[X_{k_1:n}]^2}{2k_1(k_1\,\mathbb{E}[\tilde{X}_{k_2:n}] + n\,\mathbb{E}[X_{k_1:n}])}. \qquad (2.28)\end{aligned}$$

Proof: When the interarrival times to the middle nodes, R_2 are exponential, then $Z_2 = R_2$ due to the memoryless property of the exponential distribution. In addition, we know that $R_2 = S_1$. Therefore, Z_2 is exponential with mean $\mathbb{E}[Z_2] = \mathbb{E}[S_1] = \mathbb{E}[M_1]\,\mathbb{E}[X_{k_1:n}]$. Then, $\mathrm{Var}[Z_2] = \mathbb{E}[S_1]^2 = \mathbb{E}[M_1]^2\,\mathbb{E}[X_{k_1:n}]^2$ where M_1 is geometrically distributed with $p_1 = \frac{k_1}{n}$. Combining these and noting that Z_2 and $\tilde{X}_{k_2:n}$ are independent yields the result. ■

Corollary 2.4 *For a two-hop system, assuming n is large and $n > k_1$ and $n > k_2$ and letting $k_1 = \alpha_1 n$ and $k_2 = \alpha_2 n$, under exponential interarrival assumption to the middle nodes with mean $\mathbb{E}[R_2]$, and shifted exponential service times X with (λ, c) and $\tilde{X}$ with $(\tilde{\lambda}, \tilde{c})$, the average age for the earliest k_1, k_2 scheme can be approximated as*

$$\Delta'_{(k_1,k_2)} \approx \frac{1}{\lambda} + \frac{1}{\tilde{\lambda}} + \frac{\tilde{c}}{\alpha_2} + \frac{\tilde{c}}{2} - \frac{1}{2\tilde{\lambda}}\log(1-\alpha_2) + \frac{2-\alpha_2+2\alpha_1\alpha_2}{2\alpha_1\alpha_2}c$$

$$+\frac{\tilde{\lambda}K_1^2}{2\alpha_1\lambda[\lambda\alpha_1 K_2+\tilde{\lambda}K_1]}+\frac{3\alpha_2-2\alpha_1\alpha_2-2}{2\alpha_1\alpha_2\lambda}\log(1-\alpha_1), \tag{2.29}$$

where $K_1=(\lambda c-\log(1-\alpha_1))$ *and* $K_2=(\tilde{\lambda}\tilde{c}-\log(1-\alpha_2))$.

The proof of Corollary 2.4 is similar to that of Corollary 2.2. With Corollary 2.4 we have showed that $\Delta'_{(k_1,k_2)}$ derived in Corollary 2.3 is independent of n for large n. Since it upper bounds our age expression $\Delta_{(k_1,k_2)}$, we conclude that age under the earliest k_1, k_2 stopping scheme for a two-hop multicast network is also independent of n for large n and is bounded by a constant as the number of end nodes increases.

2.5 Extension to L Hops

In this section, we extend our two-hop age results in (2.26), (2.28), and (2.29) to L hops. Considering an L-hop network, we have a single source node, n first hop receiver nodes, n^2 second hop receiver nodes and extending in this manner, n^L end (L hop) nodes. The network model for $L=2$ is shown in Fig. 2.1 and it is generalized such that each of n nodes of the first hop is tied to n further nodes making n^2 second hop nodes and similarly, each of these n^2 second hop nodes is further connected to n nodes forming n^3 end nodes, and so on, for L hops.

Remember that in the two-hop model, we denote the first hop link delays as X, and the second hop link delays as $\tilde{X}$. In this section, in order to accommodate general L hops, we change our notation so that the first hop link delay is now denoted as $X^{(1)}$, the second hop link delay is now denoted as $X^{(2)}$, the ℓth hop is denoted

as $X^{(\ell)}$, and the last hop link delay is denoted as $X^{(L)}$. For each hop ℓ, we utilize the earliest k_ℓ transmission policy such that once hop $\ell-1$ receiver nodes receive an update they begin to act as hop ℓ transmitter nodes and relay the update they have received to k_ℓ of their n children nodes (see Fig. 2.2). After the packet transmission to k_ℓ children nodes are completed, hop ℓ transmitter nodes start waiting for the next update. Here, random variable Z_ℓ denotes this waiting time upon the completion of an update until the next one arrives. Thus, the interarrival time between two consecutive updates that depart from the transmitter and start service in hop ℓ is $Y_\ell = X^{(\ell)}_{k_\ell:n} + Z_\ell$.

Overall this L-hop network implements the earliest $\{k_\ell\}_{\ell=1}^{L}$ transmission scheme. At each hop, we have random variable M_ℓ which is geometrically distributed with parameter $p_\ell = \frac{k_\ell}{n}$ that represents the number of update cycles between two successive updates that arrive at a receiver in hop ℓ. The interarrival time between two consecutive updates that leave service in hop ℓ without being preempted can be written as

$$S_\ell = \sum_{i=1}^{M_\ell} (Y_\ell)_i = \sum_{i=1}^{M_\ell} (X^{(\ell)}_{k_\ell:n} + Z_\ell). \tag{2.30}$$

A receiver node in hop ℓ immediately transmits an update it receives to its children nodes in hop $\ell+1$. Therefore, the interarrival time between two consecutive successful updates that leave service in hop ℓ, S_ℓ, is equal to the interarrival time between two consecutive updates that arrive in hop $\ell+1$, $R_{\ell+1}$. We have $R_{\ell+1} = S_\ell$.

The last hop in this L-hop network can be seen as an application of our

building block problem. In the building block problem, there is a source node and an end node. When this is applied to the last hop of the L-hop network, the source node is an arbitrary receiver node in hop $L-1$, and the end node is an arbitrary receiver node in hop L. Interarrival time of updates that arrive exogenously at the source node in the building block problem, R, is now $R = S_{L-1}$, and the interarrival time between successful updates that arrive at the end node in the building block problem, S, is now $S = S_L$. In addition, we know that $S_L = \sum_{i=1}^{M_L} (Y_L)_i$. Finally, the service time in the building block problem, $\bar{X}$, is now $\bar{X} = \sum_{\ell=1}^{L} \bar{X}_\ell$, where $\bar{X}_\ell$ is the service time at each hop. We know that for an update packet to reach the end nodes, it has to be among the earliest $k_1, \ldots, k_L$ nodes in all hops. Thus, $\sum_{\ell=1}^{L} \bar{X}_\ell$ term captures the expected time spent in the system for a successful update without being preempted until it reaches one of the end nodes. The average age of an end node for an L-hop system is given in the following theorem.

Theorem 2.3 *For the general L-hop network with the earliest $\{k_\ell\}_{\ell=1}^{L}$ stopping scheme, the average age at an individual end node at hop L is*

$$\begin{aligned}\Delta_{\{k_\ell\}_{\ell=1}^{L}} = &\sum_{\ell=1}^{L} \frac{1}{k_\ell} \sum_{i=1}^{k_\ell} \mathbb{E}[X_{i:n}^{(\ell)}] + \frac{2n - k_L}{2k_L} (\mathbb{E}[X_{k_L:n}^{(L)}] + \mathbb{E}[Z_L]) \\ &+ \frac{\mathrm{Var}[X_{k_L:n}^{(L)} + Z_L]}{2(\mathbb{E}[X_{k_L:n}^{(L)}] + \mathbb{E}[Z_L])}. \end{aligned} \tag{2.31}$$

Proof: Since the last hop of the general L-hop network can be seen as a building block, we can apply (2.10) to the setting here by inserting $Y = Y_L$ and $\bar{X} = \sum_{\ell=1}^{L} \bar{X}_\ell$. Then, (2.31) follows after similar calculations as in Theorems 2.1 and 2.2. ■

Similar to Corollary 2.3, the age of the last hop can be upper bounded by assuming that the interarrival times at each hop are exponentially distributed.

Corollary 2.5 *For the general L-hop network, assuming exponential interarrivals to each hop with mean $\mathbb{E}[R_\ell]$, average age at an end node under $\{k_\ell\}_{\ell=1}^L$ stopping scheme is given by*

$$\begin{aligned}\Delta'_{\{k_\ell\}_{\ell=1}^L} =& \sum_{\ell=1}^{L} \frac{1}{k_\ell} \sum_{i=1}^{k_\ell} \mathbb{E}[X_{i:n}^{(\ell)}] + \frac{2n-k_L}{2k_L} \sum_{\ell=1}^{L} \mathbb{E}[X_{k_\ell:n}^{(\ell)}] \prod_{i=\ell}^{L-1} \mathbb{E}[M_i] \\ &+ \frac{\mathrm{Var}[X_{k_L:n}^{(L)}] + \left(\sum_{\ell=1}^{L-1} \mathbb{E}[X_{k_\ell:n}^{(\ell)}] \prod_{i=\ell}^{L-1} \mathbb{E}[M_i]\right)^2}{2(\sum_{\ell=1}^{L} \mathbb{E}[X_{k_\ell:n}^{(\ell)}] \prod_{i=\ell}^{L-1} \mathbb{E}[M_i])}. \end{aligned} \tag{2.32}$$

Proof: Using Lemma 2.1, (2.31) can be upper bounded with exponential interarrivals to the nodes at hop $L-1$, where mean interarrival time is $\mathbb{E}[R_L] = \mathbb{E}[S_{L-1}]$. When the interarrivals are exponentially distributed, we have $Z_L = S_{L-1}$ and therefore, $\mathbb{E}[Z_L] = \mathbb{E}[S_{L-1}]$. In addition, with exponential interarrivals, Z_L is independent of $X_{k_L:n}$. Then, we have $\mathrm{Var}[X_{k_L:n}^{(L)} + Z_L] = \mathrm{Var}[X_{k_L:n}^{(L)}] + \mathrm{Var}[Z_L]$, where $\mathrm{Var}[Z_L] = \mathbb{E}[S_{L-1}]^2$. Now, the upper bound for (2.31) can be written as

$$\begin{aligned}\Delta'_{\{k_\ell\}_{\ell=1}^L} =& \sum_{\ell=1}^{L} \frac{1}{k_\ell} \sum_{i=1}^{k_\ell} \mathbb{E}[X_{i:n}^{(\ell)}] + \frac{2n-k_L}{2k_L} (\mathbb{E}[X_{k_L:n}^{(L)}] + \mathbb{E}[S_{L-1}]) \\ &+ \frac{\mathrm{Var}[X_{k_L:n}^{(L)}] + \mathbb{E}[S_{L-1}]^2}{2(\mathbb{E}[X_{k_L:n}^{(L)}] + \mathbb{E}[S_{L-1}])}. \end{aligned} \tag{2.33}$$

Now, let us calculate $\mathbb{E}[S_{L-1}]$. We can write S_{L-1} as

$$S_{L-1} = \sum_{i=1}^{M_{L-1}} (Y_{L-1})_i = \sum_{i=1}^{M_{L-1}} (X_{k_{L-1}:n}^{(L-1)} + Z_{L-1})_i, \tag{2.34}$$

where Y_{L-1} is the interarrival time between updates that start service in hop $L-1$. Then,

$$\mathbb{E}[Z_L] = \mathbb{E}[S_{L-1}] = \mathbb{E}[M_{L-1}]\,\mathbb{E}[X^{(L-1)}_{k_{L-1}:n}] + \mathbb{E}[M_{L-1}]\,\mathbb{E}[Z_{L-1}]. \tag{2.35}$$

Similarly, $\mathbb{E}[Z_{L-1}]$ can be written in terms of the variables in hop $L-2$. We continue with this recursive calculation until the second hop, where we have $\mathbb{E}[Z_2] = \mathbb{E}[M_1]\,\mathbb{E}[X^{(1)}_{k_1:n}]$. As a result, we have

$$\begin{aligned}\mathbb{E}[S_{L-1}] =& \mathbb{E}[M_{L-1}]\,\mathbb{E}[X^{(L-1)}_{k_{L-1}:n}] + \cdots + \mathbb{E}[M_{L-1}]\,\mathbb{E}[M_{L-2}]\cdots\mathbb{E}[M_2]\,\mathbb{E}[M_1]\,\mathbb{E}[X^{(1)}_{k_1:n}] \\ =& \sum_{\ell=1}^{L-1} \mathbb{E}[X^{(\ell)}_{k_\ell:n}] \prod_{i=\ell}^{L-1} \mathbb{E}[M_i].\end{aligned} \tag{2.36}$$

Adding $\mathbb{E}[X^{L-1}_{k_{L-1}:n}]$ to (2.36), we have

$$\mathbb{E}[X^{L-1}_{k_{L-1}:n}] + \mathbb{E}[S_{L-1}] = \sum_{\ell=1}^{L} \mathbb{E}[X^{(\ell)}_{k_\ell:n}] \prod_{i=\ell}^{L-1} \mathbb{E}[M_i]. \tag{2.37}$$

Finally, by inserting (2.36) and (2.37) into (2.33), we obtain (2.32). ■

Corollary 2.5 introduces an upper bound for average age of an update that is generated at the source node and delivered to any one of the end nodes in hop L. Note that the total number of nodes in the L-hop network is $\sum_{\ell=0}^{L} n^l$. The result of Corollary 2.5 holds for any n. Next, we characterize the effect of increasing n on the scaling of average age.

Corollary 2.6 *Assume n is large and $n > k_\ell$ and let $k_\ell = \alpha_\ell n$ for $\ell = 1, \ldots, L$.*

Under exponential interarrival assumption to each hop with mean $\mathbb{E}[R_\ell]$, and shifted exponential service times $X^{(\ell)}$ with (λ_ℓ, c_ℓ), the average age for the earliest $\{k_\ell\}_{\ell=1}^{L}$ scheme can be approximated as

$$\begin{aligned}\Delta'_{\{k_\ell\}_{\ell=1}^{L}} \approx & \sum_{\ell=1}^{L}\left(c_\ell + \frac{1}{\lambda_\ell} + \frac{1-\alpha_\ell}{\alpha_\ell\lambda_\ell}\log(1-\alpha_\ell)\right) \\ & + \frac{2-\alpha_L}{2\alpha_L}\sum_{\ell=1}^{L}\left[\left(c_\ell - \frac{1}{\lambda_\ell}\log(1-\alpha_\ell)\right)\prod_{i=\ell}^{L-1}\frac{1}{\alpha_i}\right] \\ & + \frac{\left(\sum_{\ell=1}^{L-1}\left[\left(c_\ell - \frac{1}{\lambda_\ell}\log(1-\alpha_\ell)\right)\prod_{i=\ell}^{L-1}\frac{1}{\alpha_i}\right]\right)^2}{2\sum_{\ell=1}^{L}\left[\left(c_\ell - \frac{1}{\lambda_\ell}\log(1-\alpha_\ell)\right)\prod_{i=\ell}^{L-1}\frac{1}{\alpha_i}\right]}. \end{aligned} \tag{2.38}$$

The proof of this corollary is similar to that of Corollaries 2.2 and 2.4. We use the first and second moments of order statistics of shifted exponential random variables and make necessary approximations for large n.

In conclusion, under the assumption that middle nodes receive update packets with exponential interarrivals with means equal to $\mathbb{E}[R_\ell]$, the average age attained at the end nodes depends on n only through ratios α_ℓ. Thus, this result together with Lemma 2.1 imply that the average age of an end node in an L-hop multicast network implementing the earliest $\{k_\ell\}$ transmission scheme is bounded by a constant as n increases.

2.6 Numerical Results

In this section, we provide simple numerical results. In order to optimize the age of information at the end nodes, we need to select appropriate k_i values, i.e., optimum ratios α_i^*, at each hop. From [113], we know that in the single-hop multicast network

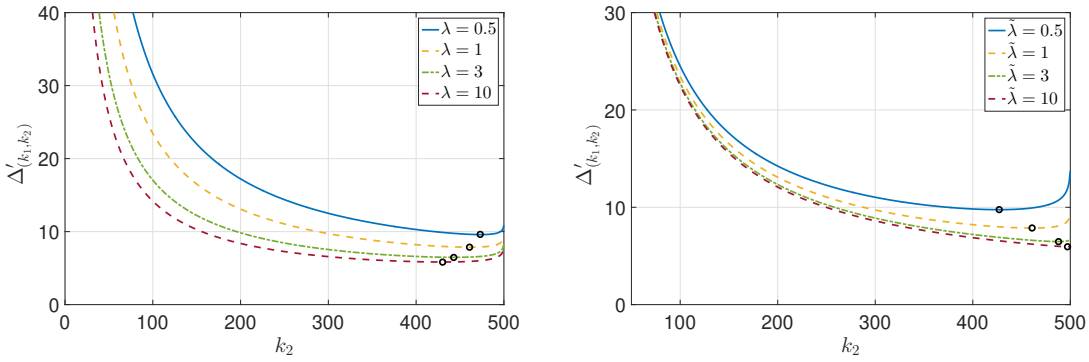
number of hops L	α_1^*	α_2^*	α_3^*	α_4^*
$L=1$	0.732	-	-	-
$L=2$	0.615	0.921	-	-
$L=3$	0.626	0.832	0.965	-
$L=4$	0.635	0.837	0.901	0.935

Table 2.1: Optimal α_i values for L-hop network when all link delays are shifted exponentials with $(1,1)$.

α^* is 0.732 when link delays are shifted exponential with parameters $(\lambda, c) = (1,1)$. For the two-hop network with $(\lambda, c) = (\tilde{\lambda}, \tilde{c}) = (1,1)$, we obtain $\alpha_1^* = 0.615$ and $\alpha_2^* = 0.921$. This shows that when all link delays are statistically identical, to achieve a good age performance, we need to be more aggressive in the second hop than the first hop (see [115]).

When we add a third hop, we see that the optimal threshold results follow similar trends. Let us denote the link delay parameters of the third hop as $(\tilde{\tilde{\lambda}}, \tilde{\tilde{c}})$ for tractability. When all link delays are statistically identical, i.e., shifted exponentials with parameters (1,1), we have $\alpha_1^* = 0.626$, $\alpha_2^* = 0.832$ and $\alpha_3^* = 0.965$. In a similar fashion, when we add a fourth hop, we have $\alpha_1^* = 0.635$, $\alpha_2^* = 0.837$, $\alpha_3^* = 0.901$ and $\alpha_4^* = 0.935$. Thus, the observation we have made for two-hop multicast networks, i.e., that we need to be more aggressive in the further hops to achieve a lower average age holds true for four-hop multicast networks as well. These results are summarized in Table 2.1.

Returning to the two-hop network, we observe that α_2 is responsive to the changes in the parameters of the first hop. This is intuitive because as k_1 varies, the mean interarrival time for the second hop changes. In Figs. 2.7(a) and 2.7(b), we plot



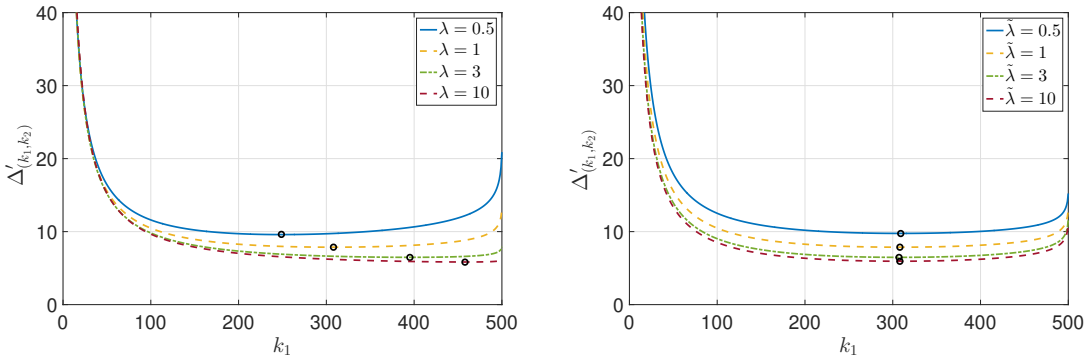
(a) (b)

Figure 2.7: $\Delta'_{(k_1,k_2)}$ as a function of k_2 for $n = 500$. $\circ$ marks the minimized average age $\Delta'_{(k_1,k_2)}$: (a) $c = 1$ and $(\tilde{\lambda}, \tilde{c}) = (1, 1)$ for varying λ, (b) $\tilde{c} = 1$ and $(\lambda, c) = (1, 1)$ for varying $\tilde{\lambda}$.

the age as a function of k_2 (equivalently α_2) for a fixed set of second hop parameters and for a fixed set of first hop parameters, respectively. As shown in Fig. 2.7(a), for the same $(\tilde{\lambda}, \tilde{c})$ pair, when the mean interarrival time gets lower by increasing λ, α_2^* gets lower as well. Knowing that the next update arrival is not far away, middle nodes tend to wait for the next one instead of sending the current packet to more and more end users when the arrivals are frequent. This is exactly what we have observed in the building block problem with exogenous update arrivals.

In Fig. 2.7(b) we observe that as the service rate of the second hop, $\tilde{\lambda}$, increases for a given λ, α_2^* increases as well. This is intuitive because when λ is fixed, the interarrival time to the middle nodes stays constant as opposed to their increasing service rate. Thus, they relay their current packet to more end nodes.

In Figs. 2.8(a) and 2.8(b), we plot the age as a function of k_1 (equivalently α_1) when the second hop parameters are fixed and when the first hop parameters are fixed, respectively. In Fig. 2.8(a), α_1^* shows a similar trend as in α_2^* in Fig. 2.7(b)



(a) (b)

Figure 2.8: $\Delta'_{(k_1,k_2)}$ as a function of k_1 for $n = 500$. $\circ$ marks the minimized average age $\Delta'_{(k_1,k_2)}$: (a) $c = 1$ and $(\tilde{\lambda}, \tilde{c}) = (1, 1)$ for varying λ, (b) $\tilde{c} = 1$ and $(\lambda, c) = (1, 1)$ for varying $\tilde{\lambda}$.

but α_1^* experiences bigger changes as λ increases. Fig. 2.8(b) shows the response of α_1^* to changes in $\tilde{\lambda}$ when λ is fixed. Here, we observe rather minor reaction from α_1^* when the service performance of the second stage varies.

In Fig. 2.9 we repeat the numerical analysis in Fig. 2.7 when the link delays are exponential, i.e., shift parameters $c = 0$ and $\tilde{c} = 0$. We observe similar trends for k_2^* as in Fig. 2.7. The only difference is now that the link delays are all exponential, we get $k_1^* = 1$ for all cases. This is also observed in [113] and in the building block problem when μ tends to ∞ and is because of the memoryless property of the exponential distribution.

In Fig. 2.10 we analyze the effects of the link delay parameters λ and $\tilde{\lambda}$ on $\Delta'_{(k_1,k_2)}$. In both cases we see that larger service rates (large λ and $\tilde{\lambda}$), i.e., lower link delays, lead to smaller average age at the end nodes.

In Fig. 2.11 we plot the three-hop version of Fig. 2.7. As shown in Fig. 2.11, when the mean interarrival time decreases at the third hop through a service rate

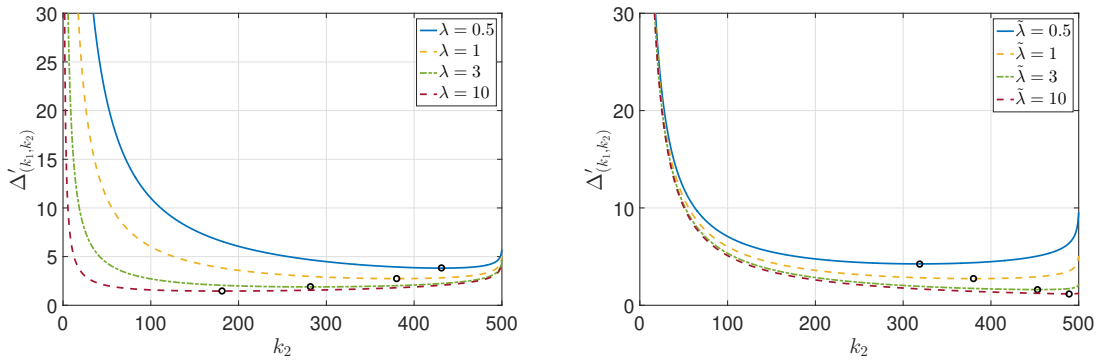
increase in either one of the first two hops α_3^* gets smaller. One other observation from Fig. 2.11 is the fact that although we change λ values in Fig. 2.11(a) and $\tilde{\lambda}$ values in Fig. 2.11(b), their effects on the third hop are quite similar. In either case we observe a similar trend, i.e., k_3 value decreases as λ or $\tilde{\lambda}$ increases.

2.7 Appendix: Proof of Lemma 2.1

In this section, we prove Lemma 2.1, first, for the two-hop scenario, then we extend the proof to the L-hop case. We use [116, Thm. 2] which, for a G/G/1/1 system, requires interarrival times to have DMRL property and service times to have NBUE property. Note that the updates that depart the service at the first hop, S_1 are considered to be the arrivals at the second hop, R_2. Then, between a transmitter and a receiver at the second hop, we have interarrivals $R_2 = \sum_{i=1}^{M_1}(X_{k_1:n})_i$ and service times $\tilde{X}_{k_2:n}$. In order to employ [116, Thm. 2], we need to show that R_2 has DMRL property and $\tilde{X}_{k_2:n}$ has NBUE property. It is sufficient to show that both R_2 and $\tilde{X}_{k_2:n}$ have log-concave density, since log-concavity implies both DMRL and NBUE properties [114].

It is given in [117, Thm. 1.C.54] that the order statistics of i.i.d. log-concave random variables is log-concave as well. Since shifted exponential has a log-concave density, we conclude that the second hop service time $\tilde{X}_{k_2:n}$ has a log-concave density.

In order to show the log-concavity of R_2, we use the fact that a function is log-concave if and only if it is a Polya Frequency function of order 2 [114, Proposition 21.B.8], which is denoted by PF_2. We know that the geometric random variable

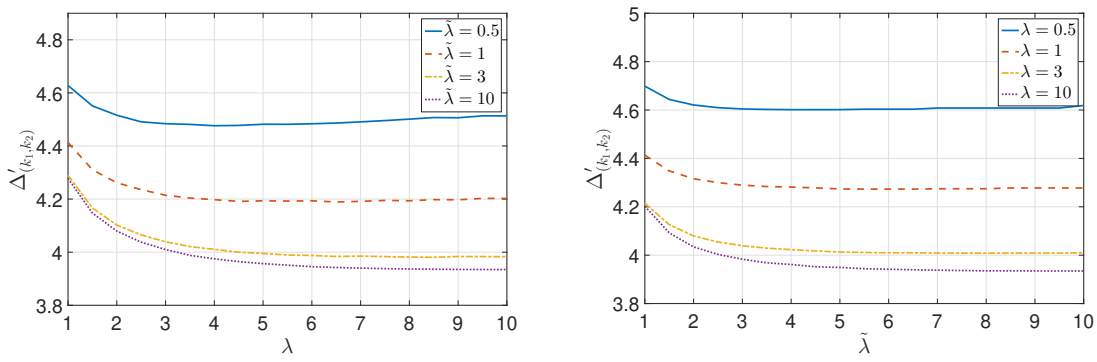


(a) (b)

Figure 2.9: $\Delta'_{(k_1,k_2)}$ as a function of k_2 for $n = 500$ when link delays are exponential. $\circ$ marks the minimized average age $\Delta'_{(k_1,k_2)}$: (a) $c = 0$ and $(\tilde{\lambda}, \tilde{c}) = (1, 0)$ for varying λ, (b) $\tilde{c} = 0$ and $(\lambda, c) = (1, 0)$ for varying $\tilde{\lambda}$.

M_1 and shifted exponential random variables $X_{k_1:n}$ are log-concave, and hence they have PF_2 densities. In addition, we know that M_1 is independent of $X_{k_1:n}$. Now, we can apply [118, Thm. 6] that states that if $X_{k_1:n}$ and M_1 have PF_k densities, and M_1 is independent of $X_{k_1:n}$, then $R_2 = \sum_{i=1}^{M_1}(X_{k_1:n})_i$ has a PF_k density as well. Since [118, Thm. 6] is stated for any k, it holds for $k = 2$, and we conclude that R_2 has a PF_2 density, which, in turn, means that R_2 is log-concave.

We have proved that interarrivals to the second hop are DMRL and service time of the second hop is NBUE. Hence, using [116, Thm. 2] average age of the second hop is upper bounded by a system where interarrivals are exponential with $\mathbb{E}[R_2]$. In the third hop, service times are i.i.d. with the service times at the second hop. Therefore, service time of the third hop is NBUE as well. The interarrivals at the third hop are now $R_3 = \sum_{i=1}^{M_2}(\tilde{X}_{k_2:n} + Z_2)_i$. In order to calculate the upper bound in the second hop, we have already assumed that the interarrivals, R_2, are exponential which resulted in $Z_2 = R_2$ to be exponential as well. Then, R_3 can



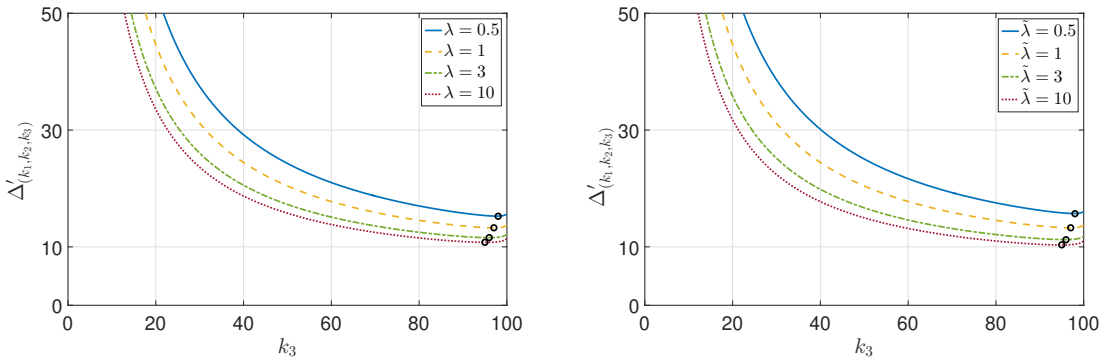
(a) (b)

Figure 2.10: $\Delta'_{(k_1,k_2)}$ as a function of link delay parameters λ and $\tilde{\lambda}$ for $n = 500$: (a) $\Delta'_{(k_1,k_2)}$ as a function of λ, (b) $\Delta'_{(k_1,k_2)}$ as a function of $\tilde{\lambda}$.

be shown to be log-concave using similar ideas above and for exponential Z_2. This reasoning holds for each hop in an L-hop system, which proves the applicability of Lemma 2.1 to an L-hop system as well.

2.8 Conclusion

We studied the age of information in a multihop multicast network with a single source updating n^L end nodes where L denotes the number of hops. We showed that when the earliest k_ℓ transmission policy is implemented at each hop ℓ, the age of information at the end nodes can be upper bounded by a constant that is independent of n. We explicitly characterized an upper bound for an L-hop multicast network, and then determined the optimal stopping thresholds k_ℓ for arbitrary shifted exponential link delays. We note that even when the link delays are exponential we find $k_1^* = 1$ and $k_\ell^* > 1$ in hops $\ell > 1$. This is because even when there is no shift in service, the random waiting time under exogenous arrivals introduces a random



(a) (b)

Figure 2.11: $\Delta'_{(k_1,k_2,k_3)}$ as a function of k_3 for $n = 100$. $\circ$ marks the minimized average age $\Delta'_{(k_1,k_2,k_3)}$: (a) $c = 1$ and $(\tilde{\lambda}, \tilde{c}) = (\tilde{\tilde{\lambda}}, \tilde{\tilde{c}}) = (1, 1)$ for varying λ, (b) $\tilde{c} = 1$ and $(\lambda, c) = (\tilde{\tilde{\lambda}}, \tilde{\tilde{c}}) = (1, 1)$ for varying $\tilde{\lambda}$.

shift to the exponential service distribution in hops $\ell > 1$.

CHAPTER 3

Age of Information in Multicast Networks with Multiple Update Streams

3.1 Introduction

In Chapter 2 and references [115, 119], we show that utilizing transmission schemes with stopping thresholds can prevent information staleness as the network grows in multihop multicast networks considering a single type of update that the users are interested in. In an extended scenario there could be many streams sharing the same network. For example, in an autonomous vehicle network, the network can carry information about the velocity, position and acceleration of a car and broadcast it to all nearby cars. Thus, often networks are used to transmit multiple update streams which include multiple different types of update packets. References [12,23,120,121] study age of information at a monitor node which receives multiple streams of update packets with or without different priorities.

In this chapter, we study the age of information in a multicast network with multiple update streams (see Fig. 3.1). Two types of updates are transmitted from the source node to n receiver nodes, namely type I and type II updates. We are

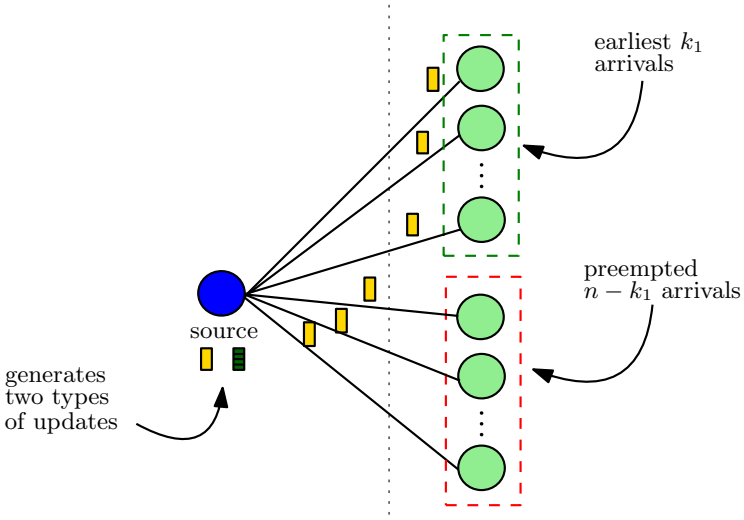


Figure 3.1: Multicast network with a single source node sending two types of updates to n nodes.

interested in the average age experienced by the receiver nodes for both of the update streams. Since the network resources are shared among two different types of updates, there is a trade-off in the ages of these two update streams. In this work, we characterize this trade-off, determine the average age of information of both of the streams and analyze the age scalability when there are large number of destination nodes. We first analyze the setting in which the updates are generated by the source node at-will and show that the average age of either of the update streams at an individual node can be made a constant independent of n using the earliest k_1, k_2 transmission scheme such that each type I update is transmitted to the earliest k_1 of n nodes and each type II update is sent to the earliest k_2 of n nodes. We determine the optimal stopping thresholds to individually and jointly optimize the average age of type I and type II updates at the receiver nodes for arbitrary shifted exponential link delays. Then, we extend these results to the case in which the updates arrive exogenously to the source node.

3.2 System Model and Age Metric

We consider a system (see Fig. 3.1), where there is a single source node broadcasting time-stamped updates to n nodes using n links with i.i.d. random delays. Each status update is one of two kinds: type I or type II. Thus, in this system two different update streams share the same network. Each of the n receiver nodes is interested in both streams. A type I update takes X time to reach from the source node to a particular receiver node whereas a type II update needs $\tilde{X}$ time to reach to a receiver node where X and $\tilde{X}$ are shifted exponential random variables with parameters (λ, c) and $(\tilde{\lambda}, \tilde{c})$, respectively, where c and $\tilde{c}$ are positive constants. Different update types have different service rates considering their possibly different lengths, compression rates etc.

We consider two variations on the operation of this network. In the first case, updates are generated by the source node at-will. At each time the source node generates a type I update with probability p_1 or a type II update with probability p_2 where $p_1 + p_2 = 1$. In the second case, however, both of the update streams arrive exogenously to the source node. Here, we model the exogenous update arrival as a Poisson process with rate μ where each arriving update is a type I update with probability p_1 or a type II update with probability p_2 such that $p_1 + p_2 = 1$. Thus, type I update stream arrives as a Poisson process with rate $\mu_1 = \mu p_1$ and type II stream arrives as a Poisson process with rate $\mu_2 = \mu p_2$.

In either way of operation, the source node adapts the earliest k_i transmission scheme where $i = 1, 2$ depending on the type of the update. Assume the jth update

that is sent from the source node is of type I. Then, the source node waits for the acknowledgment from the earliest k_1 of n receiver nodes. After it receives all k_1 acknowledgment signals, we say that update j has been completed. At this time, transmissions of the remaining $n - k_1$ packets are terminated. If the source node generates the updates itself, it generates the update $j+1$ as soon as the update j has been completed, i.e., the source node implements a zero-wait policy. However, if the updates arrive exogenously, the source node starts to wait for the next update arrival as soon as update j has been completed. When a type II update is transmitted from the source node, this process is repeated with stopping threshold k_2 instead of k_1.

Since we have two different update streams, each receiver node experiences two different age processes. Thus, age is measured for each of the n receiver nodes for each update type. Age of type I updates at node i at time t is the random process $\Delta_{I,i}(t) = t - u_{I,i}(t)$ where $u_{I,i}(t)$ is the time-stamp of the most recent type I update at that node. The metric we use, time averaged age, is given by

$$\Delta_I = \lim_{\tau \to \infty} \frac{1}{\tau} \int_0^\tau \Delta_I(t) dt, \tag{3.1}$$

where $\Delta_I(t)$ is the instantaneous age of the last successfully received type I update as defined above. Age of type II updates, Δ_{II}, is defined accordingly. We will use a graphical argument to derive the average age at an individual node for each update type. Since all link delays are i.i.d. for all nodes and packets of the same kind, each node i experiences statistically identical age processes and will have the same average age for either update type. Therefore, it suffices to focus on a single receiver

node for the age analysis.

3.3 Age Analysis

3.3.1 At-will Update Generation

The source node generates updates when the previous update is completed. Independent from previous link delays and updates, each generated update belongs to type I with probability p_1 and type II with probability p_2 such that $p_1 + p_2 = 1$. As described in Section 3.2, the source node implements the earliest k_1 and k_2 transmission policy for type I and type II updates, respectively, where $k_1, k_2 \in \{1, 2, \ldots, n\}$. Thus, in this section we extend the result from [113] to multiple update streams using the same multicast network.

We denote the time between any two update departures from the source node as the update cycle and represent it with random variable Y. Since the source node adapts a zero-wait policy and generates the next update right after the current update has been completed, update cycle of an update is equal to the transmission time of that update. If the jth update is of type I, its update cycle, Y_j, is the time needed to reach k_1 out of n receiver nodes which is equal to $X_{k_1:n}$. Correspondingly, if the jth update is of type II, Y_j is equal to $\tilde{X}_{k_2:n}$. Thus,

$$Y_j = \begin{cases} X_{k_1:n} & \text{w.p. } p_1 \\ \tilde{X}_{k_2:n} & \text{w.p. } p_2. \end{cases} \tag{3.2}$$

We denote the kth smallest of $X_1, \ldots, X_n$ as $X_{k:n}$. For a shifted exponential random variable X, we have

$$\mathbb{E}[X_{k:n}] = c + \frac{1}{\lambda}(H_n - H_{n-k}), \tag{3.3}$$

$$\mathrm{Var}[X_{k:n}] = \frac{1}{\lambda^2}(G_n - G_{n-k}), \tag{3.4}$$

where $H_n = \sum_{j=1}^{n} \frac{1}{j}$ and $G_n = \sum_{j=1}^{n} \frac{1}{j^2}$. Using these,

$$\mathbb{E}[X_{k:n}^2] = c^2 + \frac{2c}{\lambda}(H_n - H_{n-k}) + \frac{1}{\lambda^2}\left((H_n - H_{n-k})^2 + G_n - G_{n-k}\right). \tag{3.5}$$

A type I (type II) update that is sent from the source node is received by a particular node with probability $q_1 = \frac{k_1}{n}$ $\left(q_2 = \frac{k_2}{n}\right)$ since the X_i $(\tilde{X}_i)$ are i.i.d. for all receiver nodes. Noting the independence between the update generation and update transmission processes, at each cycle a particular node successfully receives a type I (type II) update with probability $p_1 q_1$ $(p_2 q_2)$. Suppose a particular node has received a type I update packet during cycle j and the next successful type I update packet delivery to that node is in cycle $j + M_1$. Then, this M_1 is a geometric random variable with success probability $p_1 q_1$ and has moments

$$\mathbb{E}[M_1] = \frac{1}{p_1 q_1} = \frac{n}{p_1 k_1}, \tag{3.6}$$

$$\mathbb{E}[M_1^2] = \frac{2 - p_1 q_1}{p_1^2 q_1^2} = \frac{2n^2}{p_1^2 k_1^2} - \frac{n}{p_1 k_1}. \tag{3.7}$$

Note that in between update cycles j and $j + M_1$ the source node may have

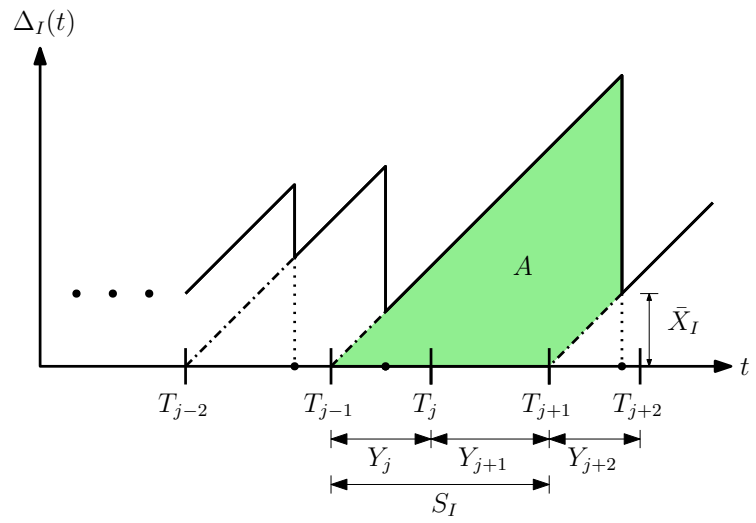


Figure 3.2: Sample age evolution $\Delta_I(t)$ of a node. Update cycle j starts at time T_{j-1} and lasts until T_j. Successful type I update deliveries are marked with $\bullet$ and in this figure, in cycles $j-1$, j and $j+2$ a type I update is delivered successfully whereas in cycle $j+1$ no type I delivery occurred.

sent type II updates. By using the success probability p_1q_1, we account for the time spent during type II update transmissions in between two successful type I update deliveries to this node. Fig. 3.2 shows a sample $\Delta_I(t)$ evolution for a particular node. Correspondingly, if a certain node has received two successive type II updates in update cycles j' and $j'+M_2$ then this M_2 is geometrically distributed with success probability p_2q_2 and its moments can be derived by changing the parameter from p_1q_1 to p_2q_2 in (3.6) and (3.7). We remark that M_1 and M_2 are independent from Y_j.

Noting the symmetry in the age of type I and type II streams at a particular receiver node, Δ_I and Δ_{II}, respectively, here we derive Δ_I and deduce Δ_{II} from it by making necessary changes. In Fig. 3.2, A denotes the shaded area and S_I is its length. In other words, S_I is the interarrival time of type I updates at a node. Inspecting Fig. 3.2, we find $\mathbb{E}[A] = \frac{1}{2}\,\mathbb{E}[S_I^2] + \mathbb{E}[S_I]\,\mathbb{E}[\bar{X}_I]$. Here, $\bar{X}_I$ denotes the

transmission time of a successful type I update such that $\mathbb{E}[\bar{X}_I] = \mathbb{E}[X_i | i \in \mathcal{K}_I]$, where $\mathcal{K}_I$ is the set of earliest k_1 nodes that receive this type I update. Then, the average age of update stream I is given by

$$\Delta_I = \frac{\mathbb{E}[A]}{\mathbb{E}[S_I]} = \mathbb{E}[\bar{X}_I] + \frac{\mathbb{E}[S_I^2]}{2\,\mathbb{E}[S_I]}. \tag{3.8}$$

Event $M_1 = m$ indicates two successive type I deliveries to a node in cycles j and $j+m$ with $Y_j = X_{k_1:n}$ and $Y_{j+m} = X_{k_1:n}$ as well as $m-1$ consecutive type I failures in between. For these $m-1$ update cycles with no type I delivery we have

$$(Y_j \mid \text{no type I delivery}) = \bar{Y}_j = \begin{cases} X_{k_1:n} & \text{w.p. } \bar{p}_1 \\ \tilde{X}_{k_2:n} & \text{w.p. } \bar{p}_2, \end{cases} \tag{3.9}$$

with $\bar{p}_1 = \frac{p_1(1-q_1)}{1-p_1q_1}$ and $\bar{p}_2 = \frac{p_2}{1-p_1q_1}$ from Bayes' rule. Thus, $S_I = X_{k_1:n} + \sum_{i=j+1}^{j+M_1-1} \bar{Y}_i$. For example, in Fig. 3.2, $M_1 = 2$ and therefore, we have $S_I = X_{k_1:n} + \bar{Y}$. Thus, we find

$$\mathbb{E}[S_I] = \mathbb{E}[X_{k_1:n}] + \mathbb{E}[M_1 - 1]\,\mathbb{E}[\bar{Y}], \tag{3.10}$$

$$\begin{aligned} \mathbb{E}[S_I^2] = {} & \mathbb{E}[X_{k_1:n}^2] + 2\,\mathbb{E}[M_1 - 1]\,\mathbb{E}[X_{k_1:n}]\,\mathbb{E}[\bar{Y}] \\ & + \mathbb{E}[M_1 - 1]\mathrm{Var}[\bar{Y}] + \mathbb{E}[(M_1-1)^2]\,\mathbb{E}[\bar{Y}]^2. \end{aligned} \tag{3.11}$$

In the following theorem, we determine the age of a type I update at an individual node using (3.8).

Theorem 3.1 *Under the earliest k_1 and k_2 transmission scheme for type I and type II updates, respectively, the average type I age at an individual node is*

$$\begin{aligned}\Delta_I =& \frac{1}{k_1}\sum_{i=1}^{k_1}\mathbb{E}[X_{i:n}] + \frac{p_1\,\mathbb{E}[X_{k_1:n}^2] + p_2\,\mathbb{E}[\tilde{X}_{k_2:n}^2]}{2p_1\,\mathbb{E}[X_{k_1:n}] + 2p_2\,\mathbb{E}[\tilde{X}_{k_2:n}]} \\ &+ \frac{p_2^2 n\,\mathbb{E}[\tilde{X}_{k_2:n}]^2 + p_1p_2(2n-k_1)\,\mathbb{E}[X_{k_1:n}]\,\mathbb{E}[\tilde{X}_{k_2:n}]}{p_1k_1(p_1\,\mathbb{E}[X_{k_1:n}] + p_2\,\mathbb{E}[\tilde{X}_{k_2:n}])} \\ &+ \frac{p_1^2(n-k_1)\,\mathbb{E}[X_{k_1:n}]^2}{p_1k_1(p_1\,\mathbb{E}[X_{k_1:n}] + p_2\,\mathbb{E}[\tilde{X}_{k_2:n}])}. \end{aligned} \tag{3.12}$$

Proof: The first term comes from $\mathbb{E}[\bar{X}_I]$ as

$$\mathbb{E}[\bar{X}_I] = \mathbb{E}[X_j | j \in \mathcal{K}_I] = \sum_{i=1}^{k_1}\mathbb{E}[X_{i:n}]Pr[j=i|j\in\mathcal{K}_I], \tag{3.13}$$

where $Pr[j=i|j\in\mathcal{K}_I] = \frac{1}{k_1}$ since we have k_1 out of n nodes selected independently and identically in $\mathcal{K}_I$. The claim follows by substituting (3.10), (3.11) and (3.13) back into (3.8), and replacing $\mathbb{E}[M_1]$ and $\mathbb{E}[M_1^2]$ by (3.6) and (3.7). Moments of $\bar{Y}$ follow from (3.9). ■

When $p_1 = 1$, i.e., the source node generates only type I updates, Δ_I given in (3.12) reduces to the single stream result in [113, Theorem 2]. From the symmetry of the network model, the average age expression of type II update stream at an individual node, Δ_{II}, can be found upon defining $\bar{Y}$ accordingly and interchanging k_1, p_1 and X with k_2, p_2 and $\tilde{X}$ in (3.12). When the service times of the packets of the same kind are i.i.d. shifted exponential random variables and n is large, we can further simplify (3.12) as follows.

Corollary 3.1 *For large n and $n > k_i$ we set $k_i = \alpha_i n$ for $i = 1, 2$. For shifted exponential transmission times X and $\tilde{X}$ with parameters (λ, c) and $(\tilde{\lambda}, \tilde{c})$ for type I and type II updates, respectively, Δ_I can be approximated as*

$$\begin{aligned}\Delta_I \approx\ & c + \frac{1}{\lambda} + \frac{1-\alpha_1}{\lambda\alpha_1}\log(1-\alpha_1) \\ & + \frac{(2-\alpha_1)p_1^2\delta_1^2(\alpha_1) + 2p_1p_2(2-\alpha_1)\delta_1(\alpha_1)\delta_2(\alpha_2)}{2p_1\alpha_1(p_1\delta_1(\alpha_1) + p_2\delta_2(\alpha_2))} \\ & + \frac{p_2(p_1\alpha_1 + 2p_2)\delta_2^2(\alpha_2)}{2p_1\alpha_1(p_1\delta_1(\alpha_1) + p_2\delta_2(\alpha_2))},\end{aligned} \tag{3.14}$$

where we denote

$$\delta_1(\alpha_1) = c - \frac{\log(1-\alpha_1)}{\lambda}, \quad \delta_2(\alpha_2) = \tilde{c} - \frac{\log(1-\alpha_2)}{\tilde{\lambda}}. \tag{3.15}$$

Proof: For the first term in (3.12) we have

$$\frac{1}{k_1}\sum_{i=1}^{k_1}\mathbb{E}[X_{i:n}] = c + \frac{H_n}{\lambda} - \frac{1}{k_1\lambda}\sum_{i=1}^{k_1}H_{n-i} \tag{3.16}$$

$$\approx c + \frac{1}{\lambda} + \frac{1-\alpha_1}{\alpha_1\lambda}\log(1-\alpha_1), \tag{3.17}$$

where (3.16) is obtained using (3.3)-(3.5). We have $\sum_{i=1}^{k_1} H_{n-i} = \sum_{i=1}^{n-1} H_i - \sum_{i=1}^{n-k_1-1} H_i$ and the series identity $\sum_{i=1}^{k_1} H_i = (k_1+1)(H_{k_1+1} - 1)$. Using these and the fact that for large n $H_i \approx \log(i) + \gamma$ yields (3.17). Note that

$$\mathbb{E}[X_{k_1:n}] = c + \frac{H_n - H_{n-k_1}}{\lambda} \approx c - \frac{\log(1-\alpha_1)}{\lambda} = \delta_1(\alpha_1), \tag{3.18}$$

$$\mathbb{E}[\tilde{X}_{k_2:n}] = \tilde{c} + \frac{H_n - H_{n-k_2}}{\tilde{\lambda}} \approx \tilde{c} - \frac{\log(1-\alpha_2)}{\tilde{\lambda}} = \delta_2(\alpha_2). \tag{3.19}$$

Lastly, we note that $\mathbb{E}[X^2_{k_1:n}] \approx (\mathbb{E}[X_{k_1:n}])^2$ for large n. This is because the sequence G_n converges to $\frac{\pi^2}{6}$ as n goes to ∞. Thus, as n increases $G_{n-k_1} = G_{(1-\alpha_1)n}$ also goes to $\frac{\pi^2}{6}$. With this, $G_n - G_{n-k_1}$ term in (3.5) approaches 0 yielding the claim. Likewise, we have $\mathbb{E}[\tilde{X}^2_{k_2:n}] \approx (\mathbb{E}[\tilde{X}_{k_2:n}])^2$. Combining all these to calculate the moments of $\bar{Y}$ and taking $k_i = \alpha_i n$ for $i = 1, 2$ gives (3.14). ■

We remark that when $p_1 = 1$, i.e., the source node only generates type I updates, Corollary 3.1 reduces to the single update stream result in [113, Corollary 2]. From the symmetry of the system, average age of type II updates at an individual node, Δ_{II}, can be also approximated as in Corollary 3.1.

We note that under the earliest k_1 and k_2 transmission scheme for updates of type I and type II, respectively, the average age is a function of ratios α_1 and α_2 for large n. Thus, the average age still converges to a constant even though the multicast network is shared across two update streams.

We can minimize Δ_I and Δ_{II} by selecting the stopping thresholds k_1, k_2. Since we have two age processes to minimize, we can consider the following optimization problem for a fixed β such that $0 \leq \beta \leq 1$

$$\begin{aligned} \min_{\{k_1,k_2\}} \quad & \beta\Delta_I + (1-\beta)\Delta_{II} \\ \text{s.t.} \quad & k_i \in \{1, \ldots, n\}, \quad i = 1, 2. \end{aligned} \tag{3.20}$$

Thus, by varying β we can assign weights to the update types I and II during optimization. While extreme cases of $\beta = 1$ and $\beta = 0$ assign absolute priority to update type I and type II, respectively, every other β value weighs the update

streams in between accordingly. Equivalently, we can solve (3.20) over α_1 and α_2. In that case, the constraint in (3.20) is replaced with $0 < \alpha_i < 1$, $i = 1, 2$.

Lemma 3.1 *When n is large, for any $k_1 < n$, to achieve the best type I average age performance, i.e., to minimize Δ_I, it is optimal to select $k_2^* = 1$.*

Proof: This is the $\beta = 1$ case. We observe that for a given $k_1 < n$ the first three terms in (3.14) and $\delta_1 = \mathbb{E}[X_{k_1:n}]$ become constant. The only parameter we select to optimize Δ_I is k_2 and for large n it only appears in (3.14) as $\delta_2(k_2) = \mathbb{E}[\tilde{X}_{k_2:n}]$. Δ_I given in (3.14) is in the following form

$$\Delta_I = c_1 + \frac{c_2\delta_1^2 + c_3\delta_1\delta_2(k_2) + c_4\delta_2^2(k_2)}{c_5\delta_1 + c_6\delta_2(k_2)}, \tag{3.21}$$

where $c_1, \ldots, c_6$ and δ_1 are positive numbers. Noting that $\delta_2(k_2)$ is also positive, (3.21) is an increasing function of $\delta_2(k_2)$ when $c_2c_6 < c_3c_5$ condition is satisfied. From (3.14), we see that this condition is met. Thus, to minimize (3.21) we need to select the smallest possible $\delta_2(k_2)$ which is given by the smallest possible k_2 since $\delta_2(k_2)$ is monotonically increasing in k_2. Thus, $k_2^* = 1$ minimizes the type I average age, Δ_I, for any $k_1 < n$. ■

Similarly, when we are only interested in minimizing the type II average age, Δ_{II}, i.e., $\beta = 0$, it is optimal to select $k_1^* = 1$. Fig. 3.3(a) shows the pareto optimal curve obtained upon numerically solving (3.20) for $p_1 = p_2 = 0.5$ and shifted exponential link delays with $(1, 1)$ for both types of updates.

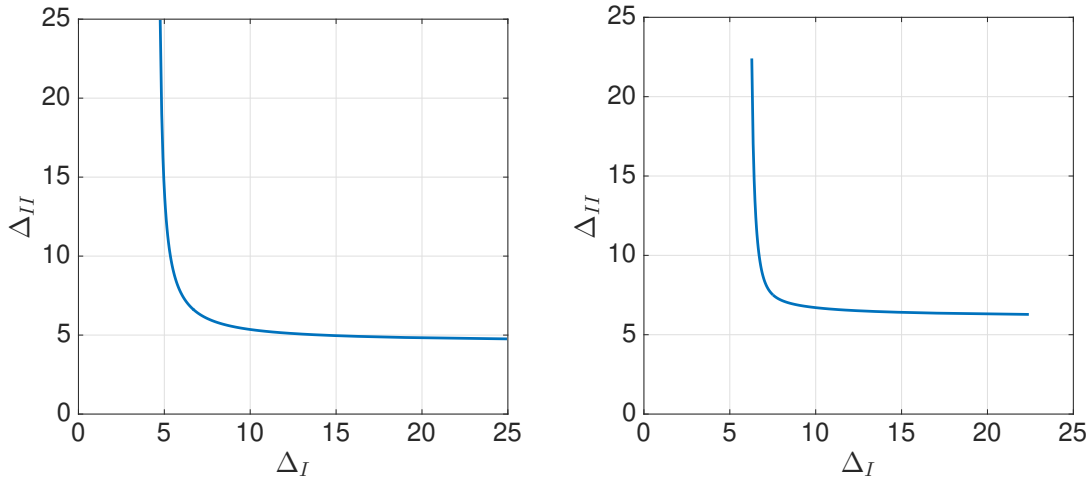


Figure 3.3: Pareto optimal curve for jointly minimizing Δ_I and Δ_{II} with $0 < \beta < 1$, $p_1 = p_2 = 0.5$ and $(\lambda, c) = (\tilde{\lambda}, \tilde{c}) = (1, 1)$ when (a) updates are generated at-will (b) updates arrive exogenously with a total rate of $\mu = 1$.

3.3.2 Exogenous Update Arrivals

Updates arrive at the source node exogenously as a Poisson process with a total rate of μ. Each arriving update is of type I with probability p_1 and of type II with probability p_2 where $p_1 + p_2 = 1$. Thus, type I update stream arrives as a Poisson process with rate $\mu_1 = \mu p_1$ and analogously, type II update stream arrives as a Poisson process with rate $\mu_2 = \mu p_2$. The source node implements the earliest k_1, k_2 transmission scheme for type I and type II update packets, respectively. When there is a packet in service, the source node discards any other arriving update packet. Thus, update types do not have priority over each other during transmission. However, during joint optimization of the average age of both streams, Δ_I and Δ_{II}, by varying β we can prioritize them.

When an update is completed, i.e., transmitted to the earliest k_i nodes, $i = 1, 2$ depending on the type of the update packet, the system stays idle until the next update packet of any kind arrives. We denote this idle period with random variable Z which is an exponential random variable with rate $\mu p_1 + \mu p_2 = \mu$. Time spent during transmission of an update is the busy period Y_B which is equivalent to (3.2) since after each idle period with probability p_1 a type I update and with probability p_2 a type II update goes into service. Thus, an update cycle, Y, is equivalent to $Y_B + Z$. Note that Z does not depend on Y_B since it is memoryless. We have random variables M_1 and M_2 which are geometrically distributed with $p_1 q_1$ and $p_2 q_2$, respectively, as in Section 3.3.1. Also we note that M_1 and M_2 are independent from update cycle Y.

Age of type I updates at an individual node is given by (3.8). However, type I update interarrival to a node, S_I, is now equal to $S_I = X_{k_1:n} + Z + \sum_{i=j+1}^{j+M_1-1} \bar{Y}_i = X_{k_1:n} + \sum_{i=1}^{M_1-1} (\bar{Y}_B)_i + \sum_{i=1}^{M_1} Z_i$ where $\bar{Y}_B$ is equivalent to (3.9). Then,

$$\mathbb{E}[S_I] = \mathbb{E}[X_{k_1:n}] + \mathbb{E}[M_1 - 1]\,\mathbb{E}[\bar{Y}_B] + \mathbb{E}[M_1]\,\mathbb{E}[Z], \tag{3.22}$$

$$\begin{aligned}\mathbb{E}[S_I^2] = {} & \mathbb{E}[X_{k_1:n}^2] + 2E[M_1 - 1]\,\mathbb{E}[X_{k_1:n}]\,\mathbb{E}[\bar{Y}_B] \\ & + 2E[M_1]\,\mathbb{E}[X_{k_1:n}]\,\mathbb{E}[Z] + \mathbb{E}[M_1]\mathrm{Var}[Z] \\ & + \mathbb{E}[M_1 - 1]\mathrm{Var}[\bar{Y}_B] + \mathbb{E}[(M_1 - 1)^2]\,\mathbb{E}[\bar{Y}_B]^2 \\ & + 2\,\mathbb{E}[M_1^2 - M_1]\,\mathbb{E}[\bar{Y}_B]\,\mathbb{E}[Z] + \mathbb{E}[M_1^2]\,\mathbb{E}[Z]^2. \end{aligned} \tag{3.23}$$

In the following theorem, we determine the age of a type I update at an

individual node when the update streams arrive exogenously at the source node.

Theorem 3.2 *Under the earliest k_1 and k_2 transmission scheme for type I and type II updates that arrive at the source node as Poisson processes with rates μ_1 and μ_2, respectively, the average type I age at an individual node is*

$$\Delta_I = \frac{1}{k_1}\sum_{i=1}^{k_1}\mathbb{E}[X_{i:n}] + \frac{\mathbb{E}[S_I^2]}{2\,\mathbb{E}[S_I]}, \tag{3.24}$$

where first and second moments of S_I are as in (3.22) and (3.23).

The proof of Theorem 3.2 follows accordingly from that of Theorem 3.1. Note that when $p_1 = 1$, (3.24) reduces to the building block result in [119, Theorem 1]. By making the corresponding replacements as in Section 3.3.1 we can obtain the average age expression of type II update stream, Δ_{II}. When the service times of the packets of the same kind are i.i.d. shifted exponential random variables and n is large, we can further simplify (3.24) as follows.

Corollary 3.2 *For large n and $n > k_i$ we set $k_i = \alpha_i n$ for $i = 1, 2$. For shifted exponential transmission times X and $\tilde{X}$ with parameters (λ, c) and $(\tilde{\lambda}, \tilde{c})$ for type I and type II updates, respectively, Δ_I can be approximated as*

$$\begin{aligned}\Delta_I \approx\, & c + \frac{1}{\lambda} + \frac{1-\alpha_1}{\lambda\alpha_1}\log(1-\alpha_1) \\ & + \frac{\mu p_1^2(2-\alpha_1)\delta_1^2(\alpha_1) + 2\mu p_1 p_2(2-\alpha_1)\delta_1(\alpha_1)\delta_2(\alpha_2)}{2p_1\alpha_1(\mu p_1\delta_1(\alpha_1) + \mu p_2\delta_2(\alpha_2) + 1)} \\ & + \frac{\mu p_2(2p_2 + p_1\alpha_1)\delta_2^2(\alpha_2)}{2p_1\alpha_1(\mu p_1\delta_1(\alpha_1) + \mu p_2\delta_2(\alpha_2) + 1)}\end{aligned}$$

$$+ \frac{2\mu p_2 \delta_2(\alpha_2) + \mu p_1(2-\alpha_1)\delta_1(\alpha_1) + 1}{\mu p_1 \alpha_1(\mu p_1 \delta_1(\alpha_1) + \mu p_2 \delta_2(\alpha_2) + 1)}, \tag{3.25}$$

where $\delta_1(\alpha_1)$ and $\delta_2(\alpha_2)$ are as in (3.15).

The proof of Corollary 3.2 follows accordingly from that of Corollary 3.1. We note that when $p_1 = 1$, Corollary 3.2 reduces to the result in [119, Corollary 2]. The corresponding approximate expression for Δ_{II} can also be derived in the same manner. We also observe that when the earliest k_1, k_2 transmission scheme is utilized for type I and type II updates, respectively, the average age of either update type is also a function of ratios α_1 and α_2. Thus, although multiple update streams that arrive exogenously use the same network, an average age that does not depend on n can be achieved for either update stream.

When we minimize $\beta\Delta_I + (1-\beta)\Delta_{II}$ with $\beta \in (0,1)$ by selecting $k_1, k_2 \in \{1, \ldots, n\}$ as in (3.20), we obtain the pareto optimal curve as shown in Fig. 3.3(b). For the cases of $\beta = 1$ and $\beta = 0$ which correspond to the individual optimization of Δ_I and Δ_{II}, respectively, we observe that when $\beta = 1$ it is optimal to select $k_2^* = 1$ to obtain the minimum Δ_I, and when $\beta = 0$ selecting $k_1^* = 1$ gives the minimum Δ_{II}. The proof of this claim is similar to that of Lemma 3.1 since random variable Z is also positive and independent of k_1 and k_2.

3.4 Conclusion

In this chapter, we studied age of information in a multicast network with multiple update streams. Each receiver node in the network is interested in two kinds

time-sensitive update packets, namely type I and type II update packets. We characterized the age of information of both update streams and showed that the average age of both streams at the receiver nodes can be made a constant independent of the network size n by utilizing an earliest k_1 and k_2 transmission scheme for type I and type II updates, respectively. That is, even if the network resources are shared among multiple update streams, careful selection of the corresponding stopping thresholds can prevent information staleness at the receiver nodes for each update stream. We determined the age-optimum k_1 and k_2 stopping thresholds for arbitrary shifted exponential link delays to individually and jointly minimize the average age of both update streams.

CHAPTER 4

Scaling Laws for Age of Information in Wireless Networks

4.1 Introduction

Most of the existing works in the age of information literature study the analysis and optimization of age of information in systems with small number of source-destination pairs. In this chapter, going beyond prior works, our aim is to analyze the age of information in a large network setting with many source-destination pairs with a focus on its scalability as a function of the network size. In early 00's, followed by the pioneering work of Gupta and Kumar [122] a similar issue had come up for scaling laws of throughput in large networks. Gupta and Kumar's work in [122] uses a multi-hop scheme that achieves a total throughput of $O(\sqrt{n})$ for the network, and hence, $O(\frac{1}{\sqrt{n}})$ throughput per-user. References [123–126] studied throughput scaling in dense and extended networks considering static and mobile nodes. This line of research has culminated in the seminal papers of Ozgur et al. [127,128] which achieved $O(1)$ throughput per-user by utilizing hierarchical cooperation between nodes. In this chapter, we study scaling of *age of information* in large wireless networks.

References that are most closely related to our work study the scaling of age of information in the broadcast setting [42, 113, 115, 119, 129] including our works on multicast networks in Chapters 2 and 3. These works study a single source node which sends status updates to multiple receiver nodes. Reference [130], on the other hand, studies age scaling in the multiaccess setting with massive number of source nodes.

In our model, we have n source-destination (S-D) pairs on a network of fixed area. Each node is both a source and a destination. Each source node wants to transmit time-sensitive data to its destination node with as little age as possible. We find an achievable transmission scheme that facilitates successful communication among these n S-D pairs and achieves the smallest average age scaling per S-D user.

As studied in [130], a straightforward way to achieve successful communication between all S-D pairs is to use a round-robin policy such that at each turn only one source transmits to its destination and stays silent while all other sources transmit successively during their respective turns. This direct method achieves an age scaling of $O(n)$ meaning that age increases linearly in n since under this policy average inter-update times at a destination node increases linearly as n grows making the updates less frequent and causing age to increase.

As in the setting of [122], a multihop scheme that involves successive transmissions between the source and destination nodes can be utilized. In that work, the network is divided into cells and transmission hops take place in between these cells such that $O(\sqrt{n})$ messages are carried by each cell. Each of these cells can be considered a queue with multiple sources. As studied in [7], the age of a single

update packet that is served by a queue with $O(\sqrt{n})$ different packet streams is also $O(\sqrt{n})$ under LCFS with preemption policy. Therefore, in the multihop scheme, after one hop, age of an update becomes $O(\sqrt{n})$ since the queue is shared by $O(\sqrt{n})$ other packets. Considering the fact that the number of hops needed is $O(\sqrt{n})$, using a multihop scheme, the average age scales as $O(n)$ as in [130].

In this chapter, considering all these previous results, we first propose a three-phase transmission scheme to serve all n S-D pairs such that the time average age of each node is small. Our scheme utilizes local cooperation between the users as in [127]. We divide the network into cells of M users each. In the first phase, nodes from the same cell communicate to create a *mega update packet*, which contains the updates of all nodes from that cell. The main idea behind the mega update packets is to serve many nodes at once to decrease inter update time. In the second phase, inter-cell communication takes place and each cell sends its mega update packet to the corresponding destination cells. In the third and final phase, individual packets are extracted from the received mega update packets and relayed to the intended recipient nodes in the cells. During all these phases, we make use of the spatial separation of the nodes to allow multiple simultaneous transmissions provided that there is no destructive interference caused by others.[1] Using this scheme, we achieve an average age scaling of $O(n^{\frac{1}{4}} \log n)$ per-user.

Next, we observe that the first and third phases of the proposed transmission scheme essentially require successful communication between pairs but among M

[1]We show in Section 4.6 that the effect of the interference under the protocol model [122] is a scaling constant; see Section 4.6 for details.

nodes rather than n. With this observation and the fact that the system is scale-invariant, we introduce hierarchy in Phases I and III to improve the age scaling result. In other words, we can further divide cells into smaller subcells and apply the proposed three-phase transmission scheme again in Phases I and III. Although hierarchical cooperation was shown to result in poor delay performance in [128], by utilizing mega update packets better age scaling can be achieved here. In fact, using this scheme, we show that an average age scaling of $O\left(n^{\alpha(h)}\log n\right)$ per-user is achievable where $\alpha(h) = \frac{1}{3\cdot 2^h+1}$ and h denotes the number of hierarchy levels. We note that when this hierarchical cooperation is not utilized, i.e., $h = 0$, we retrieve the performance of the initial scheme which achieves an age scaling of $O(n^{\frac{1}{4}}\log n)$ per-user. In the asymptotic case when $h \to \infty$, the proposed scheme with hierarchical cooperation achieves an average age scaling of $O(\log n)$.

4.2 System Model and Age Metric

We consider n nodes that are uniformly and independently distributed on a square of fixed area S. Every node is both a source and a destination. These sources and destinations are paired randomly irrespective of their locations to form n S-D pairs. Sources create time-sensitive status update packets and transmit them to their respective destinations using the common wireless channel. Each source wants to keep its destination as up-to-date as possible. Thus, destination nodes need to be updated regularly with low transmission delays. We use the age of information metric to measure the freshness of the status update packets. Age is

measured for each destination node and for node i at time t age is the random process $\Delta_i(t) = t - u_i(t)$ where $u_i(t)$ is the timestamp of the most recent update at that node. The metric we use, time averaged age, is

$$\Delta_i = \lim_{\tau \to \infty} \frac{1}{\tau} \int_0^\tau \Delta_i(t) dt, \tag{4.1}$$

for node i. We use a graphical average age analysis to derive the average age for a single S-D pair assuming ergodicity similar to [11] and [29].

Inspired by [127], we first propose a scheme based on clustering nodes and making use of what we call *mega update packets* to increase the spatial reuse of the common wireless channel. This entails dividing n users into $\frac{n}{M}$ cells with M users in each cell with high probability.[2] The users communicate locally within cells to form the mega update packets. We model the delay in these intra-cell communications as i.i.d. exponential with parameter λ. Then, mega packets are transmitted between the cells. We model the delay in these inter-cell communications as i.i.d. exponential with parameter $\tilde{\lambda}$. Finally, the individual updates are extracted from mega updates and distributed to the intended destinations within cells again via intra-cell communications. While intra-cell communications occur simultaneously in parallel across the cells (see Section 4.6 for details), inter-cell updates occur sequentially one-at-a-time.

Second, we observe that the bottleneck in average age scaling in this three-

[2] As shown in [127, Lemma 4.1], the probability of having $((1-\delta)M, (1+\delta)M)$ nodes in a cell is larger than $1 - \frac{n}{SM} e^{-\Lambda(\delta)M}$ where $\Lambda(\delta)$ is independent of n and satisfies $\Lambda(\delta) > 0$ when $\delta > 0$. Note that when $M = n^b$ with $0 < b \leq 1$ this probability tends to 1 as n grows.

phase scheme is M since during intra-cell transmissions M transmissions are needed, one for each node in a cell. Noting that each cell is a scaled-down version of the whole system, we then propose introducing hierarchy by forming subcells from the cells and applying the three-phase transmission scheme on a cell level to overcome this bottleneck. Thus, when $h = 1$ hierarchy level is utilized, this hierarchical scheme includes inter-cell, inter-subcell (within cells) and intra-subcell transmissions. We first analyze the case with $h = 1$ level of hierarchy and then generalize the result to h hierarchy levels. We again model the delay in communications as i.i.d. exponential random variables with varying parameters depending on the type, e.g., intra-subcell, inter-subcell within cells or inter-cell, of the communication (see Section 4.5 for details).

Due to i.i.d. nature of delivery times in all types of communications with or without hierarchy, all destination nodes experience statistically identical age processes and will have the same average age. Thus, we will drop user index i in the average age expression and use Δ instead of Δ_i in the following analysis.

Finally, we denote the kth order statistic of random variables $X_1, \ldots, X_n$ as $X_{k:n}$. Here, $X_{k:n}$ is the kth smallest random variable, e.g., $X_{1:n} = \min\{X_i\}$ and $X_{n:n} = \max\{X_i\}$. For i.i.d. exponential random variables X_i with parameter λ, we have [131]

$$\mathbb{E}[X_{k:n}] = \frac{1}{\lambda}(H_n - H_{n-k}), \tag{4.2}$$

$$\mathrm{Var}[X_{k:n}] = \frac{1}{\lambda^2}(G_n - G_{n-k}), \tag{4.3}$$

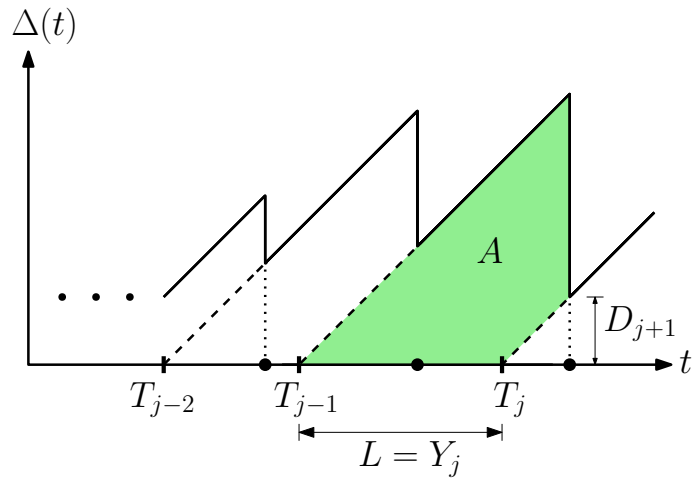


Figure 4.1: Sample age $\Delta(t)$ evolution for a single S-D pair. Update deliveries are shown with symbol $\bullet$. Session j starts at time T_{j-1} and lasts until $T_j = Y_j + T_{j-1}$.

where $H_n = \sum_{j=1}^{n} \frac{1}{j}$ and $G_n = \sum_{j=1}^{n} \frac{1}{j^2}$. Using these,

$$\mathbb{E}[X_{k:n}^2] = \frac{1}{\lambda^2} \left((H_n - H_{n-k})^2 + G_n - G_{n-k} \right). \tag{4.4}$$

Note that, for large n, we have $H_n \approx \log n + \gamma$ and $G_n \to \frac{\pi^2}{6}$ where $\gamma \approx 0.577$ is the Euler-Mascheroni constant. Since the constant γ does not affect the scaling results presented in this chapter, we take $H_n \approx \log n$ for large n in the rest of the chapter for ease of exposition. Throughout the chapter, the equality of two random variables stands for the equality of these random variables in distribution. Further, the inequalities involving random variables correspond to the usual stochastic order of random variables. That is, given random variables X and Y, X is stochastically smaller than Y if $P(X > x) \leq P(Y > x), \forall x \in \mathbb{R}$ [117].

4.3 Age Analysis of a Single S-D Pair

The network operates in sessions such that during each session all n sources successfully send their update packets to their corresponding destinations. Each session lasts for Y units of time. Here, we derive the average age of a single S-D pair (s, d) since each pair experiences statistically identical age as explained in Section 4.2.

Session j starts at time T_{j-1} and all sources including s generate their respective jth update packets. This session lasts until time $T_j = T_{j-1} + Y_j$, at which point, all n packets are received by their designated recipient nodes including node d. In other words, a session ends when the last S-D pair finishes the packet transmission at which point, all the other $n - 1$ destination nodes have already received their packets. Thus, in the proposed scheme every destination node but one receives its packet before the session ends. Fig. 4.1 shows the evolution of the age at a destination node over time. It is in the usual sawtooth shape with the age increasing linearly over time and dropping to a smaller value as the updates are received at the destination. The process repeats itself at time T_j when all sources including s generate the next update packet, namely update $j + 1$.

Using Fig. 4.1, the average age for an S-D pair is given by

$$\Delta = \frac{\mathbb{E}[A]}{\mathbb{E}[L]}, \tag{4.5}$$

where A denotes the shaded area and L is its length. From the figure, we find

$A_j = \frac{1}{2}Y_j^2 + Y_j D_{j+1}$ such that

$$\mathbb{E}[A] = \frac{\mathbb{E}[Y^2]}{2} + \mathbb{E}[D]\,\mathbb{E}[Y], \tag{4.6}$$

$$\mathbb{E}[L] = \mathbb{E}[Y], \tag{4.7}$$

since the system is ergodic and Y_j and D_{j+1} are independent. Here, D denotes the time interval between the generation of an update and its arrival at the destination node. Using these in (4.5), the average age for an S-D pair is given by

$$\Delta = \mathbb{E}[D] + \frac{\mathbb{E}[Y^2]}{2\,\mathbb{E}[Y]}. \tag{4.8}$$

Note that in some systems D may be directly equal to the link delay. However, as in our model here, D may capture some additional delays that may occur during the delivery time of an update. This will be further clarified in the next section.

4.4 Three-Phase Transmission Scheme

The proposed scheme involves clustering nodes and making use of *mega update packets* to serve many S-D pairs at once to reduce the session time. In this section, we describe the proposed three-phase transmission scheme and define mega update packets. As in [127], we divide the square network area into $\frac{n}{M}$ cells of equal area such that each cell includes M nodes with high probability which tends to 1 as n increases.[3] The transmission delays between the nodes belonging to the same

[3]We note that it is sufficient to have $O(M)$ nodes in each cell for the proposed scheme to work. However, from hereafter, we assume that each cell has exactly M nodes for ease of exposition.

cell are denoted by X_i whereas the transmission delays between the nodes from different cells are denoted by $\tilde{X}_i$. Note that X_i and $\tilde{X}_i$ are independent; X_i are i.i.d. exponential with parameter λ and $\tilde{X}_i$ are i.i.d. exponential with parameter $\tilde{\lambda}$.[4]

Phase I. Creating mega update packets: In a cell, each one of the M nodes gets a turn to distribute its current update packet to remaining $M-1$ nodes through $M-1$ links with independent random delays. This operation resembles the wait-for-all scheme studied in [113] since each node keeps transmitting until all $M-1$ nodes receive its packet. Thus, the time needed for each node to distribute its update packet to other nodes in the cell is $U = X_{M-1:M-1}$. Considering M successive transmissions for each node in the cell, this phase is completed in $V = \sum_{i=1}^{M} U_i$ units of time. By the end of this phase in a cell, each one of the M nodes has M different update packets one from each other node in that cell. Each node combines all these M packets to create what we call a *mega update packet* (see Fig. 4.2). In order to reduce the session time, cells work in parallel during Phase I (see Section 4.6 for a detailed description of this operation). This phase ends when the slowest cell among simultaneously operating cells finishes creating its mega update packet. Phase I takes $Y_I = V_{\frac{n}{M}:\frac{n}{M}}$ units of time, where Y_I denotes the duration of Phase I.

Phase II. MIMO-like transmissions: In this phase, each cell successively performs MIMO-like transmissions using the mega update packets created in Phase I. In each cell, all M source nodes send the mega update packet through the channel simultaneously to the respective destination cells in which the destina-

[4]We note that we have $\tilde{\lambda} \leq \lambda$ to reflect the increased distance and packet size, due to the utilization of mega update packets in the inter-cell transmissions of Phase II. In Section 4.8, we take $\tilde{\lambda}$ as a function of M to further account for the mega update packet size in the inter-cell transmission delays of Phase II.

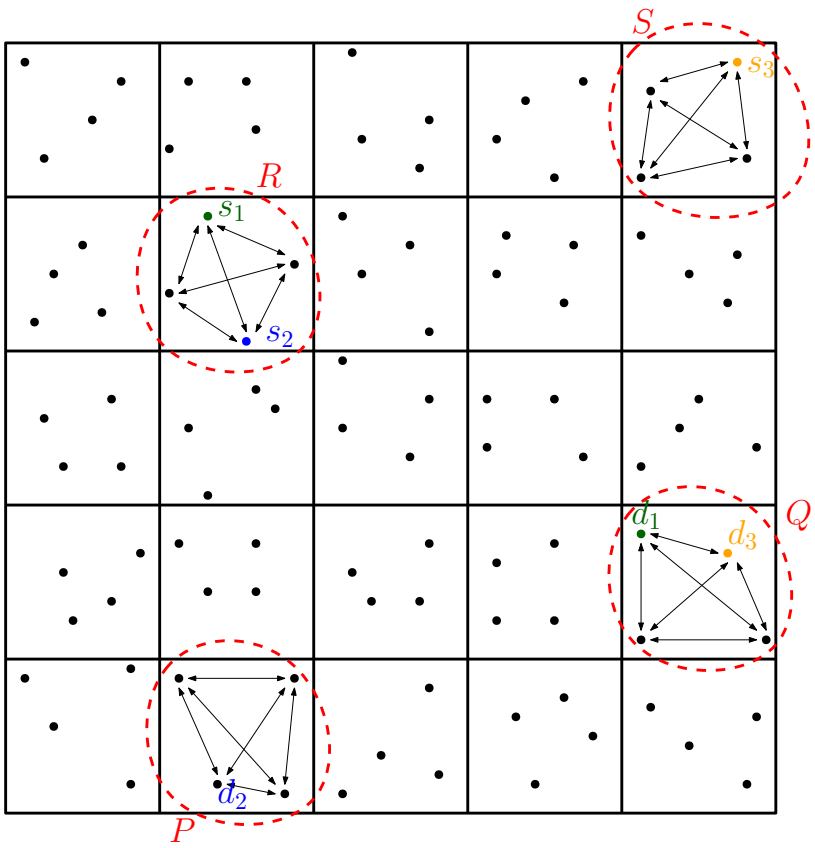


Figure 4.2: Cell formation for $M = 4$ and $n = 100$. Simultaneous intra-cell transmissions are depicted for three S-D pairs from cells P, Q, R and S.

tion nodes are located. Since every node sends the same mega packet which includes all M packets to be transmitted from that cell, this does not create interference. Thus, this is equivalent to sending update packets of all M sources with M copies each all at once (see Fig. 4.3). Hence, this significantly reduces the time needed to transmit updates of all M sources from that cell to their respective destinations. Note that this stage does not require the destination nodes to be in the same cell. In fact, considering that we have M nodes in a cell, each cell can at most have M different destination cells. Since we send M copies of each update to a destination cell in which there are M receiver nodes, only the earliest successful transmission is important. In other words, among the M^2 active links only the one with the

smallest delay is critical in delivering the mega update packet to a destination cell. Thus, it takes $\tilde{U} = \tilde{X}_{1:M^2}$ units of time for a source node s from cell j to send its update to the destination cell where the destination node d lies in.[5] This MIMO-like transmissions of cell j continues until all M destination cells receive the mega packet. Hence, for each cell, this phase lasts for $\tilde{V} = \tilde{U}_{M:M}$. We repeat this for each cell, making the session time of this phase $Y_{II} = \sum_{i=1}^{\frac{n}{M}} \tilde{V}_i$.

Phase III. In-cell relaying to the destination nodes: By the end of Phase II, each cell receives a mega packet for each one of its nodes. These packets may be received directly by their intended destination nodes. However, considering the worst case where they are received by any other node, we need to relay them to their actual designated recipient nodes. Thus, in this phase, M actual messages are extracted from the corresponding M mega update packets received during Phase II and sent to their recipients one at a time. Since this phase has intra-cell transmissions, it is performed in parallel across cells. For a single node this takes X units of time, consequently we need $\hat{V} = \sum_{i=1}^{M} X_i$ to finish this process in a cell. As in Phase I, we need to wait for the slowest cell to finish this operation. Then, this phase lasts for $Y_{III} = \hat{V}_{\frac{n}{M}:\frac{n}{M}}$ units of time.

The total session time of the proposed scheme is,

$$Y = Y_I + Y_{II} + Y_{III} = V_{\frac{n}{M}:\frac{n}{M}} + \sum_{i=1}^{\frac{n}{M}} \tilde{V}_i + \hat{V}_{\frac{n}{M}:\frac{n}{M}}, \tag{4.9}$$

[5]We note that in the MIMO-like transmissions of Phase II, since we consider the earliest transmission among M^2 active links, the speed up factor is M^2 unlike the regular MIMO transmission scheme in which the speed up factor is M. We thank one of the anonymous reviewers for raising this point. In Section 4.8, we discuss the performance of the proposed scheme when the speed up factor is only M.

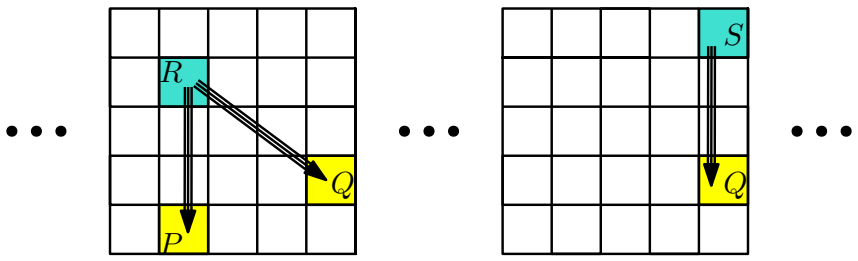


Figure 4.3: In Phase II cells take turns to perform inter-cell transmissions. These inter-cell transmissions are shown for the same three S-D pairs depicted in Fig. 4.2.

where V, $\tilde{V}$, and $\hat{V}$ are defined above. In our proposed scheme, assuming no S-D pair is in the same cell, arrivals to destination nodes occur in Phase III. Note that when an S-D pair is in the same cell, corresponding D is smaller which consequently leads to a smaller age, where as noted earlier, D denotes the time between generation of an update at certain source node till its arrival at the corresponding destination node. Therefore, by assuming no S-D pair is in the same cell, we essentially consider the worst case. Thus, any successful packet delivery will happen no earlier than the duration of the first two phases $Y_I + Y_{II}$. In addition, Phase III involves M successive in-cell transmissions for each node of a particular cell. Hence, depending on the cell that the source node lies in, as well as the realization of the transmission delay X, the corresponding destination node may receive the packet some time after Phase III starts. Let random variable Z capture when after Phase III starts that particular S-D pair is served. Then, we have,

$$D = Y_I + Y_{II} + Z. \tag{4.10}$$

For example, if a packet is the $(j+1)$th to be transmitted in Phase III, then delivery will be at $Y_I + Y_{II} + \sum_{i=1}^{j} X_i + X$. Then, the random variable Z is of the form

$Z = \sum_{i=1}^{j} X_i + X$.

Substituting (4.9)-(4.10) in (4.8) we obtain,

$$\Delta = \mathbb{E}[Y_I] + \mathbb{E}[Y_{II}] + \mathbb{E}[Z] + \frac{\mathbb{E}[Y^2]}{2\,\mathbb{E}[Y]}, \tag{4.11}$$

which is the average age of an S-D pair under the proposed transmission scheme.

Before we perform the explicit age calculation using (4.11), we make some observations to simplify our analysis. First, we note that, when the transmission delays $\tilde{X}$ are i.i.d. exponential with rate $\tilde{\lambda}$, then $\tilde{U} = \tilde{X}_{1:M^2}$ is also exponential with rate $M^2\tilde{\lambda}$ [132]. Second, we have the following upper bound for the duration of Phase I.

Lemma 4.1 *Y_I satisfies the following inequality,*

$$Y_I \leq \bar{V}, \tag{4.12}$$

where $\bar{V} = \sum_{i=1}^{M} \bar{U}_i$ and $\bar{U} = X_{n:n}$.

Proof: Recall that $Y_I = V_{\frac{n}{M}:\frac{n}{M}}$, where $V = \sum_{i=1}^{M} U_i$ and $U = X_{M-1:M-1}$. To show the inequality we make the following observation: In Phase I, $\frac{n}{M}$ cells operate simultaneously. First nodes of each of these cells start transmitting their packets to all other $M-1$ nodes of their cell at the same time. Here, the term first nodes denotes the set of arbitrarily selected nodes, one from each cell, that distribute their packet in their respective cells in the first place. Since intra-cell transmission delays are all i.i.d. across cells and packets, what we essentially have in this case is

simultaneous transmission to $\frac{n}{M}(M-1) \approx n$ nodes, and therefore all first nodes will be done in $X_{n:n}$ units of time.

We repeat this for the second nodes of each cell, i.e., nodes that distribute their packet within their respective cells in the second place, and so on to get $\bar{V} = \sum_{i=1}^{M}(X_{n:n})_i = \sum_{i=1}^{M} \bar{U}_i$. In this way of operation, a cell waits for all other cells to finish distributing the update packet of the first node and then continues with the second node and so on. In a way, for each of its nodes it waits for the slowest cell to finish. However, in our constructed scheme during Phase I, inside a cell, nodes distribute their packets to other nodes of that cell without considering other cells and phase ends when all cells finish this process for all their M nodes. Thus, $\bar{V}$ is an upper bound on Y_I. ■

Although our proposed Phase I lasts shorter than the scheme described in Lemma 4.1, for tractability and ease of calculation we worsen our scheme in terms of session time, and take the upper bound in Lemma 4.1 as our Phase I duration such that from now on $Y_I = \bar{V}$. Third, we have the following upper bound for the duration of Phase III.

Lemma 4.2 *Y_{III} satisfies the following inequality,*

$$Y_{III} \leq \bar{\bar{V}}, \tag{4.13}$$

where $\bar{\bar{V}} = \sum_{i=1}^{M} \bar{\bar{U}}_i$ and $\bar{\bar{U}} = X_{\frac{n}{M}:\frac{n}{M}}$.

We omit the proof of Lemma 4.2 since it follows similar to the proof of

Lemma 4.1. Due to the same tractability issues, we worsen Phase III as well in terms of duration and take $Y_{III} = \bar{\bar{V}}$ from now on.

As a result of Lemmas 4.1 and 4.2, (4.9) becomes

$$Y = \bar{V} + \sum_{i=1}^{\frac{n}{M}} \tilde{V}_i + \bar{\bar{V}}. \tag{4.14}$$

Now, we are ready to derive an age expression using Lemmas 4.1 and 4.2 in (4.11). This is stated in the following theorem.

Theorem 4.1 *Under the constructed transmission scheme, the average age of an S-D pair is given by,*

$$\begin{aligned}\Delta =& \frac{M}{\lambda} H_n + \frac{n}{M^3 \tilde{\lambda}} H_M + \frac{M-1}{2\lambda} H_{\frac{n}{M}} + \frac{1}{\lambda} \\ &+ \frac{\frac{M^2}{\lambda^2} H_n^2 + \frac{M}{\lambda^2} G_n}{2\left(\frac{M}{\lambda} H_n + \frac{n}{M^3 \tilde{\lambda}} H_M + \frac{M}{\lambda} H_{\frac{n}{M}}\right)} \\ &+ \frac{\frac{n^2}{M^6 \tilde{\lambda}^2} H_M^2 + \frac{n}{M^5 \tilde{\lambda}^2} G_M}{2\left(\frac{M}{\lambda} H_n + \frac{n}{M^3 \tilde{\lambda}} H_M + \frac{M}{\lambda} H_{\frac{n}{M}}\right)} \\ &+ \frac{\frac{M^2}{\lambda^2} H_{\frac{n}{M}}^2 + \frac{M}{\lambda^2} G_{\frac{n}{M}}}{2\left(\frac{M}{\lambda} H_n + \frac{n}{M^3 \tilde{\lambda}} H_M + \frac{M}{\lambda} H_{\frac{n}{M}}\right)} \\ &+ \frac{\frac{n}{M^2 \lambda \tilde{\lambda}} H_n H_M + \frac{M^2}{\lambda^2} H_n H_{\frac{n}{M}} + \frac{n}{M^2 \lambda \tilde{\lambda}} H_M H_{\frac{n}{M}}}{\frac{M}{\lambda} H_n + \frac{n}{M^3 \tilde{\lambda}} H_M + \frac{M}{\lambda} H_{\frac{n}{M}}}.\end{aligned} \tag{4.15}$$

Proof: The proof follows upon substituting (4.14) back in (4.11) and taking expectations of order statistics of exponential random variables as in Section 4.2. Doing

these, we obtain

$$\mathbb{E}[Y_I] = \frac{M}{\lambda} H_n, \qquad \mathbb{E}[Y_I^2] = \frac{M^2}{\lambda^2} H_n^2 + \frac{M}{\lambda^2} G_n, \tag{4.16}$$

$$\mathbb{E}[Y_{II}] = \frac{n}{M^3 \tilde{\lambda}} H_M, \qquad \mathbb{E}[Y_{II}^2] = \frac{n^2}{M^6 \tilde{\lambda}^2} H_M^2 + \frac{n}{M^5 \tilde{\lambda}^2} G_M, \tag{4.17}$$

$$\mathbb{E}[Y_{III}] = \frac{M}{\lambda} H_{\frac{n}{M}}, \qquad \mathbb{E}[Y_{III}^2] = \frac{M^2}{\lambda^2} H_{\frac{n}{M}}^2 + \frac{M}{\lambda^2} G_{\frac{n}{M}}. \tag{4.18}$$

Lastly, we need to calculate $\mathbb{E}[Z]$ where the random variable Z is the additional amount of time after Phase II ends until the destination node receives the update. Let us take an S-D pair (s, d) where source node s is from cell $j+1$. In Phase III, d has to wait for all other j mega packets from the first j cells to be distributed among the nodes. When its turn comes, d just needs X amount of time to get its packet. Then, d has $Z = \sum_{i=1}^{j} \bar{\bar{U}}_i + X$. Here, we have $\bar{\bar{U}}$ inside the summation as opposed to X as in the discussion preceding (4.11) because of Lemma 4.2. Taking expectation on $\bar{\bar{U}}$, j and X by noting their mutual independence we get

$$\mathbb{E}[Z] = \left(\frac{1}{M} \sum_{j=0}^{M-1} j \right) \mathbb{E}[\bar{\bar{U}}] + \mathbb{E}[X] = \frac{M-1}{2\lambda} H_{\frac{n}{M}} + \frac{1}{\lambda}. \tag{4.19}$$

Using (4.16)-(4.19) in (4.11) yields the expression. ■

Having derived the expression for the average age Δ of an S-D pair, we are now ready to work with large n.

Theorem 4.2 *For large n and with $M = n^b$, where $0 < b \leq 1$, the average age Δ*

in Theorem 4.1 approximately becomes,

$$\begin{aligned}\Delta \approx & \frac{n^b}{\lambda}\log n + \frac{n}{n^{3b}\tilde{\lambda}} b \log n + \frac{n^b - 1}{2\lambda}(1-b)\log n + \frac{1}{\lambda} \\ & + \frac{(1+(1-b)^2)\,\frac{n^{2b}}{\lambda^2}(\log n)^2}{2\left((2-b)\frac{n^b}{\lambda}\log n + \frac{n}{n^{3b}\tilde{\lambda}} b \log n\right)} \\ & + \frac{\frac{n^2}{n^{6b}\tilde{\lambda}^2} b^2 (\log n)^2 + \left(\frac{2n^b}{\lambda^2} + \frac{n}{n^{5b}\tilde{\lambda}^2}\right)\frac{\pi^2}{6}}{2\left((2-b)\frac{n^b}{\lambda}\log n + \frac{n}{n^{3b}\tilde{\lambda}} b \log n\right)} \\ & + \frac{b(2-b)\frac{n}{n^{2b}\lambda\tilde{\lambda}}(\log n)^2 + \frac{n^{2b}}{\lambda^2}(1-b)(\log n)^2}{(2-b)\frac{n^b}{\lambda}\log n + \frac{n}{n^{3b}\tilde{\lambda}} b \log n}. \end{aligned} \tag{4.20}$$

Proof: The expression follows upon substituting $M = n^b$ in (4.15) and noting that for large n, we have $H_n \approx \log n$. Further, G_n is monotonically increasing and converges to $\frac{\pi^2}{6}$. Since we have $M = n^b$, as n grows large M does too, resulting in $H_M \approx b \log n$ and G_M converging to $\frac{\pi^2}{6}$. We note that the notation $\approx$ in this theorem is mainly due to $\approx$ in H_n. ■

Theorem 4.3 *For large n, and for $\frac{1}{4} \leq b \leq 1$, the average age of an S-D pair Δ given in (4.20) reduces to,*

$$\Delta \approx c n^b \log n, \tag{4.21}$$

with a constant c. That is, age is $O(n^b \log n)$, for $\frac{1}{4} \leq b \leq 1$.

Proof: By analyzing the result of Theorem 4.2 we note that the first and third terms are $O(n^b \log n)$, and the second term is $O(n^{1-3b} \log n)$, and fourth term is a

constant independent of n. The fifth term can be written as

$$\frac{n^{2b}(\log n)^2\left(\frac{1+(1-b)^2}{\lambda^2}+\frac{b^2}{n^{2(4b-1)}\tilde{\lambda}^2}\right)}{n^b\log n\left(\frac{2(2-b)}{\lambda}+\frac{2b}{\tilde{\lambda}n^{4b-1}}\right)} + \frac{n^{2b}(\log n)^2\frac{\pi^2}{6}\left(\frac{2}{n^b(\log n)^2\lambda^2}+\frac{1}{n^{7b-1}(\log n)^2\tilde{\lambda}^2}\right)}{n^b\log n\left(\frac{2(2-b)}{\lambda}+\frac{2b}{\tilde{\lambda}n^{4b-1}}\right)}, \tag{4.22}$$

which is $O(n^b\log n)$ when $b \geq 1-3b$. Continuing similarly for the remaining term shows that it is also $O(n^b\log n)$ which gives the overall scaling result for $\frac{1}{4} \leq b \leq 1$.

■

Thus, the proposed transmission scheme, which involves intra-cell cooperation and inter-cell MIMO-like transmissions of mega update packets, allows the successful communication of n S-D pairs, and achieves an average age scaling of $O(n^{\frac{1}{4}}\log n)$ per-user.

In the next section, we propose introducing hierarchy to the proposed three-phase transmission scheme to improve the average age scaling.

4.5 Three-Phase Transmission Scheme with Hierarchy

4.5.1 Motivation and Outline of the Scheme

The three-phase transmission scheme proposed in Section 4.4 allows successful communication of n S-D pairs. Following the analysis to obtain the average age expression by substituting the first and second order moments of the phase durations given in (4.16)-(4.19) into the average age expression given in (4.8), we observe that

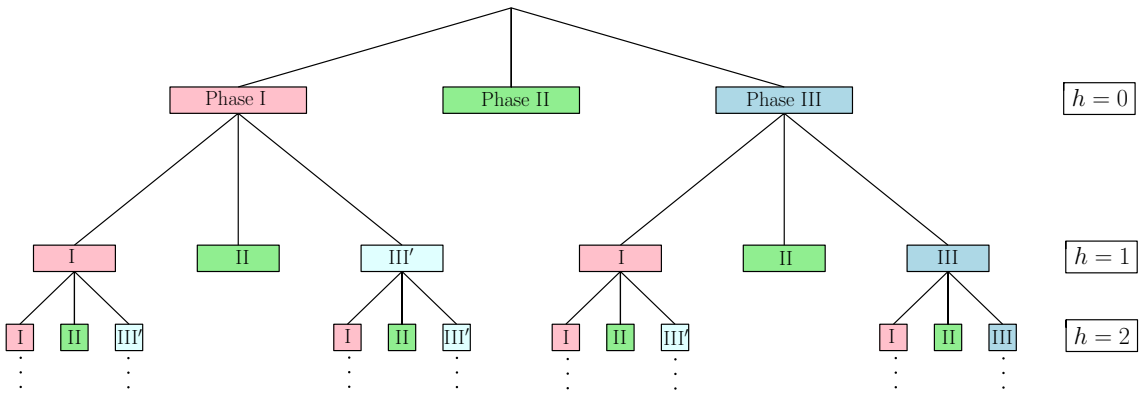


Figure 4.4: Proposed three-phase hierarchical transmission scheme.

the resulting per-user average age scaling, when n is large, with $M = n^b$ where $0 < b \leq 1$ and exponential link delays, is characterized by the expected scaling of the phases. As derived in (4.16)-(4.18) expected durations of the phases are $O(n^b \log n)$, $O(n^{1-3b} \log n)$ and $O(n^b \log n)$ which in turn result in an average age scaling of $O(n^{\frac{1}{4}} \log n)$ upon selecting $b = 1 - 3b$. Thus, to obtain a better average age scaling we need to improve the expected length of each phase. This motivates the hierarchical cooperation in the proposed scheme.

In the Phase I of Section 4.4, the communication takes place in between $M = n^b$ nodes rather than n nodes and a simple TDMA operation is performed among these nodes which results in an average scaling of $O(n^b \log n)$. Instead, we introduce the first level of hierarchy by dividing each of these cells into n^{b-a} further subcells with n^a users each where $0 < a < b$. Then, we apply the same three-phase scheme with one difference to this cell to accommodate Phase I transmissions of Section 4.4. In particular, to create the mega packet of the cell, first local communication among the nodes is performed within subcells and MIMO-like transmissions are carried out in between subcells within a cell. Then, instead of relaying the received packet to a

	$h=0$	$h=1$
Phase I	$O(n^b \log n)$	$O(n^a \log n) + O(n^{b-3a} \log n) + O(n^{b-a} \log n)$
Phase II	$O(n^{1-3b} \log n)$	$O(n^{1-3b} \log n)$
Phase III	$O(n^b \log n)$	$O(n^a \log n) + O(n^{b-3a} \log n) + O(n^a \log n)$

Table 4.1: Comparison of the expected durations of the phases with $h = 0$ as in Section 4.4 and $h = 1$ hierarchy level with $0 < a < b \leq 1$.

single node as in Phase III of Section 4.4, received packets are relayed to every other node in the subcell to create the mega update packet. With this operation, Phase I of Section 4.4 is completed in three phases, Phase I, Phase II, Phase III$'$, each of which are scaled down versions of the overall scheme with the corresponding difference in the third phase which is denoted as Phase III$'$ to highlight this difference. The expected length of the first phase with $h = 1$ level of hierarchy is then $O(n^a \log n) + O(n^{b-3a} \log n) + O(n^{b-a} \log n)$ (see Section 4.5.2 for a detailed derivation) all of which are smaller than $O(n^b \log n)$ achieved in Section 4.4.

Similarly, Phase III of Section 4.4 can also be completed in three phases under $h = 1$ level of hierarchy. However, this time in the third step we need Phase III rather than Phase III$'$ since we need to relay the received update packet within subcell to its intended recipient node to conclude the delivery.

Fig. 4.4 shows the proposed hierarchy structure in which Phases I and III of level h can be performed by applying the three-phase scheme on a smaller scale at level $h+1$ accordingly. The advantage of the hierarchical transmission is summarized in Table 4.1.

4.5.2 Detailed Description of the Scheme for $h = 1$

In this section, we describe the proposed hierarchical transmission scheme with $h = 1$ level of hierarchy in detail. Later, we generalize the average age scaling result for $h > 1$ levels of hierarchy using the fact that the system is scale-invariant. We note that the scheme in Section 4.4 does not utilize hierarchical cooperation, i.e., $h = 0$. As in Section 4.4, we start with a square network that is divided into $\frac{n}{M}$ cells of equal area with M nodes in each cell with high probability that tends to 1 as n increases. Selecting $M = n^b$ where $0 < b \leq 1$ results in n^{1-b} equal area cells with n^b users each cell. Introducing the first level of hierarchy, we further divide each cell into n^{b-a} equal area subcells to get a total of n^{1-a} subcells with n^a nodes each where $0 < a < b$.

Remember that when there is no hierarchical cooperation we denote the transmission delays within cells as X_i and in between cells as $\tilde{X}_i$. In this section, in order to accommodate hierarchical structure for $h = 1$ level, we change our notation so that transmission delays between the nodes from different cells are now denoted by $X_i^{(0)}$, between the nodes from different subcells within the same cell are denoted by $X_i^{(1)}$, and between the nodes belonging to the same subcell are denoted by $X_i^{(2)}$. Note that $X_i^{(j)}$ are independent; and $X_i^{(j)}$ are i.i.d. exponential with parameter λ_j for $j = 0, 1, 2$. Note that we have $\lambda_2 \geq \lambda_1 \geq \lambda_0$ so that link delays are proportional to the distances between nodes on average. Also note that in what follows we use Y_I, Y_{II} and Y_{III} to denote the phase durations under the three-phase scheme with hierarchy.

Phase I. Creating mega update packets: In this phase, each cell generates its mega update packet which includes all $M = n^b$ messages to be sent from that cell. Unlike the scheme in Section 4.4, we create mega update packets in three successive phases by applying the three-phase transmission scheme to each cell.

First, each node in a subcell distributes its update packet to remaining $n^a - 1$ nodes in its subcell which takes $U^I = X^{(2)}_{n^a-1:n^a-1}$ units of time. Considering n^a successive transmissions for each node of the subcell, this step is completed in a subcell in $V^I = \sum_{i=1}^{n^a} U_i^I$ units of time. This operation is analogous to the Phase I in Section 4.4 but performed among n^a nodes in a subcell rather than among n^b nodes within a cell. Upon completion of this step in a subcell, each node of that subcell has n^a different update packets one from each node. Each node combines all these update packets to form a *preliminary* mega update packet which includes all n^a messages to be sent out from this subcell. This operation is performed in parallel among all subcells in the network (see Section 4.6 for a detailed description of this operation) and ends when the slowest simultaneously operating subcell finishes creating its *preliminary* mega update packet. Hence, it takes $Y_I^I = V^I_{n^{1-a}:n^{1-a}}$ units of time, where Y_I^I denotes the duration of the first phase at $h = 1$.

When all *preliminary* mega update packets are formed, all n^{b-a} subcells of a cell perform MIMO-like transmissions among each other to distribute their *preliminary* mega update packets to remaining subcells within the cell. Since this requires cell-level transmissions in between subcells, this step is performed in parallel among cells and thus, subcells take turns. As in the Phase II of Section 4.4, all n^a nodes of a subcell start transmitting the *preliminary* mega update packet to remaining

$n^{b-a}-1$ subcells. Since every node sends the same *preliminary* mega update packet this does not create interference. This transmission continues until the earliest node in each remaining subcell receives the *preliminary* mega update packet. In other words, among the n^{2a} active links between the source and destination subcells, only the one with the smallest delay is critical. Thus, for a single subcell it takes $U^{II} = (X^{(1)}_{1:n^{2a}})_{n^{b-a}-1:n^{b-a}-1}$ units of time. Since subcells take turns, in a cell this step is completed in $V^{II} = \sum_{i=1}^{n^{b-a}} U_i^{II}$ units of time. Finally, on the network-level these MIMO-like transmissions continue until the slowest of the simultaneously operating cells finishes which corresponds to $Y_I^{II} = V^{II}_{n^{1-b}:n^{1-b}}$.

By the end of the MIMO-like transmissions among subcells, each subcell receives *preliminary* mega update packets of remaining $n^{b-a}-1$ subcells that lie in its cell. In this step, these packets are distributed within the subcell in parallel among the subcells of the network. This is identical to the operation of Phase III of Section 4.4 on subcell-level except that each *preliminary* mega update packet received is transmitted to all nodes of that subcell to successfully form the mega update packet of the corresponding cell. To highlight this difference we denote this step as Phase III$'$ in Fig. 4.4 at $h = 1$ level. Distributing one *preliminary* mega update packet takes $U^{III'} = X^{(2)}_{n^a-1:n^a-1}$ units of time. By repeating this for all *preliminary* mega update packets received this step is completed in a subcell in $V^{III'} = \sum_{i=1}^{n^{b-a}-1} U_i^{III'}$ units of time. We wait for the slowest subcell and thus on the network-level this step is completed in $Y_I^{III'} = V^{III'}_{n^{1-a}:n^{1-a}}$ units of time.

With this, each node in a subcell receives remaining $n^{b-a}-1$ *preliminary* mega update packets of n^a message each. Combining these with their own *preliminary*

mega update packet, every node in a subcell forms the mega update packet which includes all n^b messages to be sent out from that cell. Thus, the first phase lasts for $Y_I = Y_I^I + Y_I^{II} + Y_I^{III'}$ units of time.

Phase II. MIMO-like transmissions: Identical to Phase II of Section 4.4, in this phase each cell successively performs MIMO-like transmissions using the mega update packets created in Phase I. This phase requires network-level transmissions between cells. Thus, only one cell operates at a time. As in Section 4.4, a source node s from cell j needs $\tilde{U} = X^{(0)}_{1:n^{2b}}$ units of time to send its update to the destination cell where the destination node d lies in. Transmissions of cell j continue until all n^b destination cells receive the mega update packet. Hence, for each cell, this phase lasts for $\tilde{V} = \tilde{U}_{n^b:n^b}$. This operation is repeated for each cell and hence the session time of this phase $Y_{II} = \sum_{i=1}^{n^{1-b}} \tilde{V}_i$. At the end of this phase, each cell delivers its mega update packet to one node in each of the corresponding destination cells.

Phase III. In-cell relaying to the destination nodes: By the end of Phase II, each cell receives a total of n^b mega update packets, one for each node. In Section 4.4, relevant packets which have a destination node in that cell are extracted from these mega update packets and relayed to their respective designated recipient nodes by a simple TDMA operation which scales as $O(n^b \log n)$. However, as in Phase I we can introduce hierarchy to this phase and apply the three-phase scheme again. Thus, extracted relevant packets are first distributed within subcells of the nodes which received them in Phase II. Then, these packets are delivered to their corresponding destination subcells in which the destination nodes are located through MIMO-like transmissions and finally, they are relayed to the corresponding

recipient nodes within subcells.

Noting that each subcell receives on average n^a mega update packets, with one relevant packet each, distribution of these n^a packets within subcell takes $\hat{V}^I = \sum_{i=1}^{n^a} \hat{U}_i^I$ with $\hat{U}^I = X^{(2)}_{n^a-1:n^a-1}$ and on the network-level is completed in $Y^I_{III} = \hat{V}^I_{n^{1-a}:n^{1-a}}$ units of time. With this operation, the *secondary* mega update packet of that subcell is formed which includes all n^a update packets with destinations in that cell. Then, these *secondary* mega update packets are transmitted to the respective destination subcells in parallel among cells (subcells take turns) through MIMO-like transmissions until all n^a destination subcells receive them. In a cell, this is completed in $\hat{V}^{II} = \sum_{i=1}^{n^{b-a}} \hat{U}_i^{II}$ units of time where $\hat{U}^{II} = (X^{(1)}_{1:n^{2a}})_{n^a:n^a}$ and therefore, on the network-level is completed in $Y^{II}_{III} = \hat{V}^{II}_{n^{1-b}:n^{1-b}}$ when all cells finish. Thus, each subcell receives a total of n^a *secondary* mega update packets each of which includes one update destined to a node in that subcell. Finally, these packets are relayed to their actual recipient nodes within subcell. For a subcell it takes $\hat{V}^{III} = \sum_{i=1}^{n^a} \hat{U}_i^{III}$ units of time where $\hat{U}^{III} = X^{(2)}$ and hence on the network-level it is completed in $Y^{III}_{III} = \hat{V}^{III}_{n^{1-a}:n^{1-a}}$. Note that since in the last step we relay the packets to their destination node rather than all nodes in the subcell, this step is the subcell-level equivalent of Phase III of Section 4.4. As a result, the third phase lasts for $Y_{III} = Y^I_{III} + Y^{II}_{III} + Y^{III}_{III}$ and finishes when every S-D pair of the network is served.

Total session time of the proposed scheme is, therefore, $Y = Y_I + Y_{II} + Y_{III}$. Before we perform the explicit age calculation, we again make some observations to simplify our analysis.

Lemma 4.3 *Y_I satisfies the following inequality,*

$$Y_I \leq \bar{V}^I + \bar{V}^{II} + \bar{V}^{III'}, \tag{4.23}$$

where

$$\bar{V}^I = \sum_{i=1}^{n^a} \bar{U}_i^I, \qquad \bar{U}^I = X_{n:n}^{(2)}, \tag{4.24}$$

$$\bar{V}^{II} = \sum_{i=1}^{n^{b-a}} \bar{U}_i^{II}, \qquad \bar{U}^{II} = (X_{1:n^{2a}}^{(1)})_{n^{1-a}:n^{1-a}}, \tag{4.25}$$

$$\bar{V}^{III'} = \sum_{i=1}^{n^{b-a}} \bar{U}_i^{III'}, \qquad \bar{U}^{III'} = X_{n:n}^{(2)}. \tag{4.26}$$

The proof of this lemma follows similarly from that of Lemma 4.1. We show that $Y_I^I \leq \bar{V}^I$, $Y_I^{II} \leq \bar{V}^{II}$ and $Y_I^{III'} \leq \bar{V}^{III'}$ which yields (4.23).

We worsen our scheme in terms of session time and hereafter take the upper bound in Lemma 4.3 as our Phase I duration for tractability and ease of calculation. Thus, from now on $Y_I = \bar{V}^I + \bar{V}^{II} + \bar{V}^{III'}$. Next, we have the following upper bound for the duration of Phase III.

Lemma 4.4 *Y_{III} satisfies the following inequality,*

$$Y_{III} \leq \bar{\bar{V}}^I + \bar{\bar{V}}^{II} + \bar{\bar{V}}^{III}, \tag{4.27}$$

where

$$\bar{\bar{V}}^I = \sum_{i=1}^{n^a} \bar{\bar{U}}_i^I, \qquad \bar{\bar{U}}^I = X^{(2)}_{n:n}, \tag{4.28}$$

$$\bar{\bar{V}}^{II} = \sum_{i=1}^{n^{b-a}} \bar{\bar{U}}_i^{II}, \qquad \bar{\bar{U}}^{II} = (X^{(1)}_{1:n^{2a}})_{n^{1-b+a}:n^{1-b+a}}, \tag{4.29}$$

$$\bar{\bar{V}}^{III} = \sum_{i=1}^{n^a} \bar{\bar{U}}_i^{III}, \qquad \bar{\bar{U}}^{III} = X^{(2)}_{n^{1-a}:n^{1-a}}. \tag{4.30}$$

We omit the proof of Lemma 4.4 since it follows similar to the proof of Lemma 4.1. We worsen Phase III as well in terms of duration and take $Y_{III} = \bar{\bar{V}}^I + \bar{\bar{V}}^{II} + \bar{\bar{V}}^{III}$ from now on because of similar tractability issues.

As a result of Lemmas 4.3 and 4.4, total session time becomes

$$Y = \bar{V}^I + \bar{V}^{II} + \bar{V}^{III'} + Y_{II} + \bar{\bar{V}}^I + \bar{\bar{V}}^{II} + \bar{\bar{V}}^{III}. \tag{4.31}$$

Taking expectations of order statistics of exponential random variables as in (4.2)-(4.4) and using the fact that for large n, we have $H_n \approx \log n$ and G_n is monotonically increasing and converges to $\frac{\pi^2}{6}$, first two moments of the subphase and phase durations approximately become

$$\mathbb{E}\left[\sum_{i\in\mathcal{I}'} \bar{V}^{(i)}\right] = \left(\frac{n^a + n^{b-a}}{\lambda_2} + \frac{(1-a)n^{b-3a}}{\lambda_1}\right)\log n, \tag{4.32}$$

$$\mathbb{E}\left[\sum_{i\in\mathcal{I}} \bar{\bar{V}}^{(i)}\right] = \left(\frac{(2-a)n^a}{\lambda_2} + \frac{(1-b+a)n^{b-3a}}{\lambda_1}\right)\log n, \tag{4.33}$$

$$\mathbb{E}[Y_{II}] = \frac{bn^{1-3b}}{\lambda_0}\log n, \tag{4.34}$$

$$\mathbb{E}\left[Y_{II}^2\right] = \frac{n^{1-5b}}{\lambda_0^2}\frac{\pi^2}{6} + \frac{b^2 n^{2(1-3b)}}{\lambda_0^2}\log^2 n, \tag{4.35}$$

$$\mathbb{E}\left[\left(\bar{V}^I\right)^2\right] = \frac{n^a}{\lambda_2^2}\frac{\pi^2}{6} + \frac{n^{2a}}{\lambda_2^2}\log^2 n, \tag{4.36}$$

$$\mathbb{E}\left[\left(\bar{V}^{II}\right)^2\right] = \frac{n^{b-5a}}{\lambda_1^2}\frac{\pi^2}{6} + \frac{(1-a)^2 n^{2(b-3a)}}{\lambda_1^2}\log^2 n, \tag{4.37}$$

$$\mathbb{E}\left[\left(\bar{V}^{III'}\right)^2\right] = \frac{n^{b-a}}{\lambda_2^2}\frac{\pi^2}{6} + \frac{n^{2(b-a)}}{\lambda_2^2}\log^2 n, \tag{4.38}$$

$$\mathbb{E}\left[\left(\bar{\bar{V}}^I\right)^2\right] = \frac{n^a}{\lambda_2^2}\frac{\pi^2}{6} + \frac{n^{2a}}{\lambda_2^2}\log^2 n, \tag{4.39}$$

$$\mathbb{E}\left[\left(\bar{\bar{V}}^{II}\right)^2\right] = \frac{n^{b-5a}}{\lambda_1^2}\frac{\pi^2}{6} + \frac{(1-b+a)^2 n^{2(b-3a)}}{\lambda_1^2}\log^2 n, \tag{4.40}$$

$$\mathbb{E}\left[\left(\bar{\bar{V}}^{III}\right)^2\right] = \frac{n^a}{\lambda_2^2}\frac{\pi^2}{6} + \frac{(1-a)^2 n^{2a}}{\lambda_2^2}\log^2 n, \tag{4.41}$$

where in (4.32), $i \in \mathcal{I}' = \{I, II, III'\}$ and in (4.33), $i \in \mathcal{I} = \{I, II, III\}$.

Now, we are ready to derive an average age expression using (4.8). For ease of exposition, we assume that every node updates its age at the end of each session when the hierarchy is implemented and take $D_{j+1} = Y_{j+1}$. Then, (4.8) becomes

$$\Delta = \mathbb{E}[Y] + \frac{\mathbb{E}[Y^2]}{2\,\mathbb{E}[Y]}. \tag{4.42}$$

Note that this assumption can only result in a higher average age as all nodes but one receive their update packets before the session ends, i.e., $P(D \leq Y) = 1$ for all updates and nodes.

Theorem 4.4 *Under the constructed transmission scheme with $h = 1$ level of hier-*

archy, for large n, the average age of an S-D pair is given by,

$$\Delta = \mathbb{E}\left[\sum_{i\in\mathcal{I}'}\bar{V}^{(i)}\right] + \mathbb{E}[Y_{II}] + \mathbb{E}\left[\sum_{i\in\mathcal{I}}\bar{\bar{V}}^{(i)}\right] + \frac{\mathbb{E}\left[\left(\sum_{i\in\mathcal{I}'}\bar{V}^{(i)} + Y_{II} + \sum_{i\in\mathcal{I}}\bar{\bar{V}}^{(i)}\right)^2\right]}{2\left(\mathbb{E}\left[\sum_{i\in\mathcal{I}'}\bar{V}^{(i)}\right] + \mathbb{E}[Y_{II}] + \mathbb{E}\left[\sum_{i\in\mathcal{I}}\bar{\bar{V}}^{(i)}\right]\right)}. \tag{4.43}$$

The proof of Theorem 4.4 follows upon substituting (4.31) back in (4.42). Moments follow from (4.32)-(4.41).

Theorem 4.5 *For large n, with* $a = \frac{b}{2}$ *and* $\frac{1}{7} \leq a \leq \frac{1}{2}$*, the average age of an S-D pair when* $h = 1$ *hierarchy level is implemented,* Δ*, given in (4.43) reduces to,*

$$\Delta \approx \tilde{c} n^a \log n, \tag{4.44}$$

with a constant $\tilde{c}$*. That is, age is* $O(n^a \log n)$*, for* $\frac{1}{7} \leq a \leq \frac{1}{2}$*.*

Proof: Using (4.32)-(4.41) in (4.43), we observe that in the average age expression we have terms with $O(n^a \log n)$, $O(n^{b-a} \log n)$, $O(n^{b-3a} \log n)$, and $O(n^{1-3b} \log n)$. Among first three types, noting that $b - 3a < b - a$, dominating terms are $O(n^a)$ and $O(n^{b-a})$. Thus, by choosing $a = b - a$ we can minimize the resulting scaling. With this selection, the first and third terms in (4.43) are $O(n^a \log n)$ whereas the second one is $O(n^{1-6a} \log n)$. Looking at the fourth term we observe that it has the following form when $b = 2a$,

$$\frac{c_1 n^{2a} \log^2 n + c_2 n^{2(1-6a)} \log^2 n + c_3 n^{1-5a} \log^2 n}{c_4 n^a \log n + c_5 n^{1-6a} \log n}, \tag{4.45}$$

where $c_1, \ldots, c_5$ are constants. We observe that for $\frac{1}{7} \leq a \leq \frac{1}{2}$, first three terms and the fourth term given in (4.45) are $O(n^a \log n)$ which yields the result. ■

Thus, the proposed hierarchical scheme with $h = 1$ hierarchy levels achieves an average age scaling of $O(n^{\frac{1}{7}} \log n)$ per-user when $a = \frac{1}{7}$ and $b = \frac{2}{7}$. This implies that if the cells have M nodes each, each subcell has $\sqrt{M}$ nodes when $h = 1$. Note that in Section 4.4 it is shown that $\frac{1}{4} \leq b \leq 1$. Here, resulting b not only satisfies this but also gives a better scaling in the end because of the hierarchy we utilized. In Theorem 4.6 below, we generalize this scaling result to h levels of hierarchy.

Theorem 4.6 *For large n, when the proposed scheme is implemented with h hierarchy levels, the average age scaling of $O\left(n^{\alpha(h)} \log n\right)$ per-user is achievable where $\alpha(h) = \frac{1}{3 \cdot 2^h + 1}$.*

Proof: We observe that when $h = 1$ hierarchy level is utilized, the scaling result comes from $a = 1 - 6a$. Since $b = 2a$, another way to express this is $\frac{b}{2^h} = 1 - 3b$. As h increases with $b = 2a$ structure in each hierarchy level, we see that subcells at level h have $n^{\frac{b}{2^h}}$ nodes. Thus, when h levels of hierarchy is utilized, subcell transmissions take place among $n^{\frac{b}{2^h}}$ nodes and inter-subcell transmissions have $n^{\frac{b}{2^h}}$ turns. However, the second phase is still $O(n^{1-3b})$ as each cell at the top of the hierarchy has n^b nodes. Thus, $\frac{b}{2^h} = 1 - 3b$ yields $\alpha(h) = \frac{1}{3 \cdot 2^h + 1}$. ■

Thus, when hierarchy is utilized, the proposed transmission scheme, which involves local cooperation and MIMO-like inter-cell transmissions, allows the successful communication of n S-D pairs, and achieves an average age scaling of $O\left(n^{\alpha(h)} \log n\right)$ per-user where $h = 0, 1, \ldots$ is the number of hierarchy levels. Note that in the

asymptotic case when h tends to ∞, this scheme gives an average age scaling of $O(\log n)$ per-user. This is the case because as h increases, number of turns in each phase, $n^{\frac{b}{2^h}}$, decreases such that eventually $\log n$ term which comes from the fact that packets are distributed locally to all other nodes in the same subcell in Phases I and III$'$ dominates. We also observe that when the hierarchy is not utilized, i.e. $h = 0$, Theorem 4.6 yields the result in Theorem 4.3 in Section 4.4.

4.6 Note on Phases I and III

We use the protocol model introduced in [122] to model the interference such that two nodes can be active if they are sufficiently spatially separated from each other. In other words, we allow simultaneous transmissions provided there is no destructive interference caused by other active nodes. Suppose that node i transmits its update to node j. Then, node j can successfully receive this update if the following is satisfied for any other node k that is simultaneously transmitting,

$$d(j,k) \geq (1+\gamma)d(j,i), \tag{4.46}$$

where function $d(x,y)$ denotes the distance between nodes x and y and γ is a positive constant determining the guard zone.

The proposed three-phase scheme with h levels of hierarchy utilizes parallelized transmissions in Phases I and III where $h \geq 0$. When the hierarchy is not utilized, i.e., $h = 0$, parallel intra-cell transmissions take place during these phases. In order to implement these parallel communications in Phases I and III, we follow a 9-TDMA

scheme as in [127]. Specifically, $O(\frac{n}{9M})$ of the total $\frac{n}{M}$ cells work simultaneously so that Phases I and III are completed in 9 successive subphases. Using the protocol model, cells that are at least $(1+\gamma)r\sqrt{2}$ away from a cell can operate simultaneously during these phases, where $r = \sqrt{SM/n}$ is the length of each square cell and S is the network area. Noting that there are at least two inactive cells in between two active cells under a 9-TDMA operation and the maximum in-cell transmission distance is $r\sqrt{2}$, this scheme satisfies (4.46) if the guard zone parameter $\gamma \leq \sqrt{2} - 1$. We note that as the number of cells in the network increases, the number of simultaneously active nodes in Phases I and III also increases. Since the distance between the active cells, $2r$, and the in-cell transmission distance, $r\sqrt{2}$, both decrease proportionally when the number of cell increases the condition in (4.46) is still satisfied under the 9-TDMA operation given that we have $\gamma \leq \sqrt{2} - 1$.

On the other hand, when $h = 1$ level of hierarchy is utilized, the proposed scheme includes within subcell transmissions that are parallelized across subcells and within cell transmissions that are parallelized among cells (subcells take turns) in Phases I and III (or III$'$). Similar 9-TDMA scheme again is used to accommodate these simultaneous transmissions. When $\gamma \leq \sqrt{2} - 1$, parallel 9-TDMA operation among subcells is still allowed since from cell-level to subcell-level both distance terms in (4.46) decrease proportionally. Extending this, we see that for h level of hierarchy, by selecting an appropriate guard zone parameter γ, parallelized Phase I and III (or III$'$) operation under 9-TDMA scheme is allowed. Noting that 9 here is a constant and valid for any n, it does not change the scaling results.

4.7 Numerical Results

In this section, we provide simple numerical results to validate our results for the $h = 0$ case, i.e., hierarchical cooperation is not utilized.[6] Simulations are performed using MATLAB over 1000 sessions where each session is comprised of three phases. Plotted results are averaged over 10 independent simulations. In the simulations, we set $\lambda = 5$ and $\tilde{\lambda} = 2$. We recall that to make the proposed three-phase scheme analytically tractable, we worsen Phases I and III through Lemmas 4.1 and 4.2. In this section, we provide simulations for the actual proposed three-phase scheme, referred to as TPS and shown in blue solid curves throughout, and its upper-bounded version, referred to as TPS-ub and shown in purple dash-dotted curves throughout, along with our theoretical results to make comparisons and verify our results.

We first consider the case in which the nodes are placed on a grid in the network and set $b = \frac{1}{2}$. In other words, nodes are equally spaced and each of the $\frac{n}{M}$ cells has exactly M nodes. We note that since we do not consider the physical distance in between nodes in our analysis, our results still hold when the nodes are on a grid. In Fig. 4.5(a), in line with Theorem 4.3, we see that TPS-ub achieves an average age scaling of $O(n^{\frac{1}{2}} \log n)$ per-user since we set $b = \frac{1}{2}$. Here, the red dashed curve shows the theoretical result obtained from (4.16)-(4.18) whereas the yellow dotted line is obtained from (4.21) with corresponding c. We observe that these two curves coincide, even for smaller values of n. Thus, the result in (4.21) is in line with our average age analysis. Further, we observe that the actual proposed policy without

[6] Simulation complexity increases substantially when hierarchical cooperation is utilized.

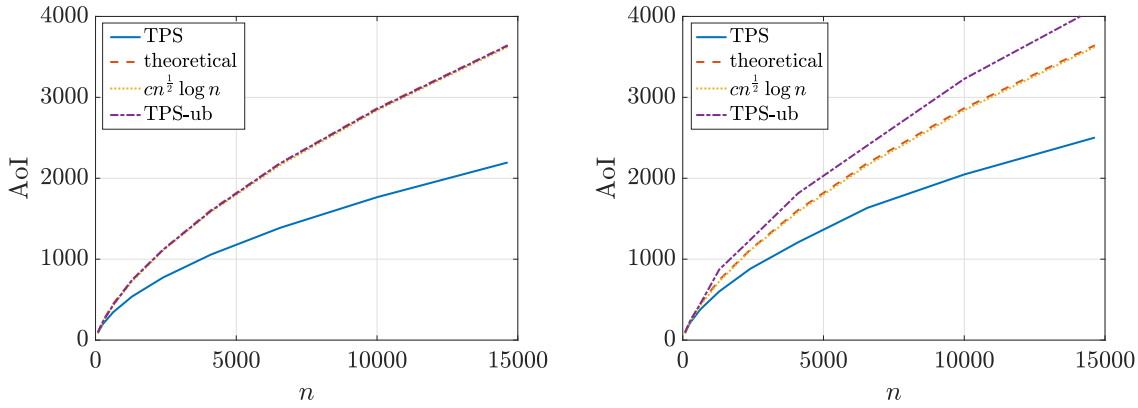


Figure 4.5: Average age scaling under the proposed three-phase hierarchical transmission scheme for $h = 0$, $b = \frac{1}{2}$, $\lambda = 5$ and $\tilde{\lambda} = 2$ for varying n when (a) nodes are on a grid, (b) nodes are randomly uniformly and independently distributed.

the upper bounds on Phases I and III, TPS, lies below the $O(n^{\frac{1}{2}}\log n)$ scaling. Thus, the proposed three-phase scheme, without the upper bounds on Phases I and III, has a better average age scaling than the upper bounded scheme, TPS-ub, which is shown to achieve $O(n^{\frac{1}{2}}\log n)$ scaling for $b = \frac{1}{2}$ in Theorem 4.3.

Next, we consider the case in which the nodes are uniformly and independently placed in the network in Fig. 4.5(b). In this case, each cell has $O(M)$ nodes rather than exactly M nodes which leads to a gap in between the TPS-ub curve and the theoretical results, which assume exactly M nodes in each cell, even though we observe that TPS-ub curve continues to have $O(n^{\frac{1}{2}}\log n)$ scaling trend. Thus, for scaling results to hold, it is enough to have $O(M)$ nodes in each cell. We also observe that age under the actual proposed scheme, TPS, slightly increases compared to Fig. 4.5(a) but still has a better scaling than $O(n^{\frac{1}{2}}\log n)$.

Throughout the analysis, we have i.i.d. transmission times for intra-cell and inter-cell transmissions of each node. Next, we analyze the performance of the

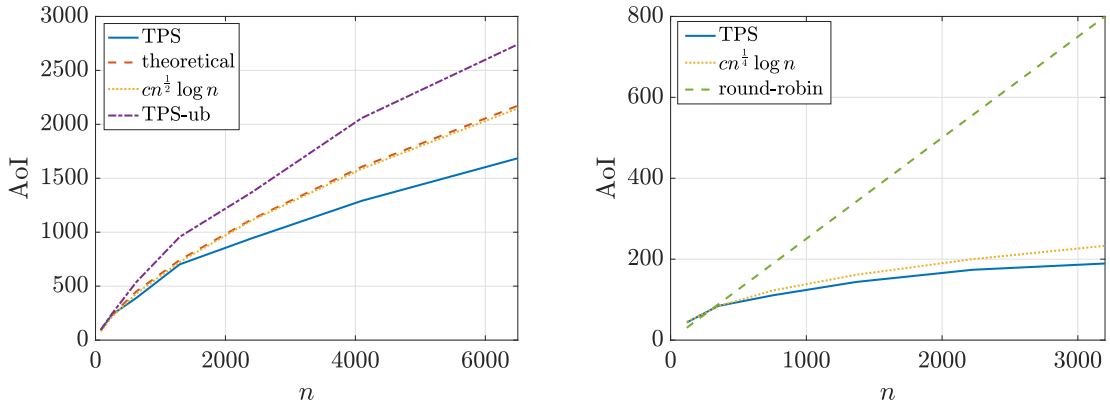


Figure 4.6: (a) Average age scaling under the proposed three-phase hierarchical transmission scheme for $h = 0$, $b = \frac{1}{2}$ when nodes have non i.i.d. transmission rates for varying n. (b) Comparison of round robin scheme and the proposed three-phase hierarchical transmission scheme for $h = 0$, $b = \frac{1}{4}$, $\lambda = 5$ and $\tilde{\lambda} = 2$ for varying n.

proposed scheme when nodes have different transmission rates. Thus, in this case, transmission times of nodes are independently yet exponentially distributed with different mean values. In the simulation we consider that λ is uniformly distributed in $[3, 7]$ and $\tilde{\lambda}$ is uniformly distributed in $[0.5, 3.5]$ for each node such that average intra-cell and inter-cell transmission rates are 5 and 2, respectively as in Fig. 4.5(b). We observe in Fig. 4.6(a) that under the non-i.i.d. transmission times, average ages for both TPS and TPS-ub increase even though the TPS curve still has a lower average age scaling than $O(n^{\frac{1}{2}}\log n)$. On the other hand, in this case, the gap between the $cn^{\frac{1}{2}}\log n$ curve and TPS-ub curve increases and these two curves are no longer parallel. That is, TPS-ub has a higher average age scaling than $O(n^{\frac{1}{2}}\log n)$ unlike the i.i.d. transmission rates setting.

Lastly, we compare the performance of the three-phase scheme with $b = \frac{1}{4}$ and the baseline round robin policy in which nodes take turns to transmit their update

packets which has per-user scaling of $O(n)$. In Fig. 4.6(b), we observe the proposed three-phase transmission scheme outperforms the round-robin policy when the total number of nodes exceeds $n = 300$.

4.8 Discussion

We note that the focus of this chapter is the scaling of age of information in large wireless networks. Thus, presented scaling results hold with high probability when the number of nodes in the network grows beyond a certain threshold. The random network model used in this chapter is rather idealized. Although it is of theoretical interest on its own right, a more realistic model may adopt a physical interference model that is based on signal-to-interference ratio requirements and possibly mobile nodes rather than static nodes. Moreover, we make use of the mega update packets in the proposed transmission scheme to serve multiple S-D pairs at once without considering the growing mega update packet size as the network population increases. Likewise, link delays are modeled as i.i.d. exponentials with constant parameters that are not affected by the transmission distance or packet size. It would be an interesting direction to analyze the effects of packet size and the distance between S-D pairs on the average age scaling. Further, the schemes discussed in this chapter are not private since a packet intended for a certain destination node is observed by other nodes in the network. To ensure privacy, updates can be encrypted in a way that the routing of a packet to the correct destination node is still maintained but only the intended destination node can fully decrypt the message.

In this sense, here, we take a careful look at the proposed three-phase transmission scheme, particularly the MIMO-like inter-cell transmissions of mega update packets in Phase II, and discuss the performance of the proposed three-phase transmission scheme under possibly more realistic and practically applicable network models.

First, we analyze the performance of the proposed three-phase scheme without the utilization of mega update packets in the next corollary.

Corollary 4.1 *When mega update packets are not utilized, the proposed three-phase transmission policy achieves an average age scaling of* $O(n^{\frac{1}{3}} \log n)$ *per-user.*

Proof: In the proposed scheme, mega update packets are formed during Phase I and are transmitted to destination cells in Phase II. By transmitting a mega update packet, all M nodes of a particular cell send their messages to the corresponding destination cells at once. When the mega update packets are not utilized, however, each node of a cell still stores all other messages received in Phase I in its buffer but instead of combining these messages to create the mega update packet, these nodes can send the individual packets to corresponding M destination cells sequentially. In other words, rather than sending all M packets as a mega packet simultaneously, nodes in a cell collectively can send the packets one by one.[7] In this operation, a cell sends out all M of its update packets to the destination cells in M successive transmissions in $\sum_{i=1}^{M}(\tilde{X}_{1:M^2})_i$ units of time where $\tilde{X}$ denotes the transmission delay

[7]We note that in this operation, each destination cell only receives the update packets that are destined to that particular cell as opposed to receiving a full mega update packet.

of a single packet and is exponentially distributed with rate $\tilde{\lambda}$.[8] Since we need a total of n transmissions for $\frac{n}{M}$ cells we have $Y_{II} = \sum_{i=1}^{n} (\tilde{X}_{1:M^2})_i$. Note that when $M = n^b$ where $0 < b \leq 1$ we have

$$\mathbb{E}[Y_{II}] = \frac{n}{M^2\tilde{\lambda}}, \qquad \mathbb{E}[Y_{II}^2] = \frac{n^2}{M^4\tilde{\lambda}^2} + \frac{n}{M^4\tilde{\lambda}^2}. \tag{4.47}$$

Repeating a similar analysis as above with (4.47) instead of (4.17) for large n yields the result. Particularly, we obtain $b = 1 - 2b$ from which the proposed scheme achieves the $O(n^{\frac{1}{3}} \log n)$ scaling result. Note that even if the mega update packets are not utilized, nodes in a cell still need enough buffer space to store all M messages received in Phase I so that these messages can be sent out one by one in Phase II.

■

Second, in the next corollary, we consider the case when the speed up factor is M in the MIMO-like transmissions of Phase II as opposed to a speed up factor of M^2 which arises since we consider the fastest of the simultaneously active M^2 links in Phase II.

Corollary 4.2 *When the speed up factor is M as in the ordinary MIMO with M transmit and M receive antennas, in the inter-cell transmissions of Phase II, the proposed three-phase transmission policy achieves an average age scaling of $O(n^{\frac{1}{3}} \log n)$ per-user.*

[8] We note that when mega update packets are not utilized, the transmission of all M update packets from a particular cell to corresponding M destination cells essentially has an Erlang distribution with rate $(M, M^2\tilde{\lambda})$ since we have sum of M i.i.d. random variables $\tilde{X}_{1:M^2}$ which is exponentially distributed with rate $M^2\tilde{\lambda}$.

Proof: In this case, we have $\tilde{U} = \tilde{X}_{1:M}$ such that the first two moments of the duration of the second phase become

$$\mathbb{E}[Y_{II}] = \frac{n}{M^2\tilde{\lambda}}H_M, \qquad \mathbb{E}[Y_{II}^2] = \frac{n^2}{M^4\tilde{\lambda}^2}H_M^2 + \frac{n}{M^3\tilde{\lambda}^2}G_M, \tag{4.48}$$

with $M = n^b$ where $0 < b \leq 1$. Repeating a similar analysis as above with (4.48) instead of (4.17) for large n yields the result. ■

We note that a gain of M in the MIMO-like transmissions can be the result of highly correlated transmission times from a node in the source cell to M nodes in the destination cell in the physical layer. That is, all M links in between a node in the source cell and nodes in the destination cell have highly correlated transmission times such that we consider the fastest of the simultaneously active M links, one for each node in the source cell, in Phase II as opposed to the fastest of M^2 links. Further, we note that the proposed transmission scheme achieves a better age scaling than the delay scaling of [128] which also uses MIMO transmissions for inter-cell transmissions and achieves a delay scaling of $O(n^{\frac{1}{2}}\log n)$ per-user while sacrificing the throughput performance.

Third, we consider the case in which the mega update packet transmission rate depends explicitly on the number of packets in the mega updates, M. So far, the inter-cell transmissions of mega update packets are modeled by i.i.d. exponential random variables with rate $\tilde{\lambda}$. Since these mega update packets are comprised of M update packets, one for each node of a particular cell, here, we scale down the exponential random variable with the mega update packet size, M. That is, we let

$\tilde{X}$ denote the transmission delay of a mega update packet such that the transmission delay is exponentially distributed with rate $\frac{\tilde{\lambda}}{M}$ instead of $\tilde{\lambda}$ and analyze the performance of the proposed three-phase transmission policy in the next corollary.

Corollary 4.3 *When the mega update packet transmission delays, $\tilde{X}$ of Phase II are modeled by exponential random variables with rate $\frac{\tilde{\lambda}}{M}$, the proposed three-phase transmission policy achieves an average age scaling of $O(n^{\frac{1}{3}} \log n)$ per-user.*

We note that, in this case, the first two moments of the duration of the second phase are again given by (4.48) which again yields the same result for large n.

Next, we consider the cases discussed in Corollaries 4.2 and 4.3 together. That is, we analyze the performance of the proposed three-phase scheme when the speedup factor is M, as in the ordinary MIMO scheme, with the inter-cell transmission of mega update packets modeled by scaled-down exponential random variables with the mega update packet size, M. In this case, the proposed scheme achieves a per-user scaling of $O(n^{\frac{1}{2}} \log n)$.

Lastly, we consider the case in which a single mega update packet transmission is modeled by an Erlang distribution with rate $(M, M^2\tilde{\lambda})$. That is, inter-cell transmission of a single packet has i.i.d. exponential delays with rate $\tilde{\lambda}$ such that MIMO-like inter-cell transmission of a single packet has an i.i.d. exponentially distributed delay with rate $M^2\tilde{\lambda}$. We note that this case is essentially equivalent to the case when mega update packets are not utilized, as discussed in Corollary 4.1, since when we have M successive transmissions for a mega packet transmission, then each packet is only sent to its destination cell. In other words, a destination cell receives

only the packets that are destined to that particular cell as opposed to receiving the whole mega update packet. Thus, when the mega update packets have Erlang service times, the proposed three-phase policy achieves an average age scaling of $O(n^{\frac{1}{3}} \log n)$ per-user.

We note that in Corollaries 4.1 to 4.3, we consider the performance of the proposed scheme without invoking the hierarchical utilization and the corresponding hierarchical extensions of these settings differ from Section 4.5 and are not discussed within the scope of this chapter.

4.9 Conclusion

Given a large wireless network of fixed area consisting of n randomly located source-destination pairs that want to send time-sensitive status update packets to each other, we have studied the scalability of age of information. To accommodate the communication between the S-D pairs, we have proposed a three-phase transmission scheme which uses local cooperation between nodes and mega update packets to achieve an average age scaling of $O(n^{\frac{1}{4}} \log n)$. Our scheme divides the network into $\frac{n}{M}$ cells of M nodes each. The first and third phases include intra-cell transmissions and take place simultaneously across all cells. The second phase includes inter-cell transmissions and therefore during this phase cells operate one at a time.

We observe that the bottleneck in the resulting age scaling result is caused by $O(M)$ transmissions in Phases I and III. Furthermore, we note that each cell is a scaled-down version of the whole network. With these, we introduce hierarchy to the

system and apply the three-phase scheme on a cell-level in these phases. In other words, Phases I and III of the hth level of the hierarchy are completed in three successive steps in the next level of hierarchy. We have shown that this scheme with hierarchical cooperation improves the scaling result and achieves an average age scaling of $O\left(n^{\alpha(h)} \log n\right)$ where $\alpha(h) = \frac{1}{3 \cdot 2^h + 1}$ and h is the number of hierarchy levels. In the asymptotic case when h tends to ∞ resulting average age per-user scales as $O(\log n)$.

CHAPTER 5

Scaling Laws for Version Age of Information in Gossip Networks

5.1 Introduction

The original age metric measures the time passed since the most recent information at the monitor was generated at the source node. This age metric increases linearly in time in the absence of update deliveries at the monitor. When an update is received, the age value drops to the age of the received update. This evolution in time demonstrates the fundamental limitation of the original age metric, which is the assumption that the age at the monitor continues to increase as time passes irrespective of any changes at the source side in the underlying source process. That is, even if the source information does not change and the monitor has the most up-to-date information, as time passes, the original age metric deems monitor's knowledge about the source process *stale*. This may not necessarily be the case in many applications, including content delivery services and surveillance systems. To overcome this inherent challenge, in the age of information literature, several variants of the original age metric have been proposed. A common feature of these

recently proposed age variants is the fact that the age of the monitor stays the same until the information at the source changes even if no updates are received at the monitor. Among these are binary freshness metric [90, 91, 133–136], age of synchronization [137], and age of incorrect information [138–140].

Similar in spirit, recently, a new age metric called *version age* has appeared in the literature [141, 142]. In the version age context, each update at the source is considered a version change so that the version age counts how many versions out-of-date the information at the monitor is, compared to the version at the source. Unlike the original age metric, the version age has discrete steps such that the version age of a monitor increases by one when the source generates a newer version, i.e., fresher information. In between version changes at the source, version age of the monitor stays the same indicating that the monitor still has the most recent information. A predecessor of version age has appeared in [51], which considers timely tracking of Poisson counting processes by minimizing the count difference, i.e., version difference, between the process and its estimate.

Recently, reference [141] has used the version age metric to characterize timeliness in memoryless gossip networks composed of n arbitrarily connected nodes. In [141], the source sends information to the receiver nodes by implementing a Poisson updating mechanism, i.e., with exponential inter-update times at each receiver node. Similar Poisson updating schemes have been investigated in the age literature in the context of social networks [42], timely tracking [51, 143], and timely cache updating [90, 91, 135]. In addition to source delivering updates to the receiver nodes, each node in [141] relays their stored version of the source information to

their neighboring nodes. Also referred to as *gossiping*, this additional information exchange among the nodes improves the age scaling at the nodes since each node can receive updates from its neighbors as well as from the source node. As a result of this gossiping, [141] shows that the average version age scales as $O(\sqrt{n})$ in a bi-directional ring network and as $O(\log n)$ in a fully connected network, where n is the number of nodes; note that the average version age would scale as $O(n)$ without gossiping, i.e., if the network is disconnected.

There have been significant efforts in the age literature to characterize and improve the average age scaling in large networks considering the classical age metric with possibly many source-destination pairs. Recent works have achieved $O(1)$ scaling in multicast networks in Chapters 2 and 3 and references [113, 144] using a centralized transmission scheme administered by the source, and $O(\log n)$ scaling in distributed peer-to-peer communication networks in Chapter 4 and references [145–147] using a hierarchical local cooperation scheme.

Inspired by these, in this chapter, our aim is to study *version age scaling* in more general gossip network models which exhibit a community structure; see Fig. 5.1. In our model, there is a single source node that generates updates following a Poisson process. Each such update constitutes a newer version of the underlying information process. The source updates multiple distinct communities regarding the underlying process. In our work, a community represents a set of receiver nodes clustered together which can only interact with each other. Each cluster has a dedicated cluster head, which serves that particular cluster. Akin to base stations in a cellular network, cluster heads act as gateways between the source and the receiver

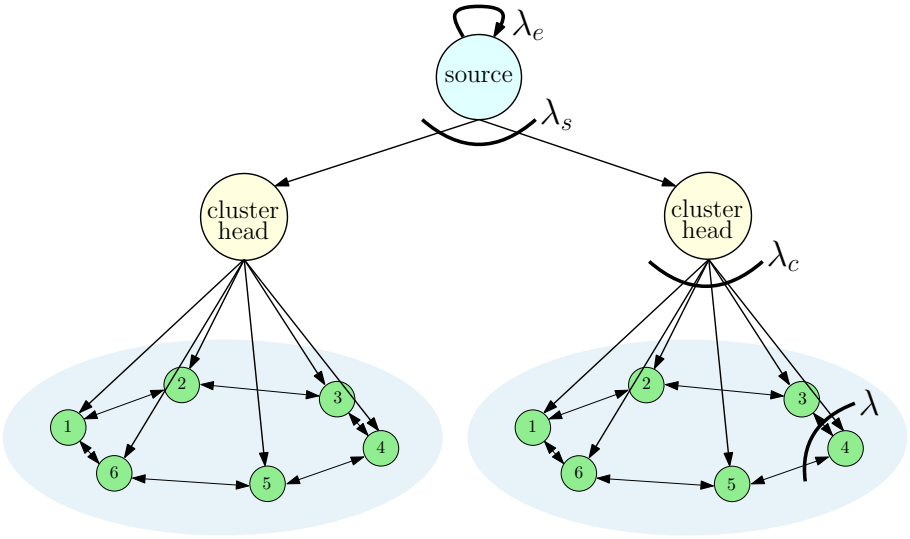


Figure 5.1: Tiered network model where blue node at the center represents the source, yellow nodes represent the cluster heads, and green nodes represent the end users. Here, nodes in each cluster form a bi-directional ring network. Other possible network topologies within a cluster are shown in Fig. 5.2.

nodes in each cluster. Unlike the model in [141], the source cannot directly deliver updates to receiver nodes in our model. Instead, source updates need to go through the corresponding cluster head to reach the receiver nodes in each cluster. There can be various degrees of gossip in each cluster, which we model by disconnected, uni-directional ring, bi-directional ring, and fully connected network topologies; see Fig. 5.2. Based on the underlying connectivity within clusters, we characterize the version age experienced by each node. In doing that, we employ the stochastic hybrid systems (SHS) approach [13, 38, 148–151] to develop recursive formulas that enable us to characterize the version age in arbitrarily connected clustered gossip networks.

Additional hop constituted by the cluster heads between the source and the end-nodes presents us with opportunities to optimize the average version age scaling by carefully tuning the number of clusters and the cluster size. Specifically, our

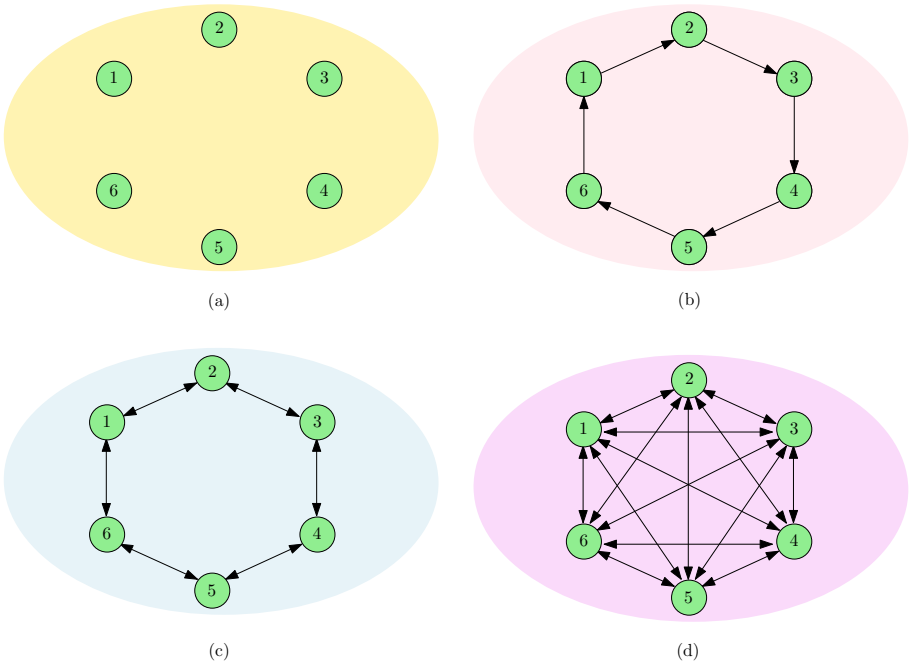


Figure 5.2: Different network topologies that can be used within each cluster: (a) disconnected, (b) uni-directional ring, (c) bi-directional ring, and (d) fully connected. Fig. 5.1 uses the one in (c). In this figure, cluster size is $k = 6$.

results indicate that even if the nodes within each community forego gossiping, i.e., disconnected networks within each cluster, we can achieve $O(\sqrt{n})$ scaling as opposed to $O(n)$. In addition, we obtain the same $O(\log n)$ scaling in the case of fully connected communities using fewer connections within clusters than [141], and further reduce the scaling result in ring networks to $O(n^{\frac{1}{3}})$ from $O(\sqrt{n})$ in [141].

So far, the cluster heads do not participate in gossiping, i.e., cluster heads among themselves form a disconnected topology. To further improve the version age at the receiver nodes, next, we characterize the average version age and its scaling when the cluster heads form a ring network among themselves and exchange information. In that case, each cluster head uses some of its update rate to relay updates to its neighboring cluster heads while its remaining update rate is used

to relay updates to the receiver nodes within its cluster. Thanks to the increased communication among the cluster heads, we can further improve the version age scaling to $O(n^{\frac{1}{3}})$ for disconnected networks within each cluster; to $O(n^{\frac{1}{4}})$ in the case of ring networks within each cluster. For the setup with a ring network in each cluster, we find the version age optimal update rate allocation at each cluster head. Interestingly, in the case of fully connected networks within each cluster, we find that the additional information exchange due to the gossip among the cluster heads does not improve the version age scaling. That is, the version age of an end user still scales as $O(\log n)$ even though cluster heads participate in gossip.

Motivated by the tiered structure in the clustered network model, next, we introduce hierarchy to our clustered network model. In this case, we forego cluster heads, and carefully place clusters of nodes in a hierarchical manner. That is, each node in a particular hierarchy level acts as a cluster head to a distinct cluster of nodes in the next hierarchy level. At the first level of hierarchy, we have a single cluster of k_1 nodes, each of which have a single cluster of k_2 nodes at the second level, and so on. Within the context of hierarchical clustered gossip networks, we consider a ring network in each cluster and show that the $O(\sqrt{n})$ scaling result of [141] and our cluster head-aided scaling result of $O(n^{\frac{1}{3}})$ for ring networks can be improved to $O(n^{\frac{1}{2h}})$ without the use of dedicated cluster heads, where h denotes the number of hierarchy levels. For convenience, we provide the summary of all scaling results for version age in Table 5.1. Finally, through numerical evaluations, we determine the version-age optimum cluster sizes for varying update rates employed by the source, cluster heads, and the nodes within each cluster.

	disconnected	ring	fully connected
no clustering as in [141]	$O(n)$	$O(\sqrt{n})$	$O(\log n)$
clustered networks	$O(\sqrt{n})$	$O(n^{\frac{1}{3}})$	$O(\log n)$
clustered networks with connected cluster heads	$O(n^{\frac{1}{3}})$	$O(n^{\frac{1}{4}})$	$O(\log n)$
h-level hierarchical clustered networks	–	$O(n^{\frac{1}{2h}})$	–

Table 5.1: The summary of the scaling of version age in gossip networks.

5.2 System Model and the Age Metric

We consider a system where a network of n nodes is divided into m clusters, each consisting of k nodes such that $n = mk$ with $k, m \in \mathbb{Z}$; see Fig. 5.1. Each cluster is served by a distinct cluster head, which takes updates from the source and distributes them across that cluster. The source process is updated as a rate λ_e Poisson process. The source has a total update injection rate of λ_s, which is uniformly distributed across cluster heads such that each cluster head is updated as a rate $\frac{\lambda_s}{m}$ Poisson process. From each cluster head to its corresponding cluster, the total update injection rate is λ_c and this rate is uniformly allocated across the nodes in that cluster. That is, each node i receives an update from its cluster head as a rate $\frac{\lambda_c}{k}$ Poisson process with $i \in \mathcal{N} \triangleq \{1, \ldots, n\}$.

Nodes in each cluster are connected to each other based on a connection graph. We consider varying levels of connectivity among nodes within each cluster. These are disconnected, uni-directional ring, bi-directional ring, and fully connected networks, which are shown in Fig. 5.2 for a cluster of $k = 6$ nodes. Updates received from the cluster head associated with each cluster are distributed across that cluster by utilizing the connections between the nodes. A node i updates another node j

as a rate λ_{ij} Poisson process. Each node in this system has a total update rate of λ, which is uniformly allocated to its neighboring nodes. That is, in the uni-directional ring, each node updates its neighbor node as a rate λ Poisson process, whereas in bi-directional ring, each node has two neighboring nodes, each of which is updated as a rate $\frac{\lambda}{2}$ Poisson process. In the fully connected cluster, each node has $k-1$ neighbors each of which is updated as a rate $\frac{\lambda}{k-1}$ Poisson process. As a result of these local connections within a cluster, a node can receive different versions of the source update from its neighboring nodes in addition to the source updates received via its cluster head.

To model the age at each node, we use the version age metric [141, 142]. We denote the version of the update at the source as $N_s(t)$, at cluster head c as $N_c(t)$, with $c \in \mathcal{C} \triangleq \{1, \ldots, m\}$, and at node i as $N_i(t)$, with $i \in \mathcal{N}$, at time t. The version age at node i is given by $\Delta_i(t) = N_s(t) - N_i(t)$. Similarly, the version age at cluster head c is $\Delta_c(t) = N_s(t) - N_c(t)$. When node i has the same version as the source, its version age becomes zero, i.e., $\Delta_i(t) = 0$. When the information at the source is updated, version ages at the cluster heads and the nodes increase by 1, e.g., $\Delta'_c(t) = \Delta_c(t) + 1$. Each node i can get updates either from its cluster head or the other nodes that it is connected to within its cluster. When node i gets an update from its cluster head, its version age becomes

$$\Delta'_i(t) = \min\{\Delta_c(t), \Delta_i(t)\} = \Delta_c(t). \tag{5.1}$$

Last equality in (5.1) follows since nodes in a cluster receive source updates through

their cluster head so that they have either the same version or older versions of the information compared to their cluster head. When node i receives an update from node j, its version age becomes

$$\Delta_i'(t) = \min\{\Delta_i(t), \Delta_j(t)\}. \tag{5.2}$$

That is, node i's version age is updated only if node j has a newer version of the source information. Otherwise, the version age at node i is not updated.

5.3 Version Age with Community Structure

In this section, we characterize the limiting version age of each node i, denoted by

$$\Delta_i = \lim_{t\to\infty} \mathbb{E}[\Delta_i(t)], \quad i \in \{1, \ldots, n\}, \tag{5.3}$$

considering various network topologies for the clusters. Since the network model in each cluster is identical and within each cluster the network is symmetric for each of the network topologies, age processes $\Delta_i(t)$ of all users are statistically identical. Thus, in the ensuing analysis, we focus on a single cluster $c \in \mathcal{C}$ and find the average version age of a node from that cluster. For this, we follow the construction in [141] and express Δ_i in terms of Δ_S, which denotes the average version age of an arbitrary subset S of the nodes in cluster c, where

$$\Delta_S(t) \triangleq \min_{j\in S} \Delta_j(t). \tag{5.4}$$

We recall the following definitions from [141]: $\lambda_i(S)$ denotes the total update rate at which a node i from cluster c updates the nodes in set S. We have

$$\lambda_i(S) = \begin{cases} \sum_{j \in S} \lambda_{ij}, & i \notin S \\ 0, & i \in S. \end{cases} \tag{5.5}$$

Similarly, $\lambda_c(S)$ denotes the total update rate of the cluster head of a particular cluster into the set S. Finally, set of updating neighbors of a set S in cluster c is

$$N_c(S) = \{i \in \mathcal{N} = \{1, \ldots, n\} : \lambda_i(S) > 0\}. \tag{5.6}$$

That is, the set $N_c(S)$ includes all updating neighbors of set S in cluster c excluding the cluster head. The total set of updating neighbors of set S is given by $N(S) = c \cup N_c(S)$.

With these definitions, next, in Theorem 5.1 below we give the resulting version age in our clustered system model as a specialization of [141, Theorem 1].

Theorem 5.1 *When the total network of n nodes is divided into m clusters, each of which consisting of a single cluster head and k nodes with $n = mk$, the average version age of subset S that is composed of nodes within a cluster c is given by*

$$\Delta_S = \frac{\lambda_e + \lambda_c(S)\Delta_c + \sum_{i \in N_c(S)} \lambda_i(S)\Delta_{S \cup \{i\}}}{\lambda_c(S) + \sum_{i \in N_c(S)} \lambda_i(S)}, \tag{5.7}$$

with $\Delta_c = m\frac{\lambda_e}{\lambda_s}$.

Proof: Proof of Theorem 5.1 follows by applying [141, Theorem 1] to our clustered network model and noting that updates arrive at the nodes through designated cluster heads. For completeness, we show the key steps in the proof below.

In our system, whenever there is an update being forwarded, a state transition occurs. We first present possible state transitions. We use $\mathcal{L}$ to denote the set of possible state transitions. Then, we have

$$\mathcal{L} = \{(s,s)\} \cup \{(s,c) : c \in \mathcal{C}\} \cup \{(c,i) : c \in \mathcal{C}, i \in \mathcal{N}\} \cup \{(i,j) : i,j \in \mathcal{N}\}, \quad (5.8)$$

where the first transition occurs when the source generates a new update, the second set of transitions occur when the source node updates a cluster head $c \in \mathcal{C}$. The third set of transitions occur when a cluster head c updates a node in its cluster and finally the last set of transitions occur when an end user updates another end user from its cluster. In clustered gossip networks, different than [141], as a result of transition (i,j), the version age of an end user evolves as

$$\Delta_k' = \begin{cases} \Delta_k + 1, & i = j = s, k \in \mathcal{N}, \\ \Delta_c, & i = c, j = k \in \mathcal{N}, \\ \min(\Delta_i, \Delta_j), & i \in \mathcal{N}, j = k \in \mathcal{N}, \\ \Delta_k, & \text{otherwise}, \end{cases} \quad (5.9)$$

where Δ_k' is the version age of node k after the transition. In (5.9), the version age of node k increases by one when the source generates a new update and becomes

equal to the version age of its cluster head when node k receives an update from its cluster head as explained in (5.1). When node k receives an update from another node in its cluster, its version age is updated only if the updating node has a newer version of the source information as shown in (5.2).

Considering an arbitrary subset S of nodes within a cluster with the version age evolution described in (5.9), we deduce that after the (s, s) transition, the version age of set S is increased by one. For all other transitions (i, j) with $j \in S$, we have

$$\Delta'_S = \min_{k \in S} \Delta'_k = \min_{k \in S \cup \{i\}} \Delta_k = \Delta_{S \cup \{i\}}. \tag{5.10}$$

When $i = c$, from (5.10), we have $\Delta'_S = \min_{k \in S \cup \{i\}} \Delta_k = \Delta_c$. If $j \notin S$, the version age of set S is affected by transition (i, j), i.e., $\Delta'_S = \Delta_S$. Using (5.9) and (5.10) and following similar steps as in [141] yields the result. ■

5.3.1 Version Age in Clustered Disconnected Networks

Nodes in a cluster are not connected to each other. Thus, the network is a two-hop multicast network, where the first hop is from the source to m cluster heads, and the second hop is from each cluster head to k nodes; combine Fig. 5.1 with Fig. 5.2(a). Multihop networks have been studied in Chapters 2 and 3 and references [113, 144] considering the classical age metric, where the source keeps sending update packets until they are received by a certain number of nodes at each hop. We do not consider such centralized management of updates, but let the source update the cluster heads as Poisson processes, and let cluster heads forward these packets to the nodes within

their clusters as further Poisson processes.

Let S_1 denote an arbitrary 1-node subset of a cluster. Subset S_1 is only connected to the cluster head, i.e., $N_c(S_1) = \emptyset$. Using the recursion given in (5.7), we find

$$\Delta_{S_1} = \Delta_c + k\frac{\lambda_e}{\lambda_c} = m\frac{\lambda_e}{\lambda_s} + k\frac{\lambda_e}{\lambda_c}, \tag{5.11}$$

where Δ_{S_1} denotes the version age of a single node from the cluster. When the network consists of two-hops, version age is additive, in that the first term in (5.11) corresponds to the first hop and is equal to the version age at the cluster head, whereas the second term in (5.11) corresponds to the version age at the second hop between the cluster head and a node.

Theorem 5.2 *In a clustered network of disconnected users, the version age of a single user scales as* $O(\sqrt{n})$.

Theorem 5.2 follows by selecting $k = \sqrt{n}$ with $m = \frac{n}{k} = \sqrt{n}$ in (5.11) for fixed λ_e, λ_s, λ_c, which do not depend on n. Theorem 5.2 indicates that when nodes are grouped into $\sqrt{n}$ clusters, an age scaling of $O(\sqrt{n})$ is achievable even though users forego gossiping. With the absence of cluster heads, i.e., when the source is uniformly connected to each of the n users, the version age scaling of each disconnected user would be $O(n)$. By utilizing clusters, we incur an additional hop, but significantly improve the scaling result from $O(n)$ to $O(\sqrt{n})$.

5.3.2 Version Age in Clustered Ring Networks

Nodes in each cluster form a ring network. We consider two types of ring clusters: uni-directional ring as shown in Fig. 5.2(b) and bi-directional ring as shown in Fig. 5.2(c).

First, we consider the uni-directional ring and observe that an arbitrary subset of j adjacent nodes S_j has a single neighbor node that sends updates with rate λ for $j \leq k-1$. Each such subset S_j receives updates from the cluster head with a total rate of $j\frac{\lambda_c}{k}$. Next, we use the recursion in (5.7) to write

$$\Delta_{S_j} = \frac{\lambda_e + j\frac{\lambda_c}{k}\Delta_c + \lambda\Delta_{S_{j+1}}}{j\frac{\lambda_c}{k} + \lambda}, \tag{5.12}$$

for $j \leq k-1$ where Δ_c is the version age at the cluster head. We note that when $j = k$ the network becomes a simple two-hop network similar to that of Section 5.3.1 and we find $\Delta_{S_k} = m\frac{\lambda_e}{\lambda_s} + \frac{\lambda_e}{\lambda_c}$.

Next, we consider the bi-directional ring and observe that an arbitrary subset S_j that consists of any adjacent j nodes has two neighbor nodes, each with an incoming update rate of $\frac{\lambda}{2}$ for $j < k-1$. When $j = k-1$, S_j has a single neighboring node that sends updates with a total rate $2\frac{\lambda}{2} = \lambda$. For $j \leq k-1$, the cluster head sends updates to subset S_j with a total rate of $j\frac{\lambda_c}{k}$. With all these, when we apply the recursion in (5.7), we obtain exactly the same formula given in (5.12).

Lemma 5.1 *Both uni-directional and bi-directional ring cluster models yield the same version age for a single node when each node in a cluster has a total update*

rate of λ.

Lemma 5.1 follows from the fact that either type of ring cluster induces the same recursion for an arbitrary subset of any adjacent j nodes within a cluster as long as the total update rate per node λ is the same. Thus, in the remainder of this chapter, we only consider the bi-directional ring cluster model.

Before focusing on age scaling in a clustered network with a ring topology in each cluster, we revisit the ring network in [141], and provide a proof of the $1.25\sqrt{n}$ age scaling result observed therein as a numerical result. We show that the approximate theoretical coefficient is $\sqrt{\frac{\pi}{2}} = 1.2533$.

Lemma 5.2 *For the ring network model considered in [141], the version age of a user scales as* $\Delta_{S_1} \approx \sqrt{\frac{\pi}{2}}\frac{\lambda_e}{\lambda}\sqrt{n}$.

Proof: From recursive application of [141, Eqn. (17)], we obtain

$$\Delta_{S_1} = \frac{\lambda_e}{\lambda}\left(\sum_{i=1}^{n-1} a_i^{(n)} + a_{n-1}^{(n)}\right), \tag{5.13}$$

where $a_i^{(n)}$ is given for $i = 1, \ldots, n-1$ as

$$a_i^{(n)} = \prod_{j=1}^{i} \frac{1}{1+\frac{j}{n}}. \tag{5.14}$$

We note that $a_i^{(n)}$ decays fast in i, and consider $i = o(n)$,

$$-\log(a_i^{(n)}) = \sum_{j=1}^{i} \log\left(1+\frac{j}{n}\right) \approx \sum_{j=1}^{i} \frac{j}{n} = \frac{i(i+1)}{2n} \approx \frac{i^2}{n} \tag{5.15}$$

where we used $\log(1+x) \approx x$ for small x, and ignored the i term relative to i^2. Thus, for small i, we have $a_i^{(n)} \approx e^{-\frac{i^2}{2n}}$. For large i, $a_i^{(n)}$ converges quickly to zero due to multiplicative terms in $\prod_{j=1}^{i} \frac{1}{1+j/n}$, and this approximation still holds. Thus, we have $\sum_{i=1}^{n-1} a_i^{(n)} \approx \sum_{i=1}^{n-1} e^{-\frac{i^2}{2n}}$. For large n, by using Riemann sum approximation with steps $\frac{1}{\sqrt{n}}$, we obtain

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n-1} a_i^{(n)} \approx \frac{1}{\sqrt{n}} \sum_{i=1}^{n-1} e^{-\frac{i^2}{2n}} = \int_0^{\infty} e^{-\frac{t^2}{2}}\, dt = \sqrt{\frac{\pi}{2}}. \tag{5.16}$$

Thus, we get $\sum_{i=1}^{n-1} a_i^{(n)} \approx \sqrt{\frac{\pi}{2}}\sqrt{n}$. By inserting this in (5.13), we obtain the age scaling of a user as $\Delta_{S_1} \approx \sqrt{\frac{\pi}{2}}\frac{\lambda_e}{\lambda}\sqrt{n}$. ■

Next, we focus on age scaling in a clustered network with a ring topology in each cluster. From recursive application of (5.12) along with Δ_{S_k}, we obtain

$$\Delta_{S_1} = \frac{\lambda_e}{\lambda} \left(\sum_{i=1}^{k-1} b_i^{(k)} \right) + \Delta_c \left(1 - b_{k-1}^{(k)} \right) + \Delta_{S_k} b_{k-1}^{(k)}, \tag{5.17}$$

where similar to (5.14), $b_i^{(k)}$ is given for $i = 1, \ldots, k-1$ as

$$b_i^{(k)} = \prod_{j=1}^{i} \frac{1}{1 + \frac{j}{k}\frac{\lambda_c}{\lambda}}. \tag{5.18}$$

When k is large, $b_{k-1}^{(k)}$ goes to zero, and Δ_{S_1} in (5.17) becomes

$$\Delta_{S_1} \approx \frac{\lambda_e}{\lambda} \left(\sum_{i=1}^{k-1} b_i^{(k)} \right) + \Delta_c \approx \sqrt{\frac{\pi}{2}} \frac{\lambda_e}{\sqrt{\lambda \lambda_c}} \sqrt{k} + m \frac{\lambda_e}{\lambda_s}, \tag{5.19}$$

where the second approximation follows as in the proof of Lemma 5.2. Terms in

(5.19) are $O(\sqrt{k})$ and $O(m)$, respectively. In [141], there is a single cluster, i.e., $m = 1$ and $k = n$, and thus, the version age scaling is $O(\sqrt{n})$. In our model, by carefully adjusting the number of clusters and the cluster sizes, we can improve this $O(\sqrt{n})$ scaling result to $O(n^{\frac{1}{3}})$.

Theorem 5.3 *In a clustered network with a ring topology in each cluster, the version age of a single user scales as* $O(n^{\frac{1}{3}})$.

Theorem 5.3 follows by selecting $m = n^{\frac{1}{3}}$ with $k = \frac{n}{m} = n^{\frac{2}{3}}$ in (5.19) for fixed λ_e, λ_s, λ_c, λ, which do not depend on n.

5.3.3 Version Age in Clustered Fully Connected Networks

Nodes in each cluster form a fully connected network where each node is connected to all the other nodes within its cluster with rate $\frac{\lambda}{k-1}$. We find the version age for a subset of j nodes S_j in a cluster. Each such subset j has $k - j$ neighbor nodes in addition to the cluster head associated with their cluster. Using the recursion given in (5.7), we find

$$\Delta_{S_j} = \frac{\lambda_e + \frac{j\lambda_c}{k}\Delta_c + \frac{j(k-j)\lambda}{k-1}\Delta_{S_{j+1}}}{\frac{j\lambda_c}{k} + \frac{j(k-j)\lambda}{k-1}}, \tag{5.20}$$

for $j \leq k-1$, where Δ_c is equal to $m\frac{\lambda_e}{\lambda_s}$. The average version age of the whole cluster is $\Delta_{S_k} = \Delta_c + \frac{\lambda_e}{\lambda_c} = m\frac{\lambda_e}{\lambda_s} + \frac{\lambda_e}{\lambda_c}$.

Next, we present bounds for Δ_{S_1}.

Lemma 5.3 *When* $\lambda_c = \lambda$, *in a clustered network with fully connected topology in*

each cluster, the version age of a single node satisfies

$$\frac{(k-1)^2+k}{k^2}\Delta_c + \frac{\lambda_e}{\lambda}\left(\frac{k-1}{k}\sum_{\ell=1}^{k-1}\frac{1}{\ell}+\frac{1}{k}\right) \leq \Delta_{S_1} \leq \Delta_c + \frac{\lambda_e}{\lambda}\left(\sum_{\ell=1}^{k}\frac{1}{\ell}\right). \tag{5.21}$$

Proof: We use steps similar to those in the proof of [141, Theorem 2] and also consider the additional hop from the source to the cluster heads. For $\lambda_c = \lambda$, we take $j = k - \ell$ and (5.20) becomes

$$\Delta_{S_{k-\ell}} = \frac{\frac{1}{k-\ell}\frac{\lambda_e}{\lambda} + \frac{1}{k}\Delta_c + \frac{\ell}{k-1}\Delta_{S_{k-\ell+1}}}{\frac{1}{k}+\frac{\ell}{k-1}}, \tag{5.22}$$

for $\ell \leq k-1$ and $\Delta_{S_k} = \Delta_c + \frac{\lambda_e}{\lambda}$, where Δ_c is the age at the cluster head. Defining $\hat{\Delta}_{S_\ell} \triangleq \Delta_{S_{k-\ell+1}}$, we get

$$\hat{\Delta}_{S_{\ell+1}} = \frac{\frac{1}{k-\ell}\frac{\lambda_e}{\lambda} + \frac{1}{k}\Delta_c + \frac{\ell}{k-1}\hat{\Delta}_{S_\ell}}{\frac{1}{k}+\frac{\ell}{k-1}}. \tag{5.23}$$

Next, one can show that $\hat{\Delta}_{S_{\ell+1}}$ satisfies the following

$$\hat{\Delta}_{S_{\ell+1}} \leq \frac{\frac{1}{k-\ell}\frac{\lambda_e}{\lambda} + \frac{1}{k}\Delta_c + \frac{\ell}{k}\hat{\Delta}_{S_\ell}}{\frac{1}{k}+\frac{\ell}{k}}. \tag{5.24}$$

Defining $\tilde{\Delta}_{S_\ell} \triangleq \frac{\ell}{k}\hat{\Delta}_{S_\ell}$ and plugging it in (5.24), we get

$$\tilde{\Delta}_{S_{\ell+1}} = \frac{\ell+1}{k}\hat{\Delta}_{S_\ell} \leq \frac{1}{k-\ell}\frac{\lambda_e}{\lambda} + \frac{1}{k}\Delta_c + \tilde{\Delta}_{S_\ell}. \tag{5.25}$$

Noting that $\tilde{\Delta}_{S_1} = \frac{\hat{\Delta}_{S_1}}{k} = \frac{\Delta_{S_k}}{k} = \frac{1}{k}\left(\Delta_c + \frac{\lambda_e}{\lambda}\right)$, we write

$$\tilde{\Delta}_{S_k} \leq \Delta_c + \frac{\lambda_e}{\lambda}\left(\sum_{\ell=1}^{k}\frac{1}{\ell}\right). \tag{5.26}$$

Since $\tilde{\Delta}_{S_k} = \hat{\Delta}_{S_k} = \Delta_{S_1}$, (5.26) presents an upper bound to the version age of a single node. For the lower bound, we start with (5.23) and observe that we have

$$\hat{\Delta}_{S_{\ell+1}} \geq \frac{k-1}{\ell+1}\left(\frac{1}{k-\ell}\frac{\lambda_e}{\lambda} + \frac{1}{k}\Delta_c + \frac{\ell}{k-1}\hat{\Delta}_{S_\ell}\right). \tag{5.27}$$

Defining $\bar{\Delta}_{S_\ell} \triangleq \frac{\ell}{k-1}\hat{\Delta}_{S_\ell}$ and using it in (5.27) gives

$$\bar{\Delta}_{S_{\ell+1}} = \frac{\ell+1}{k-1}\hat{\Delta}_{S_{\ell+1}} \geq \frac{1}{k-\ell}\frac{\lambda_e}{\lambda} + \frac{1}{k}\Delta_c + \bar{\Delta}_{S_\ell}. \tag{5.28}$$

Starting with the fact that $\bar{\Delta}_{S_1} = \frac{1}{k-1}\hat{\Delta}_{S_1} = \frac{m+1}{k-1}\frac{\lambda_e}{\lambda}$, the recursion in (5.28) yields

$$\Delta_{S_1} \geq \frac{(k-1)^2+k}{k^2}\Delta_c + \frac{\lambda_e}{\lambda}\left(\frac{k-1}{k}\sum_{\ell=1}^{k-1}\frac{1}{\ell} + \frac{1}{k}\right), \tag{5.29}$$

upon noting that $\Delta_{S_1} = \frac{k-1}{k}\bar{\Delta}_{S_k}$, which concludes the proof of the lemma. ■

From (5.21), we see that for large n with $\lambda_c = \lambda$, the version age of a single node Δ_{S_1} satisfies

$$\Delta_{S_1} \approx m\frac{\lambda_e}{\lambda_s} + \frac{\lambda_e}{\lambda}\log k. \tag{5.30}$$

Theorem 5.4 *In a clustered network with a fully connected topology in each cluster,*

the version age of a single user scales as $O(\log n)$.

Theorem 5.4 follows in multiple different ways. For instance, it follows by selecting $m = 1$ and $k = \frac{n}{m} = n$. That is, we have a single fully connected network of n users as in [141]. Theorem 5.4 also follows by selecting $m = \log n$ and $k = \frac{n}{m} = \frac{n}{\log n}$. That is, we have $\log(n)$ fully connected clusters with $\frac{n}{\log n}$ users in each cluster. Thus, version age obtained under a smaller cluster size with less connections is the same as that obtained when all nodes are connected to each other. In particular, in our model with $m = \log n$, each node has $O(\frac{n}{\log n})$ connections in comparison to $O(n)$ in [141].

Finally, we note that, a recurring theme in the analysis of clustered networks is the fact that the version age at an end-node Δ_{S_1} is *almost* additive in the version age at the cluster head Δ_c as seen in (5.11), (5.17), and (5.21). It is *exactly* additive in the case of disconnected clusters in (5.11).

5.4 Version Age with Community Structure Under Connected Cluster Heads

So far, we have studied the cases in which the cluster heads are disconnected among themselves, and consequently, they do not exchange information with each other. In this section, we model the connectivity among the cluster heads with a bi-directional ring (see Fig. 5.3).[1] [2] Thus, in this section, at the first tier, we have a ring network

[1]The model studied in Section 5.3 corresponds to the case in which the cluster heads form a disconnected topology.

[2]In addition to the bi-directional ring topology, one can study the version age considering fully connected cluster heads, which is omitted here to keep the discussion focused.

of m cluster heads, each of which is serving its own cluster. Nodes in each cluster form a disconnected, ring, or fully connected network. Our aim in this section, is to analyze the effect of additional information exchange among the cluster heads on the average version age experienced by the end nodes.

When there is no information exchange among the cluster heads, i.e., disconnected cluster heads, each cluster head updates its cluster with a total rate of λ_c. In the case of information exchange among the cluster heads, a cluster head updates its neighboring cluster heads as a rate λ_{ca} Poisson process and updates its cluster with a total rate of λ_{cb}, where $\lambda_{ca} + \lambda_{cb} = \lambda$. Thus, when the cluster heads are connected, each cluster head receives source information with a larger rate but updates its cluster with a smaller rate.

The average version age of a subset S that is composed of nodes within a cluster c is still given by (5.1) when the cluster heads exchange information in our clustered network topology. This is because when the cluster heads exchange information among themselves, the network topology within a cluster does not change, i.e., the $N_c(S)$ stays the same, and the nodes in a cluster still cannot have a lower version age than their cluster head.

The only change in (5.1) compared to Section 5.3 is the average version age of a particular cluster head c, Δ_c. As shown in Fig. 5.3, even though cluster heads are connected to the nodes in their respective clusters, each cluster head can be updated by the source node or its neighboring cluster heads. That is, to find Δ_c, we only need to look at the first tier of the network, which is the ring gossiping network presented in [141]. Thus, using Lemma 5.2 we find the average version age

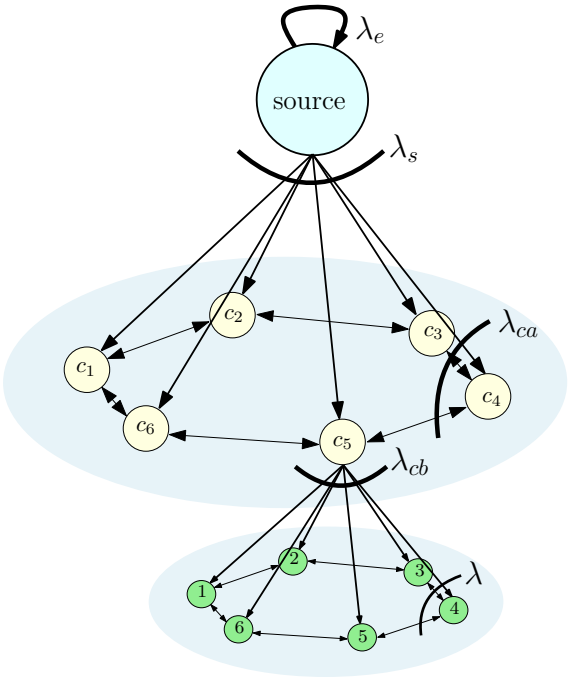


Figure 5.3: Tiered network model where blue node represents the source, yellow nodes represent the cluster heads $c_1, \ldots, c_6$, and green nodes represent the end users. Here, cluster heads form a bi-directional ring network with $m = 6$. Each cluster is associated with a cluster of $k = 6$ nodes. Here, only one such cluster is shown. Nodes in each cluster form a bi-directional ring network. Other possible network topologies within a cluster are shown in Fig. 5.2.

of a single cluster head when the cluster heads form a ring network as

$$\Delta_c \approx \sqrt{\frac{\pi}{2}} \frac{\lambda_e}{\sqrt{\lambda_s \lambda_{ca}}} \sqrt{m}. \tag{5.31}$$

We note that in (5.31), we have $\sqrt{\lambda_s \lambda_{ca}}$ in the denominator unlike [141] as λ_s and λ_{ca} are not necessarily equal in our model. We observe in (5.31) that a single cluster head's average version age approximately scales as $O(\sqrt{m})$ as opposed to $O(m)$ in Theorem 5.1 since cluster heads now form a ring network. With that, in what follows, we analyze the average version age scaling for different cluster topologies when the cluster heads form a ring network.

5.4.1 Version Age in Clustered Disconnected Networks with Connected Cluster Heads

The network model in this case is as in Section 5.3.1 except that the cluster heads form a ring network in the first hop. Nodes within a cluster are disconnected, i.e., $N_c(S_1) = \emptyset$. By invoking Theorem 5.1, we use the recursion in (5.7) and find the average version age of a single node as

$$\Delta_{S_1} = \Delta_c + k\frac{\lambda_e}{\lambda_{cb}} \approx \sqrt{\frac{\pi}{2}}\frac{\lambda_e}{\sqrt{\lambda_s\lambda_{ca}}}\sqrt{m} + k\frac{\lambda_e}{\lambda_{cb}}, \tag{5.32}$$

where the approximation follows from (5.31).

Theorem 5.5 *In a clustered network of disconnected users when the cluster heads form a ring network, the average version age of a single user scales as* $O(n^{\frac{1}{3}})$.

Theorem 5.5 follows by selecting, $k = n^{\frac{1}{3}}$ with $m = \frac{n}{k} = n^{\frac{2}{3}}$ in (5.32) for fixed λ_e, λ_s, λ_{ca} and λ_{cb} that do not depend on n. Theorem 5.5 implies that even though nodes in clusters do not gossip, by utilizing the information exchange at the cluster head level, the average version age scaling of an end user can be improved to $O(n^{\frac{1}{3}})$ from $O(\sqrt{n})$ in Theorem 5.2.

Another interesting observation is the parallelism between (5.32) and (5.19) due to the *almost* additive structure of the average version age in clustered gossip networks. In the case of (5.19), cluster heads are disconnected and nodes in each cluster form a ring network whereas in the case of (5.32) the network is reversed,

i.e., ring network in cluster heads, disconnected network within clusters. Since, we have $n = mk$, both (5.32) and (5.19) yield the same $O(n^{\frac{1}{3}})$ average version age scaling at the end users indicating that gossiping equally helps improving the average version age scaling at the end users whether it occurs at the cluster head level or within clusters at the end user level even though possibly newer versions of the source information is exchanged at the cluster head level as cluster heads are directly connected to the source node.

5.4.2 Version Age in Clustered Ring Networks with Connected Cluster Heads

Ring networks are formed both at the cluster head level and within clusters at the end user level. By invoking Theorem 5.1, we use the recursion in (5.7) and after following similar steps as in Section 5.3.2, we find

$$\Delta_{S_1} \approx \sqrt{\frac{\pi}{2}} \frac{\lambda_e}{\sqrt{\lambda_s \lambda_{ca}}} \sqrt{m} + \sqrt{\frac{\pi}{2}} \frac{\lambda_e}{\sqrt{\lambda \lambda_{cb}}} \sqrt{k}. \tag{5.33}$$

We note that (5.33) is the counterpart of (5.19) where the additional ring network topology at the cluster head level is considered.

Theorem 5.6 *In a clustered network with a ring topology in each cluster when the cluster heads form a ring network, the average version age of a single user scales as* $O(n^{\frac{1}{4}})$.

Theorem 5.6 follows by selecting $m = \sqrt{n}$ with $k = \frac{n}{m} = \sqrt{n}$ in (5.33) for

fixed λ_e, λ_s, λ_{ca}, and λ_{cb}. Here, we note that, when gossiping is employed both at the cluster head level and at the end user level within clusters through a ring topology, the average version age scaling at the end users is improved from $O(n^{\frac{1}{3}})$ in Theorem 5.1 to $O(n^{\frac{1}{4}})$ in Theorem 5.6.

Another interesting observation is the fact that since the network topology is symmetric at both levels in this case, and the average version age at the end users is *almost* additive in the average version age at the cluster heads, the average version age scaling optimal m and k values are identical.

We can also note that the selection of the update rates at the cluster heads, i.e., λ_{ca} and λ_{cb} in (5.33) is critical. After the optimal selection of $m = k = \sqrt{n}$, next, we optimize update rates at the cluster heads, i.e., λ_{ca} and λ_{cb}, to further minimize the version age in (5.33). After selecting $m = k = \sqrt{n}$, Δ_{S_1} in (5.33) becomes

$$\Delta_{S_1} \approx \sqrt{\frac{\pi}{2}} \lambda_e n^{\frac{1}{4}} \left(\frac{1}{\sqrt{\lambda_s \lambda_{ca}}} + \frac{1}{\sqrt{\lambda \lambda_{cb}}} \right). \tag{5.34}$$

The minimization of Δ_{S_1} in (5.34) is equivalent to solving the following optimization problem

$$\begin{aligned} \min_{\{\lambda_{ca}, \lambda_{cb}\}} \quad & \frac{1}{\sqrt{\lambda_s \lambda_{ca}}} + \frac{1}{\sqrt{\lambda \lambda_{cb}}} \\ \text{s.t.} \quad & \lambda_{ca} + \lambda_{cb} = \lambda_c \\ & \lambda_{ca} \geq 0, \quad \lambda_{cb} \geq 0. \end{aligned} \tag{5.35}$$

We write the Lagrangian for the optimization problem in (5.35) as,

$$\mathcal{L} = \frac{1}{\sqrt{\lambda_s \lambda_{ca}}} + \frac{1}{\sqrt{\lambda \lambda_{cb}}} + \beta(\lambda_{ca} + \lambda_{cb} - \lambda_c) - \theta_1 \lambda_{ca} - \theta_2 \lambda_{cb}, \tag{5.36}$$

where $\theta_1 \geq 0$, $\theta_2 \geq 0$, and β can be anything. We note that the problem in (5.35) is jointly convex with respect to λ_{ca} and λ_{cb}. Thus, by analyzing the KKT conditions we can find the optimal solution. We write the KKT conditions as,

$$\frac{\partial \mathcal{L}}{\partial \lambda_{ca}} = -\frac{1}{2\lambda_s} \lambda_{ca}^{-\frac{3}{2}} + \beta - \theta_1 = 0, \tag{5.37}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{cb}} = -\frac{1}{2\lambda} \lambda_{cb}^{-\frac{3}{2}} + \beta - \theta_2 = 0. \tag{5.38}$$

Then, by using the KKT conditions in (5.37) and (5.38), we find the optimal λ_{ca} and λ_{cb} as

$$\lambda_{ca} = \frac{\lambda_c \lambda^{\frac{1}{3}}}{\lambda^{\frac{1}{3}} + \lambda_s^{\frac{1}{3}}}, \tag{5.39}$$

$$\lambda_{cb} = \frac{\lambda_c \lambda_s^{\frac{1}{3}}}{\lambda^{\frac{1}{3}} + \lambda_s^{\frac{1}{3}}}. \tag{5.40}$$

We observe from (5.39) and (5.40) that when the cluster heads also form a ring network, it is optimal to choose the update rates among the cluster heads, i.e., λ_{ca}, proportional to the $\frac{1}{3}$-power of the update rate of the end users, i.e., $\lambda^{\frac{1}{3}}$. Similarly, the total update rate allocated by cluster heads to their own clusters should be proportional to the $\frac{1}{3}$-power of the update rate of the source, i.e., $\lambda_s^{\frac{1}{3}}$.[3]

[3]Similar optimization problems can be formulated for the clustered disconnected networks in Section 5.4.1, and for the clustered fully connected networks in Section 5.4.3. In order to avoid

5.4.3 Version Age in Clustered Fully Connected Networks with Connected Cluster Heads

In this case, nodes within a cluster form a fully connected network whereas each cluster head is only connected to its adjacent neighbors in a ring topology. By invoking Theorem 5.1, we use the recursion in (5.7) and after following similar steps as in Section 5.3.3, we find

$$\Delta_{S_1} \approx \sqrt{\frac{\pi}{2}} \frac{\lambda_e}{\sqrt{\lambda_s \lambda_{ca}}} \sqrt{m} + \frac{\lambda_e}{\lambda} \log k, \tag{5.41}$$

for large n, with $\lambda_{cb} = \lambda$. We note that (5.41) is analogous to (5.30) when the cluster heads form a ring network.

Theorem 5.7 *In a clustered network with a fully connected topology in each cluster when the cluster heads form a ring network, the average version age of a single user scales as* $O(\log n)$.

Theorem 5.7 follows by noting that since $k = \frac{n}{m}$ we cannot get rid of the $\log n$ term in (5.41). Thus, there are multiple (m, k) pairs that result in the same $O(\log n)$ scaling. For example, when $m = 1$ and $k = \frac{n}{m} = n$, we obtain $O(\log n)$ scaling. This implies that having a single cluster of n users as in [141]. In addition, selecting $m = \log^2 n$ and $k = \frac{n}{m} = \frac{n}{\log^2 n}$ yields Theorem 5.7 as well, parallel to the discussion after Theorem 5.4.

repetitive arguments, we skip the update rate optimizations of the cluster heads for these parts and provide the analysis only for the clustered ring networks.

An interesting observation from Theorem 5.7 is the fact that additional communication at the cluster head level does not improve the average version age scaling at the end users. In a similar fashion, one can show that, even if the cluster heads form a fully connected network among themselves while the nodes in each cluster also form a fully connected network, the average version age at the end users scales as $O(\log n)$. This is due to the fact that the number of clusters m (hence the number of cluster heads) and the number of nodes in each cluster k are such that $n = mk$. Since the level of gossip, i.e., the connectivity among the cluster heads and the nodes in each cluster, cannot be increased beyond fully connected networks, one can conclude that the average version age scaling cannot be improved further than $O(\log n)$ in the considered clustered gossip networks.

We note that once the cluster heads exchange information among themselves, essentially, what we end up with is a hierarchical gossip networks, where in the first level of hierarchy we have m cluster heads, and in the second level of hierarchy we have mk end nodes clustered into m clusters of k nodes each. Inspired by this structure, in the next section, we forego cluster heads and study the version age in hierarchical clustered gossip networks.

5.5 Version Age in Hierarchical Clustered Gossip Networks

In this section, we consider a hierarchical clustered gossip network, where at the first level of hierarchy there is a single cluster of k_1 nodes. Each node in the first level is directly updated by the source node as a rate $\frac{\lambda_s}{k_1}$ Poisson process. Total update

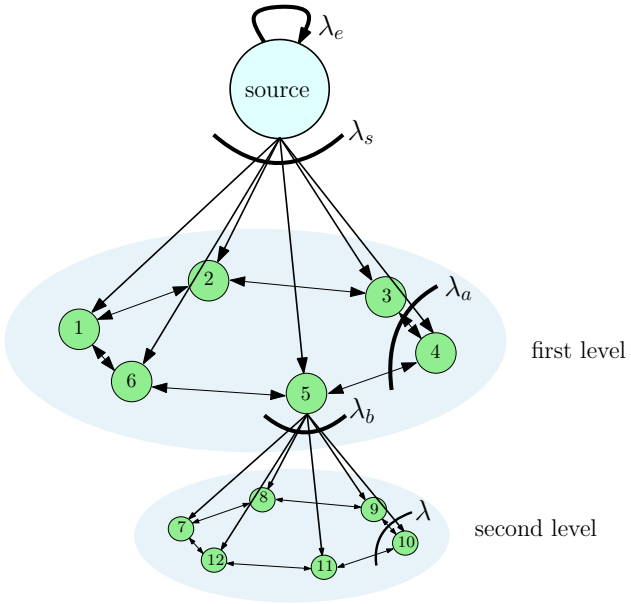


Figure 5.4: Two-level hierarchical network model where blue node represents the source and green nodes represent the end users. Here, nodes in each cluster form a bi-directional ring network of $k_1 = k_2 = 6$ nodes. We have a single cluster in the first level and $k_1 = 6$ clusters in the second level. Here, only one such second level cluster is shown. Other possible network topologies within a cluster are shown in Fig. 5.2.

rate of each node is λ. Each node spends λ_a portion of this total rate to update its neighbors within the same cluster at the same hierarchical level and spends λ_b portion of the total λ rate to update its neighbors in the next hierarchical level such that $\lambda_a + \lambda_b = \lambda$. Each node in the first level is associated with a cluster of k_2 nodes in the second level. That is, in the second level, we have k_2 clusters and a total of $k_1 k_2$ nodes. In Fig. 5.4, we show the network model for a two hierarchy levels. We note that at the last level of the network, e.g., the second level in the case of Fig. 5.4, nodes use all of their update rate λ to update their neighbors within the same cluster.

Within the scope of this section, we assume that nodes in each cluster at every

hierarchical level form a bi-directional ring network.[4] Let h denote the number of hierarchy levels in the network. Then, at level i, with $i \leq h-1$, each node updates each of its two neighbors at level i as a Poisson process with rate $\frac{\lambda_a}{2}$, whereas it updates each of its k_{i+1} child nodes at level $i+1$ as a rate $\frac{\lambda_b}{k_{i+1}}$ Poisson process. We have a total of $\prod_{j=1}^{i} k_j$ nodes at level i that are grouped into equal-sized clusters of k_i for $i \geq 2$.

Due to the symmetry in the network model, nodes in each hierarchy level i have statistically identical version age processes $\Delta^i_{S_1}$. In what follows, we find the average version age expressions of a single node at each hierarchical level.

Theorem 5.8 *When the total network of n nodes is grouped into h levels of hierarchical clusters with k_i nodes in each cluster at the ith hierarchy level such that $\sum_{i=1}^{h} \prod_{j=1}^{i} k_j = n$, the average version age of subset S that is composed of nodes within a cluster c at the ith hierarchical level is given by*

$$\Delta^i_S = \frac{\lambda_e + \lambda_{i-1}(S)\Delta^{i-1}_{S_1} + \sum_{j \in N_c(S)} \lambda_j(S)\Delta^i_{S\cup\{j\}}}{\lambda_{i-1}(S) + \sum_{j \in N_c(S)} \lambda_j(S)}, \tag{5.42}$$

where $\lambda_{i-1}(S)$ denotes the total update rate at which cluster c's parent node in level $i-1$ updates the nodes in set S.

The proof of Theorem 5.8 follows from that of Theorem 5.1 by noting that the version age of a node in level i cannot be smaller than that of its parent node in

[4]One can consider disconnected or fully connected networks within each cluster at each level as well. In our model, since the average version age under fully connected networks cannot be improved beyond $O(\log n)$ as discussed in Section 5.4 and we want to keep our discussion focused, we limit ourselves with a ring network in each cluster at every level.

level $i-1$. That is, from the perspective of a node in level i, its cluster head in the case of Theorem 5.1 corresponds to its parent node in level $i-1$.

We first present the results for $h=3$ levels of hierarchy to showcase the version age behavior in the hierarchical gossip networks and then generalize our results to h level of hierarchy.[5]

5.5.1 Version Age for $h=3$ Hierarchy Levels

In this case, we have a single cluster of k_1 nodes in the first level, k_1 clusters of k_2 nodes each in the second level, and k_2 clusters of k_3 nodes in each in the third level of hierarchy so that $n=k_1+k_1k_2+k_1k_2k_3$. By using the recursion in Theorem 5.8 and noting that at the first level of hierarchy we have the ring network model in [141], we find

$$\Delta_{S_1}^1 \approx \sqrt{\frac{\pi}{2}}\frac{\lambda_e}{\sqrt{\lambda_s\lambda_a}}\sqrt{k_1}, \tag{5.43}$$

$$\Delta_{S_1}^2 \approx \sqrt{\frac{\pi}{2}}\frac{\lambda_e}{\sqrt{\lambda_s\lambda_a}}\sqrt{k_1}+\sqrt{\frac{\pi}{2}}\frac{\lambda_e}{\sqrt{\lambda_a\lambda_b}}\sqrt{k_2}, \tag{5.44}$$

$$\Delta_{S_1}^3 \approx \sqrt{\frac{\pi}{2}}\frac{\lambda_e}{\sqrt{\lambda_s\lambda_a}}\sqrt{k_1}+\sqrt{\frac{\pi}{2}}\frac{\lambda_e}{\sqrt{\lambda_a\lambda_b}}\sqrt{k_2}+\sqrt{\frac{\pi}{2}}\frac{\lambda_e}{\sqrt{\lambda\lambda_b}}\sqrt{k_3}. \tag{5.45}$$

Theorem 5.9 *In a hierarchical clustered network with $h=3$ hierarchy levels and a ring network in each cluster, the average version age of a single user scales as $O(n^{\frac{1}{6}})$ at every hierarchy level.*

[5]For $h=2$ hierarchy levels, the resulting average version age expressions are in the same format as those in Section 5.4.2 and correspondingly yield an $O(n^{\frac{1}{4}})$ scaling at an end user. This is because the cluster heads forming a ring network at the first tier in Section 5.4.2 can essentially be thought of as the first level of hierarchy in the context of hierarchical clustered gossip networks analyzed in this section.

Theorem 5.9 follows by observing that $n = k_1 + k_1k_2 + k_1k_2k_3 = k_1(1 + k_2(1 + k_3)) \approx k_1k_2k_3$ for large n. That is, when the number of nodes in the network gets large, order-wise majority of the nodes are located at the final hierarchy level. From the symmetry of the cluster topologies at each hierarchy level and the additive structure observed in (5.43)-(5.45), we select $k_1 = k_2 = k_3 = O(n^{\frac{1}{3}})$, which yields an average version age scaling of $O(n^{\frac{1}{6}})$ for all three hierarchical levels.

We note that taking $n \approx k_1k_2k_3$ is as if we assume all the nodes are located at the last hierarchical level of the network so that the $O(n^{\frac{1}{6}})$ scaling we find in Theorem 5.9 represents a worst case scenario since nodes located at the first two hierarchy levels surely have smaller average version age than the nodes located at the last level of the hierarchy.

Theorem 5.9 shows that by implementing a three-level hierarchical clustered gossip network structure, we can improve the average version age of a single end-user in our model compared to $O(n^{\frac{1}{2}})$ in [141], $O(n^{\frac{1}{3}})$ in Section 5.3.2, and $O(n^{\frac{1}{4}})$ in Section 5.4.2. The improvement in Section 5.3.2 compared to the model in [141] originates from the use of m cluster heads with smaller ring networks under each cluster head compared to the single ring network of n nodes in [141]. When the cluster heads participate in gossip in Section 5.4.2, end users have better average version age scaling due to the additional information exchange at the cluster heads. Finally, in this section, through hierarchical placement of clusters, we obtain the same scaling as in Section 5.4.2 with $h = 2$ levels without getting help from any dedicated cluster heads and further improve the scaling result when we employ $h = 3$ hierarchy levels. That is, by carefully placing all n nodes into hierarchical clusters of

ring networks, we get the best average version age scaling at an end user compared all these network models discussed so far.[6]

5.5.2 Version Age for $h > 3$ Hierarchy Levels

In a hierarchical clustered gossip network with h hierarchy levels and a ring network topology in each cluster at every hierarchy level, the average version age of a single node located in hierarchy level i is

$$\Delta_{S_1}^{i} = \begin{cases} \sqrt{\frac{\pi}{2}} \frac{\lambda_e}{\sqrt{\lambda_s \lambda_a}} \sqrt{k_i}, & i = 1, \\ \Delta_{S_1}^{1} + \sqrt{\frac{\pi}{2}} \frac{\lambda_e}{\sqrt{\lambda_a \lambda_b}} \sum_{j=2}^{i} \sqrt{k_j}, & 1 < i < h, \\ \Delta_{S_1}^{i-1} + \sqrt{\frac{\pi}{2}} \frac{\lambda_e}{\sqrt{\lambda \lambda_b}} \sqrt{k_i}, & i = h. \end{cases} \tag{5.46}$$

Theorem 5.10 *In a hierarchical clustered network with h hierarchy levels and a ring network in each cluster, the average version age of a single user scales as $O(n^{\frac{1}{2h}})$ at every hierarchy level.*

Theorem 5.10 follows by approximating $n \approx \prod_{i=1}^{h} k_i$ for large n and taking $k_i = n^{\frac{1}{h}}$ for $i \in 1, \ldots, h$.

5.6 Numerical Results

We have seen in Sections 5.3-5.5 that the version age depends on update rates λ_e, λ_s, λ_c, and λ. In this section, we explore the effects of these rates on the version age

[6]We note that average version age scaling of an end node is improved through hierarchical clustering at the expense of increased number of connections in the network, which may incur additional operational cost to the service provider. This aspect will be discussed in Section 5.7.

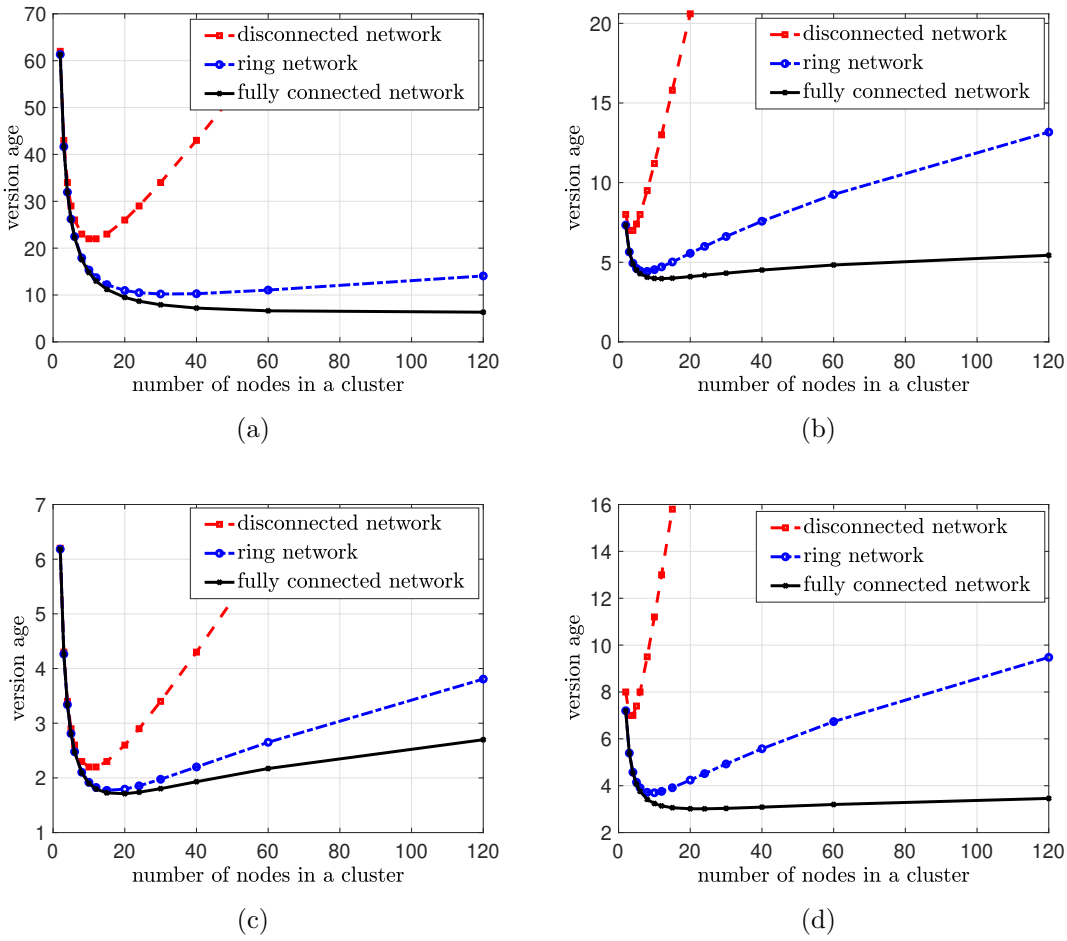


Figure 5.5: Version age of a node with fully connected, ring, and disconnected cluster models with $n = 120$, (a) $\lambda_e = 1$, $\lambda_s = 1$, $\lambda_c = 1$, and $\lambda = 1$, (b) $\lambda_e = 1$, $\lambda_s = 10$, $\lambda_c = 1$, and $\lambda = 1$, (c) $\lambda_e = 1$, $\lambda_s = 10$, $\lambda_c = 10$, and $\lambda = 1$, (d) $\lambda_e = 1$, $\lambda_s = 10$, $\lambda_c = 1$, and $\lambda = 2$.

via numerical results. In the first four simulations, we consider the model described in Section 5.3.

First, we take $\lambda_e = 1$, $\lambda_s = 1$, $\lambda_c = 1$, $\lambda = 1$, and $n = 120$. We plot the version age of a node for the considered cluster models with respect to k. We see in Fig. 5.5(a) that for the fully connected cluster model, the version age decreases with k and thus, the version age-optimal cluster size is $k^* = 120$, i.e., all n nodes are

grouped in a single cluster. In the ring cluster model, the version age is minimized when $k^* = 30$. In the disconnected cluster model, the version age is minimized when we have $k^* = 10$ (or equivalently $k^* = 12$). From these, we deduce that when the topology has less connectivity in a cluster, the optimal cluster size is smaller. Further, a topology with larger connectivity within a cluster achieves a lower version age.

Second, we consider the same setting as in Fig. 5.5(a) but take $\lambda_s = 10$ in Fig. 5.5(b). Here, the version age decreases with increasing k at first due to increasing number of connections within a cluster and the increase in the update rate between the source and each cluster head (as the number of clusters decreases with increasing k). However, as k continues to increase, the decrease in the update rate from the cluster head to the nodes starts to dominate and the version age increases for all cluster models. In Fig. 5.5(b), we see that the optimal cluster size is $k^* = 12$ in fully connected clusters, $k^* = 8$ in ring clusters, $k^* = 3$ and $k^* = 4$ in disconnected clusters.

Third, we increase the update rate of the cluster heads and take $\lambda_c = 10$. We see in Fig. 5.5(c) that the optimum value of k increases compared to the second case when cluster heads have a larger update rate in all the cluster models. We find $k^* = 20$ in fully connected clusters, $k^* = 15$ in ring clusters, and $k^* = 10$ or $k^* = 12$ in disconnected clusters.

Fourth, we study the effect of update rates among the nodes. For this, we take $\lambda_c = 10$, $\lambda_e = 1$, $\lambda_s = 1$, $\lambda = 2$. We see in Fig. 5.5(d) that as the communication rate between the nodes increases, the optimal cluster size increases, and it is equal

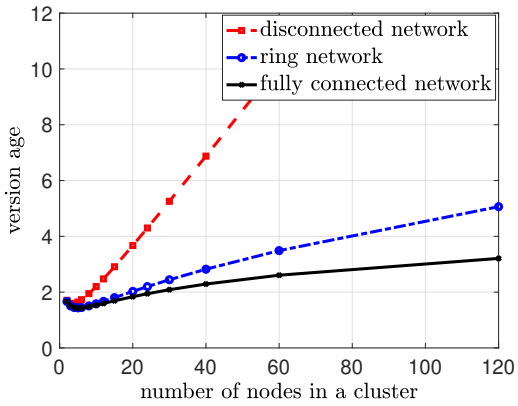


Figure 5.6: Version age of a node with fully connected, ring, and disconnected cluster models with $n = 120$, $\lambda_e = 1$, $\lambda_s = 10$, $\lambda_{ca} = 4$, $\lambda_{cb} = 6$, and $\lambda = 1$ when cluster heads form a ring network among themselves.

to $k^* = 24$ in fully connected clusters, and $k^* = 10$ in ring clusters. As there is no connection between nodes in the case of disconnected clusters, the optimum cluster size remains the same, i.e., $k^* = 3$ or $k^* = 4$, compared to Fig. 5.5(b).

Next, we look at the version age when the cluster heads form a ring network as in Section 5.4. For this simulation, we use the setup of Fig. 5.5(c) and take $\lambda_{ca} = 4$ and $\lambda_{cb} = 6$. We see in Fig. 5.6 for all network types within clusters, i.e., disconnected, ring, and fully connected networks, that version age of a single end node improves when the cluster heads exchange information among themselves. In addition, we observe that with the additional gossip at the cluster heads, the version age optimal cluster size for each case is now smaller. Comparison of the optimal cluster size and the corresponding minimum version age achieved for each type of network is given in Table 5.2 with and without gossip at the cluster heads. In Table 5.2, we observe that the biggest version age improvement is obtained in the case of disconnected clusters, as the additional communication at the cluster heads

	no gossip at cluster heads	gossiping cluster heads
disconnected	$(2.2000, 10)$	$(1.5936, 4)$
ring	$(1.7729, 15)$	$(1.4365, 5)$
fully connected	$(1.7111, 20)$	$(1.4291, 5)$

Table 5.2: Comparison of the $(\Delta^*_{S_1}, k^*)$ pairs with (as in Section 5.4) and without (as in Section 5.3) gossip at the cluster heads.

is more valuable when nodes within clusters are not connected at all.

In our last numerical result, we look at the version age in hierarchical clustered gossip networks as in Section 5.5 with $h = 3$ hierarchy levels. Here, we consider the number of nodes $n = 120$, and take $\lambda_e = 1$, $\lambda_s = 1$, and total update rate of a node as $\lambda = 5$. In this simulation, we consider different (λ_a, λ_b) pairs, and find the optimal cluster sizes at each hierarchical level k_1, k_2, and k_3 that minimize the version age of a node at the last hierarchical level as these nodes experience the highest version age in the network. We note that the selection of the (λ_a, λ_b) is important. While choosing a large λ_a increases the connectivity between the nodes within clusters, and thus can lower the version age of the nodes at the same hierarchical level, it may also increase the version age of the nodes at the higher hierarchical levels. For this reason, among the (λ_a, λ_b) pairs given in Table 5.3, we see that choosing $\lambda_a = 2$, and $\lambda_b = 3$ achieves the lowest version age with the optimum cluster sizes equal to $k_1 = 3$, $k_2 = 13$, and $k_3 = 2$. We also note that when λ_a is relatively small, i.e., $\lambda_a = 1$, most of the nodes are placed at the third hierarchical level, i.e., out of $n = 120$ nodes, $k_1k_2k_3 = 78$ nodes are placed at the third hierarchical level whereas $k_1 = 3$ nodes and $k_1k_2 = 39$ nodes are placed in the first and the second hierarchical levels, respectively. As we increase the connectivity among the nodes within the

λ_a	λ_b	k_1	k_2	k_3	$\Delta^3_{S_1}$
1	4	3	3	12	3.7992
2	3	3	13	2	3.7239
3	2	3	13	2	3.9143
4	1	8	7	1	4.3354

Table 5.3: Average version age of a single node at the third hierarchy level, $\Delta^3_{S_1}$, with $h = 3$, $n = 120$, $\lambda_e = 1$, $\lambda_s = 1$, and $\lambda_a + \lambda_b = 5$. For given (λ_a, λ_b) pairs, we find the optimum k_1, k_2, and k_3 values that minimize $\Delta^3_{S_1}$.

same level (λ_a), we see that the number of nodes at the upper hierarchical levels increases.

5.7 Conclusion

We considered a system where there is a single source and n receiver nodes that are grouped into distinct equal-sized clusters. Nodes in each cluster participate in gossiping to relay their stored versions of the source information to their neighbors. We considered four different types of connectivity among the nodes within the same cluster: disconnected, uni-directional ring, bi-directional ring, and fully connected. First, we considered the use of dedicated cluster heads in each cluster that facilitate communication between the source and the receiver nodes. For each of these network models, we found the average version age and its scaling as a function of the network size n. In particular, we showed that an average version age scaling of $O(\sqrt{n})$, $O(n^{\frac{1}{3}})$, and $O(\log n)$ is achievable per user in disconnected, ring, and fully connected cluster topologies. We then allowed information exchange among the cluster heads and showed that the version age scaling in the case of disconnected and ring networks in each cluster can be improved to $O(n^{\frac{1}{3}})$ and $O(n^{\frac{1}{4}})$, respectively, when the

cluster heads also participate in gossiping through a ring formation. Interestingly, we observed that the increased gossip among the cluster heads does not improve the version age scaling when the nodes in each cluster form a fully connected network. Finally, we implemented a hierarchical clustered gossip structure and showed that per user average version scaling of $O(n^{\frac{1}{2h}})$ is achievable in the case of ring networks in each cluster, where h denotes the number of hierarchy levels, even without the aid of the dedicated cluster heads. We numerically determined the optimum cluster sizes that minimize the version age for varying update rates at the source, cluster heads, and the nodes.

Here, the version age scaling improvement from the model in [141] to our hierarchical clustered gossip network design comes at the expense of increased number of connections at the network. For example, considering a ring network in each cluster, the $O(\sqrt{n})$ scaling result of [141] is obtained by $3n$ connections whereas our cluster head-aided $O(n^{\frac{1}{3}})$ scaling result is achieved as a result of a total $3n + n^{\frac{1}{3}}$ connections in the network. When the cluster heads also form a ring network this number increases to $3n+3\sqrt{n}$ to achieve an $O(n^{\frac{1}{4}})$ scaling result. Finally, in the case of hierarchical clustered gossip networks with h levels, we have a total of $3\sum_{i=1}^{h} n^{\frac{i}{h}}$ connections in the network which yield a per node average version age scaling of $O(n^{\frac{1}{2h}})$. Considering the operational cost of each such connection, service providers can design the network structure, i.e., the use of cluster heads, number of hierarchy levels along with the level of connectivity in each cluster, based on the operational budget to obtain the desired level of information freshness at the receiver nodes.

CHAPTER 6

Timely Distributed Computation with Stragglers

6.1 Introduction

In many real-time monitoring applications including autonomous driving, surveillance systems and predictive maintenance, time-sensitive data that are collected by sensors or mobile devices require processing to extract the embedded information. However, these devices cannot perform heavy computations due to battery related issues or their limited computational capabilities. These type of status update packets that require computation are called *computation-intensive messages*. References that are most closely related to our work are [152–158] which study queueing, packet management and scheduling in such status update systems. Common to all these works is the fact that they consider a single computation server. That is, although references [152–158] consider systems with computation-intensive status update packets, our work is the first work on age of information that considers multiple servers working in a distributed manner to process the update packets.

In this chapter, we consider a system in which there is a source node which uploads computation-intensive time-critical status updates to a computation unit

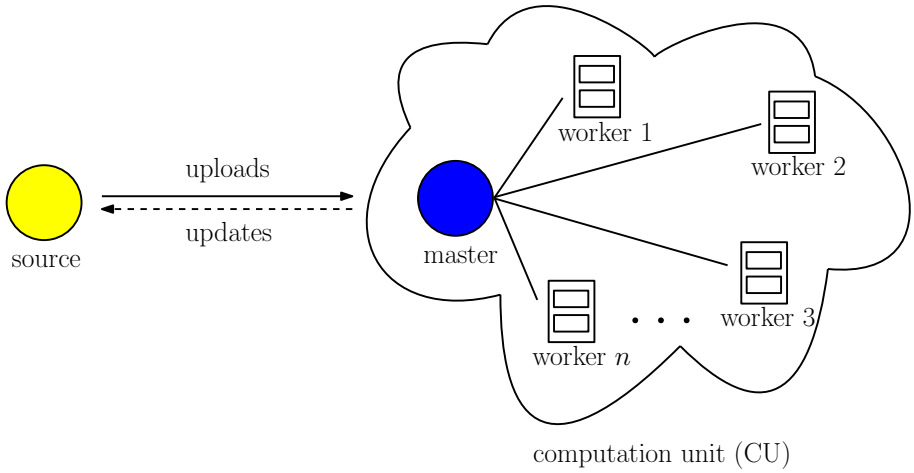


Figure 6.1: System model with a single source node and a computation unit (CU) that consists of a master node and n identical worker nodes.

(CU) which consists of a single master node and n worker nodes that perform the computations (see Fig. 6.1). We assume that the required computation on the update is a linear operation such as large matrix multiplication. This brings up the concept of computation distribution among the worker nodes. Computation distribution and scheduling problem has been extensively studied particularly in the context of machine learning with a focus on completion time and straggler threshold analysis [159–170].

Inspired by the recent distributed computation literature, we investigate the timeliness of uncoded and coded computation distribution algorithms that are used to combat the stragglers, i.e., nodes that are slower than the average. Unlike the existing distributed computation literature which uses metrics such as expected overall runtime to evaluate the performance of distributed computation systems [159], our goal is to characterize the age of information in these systems and design computation distribution algorithms that can combat stragglers as well as achieve a

minimum average age of information. The source node collects time-sensitive data and sends them to the CU for processing over a channel with random transmission delays. Arriving packets at the CU go into service (computation) if the CU is idle by the time of their arrival. Otherwise, they are dropped. Here, the master node distributes the overall computation to n worker nodes using uncoded or coded schemes. Computation time at each worker is random. Once the master node collects sufficiently many results from the worker nodes to decode the computation result, it *updates* the source node. We note that in our model destination node is also the source node. One such application is autonomous driving cars which capture images/videos of the surroundings and send them to a CU for computation. The source node in the mean time adopts a zero-wait policy such that it sends the next update packet once the current one reaches the CU.

We derive the average age for uncoded and coded schemes, and show that asymptotically MDS coded scheme outperforms the uncoded and repetition coded schemes, i.e., MDS coded scheme achieves a smaller average age. In addition, we observe that when worker nodes have multiple computations to perform (MM-MDS coded scheme), age performance of MDS coded scheme further improves. Our results also indicate that given that the source node and the CU implement zero-wait and dropping policies, respectively, when the transmission delays are i.i.d. exponentials and computation times are i.i.d. shifted exponentials, for large n, minimizing age of information is equivalent to minimizing the computation time which is not the case in general. Finally, we find the optimal repetition, MDS and MM-MDS codes that minimize the average age of information.

6.2 System Model and Age Metric

We consider a system (see Fig. 6.1), where there is a single source node which sends time-sensitive computation-intensive status updates, i.e., packets that require additional processing to extract the embedded information, to a CU which consists of a single master node and n worker nodes. Worker nodes have statistically identical computing capabilities. From the source node to the CU, update packets experience i.i.d. exponential transmission delays. Upon successful arrival of a packet to the CU, the master node distributes this computation task to n worker nodes. Here, we use status update packet and computation task interchangeably. Each worker node performs the computation, which is assumed to be a linear operation, and sends the result back to the master node. We note that one such computation task example is large matrix multiplication prevalent in machine learning applications.

When the master node receives sufficiently many responses from the worker nodes, it aggregates the results and *updates* the source node. We neglect the transmission delay from the CU back to the source node after computation as the size of the initial packet is in general much larger than the resulting update packet after computation.

Thus, in our model, packets that are able to enter the CU experience two stages: transmission and computation. Random variable D_j denotes the transmission delay of the jth update packet and is exponentially distributed with parameter λ. To model the computation times at the worker nodes, we adopt the model in [159] and assume the existence of a mother runtime distribution. This distribution cor-

responds to the computation time when the whole computation on the update is performed by a single worker, X, and has a shifted exponential distribution with (c, μ) where $c > 0$ is the shift and μ is the rate which is also the straggling parameter. When the update packet is divided into m subpackets, the computation time of each subpacket has the scaled-down (i.e., sped-up) version of the overall distribution, i.e., shifted exponential with $\left(\frac{c}{m}, m\mu\right)$. We note that computation times at the worker nodes also account for the time spent to communicate the inputs and outputs with the master node within the CU. Here, the constant shift makes sure that computation times cannot go below a certain value whereas the exponential part constitutes the tail of the computation time distribution. This is inline with the computation times observed in systems like Google Trace [168].

The source node receives an instantaneous ACK upon delivery to the CU and sends the next update as soon as the current one reaches the CU, i.e., the source node adopts a zero-wait policy. We note that, in a possibly more intuitive setting, the source node may send the next update upon receiving the computation result back from the CU in which case there is no need for a separate ACK signal which is discussed in Footnote 1 in Section 6.3. On the other hand, the CU implements a dropping policy in which when busy it neglects any update packets arriving from the source node. Thus, packets sent by the source node can only enter the computation stage if the CU is idle at the time of their arrival. Upon finishing a computation task, the CU immediately sends back the result and waits for the next packet arrival. This idle waiting time is denoted by random variable Z and is exponentially distributed with λ because of the memoryless property of the transmission delays D. We note

that, under this model, the source node always receives the most recent computation result available from the CU.

To distribute the computation task among the worker nodes upon receiving status update packets, the master node may adopt uncoded or coded distribution algorithms. In the uncoded scheme, the status update packet is divided into n equal subpackets, one for each worker node. However, in this method, the overall computation time is limited by the slowest worker node and thus, it may not be desirable especially when the computations are time-sensitive. To combat these slower straggling worker nodes, the master node can implement coding techniques to introduce redundancy to the computation task so that some straggling nodes can be tolerated. In our model, we analyze repetition and MDS (maximum distance separable) codes. Moreover, to further utilize the fastest worker nodes, we investigate the assignment of multiple MDS coded subpackets to each worker node, i.e., multi-message MDS (MM-MDS). We analyze the effects of these uncoded and coded schemes on the timeliness of the computations.

To quantify the timeliness we use the age of information metric. At time t age at the destination node, which is the source node in our model, is a random process $\Delta(t) = t - u(t)$ where $u(t)$ is the time-stamp of the most recent update at the destination node. The metric we use, long term average age, is

$$\Delta = \lim_{\tau \to \infty} \frac{1}{\tau} \int_0^{\tau} \Delta(t) dt, \tag{6.1}$$

where $\Delta(t)$ is the instantaneous age as defined above.

6.3 Age of Uncoded and Coded Task Distribution Algorithms

From the perspective of the CU, we have i.i.d. exponential interarrivals with λ. Since a dropping policy is implemented, not every arriving packet actually goes into service at the CU. We denote the packets that find the CU idle and thus go into service as the successful packets. Let T_{j-1} and T'_{j-1} denote the time at which the jth successful packet is generated at the source node and is received by the CU, respectively. Random variable Y denotes the update cycle at the CU, time in between two consecutive successful arrivals, and $Y_j = T'_j - T'_{j-1}$. As described in Section 6.2, update cycle Y_j consists of computation (service) time S_j and idle waiting time Z_j. We note that Z_j, S_j and D_j are mutually independent where D_j denotes the transmission delay experienced by the jth successful packet, i.e., $D_j = T'_{j-1} - T_{j-1}$. In our model, the interarrival process at the CU, D, and service times S are independent, and sequences $\{D_1, D_2, \dots\}$ and $\{S_1, S_2, \dots\}$ form i.i.d. processes.

We observe that Z is stochastically equal to the transmission delay D, i.e., interarrival time at the CU, due to the memoryless property of the transmission delay D. On the other hand, computation time S changes depending on the task distribution algorithm adopted by the master node. We use order statistics to express the distribution of S. We denote the kth smallest of $X_1, \dots, X_n$ as $X_{k:n}$. For a shifted exponential random variable X with (c, μ), we have [131]

$$\mathbb{E}[X_{k:n}] = c + \frac{1}{\lambda}(H_n - H_{n-k}), \tag{6.2}$$

$$\mathrm{Var}[X_{k:n}] = \frac{1}{\lambda^2}(G_n - G_{n-k}), \tag{6.3}$$

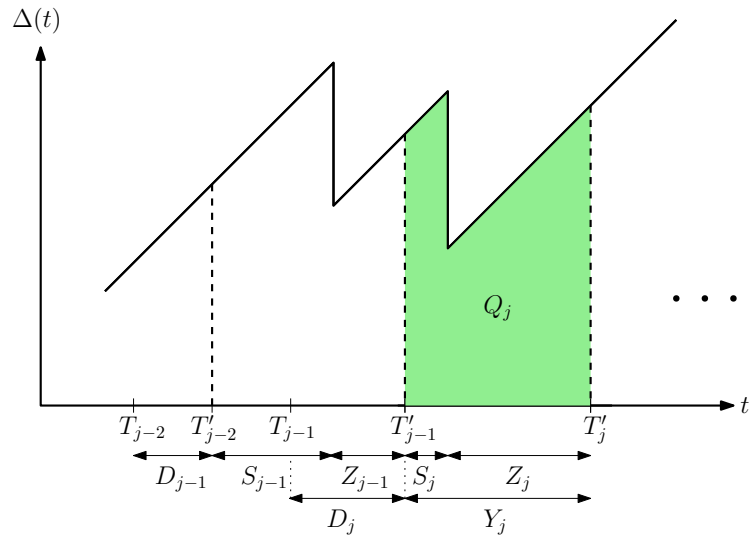


Figure 6.2: Sample age evolution $\Delta(t)$ at the destination (source) node. Successful updates are indexed by j. The jth successful update leaves the source node at T_{j-1} and arrives at the CU at T'_{j-1}. Update cycle at the CU is the time in between two successive arrivals and is equal to $Y_j = S_j + Z_j = T'_j - T'_{j-1}$.

where $H_n = \sum_{j=1}^{n} \frac{1}{j}$ and $G_n = \sum_{j=1}^{n} \frac{1}{j^2}$. Using (6.2) and (6.3),

$$\mathbb{E}[X_{k:n}^2] = \left(c + \frac{1}{\lambda}(H_n - H_{n-k}) \right)^2 + \frac{1}{\lambda^2} \left(G_n - G_{n-k} \right). \tag{6.4}$$

We note for future reference that when n is large and k is linear in n, i.e., $k = \alpha n$, for $0 < \alpha < 1$, the variance of the kth order statistic of the shifted exponential random variable, $X_{k:n}$, shown in (6.3), becomes negligibly small and tends to 0 as n increases because both G_n and G_{n-k} sequences converge to $\frac{\pi^2}{6}$. Thus, for large n, an ordered sequence of n shifted exponential random variables essentially becomes a deterministic sequence such that the kth realization takes the mean value given by (6.2). This observation will be useful in designing age optimal codes.

Our model here resembles the $M/G/1/1$ queue with blocking model analyzed

in [29] with one difference: here arriving packets at the CU have experienced a transmission delay. Therefore, they have aged by D by the time of their arrival at the CU. Noting this difference, we perform a similar graphical analysis using Fig. 6.2 to find the long term average age which is the average area under the age curve and is given by [29]

$$\Delta = \limsup_{n\to\infty} \frac{\frac{1}{n}\sum_{j=1}^{n} Q_j}{\frac{1}{n}\sum_{j=1}^{n} Y_j} = \frac{\mathbb{E}[Q]}{\mathbb{E}[Y]}, \tag{6.5}$$

where Q denotes the shaded area and Y is its length in Fig. 6.2, and the second equality follows from the ergodicity of the system. By using Fig. 6.2, we find $Y_j = S_j + Z_j$ and observe that the area Q_j can be decomposed into a right trapezoid with bases D_j and $D_j + Y_j$ and height Y_j; and a parallelogram with base $D_{j-1} + S_{j-1} + Z_{j-1} - D_j$ and height S_j. Thus, we find

$$\begin{aligned}
\mathbb{E}[Q_j] &= \mathbb{E}[D_j(S_j + Z_j)] + \frac{1}{2}\mathbb{E}[(S_j + Z_j)^2] + \mathbb{E}[S_j(D_{j-1} + S_{j-1} + Z_{j-1} - D_j)] && (6.6)\\
&= \mathbb{E}[D_j]\,\mathbb{E}[(S_j + Z_j)] + \frac{1}{2}\mathbb{E}[(S_j + Z_j)^2] \\
&\quad + \mathbb{E}[S_j]\,\mathbb{E}[D_{j-1} + S_{j-1} + Z_{j-1} - D_j] && (6.7)\\
&= \mathbb{E}[D_j]\,\mathbb{E}[(S_j + Z_j)] + \frac{1}{2}\mathbb{E}[(S_j + Z_j)^2] + \mathbb{E}[S_j]\,\mathbb{E}[S_{j-1} + Z_{j-1}], && (6.8)
\end{aligned}$$

where (6.7) follows by noting that the service process is i.i.d. and independent of the arrival process along with the fact that D_j, S_j and Z_j are mutually independent. Further, (6.8) follows by noting that $\mathbb{E}[D_{j-1}] = \mathbb{E}[D_j]$ since the system is stationary and ergodic. Since we focus on the long term behavior, we drop the subscript index

and find

$$\mathbb{E}[Q] = \mathbb{E}[D](\mathbb{E}[S] + \mathbb{E}[Z]) + \frac{1}{2}\,\mathbb{E}[(S+Z)^2] + \mathbb{E}[S](\mathbb{E}[S] + \mathbb{E}[Z]). \tag{6.9}$$

Similarly, we have

$$\mathbb{E}[Y] = \mathbb{E}[S] + \mathbb{E}[Z]. \tag{6.10}$$

Then, using (6.9) and (6.10) in (6.5), we find the long term average age as

$$\Delta = \frac{\mathbb{E}[Q]}{\mathbb{E}[Y]} = \mathbb{E}[D] + \mathbb{E}[S] + \frac{\mathbb{E}[Y^2]}{2\,\mathbb{E}[Y]}, \tag{6.11}$$

where $Y = S + Z$ as noted above. The first term in (6.11) reflects that arriving packets at the CU have aged on average by $\mathbb{E}[D]$.[1] Our goal is to minimize the average age given in (6.11) by adjusting computation (service) time S at the CU through different task distribution algorithms.

6.3.1 Uncoded Scheme

In the uncoded scheme, the master node divides the received status update packet into n subpackets, one for each worker node. Using the mother runtime distribution detailed in Section 6.2, we see that in the uncoded scheme local computation time at each worker $\tilde{X}$ follows a shifted exponential distribution with parameters $\left(\frac{c}{n}, n\mu\right)$.

[1] We note that, in a possibly more intuitive setting, if the source node sends the next update upon receiving the previous computed update back from the CU, unlike the current model in which the source node sends the next update as soon as the current one reaches the CU, the average age expression in (6.11) would stay the same as D and Z are stochastically identical.

This sped-up distribution highlights the fact that each worker node performs a part of the overall computation. Thus, in order to reveal the information in the received update packet, the master node needs to collect the results from all n worker nodes. Hence, the computation time is $S = \tilde{X}_{n:n}$. Calculating the moments of S using (6.2)-(6.4) and substituting them into (6.11) we find the average age when the uncoded scheme is utilized, Δ_{unc}, as

$$\Delta_{unc} = \frac{1}{\lambda} + \frac{c}{n} + \frac{H_n}{n\mu} + \frac{\left(\frac{c}{n} + \frac{H_n}{n\mu}\right)^2 + \frac{G_n}{n^2\mu^2} + \frac{2}{\lambda}\left(\frac{c}{n} + \frac{H_n}{n\mu}\right) + \frac{2}{\lambda^2}}{2\left(\frac{c}{n} + \frac{H_n}{n\mu} + \frac{1}{\lambda}\right)}. \tag{6.12}$$

The following theorem states the asymptotic average age performance of the uncoded scheme as the number of worker nodes n increases.

Theorem 6.1 *With i.i.d. exponential transmission delays and i.i.d. shifted exponential computation times at each worker, the average age of the uncoded distribution scheme for large n is $\frac{2}{\lambda} + O\left(\frac{\log n}{n}\right)$.*

The proof of Theorem 6.1 follows from the fact that for large n, we have $H_n \approx \log n$ and $G_n \approx \frac{\pi^2}{6}$. The constant $\frac{2}{\lambda}$ in the result reflects the sum of $\mathbb{E}[D] = \frac{1}{\lambda}$, which is the expected delay packets experience on the way from the source to the CU, and $\mathbb{E}[Z] = \frac{1}{\lambda}$, which is the expected waiting time for a new packet at the CU when it is idle. The $O\left(\frac{\log n}{n}\right)$ term in the result shows that the average age decreases with n, the number of worker nodes at the CU. Here, the $\frac{1}{n}$ term in $O\left(\frac{\log n}{n}\right)$ reflects the fact that computation tasks get smaller and consequently workers become faster as n increases, whereas the $\log n$ term in $O\left(\frac{\log n}{n}\right)$ indicates that the uncoded scheme

is limited by the performance of the slowest worker. Since the increase in the computation rates, i.e., decrease in the average straggling behavior, overcomes the increase in the computation time of the slowest worker, average age of the uncoded scheme decreases as n increases.

The main downside of the uncoded scheme is the fact that it is prone to large delays due to straggling nodes as the master node needs all of the computation results to extract the useful information from the status update packet. Therefore, if some servers are much slower than the rest, service time of the update packet increases significantly. To cope with these straggling worker nodes, redundant computation tasks may be created via coding. In what follows we analyze the effects of repetition coded, MDS coded and MM-MDS coded schemes on the average age.

6.3.2 Repetition Coded Scheme

We consider an $\frac{n}{k}$-repetition code where the packet is divided into k equal sized subpackets where $k \leq n$ and each subpacket is repeated $\frac{n}{k}$ times. In other words, each subpacket has $\frac{n}{k}$ replicas and the master node needs to collect k distinct results from n worker nodes. Thus, each worker node has a shifted exponential computation time distribution, $\tilde{X}$, with parameters $\left(\frac{c}{k}, k\mu\right)$. Since there are $\frac{n}{k}$ workers for each of the k subpackets the computation time of each subpacket is the minimum among these $\frac{n}{k}$ i.i.d. random variables which is denoted by $\bar{X} = \tilde{X}_{1:\frac{n}{k}}$. Since the minimum of shifted exponentials is also a shifted exponential with the same shift but scaled straggling parameter, $\bar{X}$ follows a shifted exponential distribution with $\left(\frac{c}{k}, n\mu\right)$.

Since we need k distinct results, the overall computation time in this case is $S = \bar{X}_{k:k}$. Using (6.11) along with the moments of order statistics given in (6.2)-(6.4), we find the average age of the repetition coded scheme, Δ_{rep}, as

$$\Delta_{rep} = \frac{1}{\lambda} + \frac{c}{k} + \frac{H_k}{n\mu} + \frac{\left(\frac{c}{k} + \frac{H_k}{n\mu}\right)^2 + \frac{G_k}{n^2\mu^2} + \frac{2}{\lambda}\left(\frac{c}{k} + \frac{H_k}{n\mu}\right) + \frac{2}{\lambda^2}}{2\left(\frac{c}{k} + \frac{H_k}{n\mu} + \frac{1}{\lambda}\right)}. \tag{6.13}$$

The following theorem states the asymptotic average age performance of the repetition coded scheme as n increases.

Theorem 6.2 *With i.i.d. exponential transmission delays and i.i.d. shifted exponential computation times at each worker, the average age of the $\frac{n}{k}$-repetition coded scheme for large n with $k = \alpha n$ where $0 < \alpha \leq 1$ is $\frac{2}{\lambda} + O\left(\frac{\log n}{n}\right)$.*

The proof of Theorem 6.2 follows similarly from that of Theorem 6.1. Here, we observe that although a coding scheme is implemented, asymptotically, we achieve the same average age performance as the uncoded scheme. Thus, repetition coded scheme is asymptotically no better than the uncoded scheme. This is because of the fact that the repetition coded scheme still suffers from the $\log n$ term in $O\left(\frac{\log n}{n}\right)$ as the runtime of this scheme is limited by the slowest of the k distinct subtask computations needed, where k is linear in n. Thus, the resulting average age scales the same as the uncoded scheme even though replication brings the $\frac{1}{n}$ term.

Next, we analyze the performance of the MDS coded schemes.

6.3.3 MDS Coded Scheme

To implement an (n,k)-MDS code where $k<n$, the update packet is first divided into k equal sized subpackets. From these k subpackets a total of n subpackets are created with coding by using $n-k$ redundant subpackets so that the master node can decode the result of the computation as soon as it receives k computation results. Since the overall computation task is divided into k subtasks as in repetition coding, each worker node completes its local task in $\tilde{X}$ which is a shifted exponential with $(\frac{c}{k}, k\mu)$. The computation time for the overall task is, however, $S=\tilde{X}_{k:n}$. Using this along with (6.2)-(6.4) in (6.11), we find the average age when an (n,k)-MDS code is implemented, Δ_{mds}, as

$$\begin{aligned}\Delta_{mds} =&\frac{1}{\lambda}+\frac{c}{k}+\frac{H_n-H_{n-k}}{k\mu}+\frac{\left(\frac{c}{k}+\frac{H_n-H_{n-k}}{k\mu}\right)^2+\frac{G_n-G_{n-k}}{k^2\mu^2}}{2\left(\frac{c}{k}+\frac{H_n-H_{n-k}}{k\mu}+\frac{1}{\lambda}\right)}\\&+\frac{\frac{2}{\lambda}\left(\frac{c}{k}+\frac{H_n-H_{n-k}}{k\mu}\right)+\frac{2}{\lambda^2}}{2\left(\frac{c}{k}+\frac{H_n-H_{n-k}}{k\mu}+\frac{1}{\lambda}\right)}.\end{aligned} \tag{6.14}$$

The following theorem gives the asymptotic average age performance of the MDS coded scheme for large n.

Theorem 6.3 *With i.i.d. exponential transmission delays and i.i.d. shifted exponential computation times at each worker, the average age of the (n,k)-MDS coded scheme for large n with $k=\alpha n$ where $0<\alpha<1$ is $\frac{2}{\lambda}+O\left(\frac{1}{n}\right)$.*

The proof of Theorem 6.3 follows similarly to that of Theorem 6.1 by noting that $H_{n-k}\approx\log(n-k)$. Thus, when $k=\alpha n$, we get $H_n-H_{n-k}=-\log(1-\alpha)$

which is independent of n. Further, $G_n - G_{n-k} = G_n - G_{(1-\alpha)n}$ tends to 0 as n increases. With these, the result follows.

We observe that the average age in Theorem 6.3 has a $O\left(\frac{1}{n}\right)$ term as opposed to $O\left(\frac{\log n}{n}\right)$ terms in Theorems 6.1 and 6.2. Thus, for large n, MDS coded scheme outperforms repetition coded and uncoded schemes in terms of average age performance. Up to now, we have investigated uncoded and coded schemes in which each worker node is assigned one subtask to compute. Although we achieve better performance in combating the straggling nodes through coding, there is still room for improvement. In all these schemes, the fastest worker nodes which finish their computations earlier stay idle. To utilize them even more, we can assign multiple subtasks to each worker node. With multiple assignments to each worker node we can utilize partial straggling worker nodes, which are the ones that cannot finish all tasks that are assigned to them but still return some partial results. In the next subsection, we consider the performance of MDS coded scheme when each worker is given multiple subtasks to compute.

6.3.4 Multi-message MDS (MM-MDS) Coded Scheme

In multi-message MDS coded scheme, each worker node is assigned ℓ subpackets to compute, where ℓ denotes the load of each worker node. That is, each worker node has a job queue of size ℓ in each update cycle. Thus, we implement an $(n\ell, k)$-MDS code. For this, the overall update packet is divided into k subtasks where $k < n\ell$ and from these subtasks $n\ell - k$ redundant subtasks are generated such

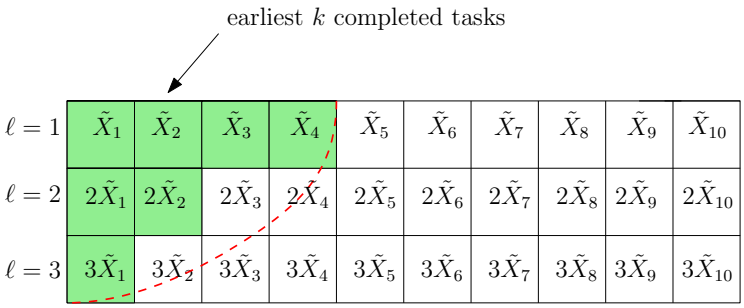


Figure 6.3: The earliest k computed tasks for $n = 10$, $\ell = 3$, and $k = 7$.

that the master node only needs to receive k computation results to extract the embedded information in the status update received from the source node. Unlike regular MDS coded scheme in which each worker has one subtask to compute, in this scheme faster workers can perform multiple computations to aid the overall computation time. Hence, we utilize partial stragglers, also called non-persistent stragglers [165], i.e., worker nodes that finish some portion of the subtasks that are assigned to them.

In line with the mother computation distribution model presented in Section 6.2, computation time of a subtask at each worker, $\tilde{X}$, has a shifted exponential distribution with $\left(\frac{c}{k}, k\mu\right)$. Following the model in [165], we assume that the duration of each computation performed by a worker during an update cycle is identical. In other words, if a worker finishes m of the ℓ subtasks during an update cycle, duration of each computation is identical which is sampled from a shifted exponential with parameters $\left(\frac{c}{k}, k\mu\right)$. Therefore, the time it takes for a worker node to perform m computations, $m\tilde{X}$, is also a shifted exponential with $(\frac{mc}{k}, \frac{k\mu}{m})$. It remains to determine $\mathbb{E}[S]$ and $\mathbb{E}[S^2]$ to calculate the average age in this setting by using (6.11).

In what follows, the mth level refers to the set of subtasks that are located in the mth position in each worker's job queue. In other words, the mth level includes a total of n subtasks which are performed in the mth position by the corresponding worker nodes upon completion of their first $m-1$ subtasks. We note that in the uncoded, repetition coded, and MDS coded schemes there is only one level as $\ell = 1$ in those schemes. Let k_m denote the number of subtasks computed in the mth level upon completion of the overall task at the CU during an update cycle. We have $\sum_{m=1}^{\ell} k_m = k$ since exactly k subtasks need to be performed to finish the overall computation. Fig. 6.3 shows an example for $n = 10$, $k = 7$, and $\ell = 3$. Here, each column represents the computation times of ℓ subtasks that a worker node is assigned and row m represents the computation times of the mth level subtasks. Without loss of generality, we order level one, i.e., $\tilde{X}_1$ in Fig. 6.3 is the smallest computation time of a level 1 subtask and $\tilde{X}_{10}$ is the largest one. Correspondingly, all other levels are ordered as well. Hence, column i in Fig. 6.3 in fact shows the computation times of the ith fastest worker node, where $i = 1, \ldots, n$. In this example, we observe that by the time the earliest $k = 7$ computations are finished, the fastest worker completed three subtasks, the second fastest worker completed two subtasks, the third and fourth fastest workers completed one subtask each, and the remaining six workers completed zero subtasks. That is, 4 first level, 2 second level and 1 third level subtasks are computed. Thus, we have $k_1 = 4$, $k_2 = 2$ and $k_3 = 1$.

For simplicity and ease of exposition, consider the case where $\ell = 2$. Assume that by the time a total of k computations are performed $k_1 < k$ computations from the first level are finished. Then, we have $k_2 = k - k_1$. When n is large,

the time it takes to compute k_1 first level subtasks, $\tilde{X}_{k_1:n}$ is equal to $\mathbb{E}[\tilde{X}_{k_1:n}]$ due to the vanishing variance property of order statistics of shifted exponentials for large n, discussed after (6.4). Similarly, k_2 second level subtasks are performed in $\mathbb{E}[2\tilde{X}_{k_2:n}]$ units of time. At this point, there can be two possible scenarios: In the first scenario, k_2 second level computations are finished before the k_1th first level subtask is finished, i.e.,

$$\mathbb{E}[2\tilde{X}_{k_2:n}] \leq \mathbb{E}[\tilde{X}_{k_1:n}] < \mathbb{E}[2\tilde{X}_{k_2+1:n}]. \tag{6.15}$$

We note that second inequality in (6.15) holds since the k_1th first level task needs to be performed before the $k_2 + 1$th second level task to be included in the earliest k subtasks. In this case, the computation duration, i.e., service time, is $S = \tilde{X}_{k_1:n}$. On the other hand, in the second scenario, the k_2th second level subtask is finished after the k_1st first level subtask but before the subtask $k_1 + 1$ of the first level, i.e.,

$$\mathbb{E}[\tilde{X}_{k_1:n}] \leq \mathbb{E}[2\tilde{X}_{k_2:n}] < \mathbb{E}[\tilde{X}_{k_1+1:n}]. \tag{6.16}$$

In this case, we have $S = 2\tilde{X}_{k_2:n}$. When k_1 and k_2 are linear in n, i.e., $k_1 = \alpha_1 n$ and $k_2 = \alpha_2 n$, with $0 < \alpha_1 < 1$ and $0 < \alpha_2 < 1$, for large n, we have

$$\alpha_1 = \frac{k_1}{n} \approx \frac{k_1 + 1}{n}, \tag{6.17}$$

$$\alpha_2 = \frac{k_2}{n} \approx \frac{k_2 + 1}{n}, \tag{6.18}$$

which imply that the bounds in (6.15) and (6.16) meet as n gets large. Thus, by the sandwich theorem $\mathbb{E}[\tilde{X}_{k_1:n}] = \mathbb{E}[2\tilde{X}_{k_2:n}]$ is satisfied for large n in either scenario. By using (6.2), for large n when both k_1 and k_2 are linear in n as defined above we have

$$\mathbb{E}[\tilde{X}_{k_1:n}] = \frac{c}{k} + \frac{H_n - H_{n-k_1}}{k\mu} = \frac{c}{2\alpha n} + \frac{1}{2\alpha n\mu} \log\left(\frac{1}{1-\alpha_1}\right), \tag{6.19}$$

and

$$\mathbb{E}[2\tilde{X}_{k_2:n}] = 2\left(\frac{c}{k} + \frac{H_n - H_{n-k_2}}{k\mu}\right) = \frac{2c}{2\alpha n} + \frac{1}{2\alpha n\mu} \log\left(\frac{1}{1-\alpha_2}\right)^2. \tag{6.20}$$

Here, $k = \alpha 2n$ with $0 < \alpha < 1$ when both k_1 and k_2 are linear in n as defined above. We note that $k_1 + k_2 = k$ is equivalent to $\alpha_1 + \alpha_2 = 2\alpha$. Equating (6.19) and (6.20) yields

$$\frac{1}{1-\alpha_1} = e^{\mu c} \frac{1}{(1-\alpha_2)^2}, \tag{6.21}$$

where $\alpha_1 + \alpha_2 = 2\alpha$. We note that (6.21) holds when MM-MDS coded scheme is implemented for $\ell = 2$ if $\alpha_2 > 0$, i.e., $k_2 > 0$. When $\alpha_2 = 0$, we have $\alpha_1 = 2\alpha$ directly. That is, in that case we have $k_1 = k$.

A similar relationship between α_ms, equivalently between k_ms, holds for the general case with $\ell > 2$ as well. When we have ℓ levels, we have at most $\binom{\ell}{2}$ inequalities like (6.15) and (6.16) to represent the ordering between the last subtasks of each level that are included in the earliest k. For example, when $\ell = 2$ we have one

inequality, either (6.15) or (6.16), to represent the relationship between the k_1th first level and the k_2th second level subtasks provided that k_2 is nonzero. When $\ell > 2$, if $k_m > 0$ subtasks are finished in level m by the time it takes to receive a total of k computations, completion time of the k_mth subtask of the mth level satisfies either one of the following inequalities for any other level $\bar{m}$ for which $k_{\bar{m}} > 0$:

$$\mathbb{E}[m\tilde{X}_{k_m:n}] \leq \mathbb{E}[\bar{m}\tilde{X}_{k_{\bar{m}}:n}] < \mathbb{E}[m\tilde{X}_{k_m+1:n}] \tag{6.22}$$

which implies that the k_mth subtask of the mth level is finished earlier than the $k_{\bar{m}}$th subtask of level $\bar{m}$ or

$$\mathbb{E}[\bar{m}\tilde{X}_{k_{\bar{m}}:n}] \leq \mathbb{E}[m\tilde{X}_{k_m:n}] < \mathbb{E}[\bar{m}\tilde{X}_{k_{\bar{m}}+1:n}], \tag{6.23}$$

which implies that the k_mth subtask of the mth level is finished after the $k_{\bar{m}}$th subtask of level $\bar{m}$ is finished. Upon writing this relationship between every $(m, \bar{m})$ pair, we take $k_m = \alpha_m n$ with $0 < \alpha_m < 1$ and proceed similarly to get

$$\frac{1}{(1-\alpha_{m-1})^{m-1}} = e^{\mu c}\frac{1}{(1-\alpha_m)^m}, \tag{6.24}$$

with $\sum_{m=1}^{\ell} \alpha_m = \ell\alpha$ similar to (6.21). We note that if after some level $m > m_0$, none of the level m subtasks are finished, then, $\alpha_m = 0$ for all $m > m_0$ and (6.24) holds for all nonzero α_ms. With these, by using (6.24) and the fact that $\sum_{m=1}^{\ell} \alpha_m = \ell\alpha$, each remaining nonzero α_m and correspondingly each remaining k_m can be determined. As a direct consequence of (6.24), we see that the time it takes to receive the earliest

k computation results is equivalent to the time it takes to receive k_m from level m such that $k_m = \alpha_m n$ and α_ms satisfy (6.24) and $\sum_{m=1}^{\ell} \alpha_m = \ell\alpha$. With such k_ms, we then have $S = \tilde{X}_{k_1:n}$. Hence, the average age when the MM-MDS coded scheme is implemented with ℓ subpackets at each node, Δ_{mm-mds}, can now be computed using (6.11) as follows

$$\begin{aligned}\Delta_{mm-mds} =& \frac{1}{\lambda} + \frac{c}{k} + \frac{H_n - H_{n-k_1}}{k\mu} + \frac{\left(\frac{c}{k} + \frac{H_n - H_{n-k_1}}{k\mu}\right)^2 + \frac{G_n - G_{n-k_1}}{k^2\mu^2}}{2\left(\frac{c}{k} + \frac{H_n - H_{n-k_1}}{k\mu} + \frac{1}{\lambda}\right)} \\ &+ \frac{\frac{2}{\lambda}\left(\frac{c}{k} + \frac{H_n - H_{n-k_1}}{k\mu}\right) + \frac{2}{\lambda^2}}{2\left(\frac{c}{k} + \frac{H_n - H_{n-k_1}}{k\mu} + \frac{1}{\lambda}\right)}. \end{aligned} \tag{6.25}$$

where $k_m = \alpha_m n$ and α_ms satisfy (6.24) and $\sum_{m=1}^{\ell} \alpha_m = \ell\alpha$.

The following theorem gives the asymptotic average age performance of the MM-MDS coded scheme for large n.

Theorem 6.4 *With i.i.d exponential transmission delays and i.i.d. shifted exponential computation times at each worker, the average age of the MM-MDS coded scheme with load ℓ, for large n with $k_m = \alpha_m n$ where $0 < \alpha_m < 1$, $m = 1, \ldots, \ell$, is $\frac{2}{\lambda} + O\left(\frac{1}{\ell n}\right)$.*

To prove Theorem 6.4 we first note that when $k_m = \alpha_m n$ for each level m, we have $k = \alpha n \ell$ where $0 < \alpha < 1$ such that $\sum_{m=1}^{\ell} \alpha_m = \alpha\ell$. With this, the proof follows similarly from that of Theorem 6.3. We note that compared to the MDS coded scheme where we have $O\left(\frac{1}{n}\right)$, here in the MM-MDS coded scheme, we have $O\left(\frac{1}{\ell n}\right)$ which reflects ℓ, the number of subtasks assigned to each worker node. Thus,

for large n, the best asymptotic performance is achieved when MM-MDS coded scheme is implemented.

The performance of repetition coded, MDS coded and MM-MDS coded schemes can be optimized through the selection of k, which will be the focus of Section 6.4.

6.4 Optimizing Age by Parameter Selection

In Section 6.3, we showed that the age in uncoded, repetition coded, MDS coded, and MM-MDS coded schemes depend on n as $O\left(\frac{\log n}{n}\right)$, $O\left(\frac{\log n}{n}\right)$, $O\left(\frac{1}{n}\right)$, and $O\left(\frac{1}{\ell n}\right)$, respectively, excluding the common constant term $\frac{2}{\lambda}$. In repetition, MDS and MM-MDS coded schemes, we have a parameter k that depends on n linearly as $k = \alpha n \ell$, where $\ell = 1$ for repetition and MDS coded schemes, and $\ell > 1$ for MM-MDS coded scheme. In this section, we consider the optimization of this parameter k, which is equivalent to the optimization of the parameter α. This is similar in spirit to the optimization of k, correspondingly the optimization of α, in Chapters 2 and 3 and references [113, 159]. Towards that goal, in order to unify our approach for all three coded schemes (repetition, MDS, and MM-MDS), we first provide the following theorem. This theorem shows that, in our model, age minimization translates into computation (service) time minimization which is not always the case in age optimization problems.

Theorem 6.5 *When the transmission delays are i.i.d. exponentials and computation times at each worker are i.i.d. shifted exponentials under the dropping policy at the CU, for large n, minimization of the average age of repetition coded, MDS*

coded and MM-MDS coded schemes, is equivalent to minimization of the average computation time.

Proof: In the repetition, MDS and MM-MDS coded schemes, the computation time S is characterized through the selection of k. Using the average age expression in (6.11), the minimization problem is

$$\begin{aligned}
&\min_k \mathbb{E}[D] + \mathbb{E}[S] + \frac{\mathbb{E}[(S+Z)^2]}{2\,\mathbb{E}[S+Z]} \\
&= \min_k \mathbb{E}[D] + \mathbb{E}[S] + \frac{\mathbb{E}[S^2] + 2\,\mathbb{E}[S]\,\mathbb{E}[Z] + \mathbb{E}[Z^2]}{2\,\mathbb{E}[S+Z]} && (6.26)\\
&\approx \min_k \mathbb{E}[D] + \mathbb{E}[S] + \frac{E^2[S] + 2\,\mathbb{E}[S]\,\mathbb{E}[Z] + \mathbb{E}[Z^2]}{2(\mathbb{E}[S] + \mathbb{E}[Z])} && (6.27)\\
&= \min_k \frac{1}{\lambda} + \mathbb{E}[S] + \frac{E^2[S] + 2\,\mathbb{E}[S]\frac{1}{\lambda} + \frac{2}{\lambda^2}}{2(\mathbb{E}[S] + \frac{1}{\lambda})} && (6.28)\\
&= \min_k \frac{1}{\lambda} + \mathbb{E}[S] + \frac{(\mathbb{E}[S] + \frac{1}{\lambda})^2 + \frac{1}{\lambda^2}}{2(\mathbb{E}[S] + \frac{1}{\lambda})} && (6.29)\\
&= \min_k \frac{3}{2\lambda} + \frac{3}{2}\,\mathbb{E}[S] + \frac{\frac{1}{\lambda^2}}{2\,\mathbb{E}[S] + \frac{2}{\lambda}}, && (6.30)
\end{aligned}$$

where (6.26) follows from the independence of S and Z, and (6.27) follows from the fact that $\mathbb{E}[S^2] \approx E^2[S]$ in all of these coding schemes due to the vanishing variance property of order statistics of shifted exponentials for large n, discussed after (6.4). The term ignored in (6.27) is $\frac{1}{2\nu^2}(G_n - G_{n-k})/(\mathbb{E}[S] + \mathbb{E}[Z])$, where ν denotes the computation rate and varies depending on the task distribution algorithm. The numerator of the vanishing term can be lower and upper bounded by 0 and $\frac{1}{\nu^2}\frac{\pi^2}{6}$, respectively, as G_n is upper bounded by $\frac{\pi^2}{6}$. We note that as $n \to \infty$, $\frac{1}{\nu^2} \to 0$ as detailed in Section 6.3 for repetition coded, MDS coded and MM-MDS coded

schemes. Thus, bounds meet for large n which yields the approximation $\mathbb{E}[S^2] \approx E^2[S]$.

In order to optimize the average age, we need to select the optimal k in the repetition coded, MDS coded and MM-MDS coded schemes in (6.30). We note that in (6.30) only $\mathbb{E}[S]$ depends on k. Although the second term in (6.30) increases in $\mathbb{E}[S]$ and the third term decreases in $\mathbb{E}[S]$, overall (6.30) is monotonically increasing in $\mathbb{E}[S]$, as the derivative of (6.30) with respect to $\mathbb{E}[S]$ is non-negative. Thus, the average age is minimized when $\mathbb{E}[S]$ is minimized. That is, the average age minimization is equivalent to the average computation time minimization. ■

For large n, average computation time is given, for the repetition coded scheme, by

$$\mathbb{E}[S_{rep}] = \frac{c}{k} + \frac{H_k}{n\mu} = \frac{c}{k} + \frac{1}{\mu n}\log k = \frac{c}{\alpha n} + \frac{1}{\mu n}\log(\alpha n), \tag{6.31}$$

for the MDS coded scheme by

$$\begin{aligned}\mathbb{E}[S_{mds}] &= \frac{c}{k} + \frac{H_n - H_{n-k}}{k\mu} = \frac{c}{k} + \frac{1}{\mu k}\log\left(\frac{n}{n-k}\right) \\ &= \frac{c}{\alpha n} + \frac{1}{\mu\alpha n}\log\left(\frac{1}{1-\alpha}\right),\end{aligned} \tag{6.32}$$

and for the MM-MDS coded scheme by

$$\begin{aligned}\mathbb{E}[S_{mm-mds}] &= \frac{c}{k} + \frac{H_n - H_{n-k_1}}{k\mu} = \frac{c}{k} + \frac{1}{\mu k}\log\left(\frac{n}{n-k_1}\right) \\ &= \frac{c}{\alpha n\ell} + \frac{1}{\mu\alpha n\ell}\log\left(\frac{1}{1-\alpha_1}\right),\end{aligned} \tag{6.33}$$

where in (6.33) k_ms satisfy (6.24) and $\sum_{m=1}^{\ell} \alpha_m = \ell\alpha$.

Reference [159] finds the optimal k for repetition coded and MDS coded schemes when k is linear in n by noting that both (6.31) and (6.32) have unique extreme points as functions of k. In [159, Lemma 1] the following optimization problem is solved to find the optimal computation time in the repetition coded scheme:

$$\min_{k} \mathbb{E}[S_{rep}] = \min_{1\leq k\leq n} \frac{c}{k} + \frac{1}{\mu n}\log k = \min_{0<\alpha\leq 1} \frac{c}{\alpha} + \frac{1}{\mu}\log \alpha. \tag{6.34}$$

Objective in (6.34) has a term that increases in α and another term that decreases in α. Depending on μ and c values, there is a unique α^* which is the extremum point

$$\alpha^* = \begin{cases} 1, & c\mu \geq 1 \\ c\mu, & c\mu < 1, \end{cases} \tag{6.35}$$

and correspondingly,

$$k^* = \begin{cases} n, & c\mu \geq 1 \\ c\mu n, & c\mu < 1, \end{cases} \tag{6.36}$$

for large n. Solutions in (6.35) and (6.36) are computation time optimum, and also average age optimum from Theorem 6.5 for $\frac{n}{k}$-repetition coded scheme. Note that, for $c\mu \geq 1$, the optimal repetition coded scheme is in fact the uncoded scheme. However, when $c\mu < 1$, repetition coded scheme outperforms the uncoded scheme.

Similarly, to find the optimal computation time in the (n, k)-MDS coded

scheme, the following optimization problem is solved in [159, Lemma 2]:

$$\min_k \mathbb{E}[S_{mds}] = \min_{1\le k<n} \frac{c}{k} + \frac{1}{\mu k}\log\left(\frac{n}{n-k}\right) = \min_{0<\alpha<1} \frac{c}{\alpha} + \frac{1}{\mu\alpha}\log\left(\frac{1}{1-\alpha}\right), \quad (6.37)$$

and it is shown that the optimum α is

$$\alpha^* = 1 + \frac{1}{W_{-1}(-e^{-\mu c-1})}, \quad (6.38)$$

and correspondingly,

$$k^* = \left[1 + \frac{1}{W_{-1}(-e^{-\mu c-1})}\right] n, \quad (6.39)$$

for large n, which is also average age optimal from Theorem 6.5. Here, $W_{-1}(\cdot)$ is the lower branch of Lambert W function.

In the MM-MDS coded scheme, as in the repetition coded and MDS coded schemes, by selecting the optimal k, correspondingly the optimal α, the computation time and by Theorem 6.5, the age can be minimized. To do that, we need to solve the following optimization problem:

$$\begin{aligned} \min_{0<\alpha,\alpha_1,\cdots,\alpha_\ell<1} \quad & \frac{c}{\alpha} + \frac{1}{\mu\alpha}\log\left(\frac{1}{1-\alpha_1}\right) \\ \text{s.t.} \quad & \frac{1}{(1-\alpha_{m-1})^{m-1}} = \frac{e^{\mu c}}{(1-\alpha_m)^m},\ m = 2,\dots,\ell \\ & \sum_{m=1}^{\ell} \alpha_m = \ell\alpha. \end{aligned} \quad (6.40)$$

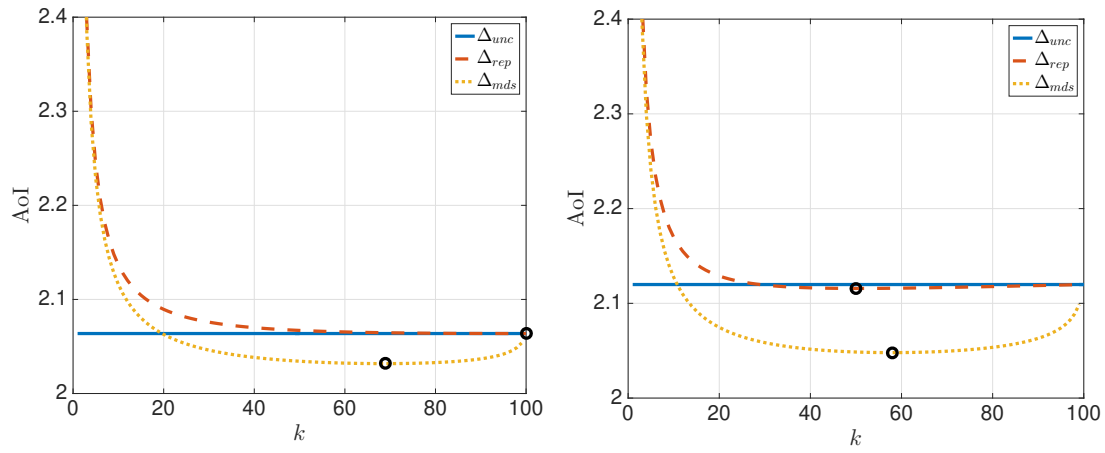
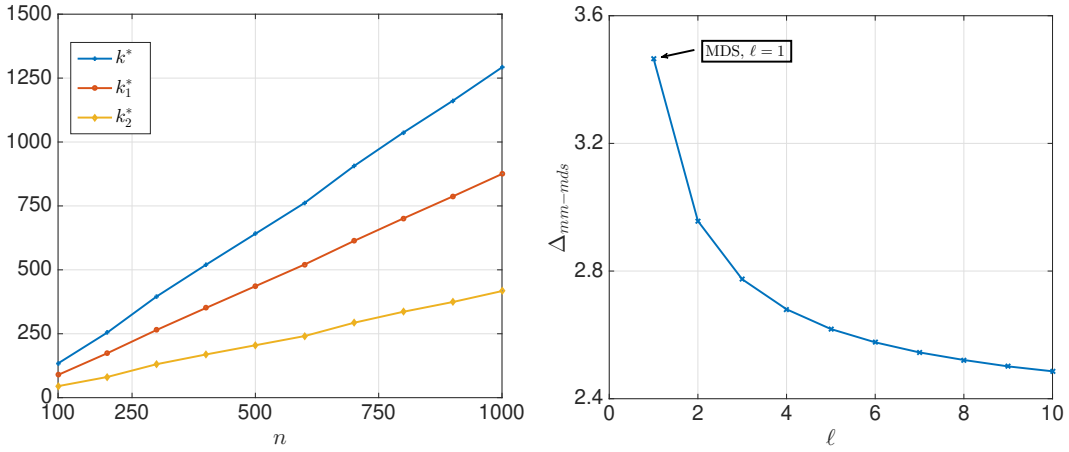


Figure 6.4: Δ_{unc}, Δ_{rep} and Δ_{mds} for varying k with $n = 100$, and $(\lambda, c) = 1$: (a) When $\mu = 1$, (b) when $\mu = 0.5$. Symbol $\circ$ marks the optimal k values.

To give an explicit example, for instance, for $\ell = 3$, the master node solves the following optimization problem:

$$\begin{aligned} \min_{0<\alpha,\alpha_1,\alpha_2,\alpha_3<1} \quad & \frac{c}{\alpha} + \frac{1}{\mu\alpha} \log\left(\frac{1}{1-\alpha_1}\right) \\ \text{s.t.} \quad & \frac{1}{1-\alpha_1} = e^{\mu c} \frac{1}{(1-\alpha_2)^2} \\ & \frac{1}{(1-\alpha_2)^2} = e^{\mu c} \frac{1}{(1-\alpha_3)^3} \\ & \alpha_1 + \alpha_2 + \alpha_3 = 3\alpha. \end{aligned} \tag{6.41}$$

We note that unlike the repetition coded and MDS coded schemes, in the MM-MDS coded scheme, the optimization problem (6.40) is more complicated. The optimization in (6.40) is over α and all α_ms. Here, a closed-form expression for k^*, or equivalently α^*, is not available unlike the former two cases. We solve the problem in (6.40) in the next section using numerical methods.



(a) (b)

Figure 6.5: (a) k^*, k_1^*, and k_2^* values as a function of n for $\ell = 2$ with $\mu = 1$, and $(\lambda, c) = 1$. Note that $k^* = k_1^* + k_2^*$. (b) Δ_{mm-mds}, as a function of load ℓ for $(n\ell, k^*)$-MDS code with $n = 100$, $\mu = 0.01$, and $(\lambda, c) = 1$.

6.5 Numerical Results

In this section, we provide simple numerical results.

First, we consider the uncoded, repetition coded and MDS coded schemes. In Figs. 6.4(a) and 6.4(b) age performance of the uncoded, repetition coded and MDS coded schemes are presented when $n = 100$, $\lambda = 1$, and $c = 1$ for $\mu = 1$ and $\mu = 0.5$, respectively, for varying k. We observe that in both Figs. 6.4(a) and 6.4(b), MDS coded scheme performs the best as expected. Optimal k values for the MDS coded scheme in these cases are $k^* = 69$ and $k^* = 58$, respectively. Moreover, we observe that when $\mu = 1$ optimal k for repetition coded scheme is in fact $k^* = n = 100$. However, when $\mu = 0.5$, we get $k^* = 0.5 * 100 = 50$. These results are in line with (6.36). We also observe in Fig. 6.4(b) that repetition coded scheme beats the uncoded scheme when $c\mu < 1$. However, as seen in Fig. 6.4(a) when $c\mu \geq 1$,

repetition coded scheme does not present any advantage over the uncoded scheme.

Next, we consider the MM-MDS coded scheme. Fig. 6.5(a) shows the optimal k values as a function of n for $\ell = 2$. We observe that as the number of worker nodes, n, increases optimal k increases as well. Fig. 6.5(b) shows the improvement in the average age performance of MDS coded scheme when worker nodes are assigned ℓ subpackets to compute with $n = 100$, $\mu = 0.01$, $c = 1$ and $\lambda = 1$. We note that when $\ell = 1$ we recover the performance of the single message MDS coded scheme analyzed in Section 6.3.3 and we observe that when multiple subpackets are assigned to each worker, we achieve a lower age than all the other schemes discussed.

6.6 Conclusion

In contrast to the initial works on age of information which assumed small sized status update packets, we have considered a status update system encountered in emerging data-intensive applications such as UAV and V2V systems, in which the updates are more complex and require processing to extract the useful information. This task is handled by a computation unit consisting of a master node and n worker nodes. We have investigated the age performance of uncoded and coded computation distribution algorithms and showed that the MDS coded task distribution scheme asymptotically outperforms the uncoded and repetition coded schemes. In addition, we observed that assigning multiple computations to each worker node (MM-MDS coded scheme) further improves the age performance of MDS coded scheme. By showing that under certain arrival and service (computation) time profiles minimiz-

ing age is equivalent to minimizing the computation time, we have characterized the age-optimal repetition, MDS and MM-MDS code parameter k (equivalently, αs).

CHAPTER 7

Timely Communication in Federated Learning

7.1 Introduction

Introduced in [171], federated learning (FL) is a distributed learning framework, where a parameter server (PS) iteratively trains a global model using rich client (user) datasets that are privacy-sensitive and large in quantity without actually storing them centrally at the data center. At each iteration, the PS distributes the current model to the clients. Each client performs the learning task locally using its own dataset and sends its model update to the PS, which then aggregates the results and updates the model (see Fig. 7.1). Some promising applications of FL are image classification and next-word prediction [171], human mobility prediction [172], news recommenders and interactive social networks [173], healthcare applications [174], and so on. Recent works in [175–181] study communication-efficient FL frameworks suitable for the limited communication between the PS and the clients considering varying channel conditions, quantization and sparsification, non-i.i.d. client datasets, and coding.

The performance of different FL frameworks are usually determined by their

convergence performance, average iteration time, and number of iterations. In certain FL applications such as social media networks and human mobility prediction where large quantities of highly temporal data are produced which diminish in value in a matter of hours, timeliness is also critical to incorporate rapidly changing data into the model in a timely manner. To illustrate this, we consider two clients, Alice and Bob, who participate in a next place forecasting task that aims to jointly train clients to predict their next location based on their current location and past habits. Such information is used to offer an enhanced experience and better recommendations in location-based social networks such as Twitter and Yelp as well as improved navigation that uses less congested routes. Assuming Bob gets moving earlier than Alice, to predict Alice's movements more accurately and consequently deliver the most relevant suggestions to her, Bob's (and all other similar clients') earlier activity should be incorporated into the model by the time Alice gets moving.

Motivated by this, in this work, we use the age of information metric to characterize information freshness in an FL framework. Recently, age of information has found new applications in reinforcement learning and distributed computation and learning [104, 105, 108–110, 182–184] including Chapters 6 and 8 of this dissertation. Particularly, [184] studies an age metric, called age of update, in the FL context to measure the staleness of each update and schedule clients at each iteration accordingly. Based on this age-based metric, authors in [184] propose a scheduling policy, which takes the staleness and communication quality of devices into account to accelerate the convergence of FL. In [184], age is used as a client scheduling metric rather than an end-to-end performance metric.

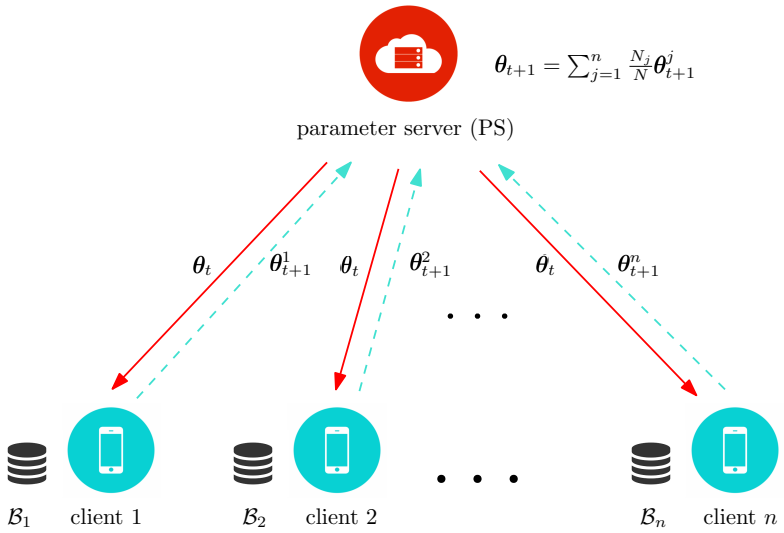


Figure 7.1: Federated learning model where a parameter server (PS) trains a learning model using n clients without actually storing clients' data centrally.

In this chapter, we propose a timely communication framework for FL that considers limited client availability and communication resources to make sure that locally generated client data are incorporated in the learning model with as little age as possible. In the proposed scheme, the PS waits for m available clients out of the total n clients at each iteration and uses local updates of the earliest k clients among these m available ones to update the global model. When k is larger, more clients update the PS at the expense of larger iteration durations. To obtain a larger k value, we can increase m which induces larger waiting times for client availability. In such a system, we characterize the average age experienced by each client and determine the age-optimal k and m values. We show that, in addition to ensuring freshness, the proposed timely communication scheme significantly improves the average iteration time compared to random client selection employed by [171] without harming the convergence of the global model.

7.2 System Model

In a typical FL setup with a single PS and n clients, each client j has the local data set $\mathcal{B}_j$, with $N_j = |\mathcal{B}_j|$ (see Fig. 7.1). The aim of the PS is to train a model parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$ using the data stored locally across the clients to minimize a particular loss function given by $L(\boldsymbol{\theta}) = \frac{1}{n}\sum_{j=1}^{n} L_j(\boldsymbol{\theta})$, where $L_j(\boldsymbol{\theta})$ denotes the application specific loss function at client j and is computed over the N_j samples in $\mathcal{B}_j$, $j \in [n]$.

At each iteration t, each client receives the current model $\boldsymbol{\theta}_t$ from the PS and performs τ-step stochastic gradient descent (SGD) for $\tau \in \mathbb{N}$ to minimize an empirical loss function with respect to its local dataset by using $\boldsymbol{\theta}_t$. At iteration t, the ℓth step of the local SGD is given by

$$\boldsymbol{\theta}_{\ell+1}^{j} = \boldsymbol{\theta}_{\ell}^{j} - \eta_{\ell}^{j} \nabla L_j(\boldsymbol{\theta}_{\ell}^{j}), \quad \ell \in [\tau] \tag{7.1}$$

where η_{ℓ}^{j} is the learning rate. Each selected device sends its model estimate after τ local steps, denoted by $\boldsymbol{\theta}_{t+1}^{j}$, with $\boldsymbol{\theta}_{t+1}^{j} = \boldsymbol{\theta}_{\tau+1}^{j}$, to the PS which updates the global model using

$$\boldsymbol{\theta}_{t+1} = \sum_{j=1}^{n} \frac{N_j}{N} \boldsymbol{\theta}_{t+1}^{j}. \tag{7.2}$$

Then, the updated global model $\boldsymbol{\theta}_{t+1}$ is shared with the clients and the whole process is repeated until convergence.

In our model, clients are not always available for the learning task.[1] Rather, clients experience exponentially separated *availability windows* to participate in the learning task. We assume that once available each client completes an iteration after which the next availability window of that client starts in an exponential duration of time. At each iteration, the PS schedules m clients such that upon completion of an iteration, the PS waits for $Z_{m:n}$ duration of time[2] to have m available clients for the next iteration, where Z is an exponential random variable with rate λ from the memoryless property.[3]

The set $\mathcal{A}_t^m$ denotes the first m available clients in iteration t to which the PS broadcasts the current model, $\boldsymbol{\theta}_t$.[4] Link delays in downlink transmissions from the PS to the clients are modeled with an exponential random variable with rate μ. We assume that the actual computation duration at clients is a deterministic duration c so that the time in between the beginning of an iteration until a participating client generates its local update is a shifted exponential random variable X with rate (c, μ).[5] The link delay back to the PS in uplink transmissions is an exponential random variable $\tilde{X}$ with rate $\tilde{\mu}$. Once the PS collects the earliest k of the m local updates, $k \leq m$, it updates the global model as in (7.2). We denote the set of the earliest k clients out of the available m clients at iteration t with $\mathcal{A}_t^k$ such that

[1]This is because, to participate in a learning task that potentially includes heavy computations, devices need to be plugged-in and running on an unmetered WiFi connection.

[2]We denote the mth smallest of random variables $Z_1, \ldots, Z_n$ as $Z_{m:n}$.

[3]To model the case in which the clients are all available, we can take $\lambda \to \infty$ in which case the PS selects m clients randomly at each iteration.

[4]Clients in $\mathcal{A}_t^m$ commit to participate in iteration t such that a client in $\mathcal{A}_t^m$ does not become unavailable within the $Z_{m:n}$ duration while waiting for others to become available.

[5]We note that each selected available client performs a local SGD using the same number of samples at each local step. When clients have identical computation capabilities we have the same computation time c for each client.

$|\mathcal{A}_t^k| = k$ and $\mathcal{A}_t^k \subseteq \mathcal{A}_t^m$.

Our aim is to design a learning framework, where the locally generated data are incorporated at the PS as timely as possible. In this work, we consider the age of information metric to measure the timeliness of the received information. The PS keeps the age of information of each client. A client's age is updated whenever its local update is received by the PS. Otherwise, its age continues to increase linearly in time. We define the long term average age of a client j as

$$\Delta_j = \lim_{T\to\infty} \frac{1}{T} \int_0^T \Delta_j(\bar{t}) d\bar{t}, \tag{7.3}$$

where $\Delta_j(\bar{t})$ represents the instantaneous age of client j at time $\bar{t}$ at the PS. We have $\Delta_j(\bar{t}) = \bar{t} - u_j(\bar{t})$, where $u_j(\bar{t})$ denotes the generation time of the most recent local update of client j that is available at the PS. We note that from the i.i.d. nature of this system, each client experiences identical age of information. Thus, in what follows, we focus on a single client and calculate its average age of information.

7.3 Average Age Analysis

The total time of an iteration, denoted by Y, is given by

$$Y = S + Z_{m:n}, \tag{7.4}$$

where S denotes the service time and is given by $S = (X + \tilde{X})_{k:m}$. A client participates in an iteration with probability $p_1 = \frac{m}{n}$ since the PS only selects the earliest

available m clients. Out of these m clients only the earliest k of them actually send back their updates to the PS for aggregation at each iteration. That is, given selected for the iteration, the probability of updating the PS is $p_2 = \frac{k}{m}$. Thus, at each iteration a client updates the PS with probability $p \triangleq p_1 p_2 = \frac{k}{n}$. The number of iterations in between two consecutive updates from a particular client is given by a geometric random variable M with rate p.

A sample age evolution of client j, $j \in [n]$ is shown in Fig. 7.2. Here, iteration t starts at time T_{t-1} when the PS starts broadcasting the tth model update to the selected available clients. Filled circles and crosses in Fig. 7.2 show the time instances at which client j generates its local updates and the PS receives those corresponding local updates, respectively. We note that upon delivery, the age of client j drops to the age of the most recent local update from client j.

In the ensuing analysis, moments of S are not easy to calculate as S is equal to an order statistic of a sum of exponential and shifted exponential random variables. To simplify, we assume that downlink transmissions are instantaneous since, in general, connection speeds are significantly asymmetric such that downlink transmissions are much faster than uplink transmissions [185]. In this case, the time it takes for each client in $\mathcal{A}_t^k$ to generate its update is c and we have $S = c + \tilde{X}_{k:m}$.

Let $\bar{Y}$ denote the time in between the generation time of two consecutive updates from client j. In Fig. 7.2, since client j updates the PS in iterations t and $t+2$, we have $M = 2$. From Fig. 7.2, we see that $\bar{Y}_t = Y_t - \bar{X}_t + Y_{t+1} + \bar{X}_{t+2}$, where $\bar{X}_t$ denotes the downlink delay of a client that is one of the earliest k clients to deliver its update at iteration t, i.e., $\bar{X} = X_j | j \in \mathcal{A}_t^k$. Since $\bar{X}_t = \bar{X}_{t+2} = c$, we

have $\bar{Y} = Y_1 + Y_2$ for $M = 2$. Then, in general, we have

$$\bar{Y} = \sum_{i=1}^{M} Y_i, \tag{7.5}$$

which is equal to the length of the shaded trapezoid in Fig. 7.2.

The metric we use, long term average age, is the average area under the age curve which is given by [29]

$$\Delta = \limsup_{T\to\infty} \frac{\frac{1}{T}\sum_{t=1}^{T} Q_t}{\frac{1}{T}\sum_{t=1}^{T} \bar{Y}_t} = \frac{\mathbb{E}[Q]}{\mathbb{E}[\bar{Y}]}. \tag{7.6}$$

By using Fig. 7.2, we find $Q_t = \frac{1}{2}\bar{Y}_t^2 + \bar{Y}_t\bar{\bar{X}}_{t+2}$. Thus, (7.6) is equivalent to

$$\Delta_j = \mathbb{E}[\bar{\bar{X}}] + \frac{\mathbb{E}[\bar{Y}^2]}{2\mathbb{E}[\bar{Y}]}, \tag{7.7}$$

where $\bar{\bar{X}}$ denotes the uplink delay of a client that is one of the earliest k clients to deliver its update at iteration t, i.e., $\bar{\bar{X}} = \tilde{X}_j | j \in \mathcal{A}_t^k$. From (7.5), we find the first and second moments of $\bar{Y}$ in terms of Y as

$$\mathbb{E}[\bar{Y}] = \mathbb{E}[M]\mathbb{E}[Y] \tag{7.8}$$

$$\mathbb{E}[\bar{Y}^2] = \mathbb{E}[M]\mathbb{E}[Y^2] + \mathbb{E}[Y]^2\mathbb{E}[M^2 - M]. \tag{7.9}$$

Inserting (7.8) and (7.9) in (7.7), we find

$$\Delta_j = \mathbb{E}[\bar{\bar{X}}] + \frac{\mathbb{E}[M^2]}{2\mathbb{E}[M]}\mathbb{E}[Y] + \frac{\mathrm{Var}[Y]}{2\mathbb{E}[Y]}. \tag{7.10}$$

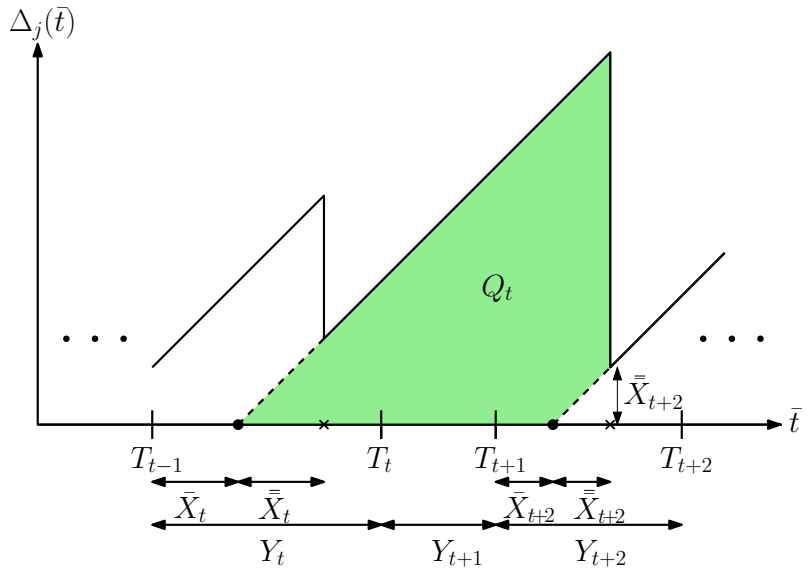


Figure 7.2: Sample age evolution at client j. Iteration t starts at time T_{t-1}. Filled circles and crosses show the time instances at which client j generates its local update and the PS receives that local update, respectively. Here, client j successfully delivers its local update to the PS in iterations t and $t+2$.

Theorem 7.1 characterizes the average age of a client under the proposed timely communication scheme using (7.10).

Theorem 7.1 *Under the proposed timely communication scheme, the average age of a client is*

$$\begin{aligned}\Delta_j =&\frac{1}{k}\sum_{i=1}^{k}\mathbb{E}[\tilde{X}_{i:m}]+\frac{2n-k}{2k}(c+\mathbb{E}[\tilde{X}_{k:m}]+\mathbb{E}[Z_{m:n}])\\&+\frac{\mathrm{Var}[\tilde{X}_{k:m}]+\mathrm{Var}[Z_{m:n}]}{2(c+\mathbb{E}[\tilde{X}_{k:m}]+\mathbb{E}[Z_{m:n}])}.\end{aligned}\tag{7.11}$$

Proof: We substitute the first two moments of M into (7.10) and note that random variables S and $Z_{m:n}$ are mutually independent to obtain

$$\Delta_j =\mathbb{E}[\bar{\bar{X}}]+\frac{2n-k}{2k}(\mathbb{E}[S]+\mathbb{E}[Z_{m:n}])+\frac{\mathrm{Var}[S]+\mathrm{Var}[Z_{m:n}]}{2(\mathbb{E}[S]+\mathbb{E}[Z_{m:n}])}.\tag{7.12}$$

The first term in (7.12) is equal to

$$\mathbb{E}[\bar{\tilde{X}}] = \mathbb{E}[\tilde{X}_j | j \in \mathcal{A}_t^k] = \sum_{i=1}^{k} \mathbb{E}[\tilde{X}_{i:m}] Pr[j = i | j \in \mathcal{A}_t^k] = \frac{1}{k} \sum_{i=1}^{k} \mathbb{E}[\tilde{X}_{i:m}]. \tag{7.13}$$

Here, noting that downlink transmissions are instantaneous, (7.13) follows from the fact that the earliest k out of m available clients is determined in an i.i.d. fashion at a certain iteration t. Together with (7.13), inserting $S = c + \tilde{X}_{k:m}$ in (7.12) and noting that $Var(S) = Var(\tilde{X}_{k:m})$ yield the result. ■

Next, we determine the average age of a client when the number of clients n is large. Here, we note that [131]

$$\mathbb{E}[Z_{m:n}] = \frac{1}{\lambda}(H_n - H_{n-m}), \tag{7.14}$$

$$\mathrm{Var}[Z_{m:n}] = \frac{1}{\lambda^2}(G_n - G_{n-m}), \tag{7.15}$$

where $H_n = \sum_{j=1}^{n} \frac{1}{j}$ and $G_n = \sum_{j=1}^{n} \frac{1}{j^2}$. First moment and variance of $\tilde{X}_{k:m}$ in (7.11) follow from (7.14) and (7.15) using $\tilde{\mu}$.

Corollary 7.1 *For large n, we set $m = \alpha n$ with $m < n$ and $k = \beta m$ with $k < m$. Then, the average age of a client given in (7.11) can be approximated as*

$$\Delta_j \approx \frac{1}{\tilde{\mu}} + \frac{(2 - \alpha\beta)c}{2\alpha\beta} - \frac{2 - \alpha\beta}{2\alpha\beta\lambda} \log(1 - \alpha) + \frac{\alpha(2 - \beta) - 2}{2\alpha\beta\tilde{\mu}} \log(1 - \beta). \tag{7.16}$$

Proof: Let δ_1, δ_2, and δ_3 denote the terms in (7.11). Then,

$$\delta_1 = \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}[\tilde{X}_{i:m}] = \frac{1}{\tilde{\mu}}H_m - \frac{1}{k\tilde{\mu}}\sum_{i=1}^{k}H_{m-i} \tag{7.17}$$

$$= \frac{1}{\tilde{\mu}} - \frac{m-k}{k\tilde{\mu}}(H_m - H_{m-k}) \tag{7.18}$$

$$\approx \frac{1}{\tilde{\mu}} + \frac{1-\beta}{\beta\tilde{\mu}}\log(1-\beta), \tag{7.19}$$

where (7.17) follows from the order statistics in (7.14). To obtain (7.18), we use the series identity $\sum_{i=1}^{k} H_i = (k+1)(H_{k+1}-1)$ [113, 119], and (7.19) follows from the fact that for large n, $H_i \approx \log(i) + \gamma$, where γ is the Euler-Mascheroni constant and is ignored here for brevity. Also,

$$\delta_2 = \frac{2n-k}{2k}(c + \mathbb{E}[\tilde{X}_{k:m}] + \mathbb{E}[Z_{m:n}]) \tag{7.20}$$

$$\approx \frac{2-\alpha\beta}{2\alpha\beta}\left(c - \frac{1}{\tilde{\mu}}\log(1-\beta) - \frac{1}{\lambda}\log(1-\alpha)\right). \tag{7.21}$$

Next, we have

$$\delta_3 = \frac{\mathrm{Var}[\tilde{X}_{k:m}] + \mathrm{Var}[Z_{m:n}]}{2(c + \mathbb{E}[\tilde{X}_{k:m}] + \mathbb{E}[Z_{m:n}])} \approx 0, \tag{7.22}$$

where (7.22) follows by using the fact that for large n, $G_n \approx \frac{\pi^2}{6}$ and hence, we have $\mathbb{E}[\tilde{X}^2_{k:m}] \approx (\mathbb{E}[\tilde{X}_{k:m}])^2$ and $\mathbb{E}[\tilde{Z}^2_{m:n}] \approx (\mathbb{E}[\tilde{Z}_{m:n}])^2$ when m is linear in n and k is linear in m. Summing δ_1, δ_2, and δ_3 yields the result. ■

Even if the PS updates the age of $k = \alpha\beta n$ clients at each iteration, the average age expression in (7.16) has terms that depend on the multiplication $\alpha\beta$ as

well as terms that only depend on either α or β. Thus, to minimize the average age, we need to optimize both α and β values. When α increases, more clients can update the PS at each iteration at the expense of longer waits for client availability. Similarly, for a given α, when β increases, more clients update the PS at the expense of longer iterations. Thus, parameters of α and β need to be carefully selected to obtain good age performance.

7.4 Numerical Results

In this section, we provide numerical results to determine the age-optimal α and β values that minimize the average age of the clients. In our simulations, we have $n = 100$ clients. In the first three simulations, we plot the average age of a client as a function of k for the age-optimal m value. That is, we first find the age-optimal (m, k) pair (equivalently, the age-optimal (α, β) pair) and then plot the average age of a client as a function of k when using the age-optimal m value.

In the first simulation, we take $\lambda = 1$, $c = 1$, and vary $\tilde{\mu}$. We observe that the average age decreases with increasing uplink transmission rates $\tilde{\mu}$. We find that the age-optimal m values are $95, 94, 92, 90, 86$ for $\tilde{\mu} = 0.1, 0.2, 0.5, 1, 5$, respectively. This shows that with increasing $\tilde{\mu}$, i.e, shorter average uplink transmission delays, the PS can afford to wait for less clients to become available at the beginning of iterations such that the age-optimal α decreases. This is because, as the transmissions become faster, the PS obtains the client updates quickly and the initial wait for client availability becomes the performance bottleneck. The corresponding age-optimal k

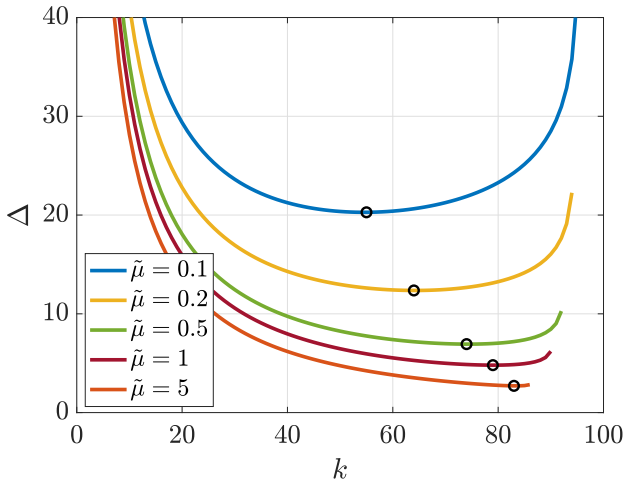


Figure 7.3: Average age experienced by a client as a function of k with $n = 100$, $\lambda = 1$, and $c = 1$ for varying $\tilde{\mu}$. In each curve we use the age-optimal m. The age-optimal k values are shown with a circle.

values are $55, 64, 74, 79, 83$ such that as the uplink transmissions become faster the PS opts for waiting more results from the clients, i.e., increasing β, instead of waiting for clients to become available in the next iteration. Further, more specifically for the low transmission rates, i.e., $\tilde{\mu} = 0.1, 0.2, 0.5$ cases, we have a familiar age curve as in the earlier works on multicast networks [113, 144] and Chapters 2 and 3 that employ an earliest k out of m idea to facilitate timely updates. In particular, we see that the average age first decreases when k increases. This is because with increasing k, clients update the PS more frequently. When k increases beyond a certain value, however, the average age starts to increase indicating that the PS waits for clients with slower links to perform the iteration.

In the second simulation, we consider the same setup as in Fig. 7.3 but take $\tilde{\mu} = 1$ and vary λ. We observe that the average age decreases for larger values of λ as the time it takes for clients to become available is less for larger λ values. Here,

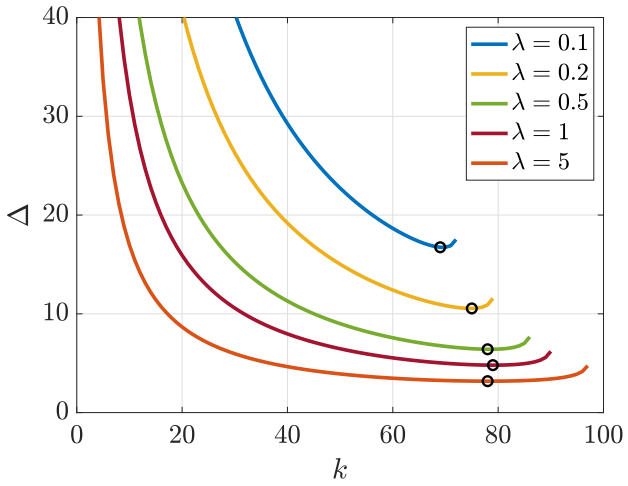


Figure 7.4: Average age experienced by a client as a function of k with $n = 100$, $\tilde{\mu} = 1$, and $c = 1$ for varying λ. In each curve we use the age-optimal m. The age-optimal k values are shown with a circle.

the age-optimal m values are $72, 79, 86, 90, 97$ for $\lambda = 0.1, 0.2, 0.5, 1, 5$, respectively, which indicate that, to facilitate timeliness, the PS selects a smaller α when the availability of the clients is more scarce. Corresponding age-optimal k values are $69, 75, 78, 79, 78$ such that we observe that, as the clients become more frequently available, the PS uses a smaller fraction of the available m clients at each iteration, i.e., β decreases with increasing λ. In this case, instead of waiting for clients with slower links to return their update, the PS chooses to start a new iteration.

In the third simulation, we consider the same setup as in Fig. 7.3 but take $\tilde{\mu} = 1$ and vary c. In this case, the age-optimal m values are $85, 90, 96, 97$ for $c = 0.1, 1, 5, 10$, respectively. Corresponding age-optimal k values are $70, 79, 91, 94$. We observe from Fig. 7.5 that as c increases both the average age and the age-optimal (k, m) values, correspondingly (α, β) values, increase. This suggests that when the fixed computation duration at the clients is larger, at each iteration, the

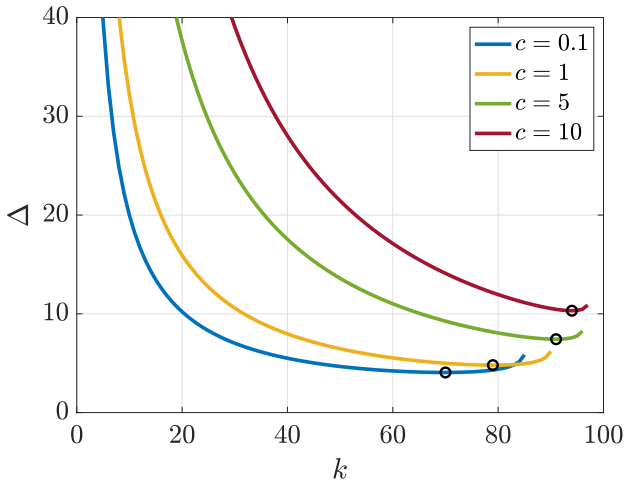


Figure 7.5: Average age experienced by a client as a function of k with $n = 100$, $\lambda = 1$, and $\tilde{\mu} = 1$ for varying c. In each curve we use the age-optimal m. The age-optimal k values are shown with a circle.

PS is more incentivized to wait for more clients to return their updates.

So far, we have found the age-optimal m and k values to minimize the average age of the clients. In practical systems, as in [184], the PS can only schedule a fixed number of clients at each iteration due to limited communication resources such as number of subchannels etc. To investigate such settings, in the fourth simulation, we fix m and analyze the average age performance by varying k. In Fig. 7.6 we observe that the age-optimal k value increases with increasing m. That is, when the PS waits for a larger number of available clients at each iteration, it is more beneficial for decreasing the age to wait for more of those available clients to return their updates. Here, the age-optimal k values are $15, 31, 48, 68, 93$ for $m = 20, 40, 60, 80, 100$, respectively. Among these m values, $m = 80$ gives the best average age result whereas $m = 60$ and $m = 100$ yield similar performance. Thus, having more communication resources is advantageous but as m increases the time

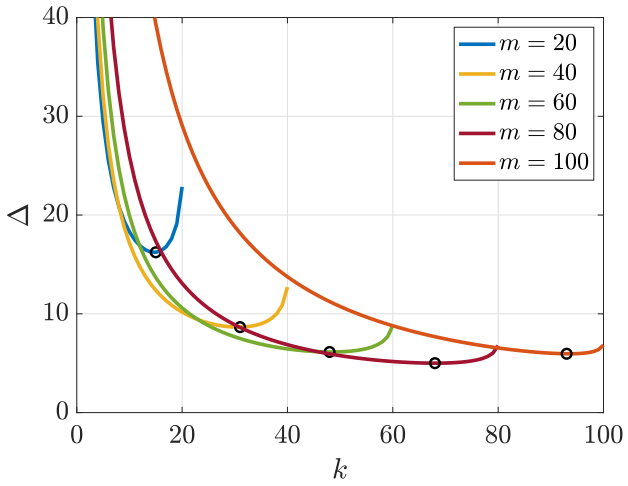


Figure 7.6: Average age experienced by a client as a function of k with $n = 100$, $\lambda = 1$, $c = 1$, and $\tilde{\mu} = 1$ for varying m. The age-optimal k values are shown with a circle.

spent in waiting for client availability starts to hurt the age performance.

Until now, we have investigated the age-optimal m and k values. Our results indicate that it is not necessarily age-optimal to get updates from each client at each iteration. In particular, we show that to get more timely updates it is better to wait for the first m available clients and then use the updates of the earliest k clients among these m available ones.

Next, we analyze the average iteration time $\mathbb{E}[Y]$ under the proposed timely communication framework. We compare its performance with two baseline schemes: random k, which selects any k clients uniformly at random at each iteration and first k, which selects the first k clients that become available at each iteration. We take $k = 10$ and $m = 20$ and use the same setup as in Fig. 7.3. In Fig. 7.7 we see that the proposed timely communication framework outperforms the random k and first k schemes. In this case, the performance improvement compared to the random

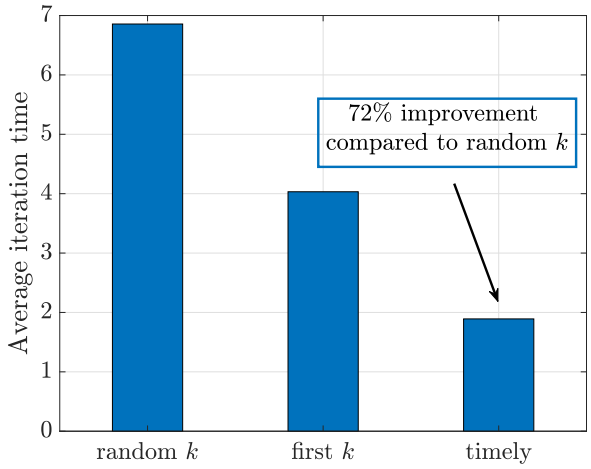


Figure 7.7: Average iteration time under different schemes when $k = 10$, $m = 20$, $n = 100$ for $c = 1$, $\tilde{\mu} = 1$, $\lambda = 1$ averaged over 50000 iterations.

k scheme is 72%. This is because random k does not consider the availability of the clients to make the client selection whereas the proposed scheme uses the client availability as well as the link delays to make the client selection. We also note that even if the clients are all available, i.e., λ tends to ∞, the proposed scheme still yields more than 50% improvement over the random k scheme. This shows that the proposed timely communication framework not only gives better age performance but also decreases the average iteration time compared to random client selection implemented in [171].

Finally, we consider the convergence performance of the proposed scheme in a learning task. The proposed timely communication operation is operationally no different than selecting a random k subset of clients at each iteration uniformly at random. In other words, under the proposed operation, at each iteration, each client updates the PS with equal probability. Thus, earlier convergence results on FL that employ a random client selection at each iteration as in [171] readily apply

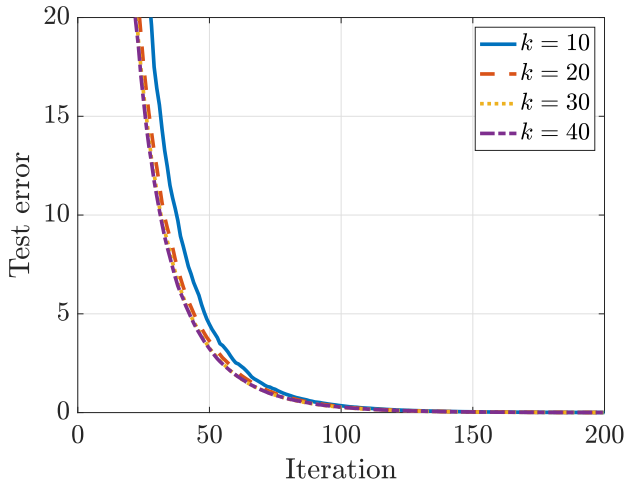


Figure 7.8: Convergence performance of the proposed scheme for varying k with $m = 40$, $n = 100$ $c = 1$, $\lambda = 1$, and $\tilde{\mu} = 1$ for a linear regression task.

in the case of the proposed timely communication scheme. To demonstrate this, we consider a simple linear regression problem over synthetically created training and test datasets as in [183]. The loss function at the clients L_j is the mean squared error and the size of the model is $d = 1000$. The dataset is randomly distributed to each client such that client j has $N_j = 20$ samples for $j \in [n]$. We have the batch size equal to 20, $\tau = 1$, and $\eta = 0.1$ for all workers and iterations. A single simulation includes $T = 200$ iterations. Results are averaged over 5 independent simulations.

Fig. 7.8 shows the convergence performance of the proposed scheme for varying k and $m = 40$. As shown in [171] for random client selection, selecting 10% of the clients, i.e., setting $k = 10$, is sufficient for achieving good convergence performance. We see from Fig. 7.6 that the age-optimal k value is 31 when $m = 40$. Here, the age-optimal k value is larger than 10 which indicates that the proposed timely communication scheme minimizes the average age of information at the clients without

slowing down the convergence.

7.5 Conclusion

In this work, we proposed a timely communication scheme for FL that is suitable for applications that include highly temporal rapidly changing client data such as social media networks, human mobility prediction systems, and news recommenders. Considering limited client availability and communication resources, in the proposed communication scheme, the PS waits until there are m available clients at the beginning of each iteration. To update the global model, the PS uses the local update of the earliest k clients from these available m clients. Under such operation, we characterized the average age of information at the clients and numerically determined the age-optimal m and k values. Our results indicate that there exists an age-optimal (m, k) pair that strikes a balance between waiting times for client availability, local update transmission times, and fresh model updates. We also showed that for the same k value, the proposed timely communication framework significantly improves the average iteration time without hurting the convergence of the learning task.

CHAPTER 8

Age-Aware Distributed Computation for Bias Reduction

8.1 Introduction

Machine learning algorithms partially owe their success to the availability of large datasets for training. However, when the size of the training datasets and the complexity of the trained models are formidable, it is not feasible to train the model on a single machine within a reasonable time frame. To overcome this computational bottleneck, distributed learning techniques are implemented across multiple machines, called *workers*.

Gradient descent (GD) methods are widely used in machine learning problems to optimize the model parameters in an iterative fashion. To speed up GD iterations, gradient computations can be distributed across multiple workers. In particular, by employing a *parameter server* (PS) type framework [186], the dataset can be divided among workers, and at each iteration, workers compute gradients based on their local data, which are aggregated by the PS. However, slow, so-called *straggling*, workers are the Achilles heel of distributed GD (DGD) since the PS has to wait for all the workers to complete an iteration. A wide range of straggler-mitigation

strategies have been proposed in recent years. In these works, the main theme is to assign redundant computations to workers to overcome the potential delays caused by straggling workers, either together with coded dataset assignment to workers, i.e., coded computation [159, 161–163, 165, 167, 169, 183, 187–199], or combined with coded local computations, i.e., coded transmission [160, 166, 200–208], or by simply using backup computations, i.e., uncoded computation [209–214]. Owing to the redundancy in these schemes, stragglers can be tolerated as the fast workers can compensate for the straggling ones.

The focus of this chapter is the coded computation for straggler mitigation. Most of the coded computation solutions designed to mitigate stragglers have two shortcomings: First, they are designed such that each worker performs a single computation per iteration, which in turn results in the under-utilization of computational resources [215]. Multi-message communication (MMC) strategy is implemented to allow workers to perform multiple computations in each iteration to seek a balance between computation and communication latency [163, 165, 167, 193, 194, 199, 205, 214]. The second shortcoming arises from the fact that most coded computation techniques aim to recover the full gradient at each iteration, which may unnecessarily increase the average completion time of an iteration. To avoid this, reference [216] combines coded computation with partial recovery (CCPR) to provide a trade-off between the average completion time of an iteration and the accuracy of the recovered gradient estimate.

If the straggling behavior is not independent and identically distributed over time and workers, which is often the case in practice, the gradient estimates recov-

ered by the CCPR scheme become biased. For example, this may happen when a worker straggles over multiple iterations. Regulating the recovery frequency of the partial computations to make sure that each partial computation contributes to the model updates as equally as possible is critical to avoid biased updates. We use the age of information (AoI) metric to track the recovery frequency of partial computations. In our work, we associate an age to each partial computation and use this age to track the time passed since the last time each partial computation has been recovered.

In this chapter, we design a dynamic encoding framework for the CCPR scheme that includes a timely dynamic order operator to prevent biased updates, and improve the performance. The proposed scheme increases the timeliness of the recovered partial computations by changing the codewords and their computation order over time. To regulate the recovery frequencies, we use the *age of partial computations (AoPC)* in the design of the dynamic order operator. We show by numerical experiments on a linear regression problem that the proposed dynamic encoding scheme increases the timeliness of the recovered computations, yields less biased model updates, and as a result, achieves better convergence performance compared to the conventional static encoding framework.

8.2 System Model and Problem Formulation

For completeness, we first present the coded computation framework and the CCPR scheme.

8.2.1 DGD with Coded Computation

We focus on the least-squares linear regression problem, where the loss function is the empirical mean squared error

$$L(\boldsymbol{\theta}) \triangleq \frac{1}{2N} \sum_{i=1}^{N} (y_i - \mathbf{x}_i^T \boldsymbol{\theta})^2, \tag{8.1}$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ are the data points with corresponding labels $y_1, \ldots, y_N \in \mathbb{R}$, and $\boldsymbol{\theta} \in \mathbb{R}^d$ is the parameter vector. The optimal parameter vector can be obtained iteratively by using the gradient descent (GD) method

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t), \tag{8.2}$$

where η_t is the learning rate and $\boldsymbol{\theta}_t$ is the parameter vector at the tth iteration. Gradient of the loss function in (8.1) is

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta_t}) = \mathbf{X}^T \mathbf{X} \boldsymbol{\theta}_t - \mathbf{X}^T \mathbf{y}, \tag{8.3}$$

where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T$ and $\mathbf{y} = [y_1, \ldots, y_N]^T$. In (8.3), only $\boldsymbol{\theta}_t$ changes over iterations. Thus, the key computational task at each iteration is the matrix-vector multiplication of $\mathbf{W}\boldsymbol{\theta}_t$, where $\mathbf{W} \triangleq \mathbf{X}^T \mathbf{X} \in \mathbb{R}^{d \times d}$. To speed up GD, execution of this multiplication can be distributed across K *workers*, by simply dividing $\mathbf{W}$ into K equal-size disjoint submatrices. However, under this naive approach, computation time is limited by the *straggling* workers [159].

Coded computation is used to tolerate stragglers by encoding the data before it is distributed among workers to achieve certain redundancy. That is, with coded computation, redundant partial computations are created such that the result of the overall computation can be obtained from a subset of the partial computations. Thus, up to a certain number of stragglers can be tolerated since the PS can recover the computation result without getting partial results from all workers. Many coded computation schemes, including MDS [159,163,192], LDPC [187], and rateless codes [194], and their various variants have been studied in the literature.

8.2.2 Coded Computation with Partial Recovery (CCPR)

In naive uncoded distributed gradient computation, straggling workers result in erasures in the gradient vector as illustrated in Fig. 8.1. The main motivation behind coded computation schemes is to find the minimum number of responsive workers to guarantee the recovery of the gradient vector without any erasures. Alternatively, the CCPR scheme [216] allows erasures on the gradient vector to reduce the computation time while controlling the number of erasures to guarantee certain accuracy for the gradient estimate. To implement the CCPR scheme, we employ a linear code structure similar to LT codes. In the case of centralized encoding, $\mathbf{W}$ is initially divided into K disjoint submatrices $\mathbf{W}_1, \ldots, \mathbf{W}_K \in \mathbb{R}^{d/K \times d}$ using which r coded submatrices, $\tilde{\mathbf{W}}_{i,1}, \ldots, \tilde{\mathbf{W}}_{i,r}$, are created and assigned to each worker i, $i \in [K]$, by

the PS, where $\tilde{\mathbf{W}}_{i,j}$ is a linear combination of the K submatrices, i.e.,

$$\tilde{\mathbf{W}}_{i,j} = \sum_{k\in[K]} \alpha_{j,k}^{(i)} \mathbf{W}_k. \tag{8.4}$$

Here, by using an analogy with LT codes, one can consider K submatrices $\mathbf{W}_1, \ldots, \mathbf{W}_K$ as symbols and their linear combinations in (8.4) as codewords, where the number of non-zero coefficients defines their degree. Following the encoding phase, at each iteration t, the ith worker performs the computations $\tilde{\mathbf{W}}_{i,1}\boldsymbol{\theta}_t, \ldots, \tilde{\mathbf{W}}_{i,r}\boldsymbol{\theta}_t$ in the given order, and sends the results one by one as soon as they are completed. In the meantime, the PS collects coded computations from all the workers until it successfully recovers $(1-q)\times 100$ percent of the gradient entries. Parameter q denotes the *tolerance*, a design parameter, which can be chosen according to the learning problem. We want to remark that due to the centralized encoding mechanism, coded submatrices are fixed throughout all the iterations. However, by considering a distributed encoding framework, as we explain next, it is possible to update coded submatrices through the iterations to have a more flexible design.

8.2.3 Distributed Dynamic Coded Submatrix Generation

Unlike the static centralized encoding scheme, inspired by the random circularly shifted (RCS) code design in [216], we consider a distributed dynamic coded submatrix generation framework. Initially all the submatrices are distributed among the workers. Then, at each iteration, workers dynamically generate their coded submatrices following a prescribed procedure. In particular, this dynamic framework

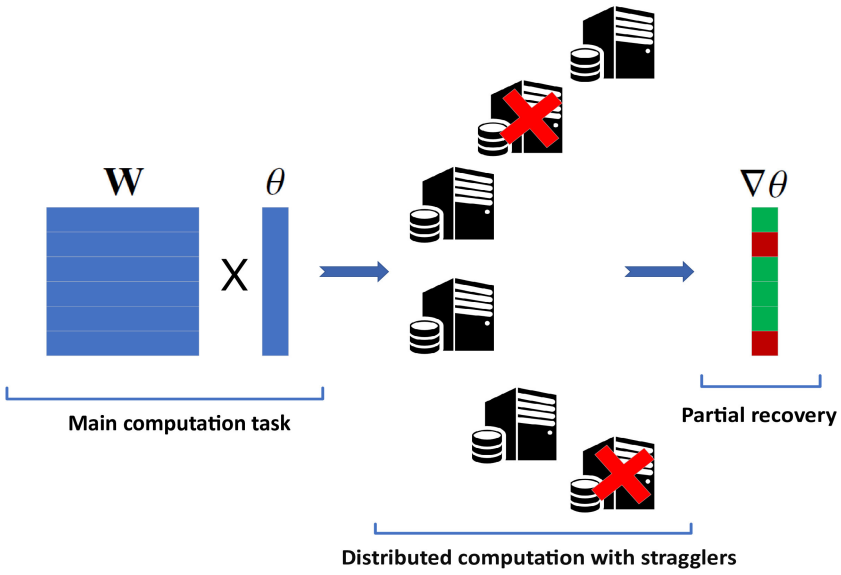


Figure 8.1: Illustration of partial recovery in a naive distributed computation scenario with 6 workers, 2 of which are stragglers.

utilizes three operators: *data partition* $D(\cdot)$, *ordering* $O(\cdot)$, and *encoding* $E(\cdot)$. The data partition operator distributes the submatrices $\mathcal{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_K\}$ among the workers such that

$$D_i(\mathcal{W}, M) : \mathcal{W} \mapsto \mathcal{W}_i, \quad |\mathcal{W}_i| \leq M, \tag{8.5}$$

where M is the given memory constraint, and $\mathcal{W}_i$ is the set of submatrices assigned to the ith worker. We assume that operator $D(\cdot)$ is executed, for each worker, only once at the beginning of the process, and $\mathcal{W}_i$, $i \in [K]$, remains the same over the iterations. The order operator $O(\cdot)$ is used to form an ordered set from the initial set $\mathcal{W}_i$ for encoding, i.e.,

$$O_{i,t}(\mathcal{W}_i) : \mathcal{W}_i \mapsto \tilde{\mathcal{W}}_{i,t}, \tag{8.6}$$

where $\tilde{\mathcal{W}}_{i,t}$ is an ordered set representing the order of computation at each iteration t for the ith worker. We remark that unlike the data partition operator, order

operator may change over time. These two operators together can be represented by an assignment matrix $\mathbf{A}_t$, whose ith column is given by $\tilde{\mathcal{W}}_{i,t}$.

Once the assignment matrix $\mathbf{A}_t$ is fixed, the encoding process is executed according to a *degree vector* $\mathbf{m}$, which identifies the degree of each codeword based on its computation order. Encoding is executed for each worker independently. The encoder operator $E(\cdot)$ maps the ordered set of data (submatrices) to ordered set of codewords of size L, where L is the length of $\mathbf{m}$, i.e.,

$$E_{i,t}(\tilde{\mathcal{W}}_{i,t}, \mathbf{m}) : \tilde{\mathcal{W}}_{i,t} \mapsto \tilde{\mathcal{C}}_{i,t} = \left\{ \mathbf{C}_{i,1}^t, \ldots, \mathbf{C}_{i,L}^t \right\}. \tag{8.7}$$

The encoding operator first divides set $\tilde{\mathcal{W}}_{i,t}$ into L disjoint subsets, $\mathcal{W}_{i,1}^t, \ldots, \mathcal{W}_{i,L}^t$, such that $|\mathcal{W}_{i,l}^t| = \mathbf{m}(l)$. We note that the degree vector is chosen according to the RCS code design, such that $\mathbf{m}(L) > \mathbf{m}(L-1) > \ldots > \mathbf{m}(1) = 1$. Then, at iteration t, the coded submatrix of the ith worker with computation order ℓ, denoted by $\mathbf{C}_{i,\ell}^t$, is constructed as

$$\mathbf{C}_{i,\ell}^t = \sum_{\mathbf{W}_k \in \mathcal{W}_{i,\ell}^t} \mathbf{W}_k. \tag{8.8}$$

An example assignment matrix $\mathbf{A}_t$ is given below for $K = 20$ and $M = 6$:

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{W}_1 & \mathbf{W}_2 & \mathbf{W}_3 & \ldots & \mathbf{W}_{20} \\ \mathbf{W}_4 & \mathbf{W}_5 & \mathbf{W}_6 & \ldots & \mathbf{W}_3 \\ \mathbf{W}_{11} & \mathbf{W}_{12} & \mathbf{W}_{13} & \ldots & \mathbf{W}_{10} \\ \mathbf{W}_{15} & \mathbf{W}_{16} & \mathbf{W}_{17} & \ldots & \mathbf{W}_{14} \\ \mathbf{W}_6 & \mathbf{W}_7 & \mathbf{W}_8 & \ldots & \mathbf{W}_5 \\ \mathbf{W}_{18} & \mathbf{W}_{19} & \mathbf{W}_{20} & \ldots & \mathbf{W}_{17} \end{bmatrix}. \tag{8.9}$$

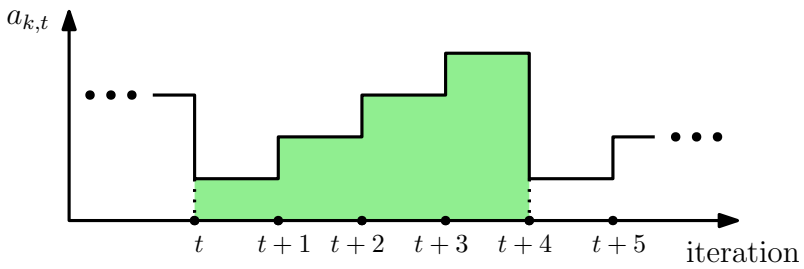


Figure 8.2: Sample age evolution of $\mathbf{W}_k\boldsymbol{\theta}_{\bar{t}}$. Time t marks the beginning of iteration $t+1$. $\mathbf{W}_k\boldsymbol{\theta}_{\bar{t}}$ is recovered at iterations $\bar{t}=t$ and $\bar{t}=t+4$.

The elements of the assignment matrix $\mathbf{A}_t$ are colored to illustrate the first step of the encoding operator, for $\mathbf{m}=[1,2,3]$, where colors blue, red and brown represent the submatrices used to generate the first, second, and third codewords, respectively. The encoding phase for the first worker at iteration t is illustrated below:

$$\tilde{\mathcal{W}}_{1,t}=\begin{bmatrix}\mathbf{W}_1\\ \mathbf{W}_4\\ \mathbf{W}_{11}\\ \mathbf{W}_{15}\\ \mathbf{W}_6\\ \mathbf{W}_{18}\end{bmatrix}\rightarrow\tilde{\mathcal{C}}_{1,t}=\begin{bmatrix}\mathbf{W}_1\\ \mathbf{W}_4+\mathbf{W}_{11}\\ \mathbf{W}_{15}+\mathbf{W}_6+\mathbf{W}_{18}\end{bmatrix}. \tag{8.10}$$

With this code, the worker first computes $\mathbf{W}_1\boldsymbol{\theta}_t$ and sends the result directly to the PS. Then, it computes $(\mathbf{W}_4+\mathbf{W}_{11})\boldsymbol{\theta}_t$ sends the result to the PS. Finally, it computes $(\mathbf{W}_{15}+\mathbf{W}_6+\mathbf{W}_{18})\boldsymbol{\theta}_t$ and sends the result to the PS.

Next, we formally state the problem using the data partition, ordering and encoding operators.

8.2.4 Problem Definition

The recovery of a partial computation $\mathbf{W}_k\boldsymbol{\theta}_t$ at iteration t depends on the data partition $\{D_i\}_{i\in[K]}$, ordering decisions $\{O_{i,t}\}_{i\in[K]}$, encoding decisions $\{E_{i,t}\}_{i\in[K]}$, computation delay statistics of the workers, $\mathbf{d}_t$, and the tolerance q, i.e.,

$$\mathbf{r}_t = R(D, O_t, E_t, d_t, q), \tag{8.11}$$

where R is the recovery operation that returns a vector $\mathbf{r}_t$ which demonstrates the recovered partial computations such that $\mathbf{r}_t(k) = 1$ if $\mathbf{W}_k\boldsymbol{\theta}_t$ is recovered at the PS for $k \in [K]$.

In the partial recovery approach, without any further control on the assigned computations, operators are fixed throughout the training process. Thus, recovered submatrix indices may be correlated over time and some partial computations may not be recovered at all. We note that this kind of recovery behavior may lead to divergence especially when q is large, since the updates become biased. Our goal is to introduce a dynamic approach for the coded computation/partial recovery procedure to regulate the recovery frequency of each partial computation. For this, we first introduce an age-based performance metric.

We define the age of partial computation $\mathbf{W}_k\boldsymbol{\theta}_t$ at iteration t, denoted by $a_{k,t}$, as the number of iterations since the last time the PS recovered that partial computation. The age for each partial computation is updated at the end of each

iteration in the following way

$$a_{k,t+1} = \begin{cases} a_{k,t} + 1, & \text{if } \mathbf{r}_t(k) = 0 \\ 1, & \text{if } \mathbf{r}_t(k) = 1 \end{cases}. \tag{8.12}$$

A sample age evolution of a partial computation is shown in Fig. 8.2 where partial computation $\mathbf{W}_k \boldsymbol{\theta}_{\bar{t}}$ is recovered at iterations $\bar{t} = t$ and $\bar{t} = t+4$. The average age of a partial computation over the training interval of T iterations is

$$a_k = \frac{1}{T} \sum_{t=1}^{T} a_{k,t}. \tag{8.13}$$

In order to make sure that each submatrix contributes to the model update as equally as possible during the training period, our goal is to keep the age of each partial computation under a certain threshold a_{th}. Thus, our objective is

$$\min_{\Pi(D,O,E)} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{K} \sum_{k=1}^{K} \mathbb{1}_{\{a_{k,t} > a_{th}\}}, \tag{8.14}$$

where $\mathbb{1}_x$ is the indicator function that returns 1 if x holds, 0 otherwise. Here, a_{th} is a design parameter that determines the desired freshness level for the partial computations and can be adjusted according to the learning problem. We note that the problem in (8.14) is over all data partitions, ordering and encoding policies, thereby is hard to optimally solve. Instead of solving (8.14) exactly, we introduce a timely dynamic ordering technique that can be used to regulate the recovery frequency of the partial computations.

8.3 Solution Approach: Timely Dynamic Ordering

In this section, we introduce timely dynamic ordering to better regulate the ages of partial computations and to avoid biased updates. We keep the data partition and encoding operators fixed and change only the ordering operator dynamically. This timely dynamic ordering is implemented by employing a vertical circular shift in the assignment matrix. With this, we essentially change the codewords and their computation order, which in turn, changes the recovered indices.

We first employ fixed vertical shifts for dynamic ordering. Then, we will dynamically adjust the shift amount based on the ages of the partial computations.

8.3.1 Fixed Vertical Shifts

In this code, which we call RCS-1, we employ one vertical shift for each worker at each iteration. That is, the order operator becomes

$$O_{i,t}(\mathcal{W}_i) : \mathcal{W}_i \mapsto circshift(\mathcal{W}_i, mod(t, L)), \tag{8.15}$$

where $circshift$ is the circular shift operator and $mod(x, y)$ is a modulo operator returning the remainder of x/y. By using vertical shifts, coded computations transmitted to the PS from a particular worker change over time to prioritize certain partial computations. For example, if worker 1 employs the ordered set $\tilde{\mathcal{W}}_{1,t}$ and codewords $\tilde{\mathcal{C}}_{i,t}$ specified in (8.10) at iteration t, after applying one vertical shift, its

computation order and codewords at iteration $t+1$, are given by

$$\tilde{\mathcal{W}}_{1,t+1} = \begin{bmatrix} \mathbf{W}_4 \\ \mathbf{W}_{11} \\ \mathbf{W}_{15} \\ \mathbf{W}_6 \\ \mathbf{W}_{18} \\ \mathbf{W}_1 \end{bmatrix} \rightarrow \tilde{\mathcal{C}}_{i,t+1} = \begin{bmatrix} \mathbf{W}_4 \\ \mathbf{W}_{11} + \mathbf{W}_{15} \\ \mathbf{W}_6 + \mathbf{W}_{18} + \mathbf{W}_1 \end{bmatrix}. \tag{8.16}$$

Here, we see that, at iteration t, the worker prioritizes the computation of $\mathbf{W}_1\boldsymbol{\theta}_t$, while in the next iteration computation of $\mathbf{W}_4\boldsymbol{\theta}_{t+1}$ is prioritized. We note that, in this method, the shift amount is fixed to one shift at each iteration, and is independent of the ages of the partial computations.

Next, we introduce an age-based vertical shift scheme to control the order of computations.

8.3.2 Age-Based Vertical Shifts

In this code, which we call RCS-adaptive, we choose the vertical shift amount based on the current ages of the partial computations. That is, instead of shifting by 1 at each iteration, the shift amount changes across iterations based on the ages of the partial computations. To effectively avoid biased updates, we focus on recovering the partial computations with the highest age at the current iteration, that is, the computations that have not been recovered in a while. In line with the problem in (8.14), we term the partial computations with age higher than the threshold

a_{th} as aged partial computations, which need to be recovered as soon as possible. To this end, a vertical shift amount is selected that places the maximum number of aged partial computations in the first position in the non-straggling workers' computation order so that they have a higher chance of recovery in the next iteration. In particular, to determine the shift amount in iteration $t+1$, the PS considers the computation order at the workers that have returned at least one computation in the previous iteration and determines a shift which places maximum number of aged partial computations in the first order in these workers. Upon determining the shift amount, every worker's assignment matrix is shifted by that amount in the next iteration. For example, if the age-based shift amount is 3 in iteration $t+1$, then the first user has

$$\tilde{\mathcal{W}}_{1,t+1} = \begin{bmatrix} \mathbf{W}_{15} \\ \mathbf{W}_{6} \\ \mathbf{W}_{18} \\ \mathbf{W}_{1} \\ \mathbf{W}_{4} \\ \mathbf{W}_{11} \end{bmatrix} \rightarrow \tilde{\mathcal{C}}_{i,t+1} = \begin{bmatrix} \mathbf{W}_{15} \\ \mathbf{W}_{6} + \mathbf{W}_{18} \\ \mathbf{W}_{1} + \mathbf{W}_{4} + \mathbf{W}_{11} \end{bmatrix}. \tag{8.17}$$

Here, in iteration $t+1$, the first worker prioritizes the computation of $\mathbf{W}_{15}\boldsymbol{\theta}_{t+1}$.

8.4 Numerical Results

In this section, we provide numerical results for comparing the proposed age-based partial computation scheme to alternative static schemes using a model-based scenario for computation latencies. For the simulations, we consider a linear regression problem over synthetically created training and test datasets, as in [188], of size of

2000 and 400, respectively. We also assume that the size of the model $d = 1000$ and the number of workers $K = 40$ while each worker can return $L = 3$ computations with $\mathbf{m} = [1, 2, 3]$. A single simulation includes $T = 400$ iterations. For all simulations, we use learning rate $\eta = 0.1$. To model the computation delays at the workers, we adopt the model in [215], and assume that the probability of completing s computations at any worker, performing s identical matrix-vector multiplications, by time t is given by

$$F_s(t) \triangleq \begin{cases} 1 - e^{-\mu(\frac{t}{s} - \alpha)}, & \text{if } t \geq s\alpha \\ 0, & \text{otherwise.} \end{cases} \tag{8.18}$$

First, we consider an extreme scenario in the straggling behavior, where we assume there are 15 persistent stragglers that are fixed for all the $T = 400$ iterations which do not complete any partial computations. For the non-persistent stragglers, we set $\mu = 10$ and $\alpha = 0.01$.[1] In Fig. 8.3, we set the tolerance level $q = 0.3$, such that at each iteration the PS aims at recovering 28 of the total 40 partial computations. We see that the proposed timely dynamic encoding strategy with one vertical shift at each iteration, RCS-1, achieves a significantly better convergence performance than the conventional static encoding with RCS. When the ages of partial computations are taken into consideration in determining the order of computation at each iteration with the proposed RCS-adaptive scheme with an age threshold of $a_{th} = 2$, we observe a further improvement in the convergence performance.

[1]To simulate the straggling behavior in our simulations, we take $\alpha = 10$ for the persistent stragglers so that effectively they do not complete any partial computations.

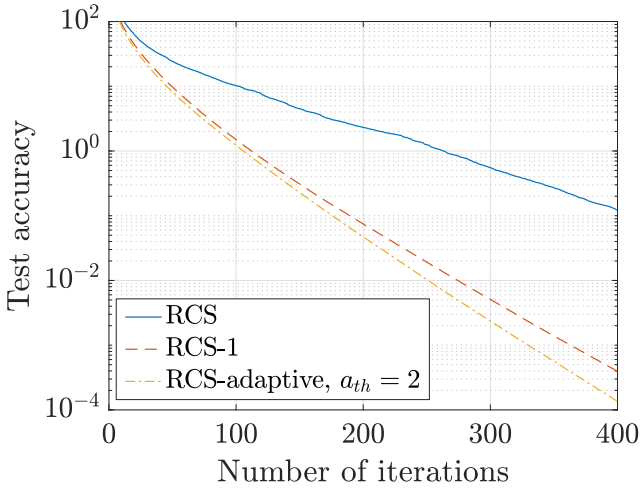


Figure 8.3: Test accuracy (log-scale) vs. number of iterations with $q = 0.3$ and 15 persistent stragglers.

An interesting observation comes from Fig. 8.4, where we plot the average ages of the partial computations. While the proposed timely dynamic encoding strategy does not result in a better average age performance for every single partial computation, it targets the partial computations with the highest average age (see computation tasks 2, 10, and 26 in Fig. 8.4). By utilizing the dynamic order operator, we essentially lower the average age of the partial computations with the worst age performance at the expense of slight increase in the age of some of the remaining partial computations. As expected, age-based vertical shift strategy further lowers the average ages of the partial computations. Here, we can draw parallels with this result and [217], which shows that as long as each component is received every p iterations, the distributed SGD can maintain its asymptotic convergence rate. From Fig. 8.4, we can see that the proposed vertical shift operator guarantees that on average each task is received every 3 iterations, since the yellow bar in Fig. 8.4 is less than 3 for each partial computation.

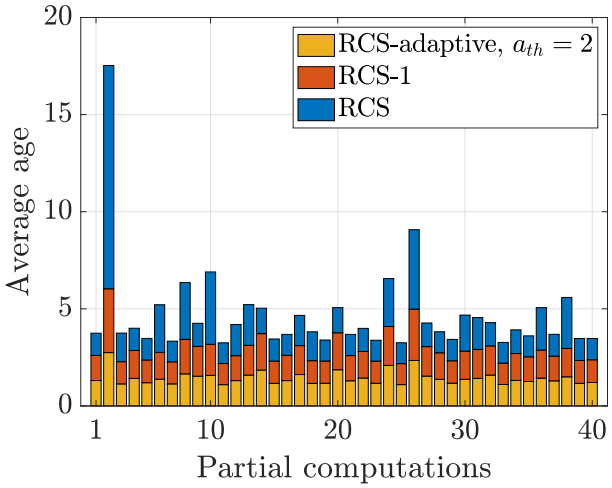


Figure 8.4: Average ages of the partial computations with $q = 0.3$ and 15 persistent stragglers.

We note that in Figs. 8.3 and 8.4 the performance gap between RCS-1 and RCS-adaptive schemes is narrow. This shows that the randomness introduced by a fixed vertical shift is already quite helpful in mitigating the biased updates with less stale partial computations.

In Table 8.1, we look at the value of the objective function in (8.14) when $a_{th} = 2$ with $\mu = 10$ and $\alpha = 0.01$ for varying tolerance levels in the case of fixed 15 persistent stragglers throughout all the iterations. We observe that, for each tolerance level q, when RCS-1 is employed, we achieve a better performance than the static RCS scheme, whereas the age-based vertical shift method RCS-adaptive results in the best performance. This is because the RCS-adaptive scheme specifically targets the computational tasks that have average age higher than the threshold a_{th} to effectively create less biased model updates where each partial computation contributes to the learning task more uniformly.

Second, we consider a more realistic scenario and model the straggling behavior

Tolerance level	RCS	RCS-1	RCS-adaptive
$q = 0.1$	0.0261	0.0180	0.0156
$q = 0.2$	0.0681	0.0476	0.0451
$q = 0.3$	0.1316	0.0970	0.0919

Table 8.1: The value of the objective function in (8.14) when $a_{th} = 2$ for varying tolerance levels q.

of workers based on a two-state Markov chain: a slow state s and a fast state f, such that computations are completed faster when a worker is in state f. This is similar to the Gilbert-Elliot service times considered in [169, 218] and Chapter 10. Specifically, in (8.18) we have rate μ_f in state f and rate μ_s in state s where $\mu_f > \mu_s$. We assume that the state changes only occur at the beginning of each iteration with probability p; that is, with probability $1-p$ the state stays the same. A low switching probability p indicates that the straggling behavior tends to stay the same in the next iteration. In Fig. 8.5, we set $p = 0.05$, $q = 0.3$, $\alpha = 0.01$, $\mu_s = 2$, and $\mu_f = 10$ and let 15 workers start at the slow state, i.e., initially we have 15 straggling workers. We note that with 15 initial stragglers and $p = 0$ we recover the setting considered in Fig. 8.3. We observe in Fig. 8.5 that the proposed timely dynamic encoding strategy improves the convergence performance even though the performance improvement is less compared to the setting in Fig. 8.3. This is because, in this scenario, the straggling behavior is less correlated over iterations, which results in less biased model updates even for the static RCS scheme. Further, we see in Fig. 8.5 that the RCS-adaptive scheme with $a_{th} = 3$ performs the best, whereas the RCS-1 scheme outperforms the RCS-adaptive scheme when $a_{th} = 2$. This shows that the age threshold a_{th} needs to be tuned to get the best performance from the RCS-adaptive

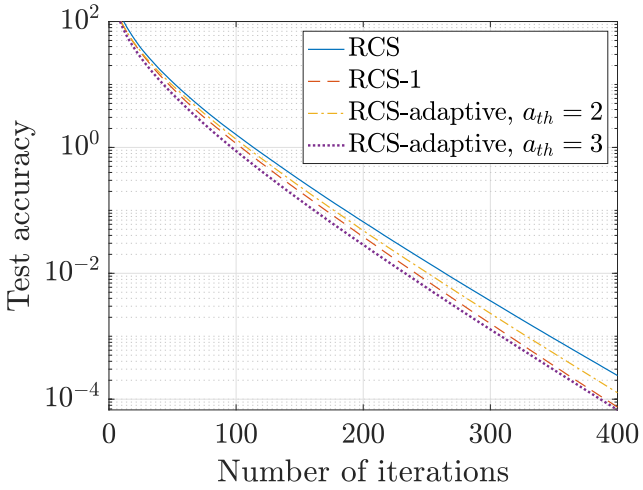


Figure 8.5: Test accuracy (log-scale) vs. number of iterations with $q = 0.3$ and straggling behavior based on a 2-state Markov chain with a state transition probability of $p = 0.05$.

scheme.

Even though we focus on the distributed coded computation scenario in this work, the proposed dynamic order operator can be applied when the computations are assigned to workers in an uncoded fashion as well. To see the performance in the case of uncoded computations with MMC, we set $\mathbf{m} = [1, 1, 1]$ and $q = 0.2$ and consider the same setup as in Fig. 8.3. In Fig. 8.6, we observe that the static partial recovery scheme fails to converge since if coding is not implemented along with partial recovery, model updates are highly biased in the presence of persistent stragglers. However, when the dynamic order operator is employed, particularly the age-aware vertical shifts with $a_{th} = 1$, convergence is achieved.

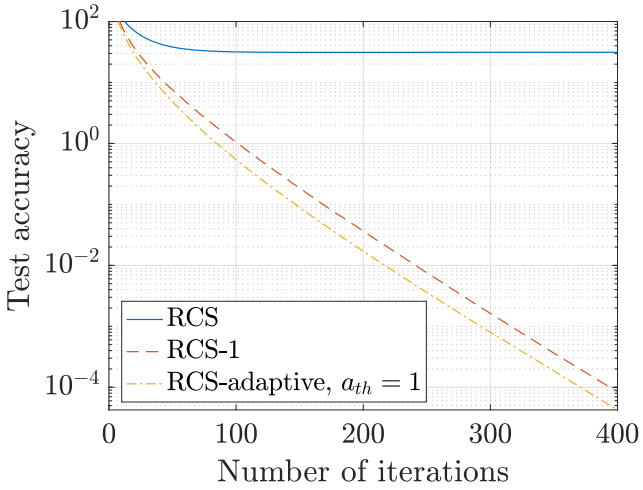


Figure 8.6: Test accuracy (log-scale) vs. number of iterations for the uncoded scheme with $q = 0.2$ and 15 persistent stragglers.

8.5 Conclusion

MMC and partial recovery are two strategies designed to enhance the performance of coded computation employed for straggler-aware distributed learning. The main drawback of the partial recovery strategy is biased model updates that are caused when the straggling behaviors of the workers are correlated over time. To prevent biased updates, we introduce a timely dynamic encoding strategy which changes the codewords and their computation order over time. We use an age metric to regulate the recovery frequencies of the partial computations. By conducting several experiments on a linear regression problem, we show that dynamic encoding, particularly an age-based encoding strategy, can significantly improve the convergence performance compared to conventional static encoding schemes. Although our main focus is on coded computation, the advantages of the proposed strategy are not limited to the coded computation scenario. The proposed timely dynamic encoding strat-

egy can be utilized for coded communication and uncoded computation scenarios as well.

CHAPTER 9

Gradient Coding with Dynamic Clustering for Straggler-Tolerant Distributed Learning

9.1 Introduction

In a typical parameter server (PS) framework with synchronous GD iterations, the dataset is distributed across the workers, and each worker computes a gradient estimate, also called a *partial gradient*, based on its own local dataset. The PS aggregates these partial gradients to obtain the full gradient and update the model. In this distributed setting, the main performance bottleneck is the slowest *straggling* workers. Many recent works have focused on developing straggler-tolerant distributed GD schemes; see Chapter 8.

In this chapter, we consider the gradient coding (GC) framework introduced in [160], where the dataset is distributed across the workers in an uncoded but redundant manner, and workers return coded computations to the PS. We note that this can also model a scenario, in which data is collected directly by the workers, instead of being distributed by the server. Redundancy can either be created by data sharing among the workers, or may be inherent due to the data collection/generation

mechanism. Thanks to the redundancy in the local datasets, partial gradients from only a subset of the workers will be sufficient to recover the full gradient. Coded combinations retrieved by the workers are designed such that any subset of responses from sufficiently many workers will allow the computation of the full gradient by the PS. Further details of GC are presented in the next section.

To improve the performance of the GC scheme, reference [166] proposes a static clustering technique, which entails dividing the workers into smaller clusters and applying the original GC scheme at the cluster level. This technique is shown to improve the average computation time compared to the original GC scheme. With clustering, unlike in the original GC scheme, the number of tolerated stragglers scales with the number of clusters when the stragglers are uniformly distributed among the clusters. However, this may not be the case in practical scenarios as evident in the measurements taken over Amazon EC2 clusters that indicate a time-correlated straggling behavior for the workers [160, 169]. In this case, the advantage of clustering diminishes since the stragglers are not uniform across clusters.

In this chapter, to mitigate this problem and to further improve the performance, we introduce a novel paradigm of dynamic coded computation, which assigns more data samples to workers than the actual computation load (per-iteration) to give them the flexibility in choosing the computations they need to carry out at each iteration. This allows the PS to choose which subset of computations each worker should try to complete at each iteration, and which coded combination it should transmit back to the PS. In particular, to reduce the potential solution space, we propose a novel GC scheme with dynamic clustering, called GC-DC, where the PS

decides on the clusters to be formed at each iteration. At each iteration in GC-DC, the PS forms the clusters such that the stragglers are distributed across the clusters as uniformly as possible based on the workers' past straggling behavior. We numerically show that the proposed GC-DC scheme significantly improves the average per-iteration completion time without an increase in the communication load under both homogeneous and heterogeneous worker environments.

9.2 Preliminaries: Gradient Coding (GC) and Clustering

In many machine learning problems, given a labeled dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots (\mathbf{x}_s, y_s)\}$, where $\mathbf{x}_1, \dots, \mathbf{x}_s \in \mathbb{R}^d$ are the data points with corresponding labels $y_1, \dots, y_s \in \mathbb{R}$, the goal is to solve the following optimization problem

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{s} l(\mathbf{x}_i, y_i, \boldsymbol{\theta}), \tag{9.1}$$

where l is the application-specific loss function and $\boldsymbol{\theta} \in \mathbb{R}^d$ is the parameter vector to be optimized. The optimal parameter vector can be obtained iteratively using GD. The *full gradient* computed over the whole dataset at iteration t is given by $\boldsymbol{g}^{(t)} = \sum_{i=1}^{s} \nabla l(\mathbf{x}_i, y_i, \boldsymbol{\theta}_t)$. When the size of the dataset, s, is large, the computation of the full gradient becomes a performance bottleneck. To speed up GD iterations, gradient computations can be distributed across multiple workers. However, in many implementations, particularly in the context of 'serverless' computing, e.g., Microsoft Azure, Amazon Web Services (AWS), the workers' completion time of assigned tasks can be highly heterogeneous and stochastic over time. In those cases, the overall

computation speed of each iteration becomes limited by the slowed *straggling* server. Coded computing techniques tackle the bottleneck due to stragglers by introducing redundant computations in a structured manner such that additional computations carried out by faster servers can compensate for the stragglers.

9.2.1 Gradient Coding (GC)

GC is a distributed coded computation technique introduced in [160] to perform distributed GD across K workers. The complete dataset $\mathcal{D}$ is divided into K non-overlapping equal-size mini-batches, $\mathcal{D}_1, \ldots, \mathcal{D}_K$, and each worker is assigned multiple mini-batches. We denote the set of indices of mini-batches assigned to the kth worker with $\mathcal{I}_k$, $k \in [K] \triangleq \{1, \ldots, K\}$. Let $\boldsymbol{g}_k^{(t)}$ denote the partial gradient for the parameter vector $\boldsymbol{\theta}_t$ evaluated over mini-batch $\mathcal{D}_k$ at the tth GD iteration, i.e.,

$$\boldsymbol{g}_k^{(t)} = \frac{1}{|\mathcal{D}_k|} \sum_{(\mathbf{x},y)\in\mathcal{D}_k} \nabla l(\mathbf{x}, y, \boldsymbol{\theta}_t). \tag{9.2}$$

We note that the *full gradient* is given by $\boldsymbol{g}^{(t)} = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{g}_k^{(t)}$. To tolerate straggling workers, GC assigns redundant mini-batches, and hence, redundant computations, to the workers.

If a mini-batch $\mathcal{D}_i$ is assigned to worker k, i.e., $i \in \mathcal{I}_k$, then the corresponding partial gradient $\boldsymbol{g}_i^{(t)}$ is computed by the kth worker. *Computation load*, r, denotes the number of mini-batches assigned to each worker, i.e., $|\mathcal{I}_k| = r$, $\forall k \in [K]$. At each iteration, each worker first computes the r partial gradients, one for each mini-batch available locally, and sends a linear combination of the results, $\boldsymbol{c}_k^{(t)} \triangleq \mathcal{L}_k(\boldsymbol{g}_i^{(t)}$:

$i \in \mathcal{I}_k$), called a *coded partial gradient*. Thus, in GC, each worker is responsible for computing a single predefined coded partial gradient. The underlying code structure in GC, which dictates the linear combinations formed by each worker, exploits the available redundancy so that the PS can recover the full gradient from only a subset of the combinations. Accordingly, from now on, we refer to the coded partial gradients formed by the workers simply as *codewords.* As shown in [160], the GC scheme can tolerate up to $r-1$ persistent stragglers[1] at each iteration. Formally, for any set of non-straggling workers $\mathcal{W} \subseteq [K]$ with $|\mathcal{W}| = K - r + 1$, there exists a set of coefficients $\mathcal{A}_{\mathcal{W}} = \left\{a_k^{(t)} : k \in \mathcal{W}\right\}$ such that

$$\sum_{k\in\mathcal{W}} a_k^{(t)} \boldsymbol{c}_k^{(t)} = \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{g}_k^{(t)}. \tag{9.3}$$

Thus, at each iteration t, the full gradient $\boldsymbol{g}^{(t)}$ can be recovered from any $K - r + 1$ codewords.

Next, we present the idea of *clustering* that was introduced in [166] to reduce the average per-iteration completion time of the GC scheme.

9.2.2 Gradient Coding with Static Clustering (GC-SC)

In GC with clustering, we divide the workers into P disjoint equal-size clusters. Let $\mathcal{K}_p \subset [K]$ denote the set of workers in cluster p, $p \in [P]$, where $\mathcal{K}_q \cap \mathcal{K}_p = \emptyset$ for $q \neq p$, and $\bigcup_{p\in[P]} \mathcal{K}_p = [K]$. We denote the cluster size by $\ell \triangleq \frac{K}{P}$, where we assume that K is divisible by P for simplicity. The assignment of the workers to the clusters

[1]These are the straggler workers that either cannot complete any computation or whose computations are not used while recovering the full gradient [166].

is dictated by an $\ell \times p$ worker assignment matrix, denoted by $\mathbf{A}_{cluster}$, where each column corresponds to a different cluster and the entries in each column correspond to indices of the workers assigned to that cluster. This worker assignment matrix is fixed throughout the training process, hence the name *static clustering*. From now on, we refer to the GC with static clustering scheme as GC-SC.

In GC-SC, each worker is assigned r mini-batches based on its cluster. This is represented by an $r \times k$ data assignment matrix $\mathbf{A}_{data}$, where each column corresponds to a different worker, and the entries in column i, $i \in [K]$, represent the mini-batches (correspondingly the partial gradient computations) assigned to the ith worker. Equivalently, data assignment can be represented by a $1 \times k$ codeword assignment matrix $\mathbf{A}_{code}$, which represents the codewords assigned to the workers, where the codeword assigned to the ith worker in the pth cluster is denoted by $c_{p,i}$, for $p \in [P]$, $i \in [\ell]$. Let $\mathcal{I}_{\mathcal{K}_p}$ denote the set of mini-batches assigned to the workers in the pth cluster, i.e., $\mathcal{I}_{\mathcal{K}_p} = \bigcup_{k \in \mathcal{K}_p} \mathcal{I}_k$. In GC-SC, the GC scheme is applied to each cluster separately and the workers in cluster p aim at computing

$$\frac{1}{|\mathcal{I}_{\mathcal{K}_p}|} \sum_{k \in \mathcal{I}_{\mathcal{K}_p}} \boldsymbol{g}_k^{(t)}. \tag{9.4}$$

To illustrate the advantage of the clustering technique, consider $K = 12$, $r = 2$, and $P = 4$. Here, the workers are divided into 4 clusters, each consisting of $\ell = 3$ workers, and each cluster is responsible for computing 3 of the total 12 partial gradients. Since $r = 2$, each worker aims at computing the assigned 2 partial gradients.

In our example, the worker assignment can be specified by the following matrix:

$$\mathbf{A}_{cluster} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 6 & 7 & 8 & 5 \\ 9 & 10 & 11 & 12 \end{bmatrix}. \tag{9.5}$$

In this assignment, workers $1, 6$ and 9 are in the first cluster, workers $2, 7$ and 10 are in the second cluster, and so on. The corresponding $\mathbf{A}_{data}$ is given in (9.6) for the cluster assignment in (9.5). In (9.6), workers in each cluster are represented by a different color. We use blue, red, magenta, and green for clusters 1, 2, 3, and 4, respectively. The corresponding $\mathbf{A}_{code}$ for the cluster assignment in (9.5) is given in (9.7), where, codewords corresponding to different clusters are shown in different colors. Each codeword in $\mathbf{A}_{code}$ is a linear combination of $r = 2$ partial gradients. For example, $c_{1,1}$ is a linear combination of partial gradients g_1 and g_2; $c_{1,2}$ is a linear combination of partial gradients g_2 and g_3, and $c_{1,3}$ is a linear combination of partial gradients g_3 and g_1. Thus, given $\mathbf{A}_{cluster}$, either $\mathbf{A}_{data}$ or $\mathbf{A}_{code}$ is sufficient to completely characterize the partial computations that will be carried out by each worker.

In the original GC scheme, the PS waits until it receives $K - r + 1 = 11$ results at each iteration; hence only $r - 1 = 1$ straggler can be tolerated. With clustering, the PS needs to receive at least $\ell - r + 1 = 2$ results from each cluster to recover the full gradient. Thus, the non-straggling threshold is still $K - r + 1$, since more than one straggler cannot be tolerated if they are in the same cluster. However, the

$$\mathbf{A}_{data} = \begin{bmatrix} g_1 & g_4 & g_7 & g_{10} & g_{11} & g_2 & g_5 & g_8 & g_3 & g_6 & g_9 & g_{12} \\ g_2 & g_5 & g_8 & g_{11} & g_{12} & g_3 & g_6 & g_9 & g_1 & g_4 & g_7 & g_{10} \end{bmatrix} \tag{9.6}$$

$$\mathbf{A}_{code} = \begin{bmatrix} c_{1,1} & c_{2,1} & c_{3,1} & c_{4,1} & c_{4,2} & c_{1,2} & c_{2,2} & c_{3,2} & c_{1,3} & c_{2,3} & c_{3,3} & c_{4,3} \end{bmatrix} \tag{9.7}$$

non-straggling threshold represents a worst case scenario. With clustering, up to 4 stragglers can be tolerated if they are uniformly distributed across clusters, i.e., one straggler per cluster, as shown in "Realization 1" in Fig. 9.1. This shows that, with clustering, the full gradient can be recovered in a much larger set of realizations compared to the original GC scheme. Thus, even if the non-straggling threshold (which corresponds to the worst case scenario) remains the same, clustering will reduce the average per-iteration completion time.

Formally, with clustering, it is possible to tolerate $r-1$ stragglers in each cluster in the best case scenario, which is when the stragglers are uniformly distributed among the clusters. In this case, it is possible to tolerate $P(r-1)$ stragglers in total. However, this advantage of clustering diminishes in the case of non-uniform distributed stragglers among the clusters, which may be the case in practice. As shown in "Realization 2" in Fig. 9.1, even though there are still 8 non-straggling workers, the PS cannot compute the full gradient (in the case of persistent stragglers) when the stragglers are not uniformly distributed across the clusters. To this end, in the next section, we introduce the concept of dynamic codeword assignment, which dynamically changes codewords computed by the workers at each iteration based on the past straggler behavior to further improve the performance of the clustering

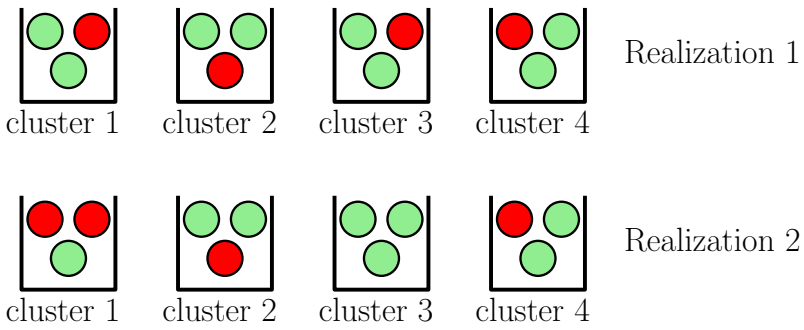


Figure 9.1: Two possible straggler realizations where red and green circles represent the straggling and non-straggling workers, respectively.

technique.

9.3 GC with Dynamic Codeword Assignment

In the conventional coded computation approaches, including GC, the assignment of the dataset to the workers and the code to be used are static and set at the beginning of the training. That is, at every iteration, a worker tries to compute the gradient estimates for all the mini-batches assigned to it, and returns their exact same linear combination. Thus, in order to recover the desired computation result at each iteration, the codes are designed for the worst case scenario. The core idea behind dynamic codeword assignment is to change the codewords assigned to the workers dynamically based on the observed straggling behavior. Dynamic codeword assignment is driven by two policies; namely, *data assignment* and *codeword assignment*. The data assignment policy, denoted by Π_d, is executed only once at the beginning of training and assigns up to m mini-batches to each worker, where m

denotes the memory constraint, i.e.,

$$\Pi_d : \mathcal{D} \mapsto \{\mathcal{I}_1, \ldots, \mathcal{I}_K : |\mathcal{I}_k| \leq m\}\,. \tag{9.8}$$

Even though each worker can be allocated up to m mini-batches, each will compute only r of them at each iteration; hence, the computation load at each iteration remains the same. We can have $\binom{m}{r}$ codewords that can be assigned to each worker depending on which subset of r computations it carries out among m possibilities. Here, we introduce $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$, where $\mathcal{C}_k$ denotes the set of feasible codewords corresponding to dataset $\mathcal{I}_k$. That is, $\mathcal{C}_k$ denotes the set of codewords that may be assigned to the kth worker at each iteration, where each codeword is a linear combination of r gradient estimates that can be computed by this worker.

We would like to highlight that with dynamic codeword assignment, the PS will specify at each iteration which codeword must be computed by each worker. This introduces additional communication requirement compared to the static schemes, such as GC and GC-SC. On the other hand, this information can be piggybacked on other control information that must be communicated from the PS to the workers at each iteration, such as signalling the end of an iteration and the transmission of the updated model parameters. However, it is still important to keep this additional information minimal by designing a codebook with minimal $|\mathcal{C}_k|$.

At the beginning of each iteration t, codeword assignment policy Π_a is executed by the PS based on the past straggler behavior of the workers up to iteration t, $\mathbf{S}^{[t-1]}$,

i.e.,

$$\Pi_a^{(t)}(\mathbf{S}^{[t-1]}, \Pi_d) : \mathcal{C} \mapsto \mathbf{c}^t = \left\{c_1^t, \ldots, c_K^t\right\}, \tag{9.9}$$

where $c_k^t \in \mathcal{C}_k$ is the codeword assigned to the kth worker at iteration t and $\mathbf{S}^{[t-1]} \triangleq (\mathbf{S}^1, \ldots, \mathbf{S}^{t-1})$, while $\mathbf{S}^t = (S_1^t, \ldots, S_K^t)$ denotes the straggler behavior at each iteration t, where $S_k^t = 0$ if the kth worker is a straggler at iteration t, and $S_k^t = 1$ otherwise.[2]

The completion time of iteration t for a given data assignment policy Π_d depends on the codeword assignment $\mathbf{c}_t$ and the straggler realization $\mathbf{S}^t$. Here, our objective is to minimize the expected completion time of each iteration based on the past straggler behavior for a given Π_d:

$$\min_{\Pi_a^{(t)}} \mathbb{E}_{\mathbf{S}^t|\mathbf{S}^{[t-1]},\Pi_d} Q(\mathbf{c}^t, \mathbf{S}^t), \tag{9.10}$$

where $Q(\mathbf{c}^t, \mathbf{S}^t)$ is the completion time of iteration t under codeword assignment $\mathbf{c}_t$ and the straggler realization $\mathbf{S}^t$.

We remark that the codeword assignment policy $\Pi_a^{(t)}$ highly depends on the data assignment policy Π_d since in most of the coded computation scenarios the data assignment policy is driven by the employed coding strategy. Thus, designing a data assignment policy Π_d without any prior knowledge on the coding strategy is a challenging task. To this end, in the next section, we reformulate the dynamic

[2]In this work, we assume an on/off straggling behavior for each worker such that a worker's straggling status can change over iterations. Workers can still deliver computation results in the straggling state but their computations are much slower. This type of two-state straggling behavior is observed in empirical studies over Amazon EC2 clusters [160, 169].

codeword assignment problem where the coding strategy, consequently the set of codewords, are fixed at the beginning and data assignment is performed based on the underlying coding strategy.

9.4 GC with Dynamic Clustering (GC-DC)

In this section, we reformulate the dynamic codeword assignment problem, and introduce the GC-DC scheme. For the construction of the GC-DC scheme, we perform three steps; namely, codeword construction, codeword distribution, and dynamic clustering, where the first two steps are executed once at the beginning of training and the last one is executed at each iteration. Our code construction will be based on GC-SC presented in Section 9.2.2, and we will transform the dynamic codeword assignment problem into a dynamic clustering problem. We note that the number of clusters P is fixed and decided at the beginning of the training.

9.4.1 Codeword Construction

In the GC-DC scheme, we will request each worker to compute and return a codeword at each iteration. Remember that each codeword is a specified linear combination of the gradient estimates for a subset of r mini-batches, and the PS and the workers need to agree on how to form these linear combinations in advance. Here, the set of codewords $\mathcal{C}$ is a union of smaller disjoint codeword sets, i.e., $\mathcal{C} = \bigcup_{p=1}^{P} \mathcal{C}^p$, such that the codewords in each set $\mathcal{C}^p$, $p \in [P]$, are encoded and decoded independently and correspond to a particular cluster. For example, in (9.7), $\mathcal{C}^1 = \{c_{1,1}, c_{1,2}, c_{1,3}\}$,

where $\mathcal{C}^1$ is disjoint from the rest of the codeword set.

9.4.2 Codeword Distribution

The codewords in $\mathcal{C}$ are distributed among the workers according to a policy Π_c, i.e.,

$$\Pi_c(\mathcal{C}) : \mathcal{C} \mapsto \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}, \tag{9.11}$$

where we remark that $\mathcal{C}_k$ denotes the set of codewords that can be assigned to the kth worker at each iteration. Now, let $\mathcal{I}(c) \subseteq \mathcal{D}$ be the minimal subset of mini-batches that is sufficient to construct codeword c, where we have $|\mathcal{I}(c)| \leq r$. Given the codeword distribution policy Π_c, any feasible data assignment policy Π_d should satisfy the following constraint

$$\mathcal{I}_k \supseteq \bigcup_{c \in \mathcal{C}_k} \mathcal{I}(c), \quad \forall k \in [K]. \tag{9.12}$$

Based on this constraint, we observe that, given $\Pi_c(\mathcal{C})$, the minimum memory is used when $\mathcal{I}_k = \bigcup_{c \in \mathcal{C}_k} \mathcal{I}(c), \forall k \in [K]$. Thus, we note that the data assignment policy Π_d is determined according to the codeword distribution policy Π_c. In other words, we first perform codeword distribution and then assign the corresponding mini-batches to the workers.

Next, we describe the codeword distribution policy Π_c in (9.11) for the proposed GC-DC scheme. We first assign each worker to n clusters. Each cluster p corresponds to a set of codewords $\mathcal{C}^p$ with $|\mathcal{C}^p| = \ell$. We say that a worker is in

cluster p, if that worker is assigned all ℓ codewords in $\mathcal{C}^p$. Hence, in the proposed scheme, each worker is assigned codewords from an n-subset of $\{\mathcal{C}^1, \ldots, \mathcal{C}^P\}$.[3] With this, we form a worker cluster assignment matrix $\mathbf{A}_{cluster}$ of size $\ell n \times P$. The pth column of $\mathbf{A}_{cluster}$ shows the workers assigned to the pth cluster, where w_k denotes the kth worker, $k \in [K]$. An example $\mathbf{A}_{cluster}$ for our continuing example is given in (9.13) for $n = 2$,

$$\mathbf{A}_{cluster} = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_6 & w_7 & w_8 & w_5 \\ w_9 & w_{10} & w_{11} & w_{12} \\ w_4 & w_1 & w_2 & w_3 \\ w_7 & w_8 & w_5 & w_6 \\ w_{10} & w_{11} & w_{12} & w_9 \end{bmatrix}. \tag{9.13}$$

When assigning workers to clusters, we start by dividing workers into groups of P according to their indices. For example, in our continuing example for $P = 4$ and $K = 12$, these groups are $\{w_1, \ldots, w_4\}$, $\{w_5, \ldots, w_8\}$, and $\{w_9, \ldots, w_{12}\}$. Then, we utilize a circular shift operator and sample n shift amounts in $\{0, \ldots, P-1\}$ uniformly at random without replacement for each of these groups. We circularly shift each of these groups according to the corresponding sampled shift amounts and form the worker cluster assignment matrix $\mathbf{A}_{cluster}$. For example, in the first and fourth rows of (9.13), the shift amounts for workers $\{w_1, \ldots, w_4\}$ are 0 and 1,

[3] That is, under the proposed GC-DC scheme, we have $|\mathcal{C}_k| = n\ell$ such that each worker may be assigned all ℓ codewords for each of the clusters that it belongs to.

respectively. As a result of these random shifts, worker w_1 is assigned to the first and second clusters, worker w_2 is assigned to the second and third clusters, and so on. Similarly, from the second and fifth rows of (9.13), we observe that the shift amounts for workers $\{w_5, \ldots, w_8\}$ are 3 and 2, respectively. We note that, since the random shifts for the same set of workers, e.g., workers $\{w_1, \ldots, w_4\}$, are sampled without replacement, each worker is assigned to exactly $n = 2$ distinct clusters.

We remark that, given n, the memory requirement m of the proposed GC-DC scheme is given by $m = n\ell$. Thus, for $n = 2$ and $\ell = 3$, each worker stores 6 mini-batches in this example.

By constructing $\mathbf{A}_{cluster}$, we essentially perform the codeword distribution as each worker is assigned all ℓ codewords for each of the n clusters that it is associated with. For example, from (9.13) we deduce that worker 1 has all the codewords in sets $\mathcal{C}^1$ and $\mathcal{C}^2$, i.e., $\mathcal{C}_1 = \mathcal{C}^1 \cup \mathcal{C}^2 = \{c_{1,1}, c_{1,2}, c_{1,3}, c_{2,1}, c_{2,2}, c_{2,3}\}$. With this, we perform the data assignment and assign corresponding mini-batches to each worker to form the data assignment matrix such that the constraint in (9.12) is satisfied with equality. Correspondingly, $\mathcal{I}_1 = \{\mathcal{D}_1, \ldots, \mathcal{D}_6\}$ so that worker 1 can compute partial gradients $g_1, \ldots, g_6$ to form any one of these 6 codewords.

9.4.3 Dynamic Clustering

The key idea behind dynamic clustering is to associate each worker to more than one cluster by assigning more than r mini-batches to each worker. Assuming that a worker is associated with n clusters, each worker is assigned a total of $n\ell$ codewords

so that a worker can replace any worker in the n clusters it is associated with by computing a codeword that would be computed by the worker to be replaced in the original GC scheme with clustering. Then, at each iteration the PS selects one of the $n\ell$ codewords for each worker based on the previous straggler realization through a codeword assignment policy Π_a given in (9.9). We note that, even though more than one codeword is assigned to each worker, computation load is still r as in the original GC scheme, and each worker still computes only one codeword consisting of r partial gradient computations at each iteration.

To see the benefit of the proposed GC-DC scheme, we consider $\mathbf{A}_{cluster}$ and corresponding codewords for a particular straggler realization $\mathbf{S} = [1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1]$, where, colors follow the cluster assignment in the static clustering case, i.e., $\mathbf{A}_{cluster}$ given in (9.5). Under the GC-SC scheme, it is not possible to recover partial gradients corresponding to the third cluster as we do not have $\ell - r + 1 = 2$ non-straggling workers in that cluster.[4] Moreover, if this straggling behavior persists for a substantial duration of time, the overall computation time will suffer drastically. To mitigate this, in the case of dynamic clustering, we observe in (9.13) that worker w_5 can replace worker w_3 since it can compute codeword $c_{3,1}$ which is the codeword that was originally assigned to worker w_3 in (9.7) in the GC-SC scheme. This does not affect the recoverability of the partial gradients assigned to the fourth cluster, to which worker w_5 initially belongs, since that cluster has 2 more non-straggling workers, workers w_4 and w_{12}. Further, worker w_2 can replace worker w_8 so that all

[4]We note that this is the case assuming straggling workers do not return any computation results. Even if they do, whenever there are less than $\ell - r + 1$ non-straggling workers in a cluster, the PS has to wait for at least one of the straggling workers to return its computation which may incur a significant delay in the completion time of that iteration.

partial gradients can be recovered successfully. Equivalently, we have assigned the clusters such that non-straggling workers w_2 and w_5 now belong to the 3rd cluster by ensuring that all other clusters still have at least $\ell - r + 1 = 2$ non-straggling workers. Thus, dynamic clustering increases the set of straggler realizations for which the full gradient recovery is possible compared to static clustering.

Since each worker can replace any worker in all the n clusters that it is assigned to, we essentially form the clusters, dynamically at each iteration through codeword assignments, hence the name *dynamic clustering.* That is, based on the codeword distribution presented in Section 9.4.2, we can assign ℓ workers to each cluster according to the given worker cluster assignment matrix $\mathbf{A}_{cluster}$ without explicitly stating which worker will compute which codeword. With this, our aim is to dynamically form clusters at each iteration to minimize the average completion time of an iteration given the past straggler behavior and the worker-cluster assignment matrix $\mathbf{A}_{cluster}$.

Next, we characterize the average completion time of an iteration for a given cluster assignment. We denote the kth smallest of random variables $Y_1, \ldots, Y_n$ as $Y_{k:n}$. The completion time of iteration t for cluster p is given by the time the PS receives the earliest $\ell - r + 1$ results from that cluster such that

$$Q^p(\mathbf{c}^t, \mathbf{S}^t) = \{X^p_{1,r}, \ldots, X^p_{\ell,r}\}_{\ell-r+1:\ell}, \quad p \in [P], \tag{9.14}$$

where $\mathbf{c}^t$ is the set of codewords assigned to the workers as in (9.9) and $X^p_{k,r}$, $k \in [\ell]$, is the computation duration of the kth worker of cluster p, i.e., the time it takes for

that worker to compute r partial gradients. Noting that iteration t ends when each cluster recovers its corresponding partial gradients, completion time of iteration t is given by

$$Q(\mathbf{c}^t, \mathbf{S}^t) = \max_{p \in [P]} Q^p(\mathbf{c}^t, \mathbf{S}^t). \tag{9.15}$$

Since some of the workers are stragglers, computation capabilities of the workers are not identical. In this case, minimizing the iteration completion time given in (9.15) through cluster assignments is not an analytically tractable problem. Instead, in the next section, we propose a greedy dynamic clustering strategy that aims to uniformly place stragglers across clusters at each iteration to speed up GC.

9.5 Greedy Dynamic Clustering Strategy

In line with the observations on Amazon EC2 instances in [160, 169], in this section, we consider a stochastic straggling behavior for the workers. In particular, we assume that workers' computation statistics are independent from each other, and follow a two-state Markov process. That is, at each iteration a worker can be either in a straggling or a non-straggling state. Once a worker starts straggling, it operates significantly slower than the non-straggling performance and remains straggling for a while. This may model an increased load at a worker for a period of time, which reduces the computational resources that can be allocated for the specific computation task. Our proposed greedy algorithm utilizes this time-correlated straggling behavior to assign straggling workers to different clusters. At each iteration, the

PS identifies the stragglers based on the past observations and implements a greedy dynamic clustering strategy to uniformly distribute the stragglers across clusters to improve the completion time of each iteration. We note that the performance gain of the proposed GC-DC scheme is prominent when the computation speeds of the workers are not identically distributed over iterations, e.g., they exhibit time-correlated straggling behavior, as the GC-DC scheme gains from adapting to the straggling behavior by carefully placing the workers to clusters at each iteration.

Inspired by the bin packing problem [219], we consider clusters as bins and workers as balls as in Fig. 9.1. Unlike the bin packing problem, which aims to place balls of different volumes into a minimum number of bins of finite volume, in our setting, the number of bins (clusters) is fixed and our aim is to distribute the straggling workers as uniformly as possible to clusters using the worker cluster assignment matrix $\mathbf{A}_{cluster}$. Our dynamic clustering algorithm has two phases: in the first phase, based on the previous straggler realization, we place straggler and non-straggler workers into clusters separately following a specific order, and in the second phase, any placement conflict that may happen in the first phase (i.e., if a worker cannot be placed into any of the remaining clusters) is resolved through worker swap between the corresponding clusters. During worker placement, clusters take turns based on a specified order and we implement a greedy policy such that, once its turn comes, each cluster selects the first available worker that can be assigned to that cluster based on the given worker cluster assignment matrix $\mathbf{A}_{cluster}$.

In what follows we describe in detail the proposed dynamic clustering strategy, which is also presented in Algorithm 1 shown at the end of this chapter. Given the

worker cluster assignment matrix $\mathbf{A}_{cluster}$, without loss of generality, we first reorder workers in each cluster according to their indices such that

$$\mathbf{A}_{cluster}(i,p) < \mathbf{A}_{cluster}(j,p), \quad i < j, \quad p \in [P], \tag{9.16}$$

where $\mathbf{A}_{cluster}(i,p)$ denotes the index of the worker in the ith position in cluster p. For example, in $\mathbf{A}_{cluster}$ given in (9.13), $\mathbf{A}_{cluster}(1,2)$ is 2 since it corresponds to worker w_2. Once its turn comes, each cluster starts selecting workers with the lowest indices first. We note that, if the workers have heterogeneous computing capabilities, then in this step we order workers according to their speed of computation, such that the fastest workers are selected first, which we will consider in Section 9.6.2. For ease of exposition, here, we provide the algorithm when all the straggling workers have identical computation statistics, and similarly all the non-straggling workers have the same computation statistics with each other. Therefore, there is no preference among workers within each group, and ordering them according to their indices is appropriate.

We assume that at the end of each iteration, each worker accurately detects its straggling status and informs the PS using an instantaneous feedback. The straggling state information is in general not available to the worker before that iteration ends due to the unpredictable and highly varying nature of computing resources in distributed computing systems. Since the current straggling behavior is random following the underlying Markov process, at iteration t, the algorithm starts by deducing the sets of non-straggling and straggling workers $\mathcal{K}_f$ and $\mathcal{K}_s$ from $\mathbf{S}^{t-1}$.

We note that, at each iteration, $\mathcal{K}_f \cup \mathcal{K}_s = [K]$. The proposed algorithm uses the straggler statistics from iteration $t-1$ to perform dynamic clustering at iteration t, which makes this algorithm suitable for Markovian straggling models.

9.5.1 Phase I - Worker Placement

We place straggling and non-straggling workers separately to the clusters following a specific order. If the number of non-straggling workers is higher than the stragglers, i.e., $|\mathcal{K}_f| \geq |\mathcal{K}_s|$, we start by placing the non-stragglers and vice-versa.

For the sake of demonstration, we assume $|\mathcal{K}_f| \geq |\mathcal{K}_s|$ and place the non-straggling workers first. Let O_f denote the order in which the clusters select workers such that $O_f(p)$ gives the order in which the pth cluster selects workers. To determine the exact order, we define $\mathcal{K}_f^p$ and $\mathcal{K}_s^p$, which denote the set of non-straggling and straggling nodes that can be assigned to cluster p, respectively. We remark that worker k can be assigned to cluster p if it is in column p of $\mathbf{A}_{cluster}$, i.e., $w_k \in \mathbf{A}_{cluster}(:, p)$. With this, we determine the order vector such that

$$O_f(p) < O_f(\bar{p}) \text{ if } |\mathcal{K}_f^p| < |\mathcal{K}_f^{\bar{p}}| \quad p, \bar{p} \in [P]. \tag{9.17}$$

That is, clusters with less availability select workers first. In the case of equal availability, i.e., $|\mathcal{K}_f^p| = |\mathcal{K}_f^{\bar{p}}|$, cluster with the smaller index selects first, i.e., $O_f(p) < O_f(\bar{p})$ for $p < \bar{p}$. The order for straggler placement O_s is determined accordingly using $\mathcal{K}_s^p$, for $p \in [P]$.

Once the order O_f is determined, non-straggling workers are placed into clus-

ters following O_f. As stated in lines 16-23 of Algorithm 1, once its turn comes, each cluster p with an open spot, i.e., each cluster p that currently has less than ℓ workers, selects the first available non-straggling worker from $\mathbf{A}_{cluster}(:, p)$, $p \in [P]$. Once a non-straggling worker is assigned to a cluster, we remove it from $\mathcal{K}_f$ and $\mathbf{A}_{cluster}$. We note that this assignment continues until there is no unassigned non-straggling worker left in $\mathcal{K}_f$ or a placement conflict is observed. Then, the straggler workers are placed following a similar procedure with the order vector O_s.

During Phase I, the algorithm makes at most M such placement attempts, where $M > 0$ is a sufficiently large number. If after M turns, a worker cannot be assigned to any of the remaining clusters, this indicates a placement conflict and we move on to the second phase of the algorithm.

9.5.2 Phase II - Conflict Resolution

Assume that there is a placement conflict at the end of Phase I such that worker k cannot be placed to the remaining cluster p. That is, all of the n clusters that worker k can be assigned to are full, i.e., already have ℓ workers, and cluster p needs one more worker. In such a case, the second conflict resolution phase of the algorithm starts.

Let $\mathcal{P}_k$ denote the set of possible clusters for worker k such that $|\mathcal{P}_k| = n$. In the conflict resolution step, as stated in lines 26-35 of Algorithm 1, we look for a worker $\bar{k}$, which has been assigned to one of the clusters in $\mathcal{P}_k$ in Phase I such that $w_{\bar{k}} \in \mathbf{A}_{cluster}(:, p)$. That is, even though worker $\bar{k}$ has been assigned to cluster

$\bar{p} \in \mathcal{P}_k$ during Phase I, it can be assigned to cluster p as well. Once we detect first such worker, we swap its position with worker k. That is, we assign worker k, the conflicted worker, to cluster $\bar{p}$ and worker $\bar{k}$ to cluster p, the conflicted cluster.

We note that there might be multiple placement conflicts at the end of Phase I, in which case the conflict resolution step is repeated until all cases are resolved.

To illustrate the proposed worker replacement policy in detail, we consider the cluster assignment matrix in (9.13), and without loss of generality, order workers in an increasing index order in each column to obtain

$$\mathbf{A}_{cluster} = \begin{bmatrix} w_1 & w_1 & w_2 & w_3 \\ w_4 & w_2 & w_3 & w_4 \\ w_6 & w_7 & w_5 & w_5 \\ w_7 & w_8 & w_8 & w_6 \\ w_9 & w_{10} & w_{11} & w_9 \\ w_{10} & w_{11} & w_{12} & w_{12} \end{bmatrix}, \tag{9.18}$$

where the straggling workers are shown in red. The straggler realization for this example is $\mathbf{S} = [1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1]$. Here, we have 5 straggling and 7 non-straggling workers, i.e., $|\mathcal{K}_s| = 5$ and $|\mathcal{K}_f| = 7$.

Since there are more non-straggling workers than stragglers, we place the non-straggling workers first. To determine a non-straggling worker placement order, we find the number of available non-straggling workers in each cluster. One can observe in (9.18) that, cluster 1 and cluster 2 have 4 available non-straggling workers that

can be assigned to these clusters whereas cluster 3 and cluster 4 have 3 available non-straggling workers. That is, we have $|\mathcal{K}_f^1| = |\mathcal{K}_f^2| = 4$ and $|\mathcal{K}_f^3| = |\mathcal{K}_f^4| = 3$. Based on these, we deduce a placement order $O_f = [3, 4, 1, 2]$ such that clusters take turns based on this placement order.[5] At each turn of a particular cluster, a single worker is assigned to that cluster according to the aforementioned greedy policy. In our example, we start with the third cluster and w_2 is assigned to this cluster. Then, the fourth cluster gets w_4 and so on. This process continues until all the non-straggling workers are placed into clusters (or until a placement conflict is observed). If a cluster is assigned $\ell = 3$ workers, we say that cluster is full and do not assign any more workers to that cluster. Next, we determine the placement order of straggling workers in a similar fashion. One can deduce from (9.18) that the order of placement for the stragglers is $O_s = [1, 2, 3, 4]$ as clusters 1 and 2 have the least availability. Based on this order, stragglers are also placed using the greedy policy described above and the first phase terminates with the worker placement shown in Fig. 9.2. Here, we observe a placement conflict as w_{12} has not been assigned to any cluster whereas cluster 1 needs one more worker, but w_{12} cannot be assigned there.

We start the second phase of the proposed worker placement algorithm to place w_{12} into a cluster that has a worker which can be assigned to the first cluster. We see from (9.18) that w_{12} can be assigned to clusters 3 or 4. None of the workers which has been assigned to cluster 3 in Phase I can be assigned to the first cluster. Then, the algorithm looks as cluster 4 and identifies that w_4, which has been assigned to

[5] In a more refined implementation, this order can dynamically change after each round of worker placement, i.e., after all clusters select one worker, to better reflect the clusters with less availability as worker placement continues.

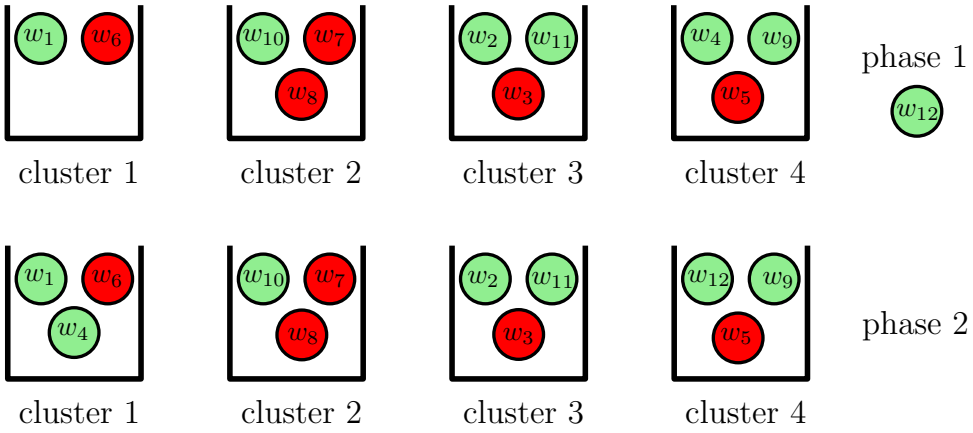


Figure 9.2: The proposed worker placement strategy.

the fourth cluster in the first phase, can go to the first cluster. With this, we swap workers w_4 and w_{12}, which yields the final placement in Fig. 9.2.

At the end of the algorithm we see that the stragglers are placed into the clusters as uniformly as possible: cluster 2 has two stragglers while the remaining clusters have only 1 straggler each. We note that since we have only 7 non-straggling workers, less than the worst case scenario of $P(\ell - r + 1) = 8$ non-stragglers, the full recovery is not possible for the static clustering scheme. Thus, the proposed dynamic clustering scheme does not improve the worst case scenario. Rather, it speeds up the GC scheme by uniformly placing the stragglers across clusters. This process is repeated at each iteration to dynamically change the clusters based on the straggler observations.

We note that at the end of the first phase, there are 4 other workers, namely workers w_4, w_7, w_9, and w_{10}, that can be placed into the first cluster, which had placement conflict at the end of Phase I of the algorithm. Even if $\ell = 3$ of them would have been assigned to cluster 2, which worker w_{12} cannot be assigned, the remaining one of them still would have been assigned to either cluster 3 or 4. Thus,

it is guaranteed that cluster 3 and cluster 4 have at least one worker that can be assigned to cluster 1 so that the placement conflict can be resolved. The next lemma formally states this guarantee.

Lemma 9.1 *Assume that we have a conflicted worker k which cannot be assigned to the remaining cluster p in Phase I. Then, if*

$$n > \frac{P(K-1)}{2K}, \tag{9.19}$$

it is guaranteed that at least one worker in one of the clusters in $\mathcal{P}_k$ can be assigned to cluster p so that the placement conflict can be resolved.

Proof: In the proof we consider the worst case scenario such that $\ell - 1$ workers have already been assigned to cluster p in Phase I. Thus, in the remaining $P-1$ clusters other than cluster p, there are $n\ell - \ell + 1$ workers that can be assigned to cluster p. We want to make sure that, at the end of Phase I of the algorithm, at least one of those workers is assigned to a cluster in set $\mathcal{P}_k$, which, as previously stated, denotes the set of clusters that worker k, the conflicted worker, can be assigned to. Except cluster p, there are $P - n - 1$ clusters that worker k cannot be assigned to. These $P - n - 1$ clusters can at most have $(P - n - 1)\ell$ workers after Phase I. Thus, as long as $n\ell - \ell + 1 > (P - n - 1)\ell$, there is at least one worker that can be assigned to cluster p in one of the clusters in $\mathcal{P}_k$, which yields (9.19) since $\ell = \frac{K}{P}$. ■

In the previous example, (9.19) is satisfied since $K = 12$, $P = 4$, and $n = 2$ such that $n > \frac{11}{6}$.

In the next section, we analyze the performance of this dynamic clustering strategy through numerical simulations.

9.6 Numerical Results

In this section, we provide numerical results comparing the proposed GC-DC scheme with GC-SC as well as the original GC scheme using a model-based scenario for computation latencies.[6] For the simulations, we consider a linear regression problem over synthetically created training and test datasets, as in [188], of sizes 2000 and 400, respectively. We set the size of the model to $d = 1000$. A single simulation consists of $T = 400$ iterations. For all the simulations, we use learning rate $\eta = 0.1$. To model the computation delays at the workers, we adopt the commonly used shifted exponential model [215], and assume that the probability of completing r partial gradient computations at worker k by time t is given by

$$\mathbb{P}[X_{k,r} \leq t] \triangleq \begin{cases} 1 - e^{-\mu_k(\frac{t}{r} - \alpha_k)}, & \text{if } t \geq r\alpha_k, \\ 0, & \text{otherwise,} \end{cases} \tag{9.20}$$

where $\alpha_k > 0$ is a constant shift indicating that a single computation duration cannot be smaller than α_k and $\mu_k > 0$ denotes the straggling effect. We consider two different models for the time-correlated straggling behavior: the homogeneous and heterogeneous worker models, which we discuss next.

[6] The fractional repetition scheme can also be used for GC in our work in addition to the cyclic GC we use as a baseline. However, the proposed cyclic GC scheme is preferred as it does not impose any constraint on the (K, r) pairs.

9.6.1 Gilbert-Elliot Model with Homogeneous Workers

We model the straggling behavior of the workers based on a two-state Markov chain: a slow state s and a fast state f, such that computations are completed faster when a worker is in state f. Specifically, in (9.20) we have rate μ_f in state f and rate μ_s in state s, where $\mu_f > \mu_s$ as in Chapters 8 and 10. That is, each worker has two possible rates based on its straggling statistics. We assume that the state transitions only occur at the beginning of each iteration with probability p; that is, with probability $1-p$ the state remains the same. A low switching probability p indicates that the straggling behavior tends to remain the same in consecutive iterations with occasional transitions. We set $p = 0.05$ or $p = 0.2$, $\alpha = 0.01$, $\mu_s = 0.1$, and $\mu_f = 10$. We assume that the transition probability p along with the computation rates μ_s and μ_f are known to the PS. At the end of each iteration, workers inform the PS regarding their straggling status before the next iteration starts. With this information along with the knowledge of transition probability p, the PS performs the dynamic clustering accordingly. For example, when $p \leq 0.5$, the PS assumes that each worker will continue with the same straggling behavior from the past iteration.[7]

In the first simulation, we set $K = 20$, $P = 5$, $r = 3$, and $n = 3$. We start with 10 stragglers initially. In Fig. 9.3, we plot the average per-iteration completion time of the original GC scheme, GC scheme with static clustering (GC-SC), GC scheme with the proposed dynamic clustering (GC-DC), and a lower bound, denoted by LB.

[7]After a sufficiently long observation period, the PS can accurately estimate the transition probability p as it is the same for all the workers and iterations.

Here, the lower bound is obtained by assuming that the full gradient is recovered as soon as the earliest $P \times (\ell - r + 1)$ workers finish their computations at each iteration, independently of the codeword assignment matrix. We remark that this lower bound is rather an idealistic scenario as it requires the perfect knowledge of computation times at each iteration as well as $n = P$, i.e., all workers can be assigned to all the clusters. We observe in Fig. 9.3(a) that clustering schemes significantly improve the performance compared to the original GC scheme. The best performance is achieved when the dynamic clustering, the GC-DC scheme, is implemented and the performance improvement compared to the GC-SC scheme is approximately 34%.

In Fig. 9.3(a), we have considered the case in which the PS does not know the exact straggler realization at the beginning of an iteration, and uses previous observation to implement the dynamic clustering strategy. In the second simulation in Fig. 9.3(b), we consider the same scenario as in the first simulation, but assume that the PS knows the exact straggler realization at the beginning of each iteration, which we call perfect straggler state information (SSI). That is, in the case of perfect SSI, the PS knows exactly which workers will straggle in the current iteration, and therefore, the proposed dynamic clustering algorithm does not suffer from transitions in the straggling behavior from one iteration to the next. In this case we see similar trends as in Fig. 9.3(a), but observe that the GC-DC scheme results in a larger improvement in the average per-iteration completion time (around 45%) than that of the imperfect SSI case.

Finally, in Fig. 9.4 we consider a case in which the straggler state transitions occur more frequently and set $p = 0.2$. We see in Fig. 9.4(a) that under imperfect

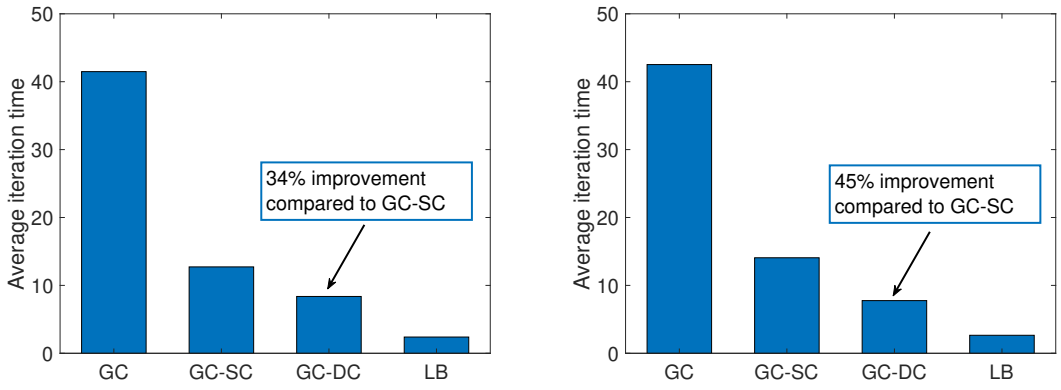


Figure 9.3: Average per-iteration completion time under the Gilbert-Elliot model with homogeneous workers for $K = 20$, $P = 5$, $r = 3$, $n = 3$, and $p = 0.05$ (a) under imperfect SSI, (b) under perfect SSI.

SSI, with a larger p value, GC-DC still performs the best, but the improvement over GC-SC is less compared to Fig. 9.3(a) when $p = 0.05$. On the other hand, under perfect SSI, i.e., the PS knows the exact straggler realization at the beginning of each iteration, the effect of increased p is not observed and we have approximately 45% improvement over GC-SC as in Fig. 9.3(b).

9.6.2 Heterogeneous Worker Model

In this model, we assume that workers have different computation rates μ_k, $k \in [K]$. In this case, we specify a straggling threshold $\tau > 0$, and a worker k is treated as a straggler if $\mu_k < \tau$.

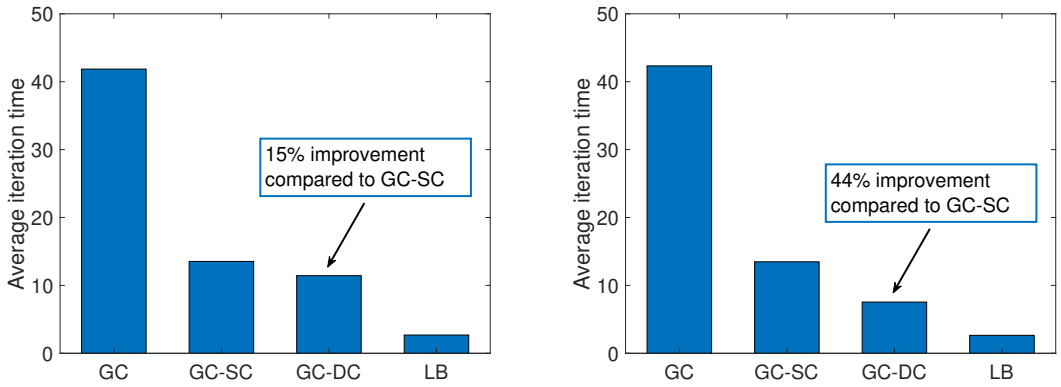


Figure 9.4: Average per-iteration completion time under the Gilbert-Elliot model with homogeneous workers for $K = 20$, $P = 5$, $r = 3$, $n = 3$, and $p = 0.2$ (a) under imperfect SSI, (b) under perfect SSI.

9.6.2.1 Gilbert-Elliot Model with Heterogeneous Workers

We study the case in which each worker's straggling behavior is modeled by a two-state Markov chain such that $\mu_k = \mu_{k,f}$ if worker k is not straggling and $\mu_k = \mu_{k,s}$ if worker k is a straggler. At the beginning of each iteration, a worker's straggling mode switches with probability p. Here, first we sample the non-straggling computation rates of each worker $\mu_{k,f}$ uniformly at random from the interval $[0, 5]$ and set $\alpha_k = 0.01$, $p = 0.05$ or $p = 0.2$, for $k \in [K]$. We model the straggling computation rates of workers $\mu_{k,s}$ such that for worker k we have $\mu_{k,s} = \frac{\mu_{k,f}}{10}$, $k \in [K]$. That is, in the straggling mode, each worker is $10\times$ slower than its typical non-straggling performance, which is motivated by the measurements taken over Amazon EC2 clusters that indicate a similar performance drop in the straggling mode [169]. With this, computation rates of the workers in the straggling mode are uniformly distributed

in $[0, 0.5]$. We assume that the non-straggling computation rates $\mu_{k,f}$ are known to the PS for $k \in [K]$ after a certain number of iterations and from these, the PS can deduce the straggling computation rates $\mu_{k,s}$.

Equipped with these, after each iteration, the PS is informed about the straggling status of each worker and performs the proposed greedy dynamic clustering scheme with a modification as follows: Instead of ordering the workers according to (9.16), we order them according to their rates μ_k, $k \in [K]$. In this case, once its turn comes, each cluster selects the fastest available worker first rather than selecting the one with the smallest index first.

We note that since the computation rates are sampled randomly, a worker's straggling computation rate can still be higher than another worker's non-straggling rate. To account for these scenarios, we set the straggling threshold $\tau = 0.5$. That is, as long as a worker's rate is below 0.5 we treat that worker as a straggler. We did not utilize such a threshold in the homogeneous worker model since in that case workers have identical computation rates μ_f and μ_s in the non-straggling and straggling states, respectively, such that $\mu_s < \mu_f$.

Simulation results for this setup are provided in Fig. 9.5. We average the results over 30 independent simulations for a fixed $\mathbf{A}_{cluster}$ that is generated according to the procedure described in Section 9.4.2. We observe in Figs 9.5(a) and 9.5(b) for $p = 0.05$, and in Figs 9.6(a) and 9.6(b) for $p = 0.2$ that the GC-DC scheme outperforms the static clustering schemes, namely GC and GC-SC. The performance improvement is larger in the case of perfect SSI and when $p = 0.05$.

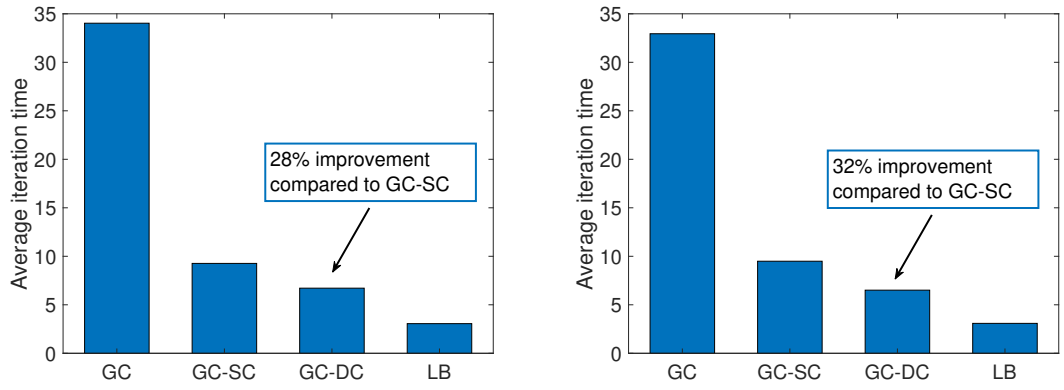


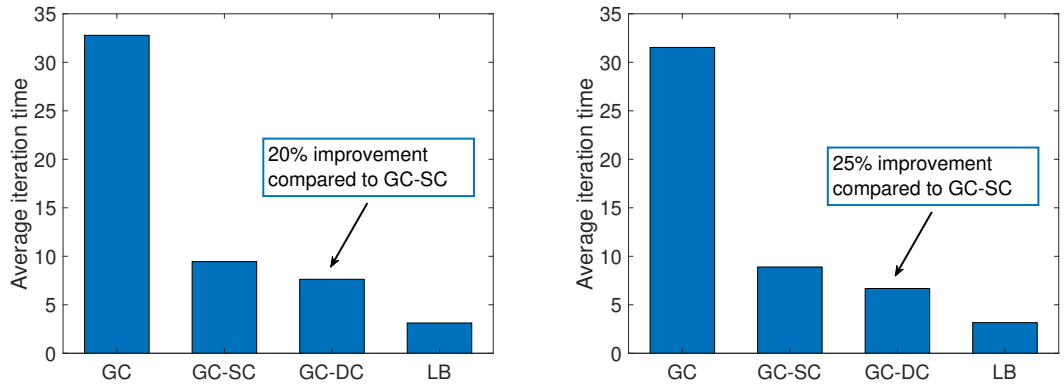
Figure 9.5: Average per-iteration completion time under the Gilbert-Elliot model with heterogeneous workers for $K = 20$, $P = 5$, $r = 3$, $n = 3$, $p = 0.05$, and $\tau = 0.5$ (a) under imperfect SSI, (b) under perfect SSI.

9.6.2.2 Heterogeneous Workers with Time-Varying Rates

So far, we have modeled the straggling behavior based on a Gilbert-Elliot mode. In this subsection, instead of a two-state Markov chain model, we consider that the straggling parameters of the workers are time-varying. We assume that each worker samples its rate uniformly at random from the interval $[0, 5]$ and set $\alpha_k = 0.01$ for all $k \in [K]$. We assume that at the beginning of each iteration, each worker re-samples its rate with probability p such that with probability $1 - p$ its rate stays the same. That is, we have

$$\mu_{k,t+1} = (1 - a_{t+1})\mu_{k,t} + a_{t+1} \cdot U[0, 5], \tag{9.21}$$

where, $\mu_{k,t}$ denotes the rate of worker k at iteration t, a_t is an i.i.d. Bernoulli(p) random variable, i.e., $\mathbb{P}(a_t = 1) = p, \forall t$, and $U[a, b]$ denotes a uniform random



(a) (b)

Figure 9.6: Average per-iteration completion time under the Gilbert-Elliot model with heterogeneous workers for $K = 20$, $P = 5$, $r = 3$, $n = 3$, $p = 0.2$, and $\tau = 0.5$ (a) under imperfect SSI, (b) under perfect SSI.

variable over interval $[a, b]$. In simulations, we use the scenario in the Fig. 9.3 and start with 10 stragglers. We initialize the rates of stragglers with $\mu_{k,0} = U[0, \tau)$ and rates of non-straggling workers with $\mu_{k,0} = U[\tau, 5]$. In this setup, we set $p = 0.05$ or $p = 0.2$.

Since the computation capabilities of the workers are not identical, we apply the proposed greedy dynamic clustering scheme with the same modification as above. We note that this model requires the workers to accurately detect their computation rates at the end of each iteration and send them to the PS before the next iteration starts.

First, we consider the case in which $\tau = 1$. In this case, we observe in Figs. 9.7(a) and (b) that the GC-DC scheme outperforms the GC and GC-SC schemes but the improvement compared to the GC-SC scheme is not significant. In fact, we see that in the case of perfect SSI the improvement is around 20% com-

pared to the GC-SC scheme whereas when the straggler realizations are not known to the PS in advance this improvement drops to approximately 16%.

Next, we set $\tau = 0.1$ such that the proposed greedy dynamic clustering scheme specifically targets the slowest workers and carefully places them across clusters. In Figs. 9.8(a) and (b), we observe for $p = 0.05$ that the GC-DC scheme performs the best and the improvement compared to the GC-SC scheme is more significant. We also note that in Fig. 9.8, the performance improvement is larger but the average iteration times are also larger for all three schemes compared to the case in Fig. 9.7. This is because when $\tau = 0.1$, we initialize the rates of the workers considering $10\times$ slower stragglers compared to when $\tau = 1$. We set $p = 0.2$ in Fig. 9.9, and observe that the GC-DC scheme still performs the best compared to the static GC schemes even though the improvement in performance decreases as the transitions occur more frequently compared to the case in which $p = 0.05$. We finally note that all the simulation results given in Figs. 9.7, 9.8, and 9.9 are averaged over 30 independent simulations for a fixed $\mathbf{A}_{cluster}$ that is generated according to the procedure described in Section 9.4.2.

9.6.3 Simulations for Shared Access Scenario

In general, the concept of a straggler refers to a state in which a worker either does not respond at all or responds with a certain delay. The delay mentioned here might be due to several factors, such as the internal delay of the processing units, communication delay due to possible link failures, or due to overloading of

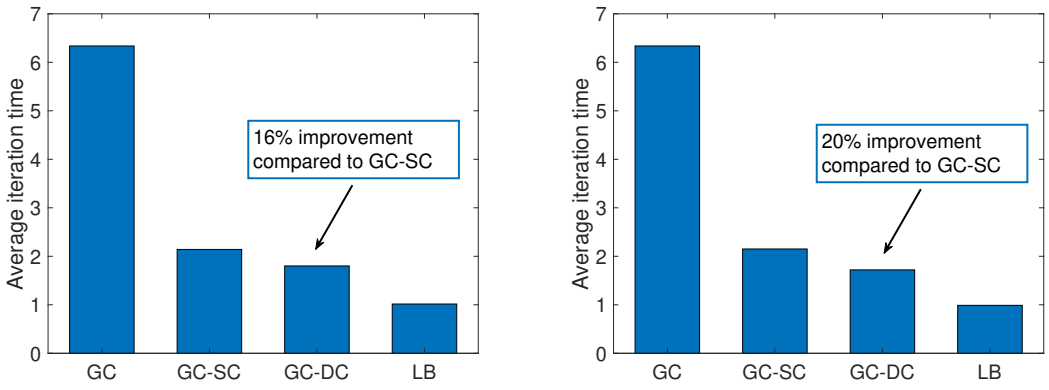


Figure 9.7: Average per-iteration completion time under the heterogeneous worker model with time-varying rates for $K = 20$, $P = 5$, $r = 3$, $n = 3$, $p = 0.05$, and $\tau = 1$ (a) under imperfect SSI, (b) under perfect SSI.

the workers. The latter is often observed when the computational resources are open to access from multiple users without any central entity to regulate the user requests or to perform resource allocation. In such cases, workers with excessive user requests might be identified as stragglers due to their long response time. To this end, we perform experiments to analyze the response time of the workers with respect to the number of ongoing computational requests. The experiments are conducted in our clusters where GeForce RTX 2080 Ti Graphics Cards with CUDA toolkit 11.0 are employed as workers. In the experiments, we consider training of ResNet-20 architecture for classification on CIFAR-10 dataset with a batch size of 64 as the computational task. To measure the impact of the workload on the workers, we generate multiple users with the same computational task and measure the latency for a single iteration with SGD framework implemented with PyTorch. Consequently, we observe that, when there is a single request, the completion time

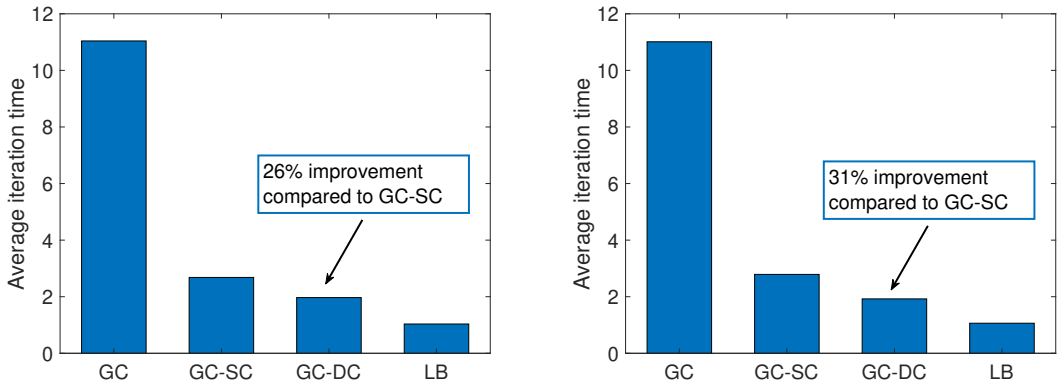
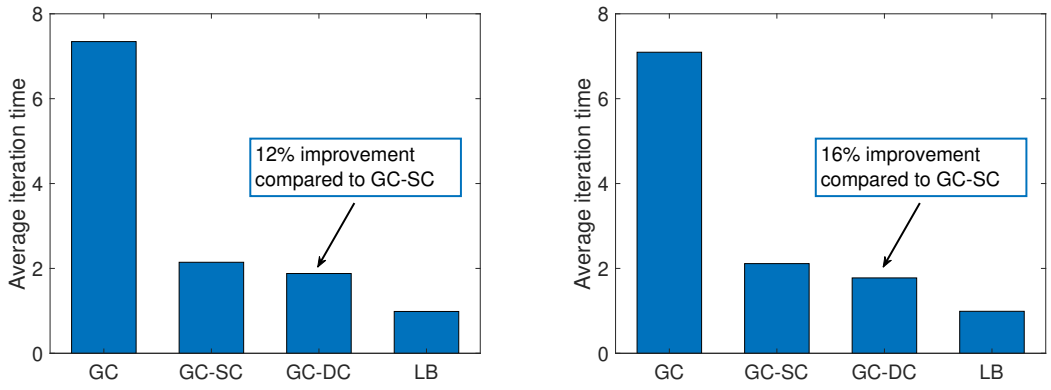


Figure 9.8: Average per-iteration completion time under the heterogeneous worker model with time-varying rates for $K = 20$, $P = 5$, $r = 3$, $n = 3$, $p = 0.05$, and $\tau = 0.1$ (a) under imperfect SSI, (b) under perfect SSI.

of a single iteration is 20 milliseconds (ms), however, this latency increases linearly with the number of ongoing tasks. Based on this observation, we simulate a scenario in which the users arrive at the system according to a certain probabilistic model and stay in the system for a random period of time depending on the complexity of the task.

Here, we assume that requests arrive at the workers according to a Poisson distribution with rate $\lambda = 0.1$. Each arriving task stays in the system for a number of iterations that is sampled uniformly at random from the interval $[2, 5]$. We assume that each worker can serve at most $C = 10$ users at a time. This means that, for a single worker, a single iteration may take between 20 ms and 200 ms based on our measurements. We denote the straggling threshold in this case by γ and set $\gamma = 7$. That is, a worker is considered straggling if it has 7 or more ongoing task computations. Simulations results for this setup are provided in Fig. 9.10. These



(a) (b)

Figure 9.9: Average per-iteration completion time under the heterogeneous worker model with time-varying rates for $K = 20$, $P = 5$, $r = 3$, $n = 3$, $p = 0.2$, and $\tau = 0.1$ (a) under imperfect SSI, (b) under perfect SSI.

results are averaged over 30 independent simulations for a fixed $\mathbf{A}_{cluster}$. In Fig. 9.10, we observe that the proposed GC-DC scheme outperforms the static GC schemes under this more practical setup. The improvement is more visible in the case of perfect SSI, i.e., when the PS knows the number of ongoing computations at each worker. In this setup, the performance of the GC-DC scheme is much closer to LB compared to the results in Section 9.6. Especially in the perfect SSI case, the gap is less than 10%, which indicates that the proposed GC-DC algorithm promises to improve the performance in more practical shared access scenarios over computing clusters.

9.7 Conclusion

In this chapter, we considered coded computing for large-scale distributed learning problems in the presence of straggling workers, and introduced a novel scheme,

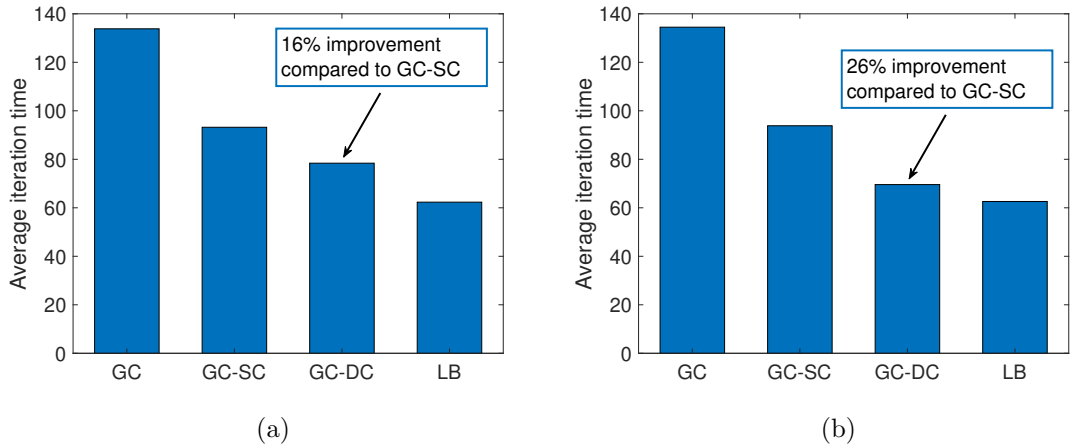


Figure 9.10: Average per-iteration completion time under the shared access model for $K = 20$, $P = 5$, $r = 3$, $n = 3$, $\lambda = 0.1$, $\gamma = 7$, $C = 10$ (a) under imperfect SSI, (b) under perfect SSI.

called GC-DC, to reduce the average per-iteration completion time of the static GC schemes. GC-DC employs the GC scheme with clustering introduced in [166], and assigns additional data to the workers without increasing the per-iteration computation load at each worker compared to the original GC scheme. By utilizing the extra degree-of-freedom offered by additional data, but without increasing the computation load at each iteration, the proposed GC-DC scheme dynamically assigns workers to different clusters at each iteration, in order to distribute the stragglers to clusters as uniformly as possible. Under a time-correlated straggler model, GC-DC can improve the overall computation speed by dynamically adapting to the straggling behavior. We showed through numerical simulations, for both homogeneous and heterogeneous worker models, that the proposed GC-DC scheme can drastically improve the average per-iteration completion time without an increase in the communication load.

Algorithm 1 Proposed dynamic clustering strategy

1: Given $\mathbf{A}_{cluster}$, K, P, n, $\mathbf{S}^0$ such that w.l.o.g. $\mathbf{A}_{cluster}(i,p) < \mathbf{A}_{cluster}(j,p)$ for $i < j$, $p \in [P]$
2: **for** $t = 1, \ldots, T$ **do**
3: Observe $\mathbf{S}^{t-1}$ and deduce $\mathcal{K}_f$ and $\mathcal{K}_s$, i.e., sets of non-straggling and straggling workers in iteration $t-1$
4: **Phase I:**
5: Place workers to clusters following an order
6: **if** $|\mathcal{K}_f| \geq |\mathcal{K}_s|$ **then**
7: Place non-stragglers first
8: **else**
9: Place stragglers first
10: **Phase II:**
11: Conflict resolution in the case of an assignment problem in Phase I
12: **Order determination:**
13: $O_f(p) < O_f(\bar{p})$ if $|\mathcal{K}_f^p| < |\mathcal{K}_f^{\bar{p}}|$ or $(|\mathcal{K}_f^p| = |\mathcal{K}_f^{\bar{p}}|$ and $p < \bar{p})$ for $p, \bar{p} \in [P]$
14: Use O_s in the case of straggler placement with $\mathcal{K}_s^p$ for $p \in [P]$
15: **Non-straggler placement:**
16: $i = 1$
17: **while** $|\mathcal{K}_f| > 0$ and $i < M$ **do**
18: $j =$ mod (i, P) with $j \leftarrow P$ when mod $(i, P) = 0$
19: Cluster to assign is $\bar{p}$ such that $O_f(\bar{p}) = j$
20: **if** $size($cluster $\bar{p}) < \ell$ **then**
21: Assign the first non-straggling worker from $\mathbf{A}_{cluster}(:, \bar{p})$ to cluster $\bar{p}$
22: Remove the assigned worker from $\mathcal{K}_f$ and $\mathbf{A}_{cluster}$
23: $i = i + 1$
24: **Straggler placement:**
25: Follow steps 16-23 using $\mathcal{K}_s$ and O_s
26: **Conflict resolution:**
27: Given a conflicted worker k and corresponding conflicted cluster p
28: Identify the clusters $\mathcal{P}_k$ that worker k can be assigned to such that $|\mathcal{P}_k| = n$
29: $i = 1$
30: **while** Worker k is not assigned to any cluster **do**
31: Select cluster $\bar{p}$ such that $\bar{p} = \mathcal{P}_k(i)$
32: **if** There is a worker $\bar{k}$ in cluster $\bar{p}$ such that $w_{\bar{k}} \in \mathbf{A}_{cluster}(:, p)$ **then**
33: Assign worker $\bar{k}$ to cluster p
34: Assign worker k to cluster $\bar{p}$
35: $i = i + 1$

CHAPTER 10

Age of Information with Gilbert-Elliot Servers and Samplers

10.1 Introduction

In most of the existing works in the age of information literature, there is an underlying i.i.d. structure in the system. The service times and packet interarrivals are i.i.d. processes and the focus is on analyzing and optimizing the resulting age of information. There may be scenarios in which these processes are correlated over time or over different status update packets. Reference [40] models transmission times as a stationary and ergodic Markov chain to analyze the effect of the temporal correlation between transmission times on age-optimal scheduling. References [220, 221] study information freshness over Markovian channels. Specifically, reference [220] models the channel using a Gilbert-Elliot model and introduces the concept of channel information age. This metric is used to express the utility and analyze the effect of aging on the probability of error in estimating the channel state. Reference [47] studies the freshness over a network with a Markov source and proposes an effective age metric which captures estimation error as well as timeliness. Reference [24] studies age of information with a two-state Markov modulated service process and

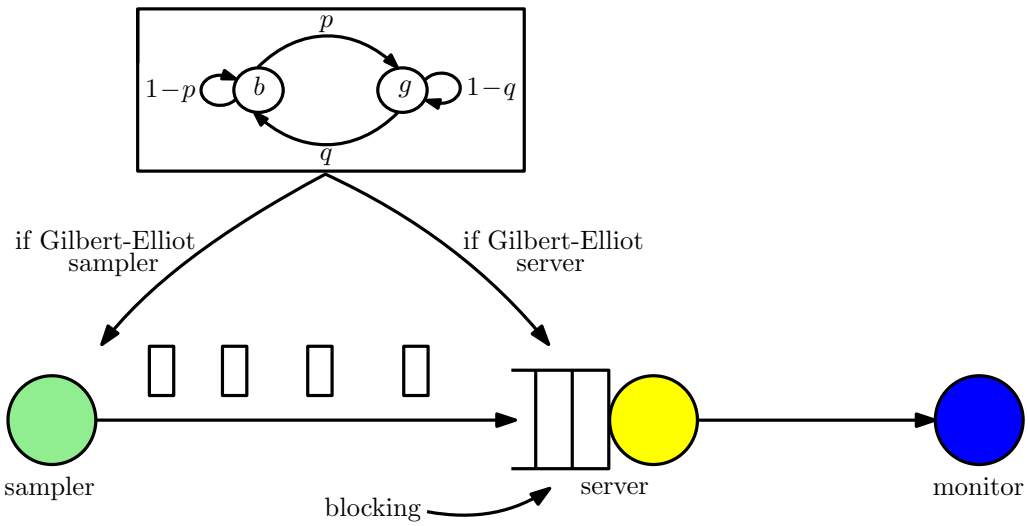


Figure 10.1: A single sampler sends time-sensitive status updates to a single monitor node through a server node. We consider Gilbert-Elliot server and Gilbert-Elliot sampler settings.

characterizes the average age for an FCFS operation under infinite and zero buffer size settings.

In this chapter, we consider a status updating system in which there is a single sampler which takes samples from an observed phenomenon and sends them to an interested monitor node in the form of status update packets through a single server node (see Fig. 10.1). We study age of information with Gilbert-Elliot servers and samplers under blocking packet management policy at the server node. We first analyze the case in which the service times follow a finite state Markov chain with two states: *bad* state b and *good* state g such that in state g, the service performance is faster than that in state b. The motivation for studying this kind of a service profile comes from the measurements over Amazon EC2 clusters which show high variability in computing speeds of the servers over time [169]. These measurements indicate that when a server is in a certain state it tends to stay in that

state in the next rounds of the computation which implies a dependence in service times over time. In addition, [222] shows that the channel quality and reliability in cooperative driving follows a Markov-modulated process which depends on the number of interfering vehicles thereby affecting the service performance. Further, in sensor networking applications, energy constraints on servers may prevent them from operating in faster states all the time, and may force them to switch between faster and slower states.

We derive the time average age under this Markovian service profile and characterize the age-optimal state transition matrix of the underlying Markov chain first without considering any constraints on the operation of the system. Next, we consider a more realistic scenario in which each state has an operational cost and the system is subject to an overall budget.

Next, we consider the case in which the sampler, i.e., the source node, follows a Gilbert-Elliot model based operation. The sampler takes samples based on a two-state Markov chain, as in Fig. 10.1. When in state g, it samples the observed phenomenon more frequently whereas in state b, samples are taken more sparsely. Thus, under this operation, interarrivals to the server node are no longer i.i.d. but follow a two-state Markov chain. This non i.i.d. sampling operation is particularly relevant when the sampler's operation cost is considered as the sampler may not be able to afford taking frequent samples all the time and may switch to a low-cost operation to save energy. Further, the sampler may choose to sample the process more often when the observed process varies above a certain threshold or varies too fast. We characterize the average age under such Gilbert-Elliot sampling and find

the age-optimal state transition matrix P.

10.2 System Model and Age Metric

We consider a communication system (see Fig. 10.1), where there is a single sampler that takes samples from an observed phenomenon at random and immediately transmits these samples to a monitor node through a single server node in the form of status update packets. The server node implements a blocking policy in which an arriving update packet goes directly into service only if the server is idle. Update packets arriving when the server node is busy are discarded.

Unlike most of the literature, we consider non i.i.d. service and interarrival profiles. Here, we use a simple Gilbert-Elliot model to introduce a non i.i.d. structure to the system. We consider two scenarios: Gilbert-Elliot service times and i.i.d. interarrival times; and i.i.d. service times and Gilbert-Elliot interarrival times.

When the server follows a Gilbert-Elliot model, service times S follow a two state Markov chain with states *bad* (b) and *good* (g) such that in state b, the server node is slower and the service takes longer than the service in state g. We model the service times with exponential random variables S_b with rate μ_b in state b and S_g with rate μ_g in state g where $\mu_g > \mu_b$ as in [24]. In this case, we model the update arrivals at the server node as a Poisson process with rate λ. We adopt an event-triggered Markov chain in which the state change only occurs when a new packet enters service. Thus, during the service of an update packet, service performance remains the same. The transition probability from state b to state g is p and the

transition probability from state g to state b is q where $p \in (0,1)$ and $q \in (0,1)$ (see Fig. 10.1). State transition matrix P of this Markov chain is

$$P = \begin{bmatrix} 1-p & p \\ q & 1-q \end{bmatrix}. \tag{10.1}$$

We note that this Markov chain is irreducible, aperiodic, and positive recurrent. Thus, service times constitute an ergodic Markov chain with a stationary distribution,

$$P(S = S_b) = \frac{q}{p+q}, \qquad P(S = S_g) = \frac{p}{p+q}, \tag{10.2}$$

where S is the service time of a packet that enters service.

When the sampler follows a Gilbert-Elliot model, this time, update interarrivals at the server node constitute a Markov chain with the state transition matrix in (10.1) where in state g interarrival times are exponential random variables with rate λ_g and in state b interarrival times are exponential random variables with rate λ_b where $\lambda_g > \lambda_b$ to reflect the increased sampling frequency in state g. The Markov chain is again event-triggered such that the sampler's state changes whenever an update packet enters service at the server node.

To quantify the timeliness in the system, we use the age of information metric. At time t age at the monitor node is the random process $\Delta(t) = t - u(t)$ where $u(t)$ is the time-stamp of the most recent update at the destination node. The metric

we use, time averaged age, is given by

$$\Delta = \lim_{\tau \to \infty} \frac{1}{\tau} \int_0^{\tau} \Delta(t) dt, \tag{10.3}$$

where $\Delta(t)$ is the instantaneous age at the monitor node.

In the next section, we derive an average age expression for the cases of Gilbert-Elliot servers and samplers.

10.3 Average Age Analysis

The sampler generates status update packets and immediately sends them through a delay-free link to the server node. Since a blocking policy is implemented, only the packets that find the server idle go into service. We denote such packets that enter service at the server node as successful packets. Let T_{j-1} denote the time at which the jth successful update packet is generated at the sampler. Since newly generated packets are assumed to be instantaneously available to the server node, T_{j-1} also marks the time at which the jth successful update packet arrives at the server node. Random variable Y denotes the update cycle at the server node, the time in between two successful arrivals, where $Y_j = T_j - T_{j-1}$.

Update cycle Y_j consists of service time S_j and idle waiting time Z_j as the server needs to wait for the next arrival upon an update delivery to the monitor node such that

$$Y_j = S_j + Z_j. \tag{10.4}$$

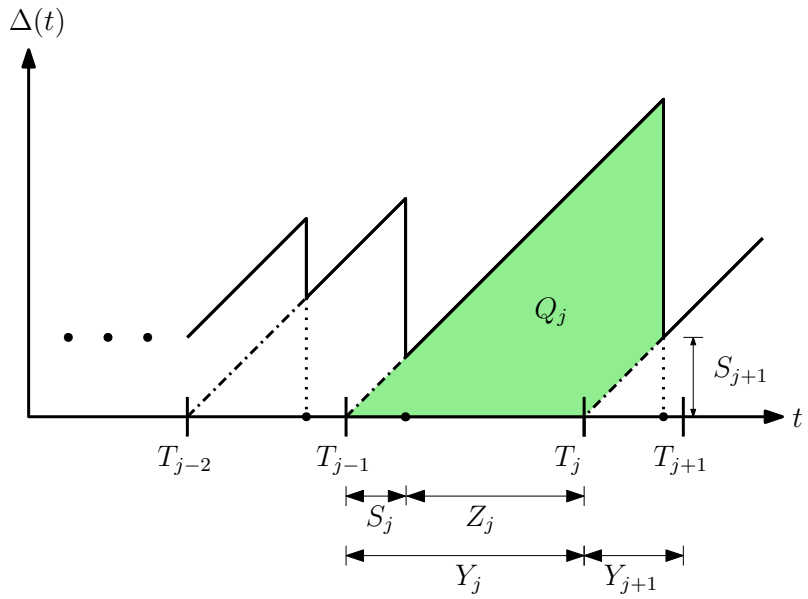


Figure 10.2: Sample age evolution $\Delta(t)$ at the monitor node. Successful updates are indexed by j. The jth successful update arrives at the server node at T_{j-1}. Update cycle at the server node is the time in between two successive arrivals and is equal to $Y_j = S_j + Z_j = T_j - T_{j-1}$.

We note that S_j and Z_j are mutually independent as the arrival and service processes are independent. Sample age evolution at the destination node is given in Fig. 10.2. Here, Q_j denotes the area under the instantaneous age curve in update cycle j and Y_j denotes the length of the jth update cycle as defined earlier. The metric we use, long term average age, is the average area under the age curve which is given [1, 7] by

$$\Delta = \limsup_{n\to\infty} \frac{\frac{1}{n}\sum_{j=1}^{n} Q_j}{\frac{1}{n}\sum_{j=1}^{n} Y_j}. \tag{10.5}$$

In the next two subsections, we find the average age for Gilbert-Elliot service times and interarrival times, respectively.

10.3.1 Gilbert-Elliot Service Times and I.i.d. Interarrival Times

Status update packets arrive at the server node as a Poisson process with rate λ. Thus, update packet interarrivals at the server node are i.i.d. exponential random variables with rate λ. Due to the memoryless property of the update arrivals at the server node, Z is also exponentially distributed with rate λ. Service times S follow a two-state Markov chain. Thus, two consecutive service times are dependent through the state transition matrix P given in (10.1).

Conditioned on S_j and S_{j+1}, the jth update cycle and the area under the age curve in this cycle are characterized by

$$(Y_j, Q_j) = \begin{cases} (Y_b, Q_{bb}), & \text{if } S_j = S_b, \quad S_{j+1} = S_b \\ (Y_b, Q_{bg}), & \text{if } S_j = S_b, \quad S_{j+1} = S_g \\ (Y_g, Q_{gb}), & \text{if } S_j = S_g, \quad S_{j+1} = S_b \\ (Y_g, Q_{gg}), & \text{if } S_j = S_g, \quad S_{j+1} = S_g. \end{cases} \tag{10.6}$$

We note that when the service times and interarrival times are i.i.d., we have a renewal process with inter-renewal time equal to the update cycle Y. However, in our model, update cycles Y_j do not form an i.i.d. sequence unlike the prior models considered in the literature. Rather, each (Y_j, Q_j) is characterized as in (10.6) depending on the state of the service time in update cycles j and $j+1$. Since the underlying Markov chain is stationary and ergodic, we have $(Y_j, Q_j) \sim (Y, Q)$ over

all update cycles and (10.5) reduces to

$$\Delta = \frac{\mathbb{E}[Q]}{\mathbb{E}[Y]} = \frac{q\,\mathbb{E}[Q_b] + p\,\mathbb{E}[Q_g]}{q\,\mathbb{E}[Y_b] + p\,\mathbb{E}[Y_g]}, \tag{10.7}$$

where the first equality follows from [40, Appendix A] and the second equality follows from the law of total probability where we define $\mathbb{E}[Q_b] = (1-p)\,\mathbb{E}[Q_{bb}] + p\,\mathbb{E}[Q_{bg}]$ and $\mathbb{E}[Q_g] = q\,\mathbb{E}[Q_{gb}] + (1-q)\,\mathbb{E}[Q_{gg}]$. From this, along with Fig. 10.2, we get

$$\mathbb{E}[Q_b] = \frac{1}{2}\,\mathbb{E}[(S_b + Z)^2] + (\mathbb{E}[S_b] + \mathbb{E}[Z])\,\mathbb{E}[\bar{S}], \tag{10.8}$$

$$\mathbb{E}[Q_g] = \frac{1}{2}\,\mathbb{E}[(S_g + Z)^2] + (\mathbb{E}[S_g] + \mathbb{E}[Z])\,\mathbb{E}[\bar{\bar{S}}], \tag{10.9}$$

where

$$\mathbb{E}[\bar{S}] = p\,\mathbb{E}[S_g] + (1-p)\,\mathbb{E}[S_b], \tag{10.10}$$

$$\mathbb{E}[\bar{\bar{S}}] = q\,\mathbb{E}[S_b] + (1-q)\,\mathbb{E}[S_g]. \tag{10.11}$$

In addition, from (10.6) we have,

$$\mathbb{E}[Y_b] = \mathbb{E}[S_b] + \mathbb{E}[Z], \tag{10.12}$$

$$\mathbb{E}[Y_g] = \mathbb{E}[S_g] + \mathbb{E}[Z], \tag{10.13}$$

since $Y = S + Z$ as defined earlier.

Substituting (10.8)-(10.13) in (10.7) yields the average age expression under Gilbert-Elliot service times and i.i.d. interarrival times. We note that the numerator

of (10.7) has pq terms whereas the denominator of (10.7) is linear in p and q.

We note that if the server operates only in state g without switching to state b, the average age given in (10.7) becomes

$$\Delta_g = \frac{\mathbb{E}[Q_g]}{\mathbb{E}[Y_g]} = \frac{1}{\lambda} + \frac{2}{\mu_g} - \frac{1}{\lambda + \mu_g}, \tag{10.14}$$

which is the result derived in [15] for an M/M/1 queue with blocking. A similar average age, Δ_b, is achieved if the server node only operates in state b.

10.3.2 Gilbert-Elliot Interarrival Times and I.i.d. Service Times

Service times at the server node are i.i.d. exponential random variables with rate μ whereas the interarrival times follow a two-state Markov chain characterized by the state transition matrix in (10.1). State changes occur upon successful entry to the server node. Let Z_g denote the waiting time until the next arrival when the interarrival state is g, and let Z_b denote the waiting time when the interarrival state is b. Thus,

$$(Y_j, Q_j) = \begin{cases} (Y_b, Q_b), & \text{if } Z_j = Z_b \\ (Y_g, Q_g), & \text{if } Z_j = Z_g. \end{cases} \tag{10.15}$$

Here, similar stationarity and ergodicity arguments apply and the average age is again given by

$$\Delta = \frac{\mathbb{E}[Q]}{\mathbb{E}[Y]} = \frac{q\,\mathbb{E}[Q_b] + p\,\mathbb{E}[Q_g]}{q\,\mathbb{E}[Y_b] + p\,\mathbb{E}[Y_g]}. \tag{10.16}$$

By inspecting Fig. 10.2, we find

$$\mathbb{E}[Q_b] = \frac{1}{2}\,\mathbb{E}[(S+Z_b)^2] + \mathbb{E}[S]^2 + \mathbb{E}[S]\,\mathbb{E}[Z_b], \tag{10.17}$$

$$\mathbb{E}[Q_g] = \frac{1}{2}\,\mathbb{E}[(S+Z_g)^2] + \mathbb{E}[S]^2 + \mathbb{E}[S]\,\mathbb{E}[Z_g]. \tag{10.18}$$

In addition, we have,

$$\mathbb{E}[Y_b] = \mathbb{E}[S] + \mathbb{E}[Z_b], \tag{10.19}$$

$$\mathbb{E}[Y_g] = \mathbb{E}[S] + \mathbb{E}[Z_g]. \tag{10.20}$$

by using (10.15) and the fact that $Y = S + Z$.

Substituting (10.17)-(10.20) in (10.16) yields the average age expression under Gilbert-Elliot interarrival times and i.i.d. service times. We note that both the numerator and denominator of (10.16) are linear in p and q.

So far, we derived average age expressions for Gilbert-Elliot servers and samplers for a given state transition matrix P. We optimize this matrix P in the next section to achieve minimum average age at the monitor node in both scenarios.

10.4 Age-Optimal Transition Matrix P

In what follows we characterize the age-optimal state transition matrix P for Gilbert-Elliot service times and Gilbert-Elliot interarrival times.

10.4.1 Gilbert-Elliot Service Times and I.i.d. Interarrival Times

In Section 10.3.1 the average age expression (10.7) for given state transition matrix P under Gilbert-Elliot service times is derived. Next two lemmas characterize the behavior of (10.7) with respect to the state transition probabilities p and q.

Lemma 10.1 *Under Gilbert-Elliot service times, the average age in (10.7) monotonically decreases in p.*

Proof: To prove the lemma, we take the derivative of (10.7) with respect to p and show that it is negative. The numerator of the derivative of (10.7) is

$$\frac{1}{2}\,\mathbb{E}[Y_b]\,\mathbb{E}[Y_g^2] - \frac{1}{2}\,\mathbb{E}[Y_g]\,\mathbb{E}[Y_b^2] + \mathbb{E}[Y_b](\mathbb{E}[S_g] - \mathbb{E}[S_b])[(1-q)\,\mathbb{E}[Y_g] + q\,\mathbb{E}[Y_b]]. \tag{10.21}$$

In (10.21), the last term is already negative as $\mathbb{E}[S_g] < \mathbb{E}[S_b]$ as stated in Section 10.2. Thus, we need to show that $\mathbb{E}[Y_b]\,\mathbb{E}[Y_g^2] - \mathbb{E}[Y_g]\,\mathbb{E}[Y_b^2] < 0$. This is indeed true for exponential interarrival times with rate λ and exponential service times S_g with rate μ_g in state g and S_b with rate μ_b in state b where $\mu_g > \mu_b$. With this, the result follows. ■

Thus, as p increases, a better age performance is achieved at the monitor node.

This is an intuitive result as larger p indicates that the service state spends more time in the good state g as implied by (10.2). We note that this result does not depend on q and is valid for any $q \in (0, 1)$.

Lemma 10.2 *Under Gilbert-Elliot service times, the average age in (10.7) monotonically increases in q.*

The proof of Lemma 10.2 follows similarly to that of Lemma 10.1. Thus, as q decreases, a better age performance is achieved at the monitor node. Similar to Lemma 10.1, Lemma 10.2 holds true for any p value in $(0, 1)$.

From Lemmas 10.1 and 10.2, we observe that, to achieve the minimum average age under Gilbert-Elliot service times, we need to maximize the time spent in state g.

10.4.2 Gilbert-Elliot Interarrival Times and I.i.d. Service Times

In Section 10.3.2 the average age expression (10.16) for given state transition matrix P under Gilbert-Elliot interarrival times is derived. Next two lemmas, which follow similar to Lemmas 10.1 and 10.2, characterize the behavior of (10.16) with respect to the state transition probabilities p and q.

Lemma 10.3 *Under Gilbert-Elliot interarrival times, the average age in (10.16) monotonically decreases in p.*

Thus, as p increases, a better age performance is achieved at the monitor node as in Gilbert-Elliot service times scenario.

Lemma 10.4 *Under Gilbert-Elliot interarrival times, the average age in (10.16) monotonically increases in q.*

Thus, as q decreases, a better age performance is achieved at the monitor node as in Gilbert-Elliot service times scenario.

From Lemmas 10.3 and 10.4, we observe that, to achieve the minimum average age under Gilbert-Elliot interarrival times, we need to maximize the time spent in state g. Although more frequent sampling may overwhelm the network and incur higher age in FCFS queueing systems as shown in [1], in our model, since a dropping policy is implemented, more frequent sampling is desirable to obtain lower average age.

In the next section, we find the age-optimal state transition matrix P when there is an average cost constraint.

10.5 Age-Optimal Transition Matrix P under Average Cost Constraint

In Section 10.4, the age optimization is over all possible p and q pairs, i.e., $(p, q) \in (0, 1) \times (0, 1)$ and we showed that as $p \to 1$ and $q \to 0$, the minimum age is achieved in both scenarios. However, when there is a constraint on the operation of the system, all of $(0, 1) \times (0, 1)$ region may not be feasible. To explore the age-optimal state transition probabilities in such a scenario, here, we consider a constraint on the average cost which may correspond possibly to limited energy budget for the system. Let c_b and c_g denote the cost of operating in state b and state g, respectively,

where $c_g \geq c_b$ as faster operation requires higher cost (e.g., more energy). When the overall budget is c units, we need to satisfy

$$\frac{q}{p+q}c_b + \frac{p}{p+q}c_g \leq c. \tag{10.22}$$

Then, the problem to solve becomes,

$$\begin{aligned} \min_{\{p,q\}} \quad & \frac{q\,\mathbb{E}[Q_b] + p\,\mathbb{E}[Q_g]}{q\,\mathbb{E}[Y_b] + p\,\mathbb{E}[Y_g]} \\ \text{s.t.} \quad & q(c_b - c) + p(c_g - c) \leq 0, \end{aligned} \tag{10.23}$$

where expectations are as in (10.8)-(10.13) for Gilbert-Elliot service times and as in (10.17)-(10.20) for Gilbert-Elliot interarrival times and the constraint follows from (10.22). The trivial case is when $c \geq c_g \geq c_b$ for which the feasible region is $(0,1) \times (0,1)$ and the results from Section 10.4 apply. Thus, in this section, we consider $c_g \geq c \geq c_b$ for which the feasible set is shown in Fig. 10.3. Next, we show that the constraint in (10.23) needs to be satisfied with equality.

Lemma 10.5 *Age-optimal (p,q) pair satisfies the constraint in (10.23) with equality, i.e., $q(c_b - c) + p(c_g - c) = 0$.*

Proof: Given a point β in the feasible set as shown in Fig. 10.3, a lower average age can be obtained as we move along direction I to increase p or along direction II to decrease q as shown in Lemmas 10.1 and 10.2 under Gilbert-Elliot service times and in Lemmas 10.3 and 10.4 under Gilbert-Elliot interarrival times. Thus, no point

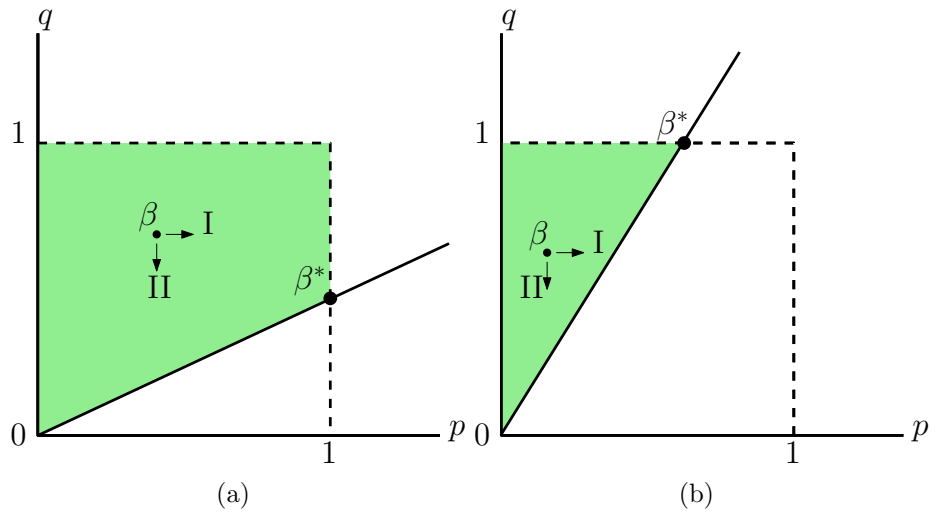


Figure 10.3: Feasible (p, q) pairs for the problem in (10.23) when (a) $\frac{c-c_b}{c_g-c} > 1$ and (b) $\frac{c-c_b}{c_g-c} < 1$ where the line is $q(c_b - c) + p(c_g - c) = 0$ in both cases.

that is not along the $q(c_b - c) + p(c_g - c) = 0$ line can be optimal as we can achieve a lower average age by moving towards this line. ■

Thus, the age-optimal (p, q) pair is such that $q = \alpha p$ where $\alpha = \frac{c_g - c}{c - c_b}$. With this, the problem in (10.23) reduces to a minimization over probability p only. That is, the objective function in (10.23) becomes

$$\Delta(p) = \frac{\alpha\,\mathbb{E}[Q_b] + \mathbb{E}[Q_g]}{\alpha\,\mathbb{E}[Y_b] + \mathbb{E}[Y_g]}, \tag{10.24}$$

where α is fixed. Next, we solve this problem for Gilbert-Elliot service times and Gilbert-Elliot interarrival times and find the age-optimal (p, q) pair that minimizes the average age.

10.5.1 Gilbert-Elliot Service Times and I.i.d. Interarrival Times

We note that, under Gilbert-Elliot service times, the denominator of (10.24) does not depend on p whereas the numerator depends on p linearly. One can show that (10.24) is a decreasing function of p with an argument similar to that of Lemma 10.1. Thus, provided that $\alpha < 1$, the age-optimal transition matrix P under average cost constraint is such that $p \to 1$ and $q \to \alpha$ which is the β^* point in Fig. 10.3(a). This result is intuitive as it maximizes the transition probability from state b to state g and spends fraction of time in state b to satisfy the budget requirement. For example, when $c_g = 2$, $c = 1.8$ and $c_b = 1$ we find $\alpha = \frac{1}{4}$ and the optimal selection is $p \to 1$ and $q \to \frac{1}{4}$.

Otherwise, when $\alpha > 1$, the age-optimal selection is $p \to \frac{1}{\alpha}$ and $q \to 1$ which is the β^* point in Fig. 10.3(b). This result tells us that in the age minimizing operation the transition probability from state g to state b approaches 1 since the transition probability from state b to state g is already limited by $\frac{1}{\alpha}$. For example, when $c_g = 2$, $c = 1.2$ and $c_b = 1$ we find $\alpha = 4$ and the age-optimal selection is $p \to \frac{1}{4}$ and $q \to 1$.

10.5.2 Gilbert-Elliot Interarrival Times and I.i.d. Service Times

We note that, under Gilbert-Elliot interarrival times, both the numerator and the denominator of (10.24) do not depend on p. Thus, any p and corresponding $q = \alpha p$ for the given α yields the same average age. In other words, as long as we operate in $(0, \beta^*)$ in Fig. 10.3, i.e., satisfy (10.22) with equality, we obtain the optimal average age since (10.16) depends on p and q only through the stationary probabilities of

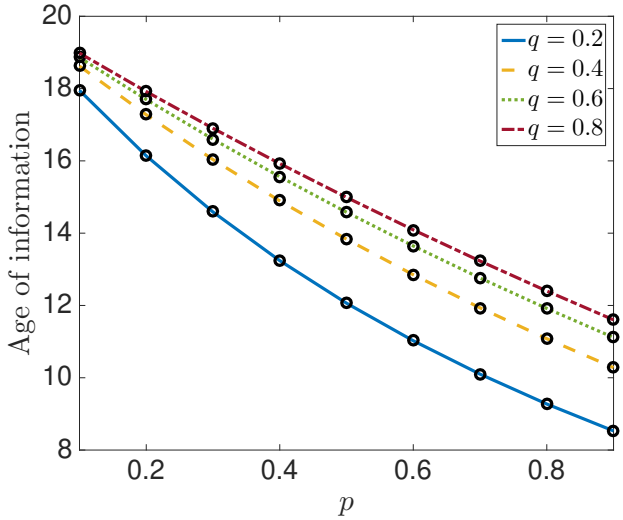


Figure 10.4: Age of information as a function of probability p for Gilbert-Elliot service times. Symbol $\circ$ marks the simulation results and curves indicate the values obtained from (10.7).

the states given in (10.2) as expectations in (10.17)-(10.20) do not depend on p and q.

10.6 Numerical Results

In this section, we provide simple numerical results to validate our theoretical results for a system with arbitrary exponential interarrival and service times.

We consider Gilbert-Elliot service times and take $\lambda = 1$ which is the rate of Poisson arrivals to the server node. We model the service times with an exponential random variable with rate $\mu_b = 0.1$ in state b and with rate $\mu_g = 1$ in state g. In Figs. 10.4 and 10.5, we plot the average age of information under Gilbert-Elliot service times as a function of the state transition probabilities p and q, respectively. In both of the figures, we plot simulation results, marked with $\circ$ symbol, along with results obtained from (10.7) and observe that the results match. Fig. 10.4 shows

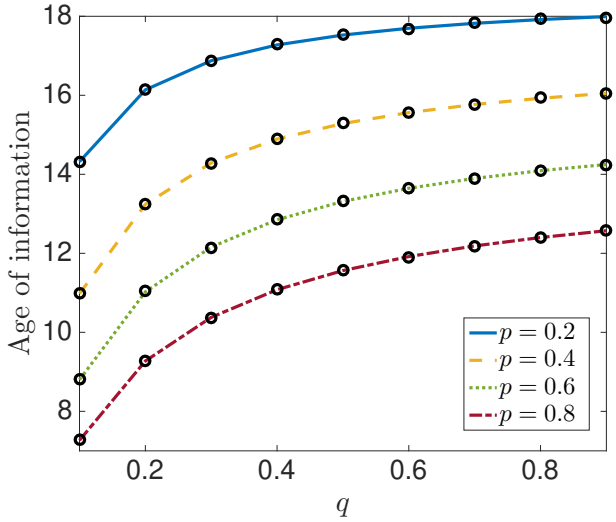


Figure 10.5: Age of information as a function of probability q for Gilbert-Elliot service times. Symbol $\circ$ marks the simulation results and curves indicate the values obtained from (10.7).

that the average age decreases monotonically as probability p gets larger as shown in Lemma 10.1. Here, we also note that as q gets lower for a fixed p value, we achieve a lower average age as discussed earlier in Section 10.4. Fig. 10.5 shows that the age of information increases monotonically as probability q increases. We also observe that for a fixed q value, the best age is obtained when the p probability is the largest. When the server node only operates in the good state g, we find that $\Delta = 2.5$. We observe that this value is lower than the age values in Figs. 10.4 and 10.5 as the server does not slow down by switching to the bad state b. Similarly, if the server node only operates in the bad state b, we find that $\Delta = 20.09$ which is strictly larger than the age values shown in Figs. 10.4 and 10.5. Thus, in this case, the server node benefits from switching to the good state g.

10.7 Conclusion

In this chapter, we considered an information update system in which status update packets are generated by a sampler and sent to a monitor node through a server node. We considered two scenarios: Gilbert-Elliot service times and i.i.d. interarrival times; and Gilbert-Elliot interarrival times and i.i.d. service times. In these scenarios, either the server or the sampler follows a two-state Markov chain with the good state g and the bad state b where the operation is faster in state g. We determined the average age at the monitor node for both scenarios and characterized the age-optimal state transition matrix for the underlying Markov chain with and without an average cost constraint on the operation of the system.

CHAPTER 11

Conclusions

In this dissertation, we studied information timeliness in large communication networks, distributed computation and learning systems.

In Chapter 2, we studied a multihop multicast network, in which a single source node sends time-sensitive status updates to n^L end nodes where L denotes the number of hops. Each update from the source node goes through relay nodes in each hop to reach the end nodes in the Lth hop. We showed that by carefully selecting stopping thresholds in each hop, the average age at the end nodes can be made a constant independent of n. We then found the optimum stopping thresholds k_ℓ for each hop ℓ for arbitrary shifted exponential link delays in each hop.

In Chapter 3, we considered age of information in a multicast network with two types of updates that share the same network, namely type I and type II updates. We showed that by utilizing an earliest k_1 and k_2 transmission scheme for type I and type II updates, respectively, the age of both update streams can be made a constant independent of the network size. We then characterized the k_1 and k_2 stopping thresholds to individually and jointly minimize the average age of both

update streams for arbitrary shifted exponential link delays.

In Chapter 4, we studied the scaling of average age of information in a large peer-to-peer network with n source-destination (S-D) pairs. We first proposed a three-phase transmission scheme that uses local cooperation between nodes as well mega update packets to achieve an average age scaling of $O(n^{\frac{1}{4}} \log n)$ per user. Then, we introduced hierarchy to the proposed scheme that entails applying the proposed three-phase scheme in smaller scales and achieved a per-user average age scaling of $O\left(n^{\alpha(h)} \log n\right)$ where $\alpha(h) = \frac{1}{3 \cdot 2^h + 1}$ and h is the number of hierarchy levels. In the asymptotic case when h tends to ∞, the resulting average age per-user scales as $O(\log n)$.

In Chapter 5, we studied version age scaling in a network consisting of n nodes that are grouped into equal-sized clusters, each equipped with a cluster-head that facilitates communication with the source. Unlike Chapters 2-4, nodes are allowed to share their stored versions of the source information with their neighbors within clusters, i.e., gossiping. We showed that average version age scalings of $O(\sqrt{n})$, $O(n^{\frac{1}{3}})$, and $O(\log n)$ are achievable per user in disconnected, ring, and fully connected cluster topologies. We then showed that, once the cluster heads exchange information among themselves following a ring network, per-node average age scalings of $O(n^{\frac{1}{3}})$, $O(n^{\frac{1}{4}})$, and $O(\log n)$ in disconnected, ring, and fully connected cluster models, respectively, are achievable. We then implemented a hierarchical gossip structure and showed that, for h levels of hierarchy, per user average age scaling of $O(n^{\frac{1}{2h}})$ is achievable in the case of ring networks in each cluster across all hierarchy levels. We finally found the version age-optimum cluster sizes as a

function of the update rates at the source, cluster heads, and among the nodes.

In Chapter 6, we studied age of information considering computationally-intensive status update packets that need additional processing to reveal the embedded useful information. This processing is handled by a computation unit (CU) using n worker nodes in a distributed manner. We analyzed the age performance of the uncoded and coded computation distribution algorithms. We first showed that the MDS coded task distribution scheme outperforms the uncoded and repetition coded schemes for large n. Next, we showed that, when workers are assigned multiple computation tasks (MM-MDS scheme), the age performance of the system further improves. We then found the age-optimal repetition, MDS, and MM-MDS coded schemes that minimize the age of information at the receiver node.

In Chapter 7, we studied age of information in federated learning and proposed a novel timely communication scheme for applications that involve highly temporal rapidly changing client datasets. Our scheme aims to reflect the fast changing client data in global model updates with as little age as possible without harming convergence by considering the limited client availability and communication resources. We showed that the proposed timely communication scheme not only increases the timeliness of the system but also significantly improves the average iteration time without sacrificing convergence of the learning task.

In Chapter 8, we proposed an age-based coded computation strategy for distributed learning systems that utilize partial recovery for straggler mitigation. In order to prevent biased model updates that are caused by partial recovery in the case of correlated straggler behavior among the workers, we designed an age-based

timely encoding strategy that changes the codewords at the workers to regulate the recovery frequency of the partial computations. We showed, through extensive simulations on a linear regression problem, that the proposed age-based encoding strategy significantly improves the convergence performance compared to conventional static encoding schemes.

In Chapter 9, we studied a gradient coding (GC) framework and proposed a novel scheme called the GC-DC to improve the average iteration time of the existing static GC schemes. Our proposed GC-DC scheme dynamically forms clusters from the workers and applies the GC scheme in each cluster separately. In doing that, it aims to distribute the stragglers to clusters as uniformly as possible in each iteration under a time-correlated straggling behavior for the workers. Through extensive simulations, we showed that the GC-DC scheme significantly improves average per-iteration completion time without an increase in the communication load by dynamically adapting to the straggling behavior of the workers.

In Chapter 10, we considered a system where a single sampler updates a receiver node through a single server. We studied Gilbert-Elliot service times and i.i.d. interarrival times; and Gilbert-Elliot interarrival times and i.i.d. service times. We characterized the average age of information at the receiver node for both cases and determined the age-optimal state transition matrix of the underlying Markov chain with and without cost constraints on the operation of the system.

The contents of Chapter 2 are published in [115, 119], Chapter 3 in [129], Chapter 4 in [145–147], Chapter 5 in [223, 224], Chapter 6 in [182, 225], Chapter 7 in [226], Chapter 8 in [183], Chapter 9 in [227, 228], Chapter 10 in [218].

Bibliography

[1] S. K. Kaul, R. D. Yates, and M. Gruteser. Real-time status: How often should one update? In *IEEE Infocom*, March 2012.

[2] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano. Minimizing the age of information in broadcast wireless networks. In *Allerton Conference*, September 2016.

[3] N. Pappas, J. Gunnarsson, L. Kratz, M. Kountouris, and V. Angelakis. Age of information of multiple sources with queue management. In *IEEE ICC*, June 2015.

[4] Y. Sun, I. Kadota, R. Talak, and E. Modiano. Age of information: A new metric for information freshness. *Synthesis Lectures on Communication Networks*, 12(2):1–224, December 2019.

[5] A. Kosta, N. Pappas, and V. Angelakis. Age of information: A new concept, metric, and tool. *Foundations and Trends in Networking*, 12(3):162–259, 2017.

[6] R. D. Yates, Y. Sun, D. R. Brown III, S. K. Kaul, E. Modiano, and S. Ulukus. Age of information: An introduction and survey. *IEEE Journal on Selected Areas in Communications*, 39(5):1183–1210, May 2021.

[7] R. D. Yates and S. K. Kaul. Real-time status updating: Multiple sources. In *IEEE ISIT*, July 2012.

[8] C. Kam, S. Kompella, and A. Ephremides. Effect of message transmission diversity on status age. In *IEEE ISIT*, June 2014.

[9] C. Kam, S. Kompella, and A. Ephremides. Age of information under random updates. In *IEEE ISIT*, July 2013.

[10] C. Kam, S. Kompella, G. D. Nguyen, and A. Ephremides. Effect of message transmission path diversity on status age. *IEEE Transactions on Information Theory*, 62(3):1360–1374, March 2016.

[11] S. K. Kaul, R. D. Yates, and M. Gruteser. Status updates through queues. In *CISS*, March 2012.

[12] L. Huang and E. Modiano. Optimizing age-of-information in a multi-class queueing system. In *IEEE ISIT*, June 2015.

[13] R. D. Yates and S. K. Kaul. The age of information: Real-time status updating by multiple sources. *IEEE Transactions on Information Theory*, 65(3):1807–1827, March 2019.

[14] M. Costa, M. Codrenau, and A. Ephremides. Age of information with packet management. In *IEEE ISIT*, June 2014.

[15] M. Costa, M. Codreanu, and A. Ephremides. On the age of information in status update systems with packet management. *IEEE Transactions on Information Theory*, 62(4):1897–1910, April 2016.

[16] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides. Age of information with a packet deadline. In *IEEE ISIT*, July 2016.

[17] A. M. Bedewy, Y. Sun, and N. B. Shroff. Optimizing data freshness, throughput, and delay in multi-server information-update systems. In *IEEE ISIT*, July 2016.

[18] A. M. Bedewy, Y. Sun, and N. B. Shroff. Minimizing the age of information through queues. *IEEE Transactions on Information Theory*, 65(8):5215–5232, August 2019.

[19] A. Kosta, N. Pappas, A. Ephremides, M. Kountouris, and V. Angelakis. Age and value of information: Non-linear age case. In *IEEE ISIT*, June 2017.

[20] Y. Inoue. Analysis of the age of information with packet deadline and infinite buffer capacity. In *IEEE ISIT*, June 2018.

[21] K. Chen and L. Huang. Age-of-information in the presence of error. In *IEEE ISIT*, July 2016.

[22] E. Najm and R. Nasser. Age of information: The gamma awakening. In *IEEE ISIT*, July 2016.

[23] E. Najm and E. Telatar. Status updates in a multi-stream M/G/1/1 preemptive queue. In *IEEE Infocom*, April 2018.

[24] L. Huang and L. P. Qian. Age of information for transmissions over Markov channels. In *IEEE Globecom*, December 2017.

[25] V. Tripathi, R. Talak, and E. Modiano. Age of information for discrete time queues. January 2019. Available on arXiv:1901.10463.

[26] P. Zou, O. Ozel, and S. Subramaniam. On the benefits of waiting in status update systems. In *IEEE Infocom*, April 2019.

[27] A. Maatouk, M. Assaad, and A. Ephremides. Age of information with prioritized streams: When to buffer preempted packets? In *IEEE ISIT*, July 2019.

[28] A. M. Bedewy, Y. Sun, S. Kompella, and N. B. Shroff. Age-optimal sampling and transmission scheduling in multi-source systems. In *ACM MobiHoc*, July 2019.

[29] E. Najm, R. D. Yates, and E. Soljanin. Status updates through M/G/1/1 queues with HARQ. In *IEEE ISIT*, June 2017.

[30] A. Soysal and S. Ulukus. Age of information in G/G/1/1 systems: Age expressions, bounds, special cases, and optimization. *IEEE Transactions on Information Theory*, 67(11):7477–7489, November 2021.

[31] A. M. Bedewy, Y. Sun, and N. B. Shroff. Age-optimal information updates in multihop networks. In *IEEE ISIT*, June 2017.

[32] A. M. Bedewy, Y. Sun, and N. B. Shroff. The age of information in multihop networks. *IEEE/ACM Transactions on Networking*, 27(3):1248–1257, June 2019.

[33] R. Talak, S. Karaman, and E. Modiano. Minimizing age-of-information in multi-hop wireless networks. In *Allerton Conference*, October 2017.

[34] S. Farazi, A. G. Klein, and D. R. Brown III. Fundamental bounds on the age of information in multi-hop global status update networks. *Journal of Communications and Networks, special issue on Age of Information*, 21(3):268–279, July 2019.

[35] S. Farazi, A. G. Klein, and D. R. Brown III. Age of information with unreliable transmissions in multi-source multi-hop status update systems. In *Asilomar Conference*, November 2019.

[36] S. Farazi, A. G. Klein, and D. R. Brown III. Average age of information in multi-source self-preemptive status update systems with packet delivery errors. In *Asilomar Conference*, November 2019.

[37] K. S. Ashok Krishnan and V. Sharma. Minimizing age of information in a multihop wireless network. In *IEEE ICC*, June 2020.

[38] R. D. Yates. The age of information in networks: Moments, distributions, and sampling. *IEEE Transactions on Information Theory*, 66(9):5712–5728, September 2020.

[39] Y. Inoue, H. Masuyama, T. Takine, and T. Tanaka. A general formula for the stationary distribution of the age of information and its application to single-server queues. *IEEE Transactions on Information Theory*, 65(12):8305–8324, December 2019.

[40] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff. Update or wait: How to keep your data fresh. *IEEE Transactions on Information Theory*, 63(11):7492–7508, November 2017.

[41] P. Zou, O. Ozel, and S. Subramaniam. Waiting before serving: A companion to packet management in status update systems. *IEEE Transactions on Information Theory*, 66(6):3864–3877, June 2020.

[42] S. Ioannidis, A. Chaintreau, and L. Massoulie. Optimal and scalable distribution of content updates over a mobile social network. In *IEEE Infocom*, April 2009.

[43] M. Wang, W. Chen, and A. Ephremides. Real-time reconstruction of a counting process through first-come-first-serve queue systems. *IEEE Transactions on Information Theory*, 66(7):4547–4562, July 2020.

[44] Y. Sun, Y. Polyanskiy, and E. Uysal-Biyikoglu. Remote estimation of the Wiener process over a channel with random delay. In *IEEE ISIT*, June 2017.

[45] Y. Sun and B. Cyr. Information aging through queues: A mutual information perspective. In *IEEE SPAWC*, June 2018.

[46] J. Chakravorty and A. Mahajan. Remote estimation over a packet-drop channel with Markovian state. *IEEE Transactions on Automatic Control*, 65(5):2016–2031, May 2020.

[47] C. Kam, S. Kompella, G. D. Nguyen, and J. E. Wieselthier. Towards an effective age of information: Remote estimation of a Markov source. In *IEEE Infocom*, April 2018.

[48] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides. Towards an "effective age" concept. In *IEEE SPAWC*, June 2018.

[49] S. Poojary, S. Bhambay, and P. Parag. Real-time status updates for correlated sources. In *IEEE ITW*, November 2017.

[50] T. Z. Ornee and Y. Sun. Sampling for remote estimation through queues: Age of information and beyond. In *IEEE WiOpt*, June 2019.

[51] M. Bastopcu and S. Ulukus. Who should Google Scholar update more often? In *IEEE Infocom*, July 2020.

[52] Q. He, D. Yuan, and A. Ephremides. Optimizing freshness of information: On minimum age link scheduling in wireless systems. In *IEEE WiOpt*, May 2016.

[53] Q. He, D. Yuan, and A. Ephremides. Optimal link scheduling for age minimization in wireless systems. *IEEE Transactions on Information Theory*, 64(7):5381–5394, July 2018.

[54] Q. He, D. Yuan, and A. Ephremides. On optimal link scheduling with min-max peak age of information in wireless systems. In *IEEE ICC*, May 2016.

[55] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano. Scheduling policies for minimizing age of information in broadcast wireless networks. *IEEE/ACM Transactions on Networking*, 26(6):2637–2650, December 2018.

[56] Y. P. Hsu. Age of information: Whittle index for scheduling stochastic arrivals. In *IEEE ISIT*, June 2018.

[57] Y. P. Hsu, E. Modiano, and L. Duan. Age of information: Design and analysis of optimal scheduling algorithms. In *IEEE ISIT*, June 2017.

[58] Y. P. Hsu, E. Modiano, and L. Duan. Scheduling algorithms for minimizing age of information in wireless broadcast networks with random arrivals. *IEEE Transactions on Mobile Computing*, 19(12):2903–2915, December 2020.

[59] Y. Sun, E. Uysal-Biyikoglu, and S. Kompella. Age-optimal updates of multiple information flows. In *IEEE Infocom*, April 2018.

[60] V. Tripathi and S. Moharir. Age of information in multi-source systems. In *IEEE Globecom*, December 2017.

[61] P. R. Jhunjhunwala and S. Moharir. Age-of-information aware scheduling. In *SPCOM*, July 2018.

[62] I. Kadota, A. Sinha, and E. Modiano. Scheduling algorithms for optimizing age of information in wireless networks with throughput constraints. *IEEE/ACM Transactions on Networking*, 27(4):1359–1372, August 2019.

[63] S. K. Kaul and R. D. Yates. Status updates over unreliable multiaccess channels. In *IEEE ISIT*, June 2017.

[64] R. Talak, S. Karaman, and E. Modiano. Distributed scheduling algorithms for optimizing information freshness in wireless networks. In *IEEE SPAWC*, June 2018.

[65] R. Talak, S. Karaman, and E. Modiano. Optimizing age of information in wireless networks with perfect channel state information. In *IEEE WiOpt*, May 2018.

[66] R. Talak, I. Kadota, S. Karaman, and E. Modiano. Scheduling policies for age minimization in wireless networks with unknown channel state. In *IEEE ISIT*, June 2018.

[67] R. Talak, S. Karaman, and E. Modiano. Optimizing information freshness in wireless networks under general interference constraints. In *MobiHoc*, June 2018.

[68] N. Lu, B. Ji, and B. Li. Age-based scheduling: Improving data freshness for wireless real-time traffic. In *MobiHoc*, June 2018.

[69] M. Bastopcu and S. Ulukus. Age of information with soft updates. In *Allerton Conference*, October 2018.

[70] M. Bastopcu and S. Ulukus. Minimizing age of information with soft updates. *Journal of Communications and Networks, special issue on Age of Information*, 21(3):233–243, July 2019.

[71] M. Bastopcu and S. Ulukus. Age of information for updates with distortion. In *IEEE ITW*, August 2019.

[72] M. Bastopcu and S. Ulukus. Age of information for updates with distortion: Constant and age-dependent distortion constraints. December 2019. *IEEE/ACM Transactions on Networking,* to appear. Available on arXiv:1912.13493.

[73] N. Rajaraman, R. Vaze, and G. Reddy. Not just age but age and quality of information. *IEEE Journal on Selected Areas in Communications*, 39(5):1325–1338, May 2021.

[74] M. Bastopcu and S. Ulukus. Timely group updating. In *CISS*, March 2021.

[75] R. D. Yates. Lazy is timely: Status updates by an energy harvesting source. In *IEEE ISIT*, June 2015.

[76] A. Arafa and S. Ulukus. Age minimization in energy harvesting communications: Energy-controlled delays. In *Asilomar Conference*, October 2017.

[77] A. Arafa and S. Ulukus. Age-minimal transmission in energy harvesting two-hop networks. In *IEEE Globecom*, December 2017.

[78] B. T. Bacinoglu, E. T. Ceran, and E. Uysal-Biyikoglu. Age of information under energy replenishment constraints. In *UCSD ITA*, February 2015.

[79] B. T. Bacinoglu and E. Uysal-Biyikoglu. Scheduling status updates to minimize age of information with an energy harvesting sensor. In *IEEE ISIT*, June 2017.

[80] X. Wu, J. Yang, and J. Wu. Optimal status update for age of information minimization with an energy harvesting source. *IEEE Transactions on Green Communications and Networking*, 2(1):193–204, March 2018.

[81] A. Arafa, J. Yang, S. Ulukus, and H. V. Poor. Age-minimal online policies for energy harvesting sensors with incremental battery recharges. In *UCSD ITA*, February 2018.

[82] A. Arafa, J. Yang, and S. Ulukus. Age-minimal online policies for energy harvesting sensors with random battery recharges. In *IEEE ICC*, May 2018.

[83] A. Arafa, J. Yang, S. Ulukus, and H. V. Poor. Online timely status updates with erasures for energy harvesting sensors. In *Allerton Conference*, October 2018.

[84] S. Feng and J. Yang. Minimizing age of information for an energy harvesting source with updating failures. In *IEEE ISIT*, June 2018.

[85] A. Baknina and S. Ulukus. Coded status updates in an energy harvesting erasure channel. In *CISS*, March 2018.

[86] A. Baknina, O. Ozel, J. Yang, S. Ulukus, and A. Yener. Sending information through status updates. In *IEEE ISIT*, June 2018.

[87] R. D. Yates, P. Ciblat, A. Yener, and M. Wigger. Age-optimal constrained cache updating. In *IEEE ISIT*, June 2017.

[88] C. Kam, S. Kompella, G. D. Nguyen, J. Wieselthier, and A. Ephremides. Information freshness and popularity in mobile caching. In *IEEE ISIT*, June 2017.

[89] H. Tang, P. Ciblat, J. Wang, M. Wigger, and R. Yates. Age of information aware cache updating with file- and age-dependent update durations. In *IEEE WiOpt*, June 2020.

[90] M. Bastopcu and S. Ulukus. Maximizing information freshness in caching systems with limited cache storage capacity. In *Asilomar Conference*, November 2020.

[91] M. Bastopcu and S. Ulukus. Information freshness in cache updating systems. *IEEE Transactions on Wireless Communications*, 20(3):1861–1874, March 2021.

[92] M. Bastopcu and S. Ulukus. Cache freshness in information updating systems. In *CISS*, March 2021.

[93] Y. Gu, Q. Wang, H. Chen, Y. Li, and B. Vucetic. Optimizing information freshness in two-hop status update systems under a resource constraint. *IEEE Journal on Selected Areas in Communications*, 39(5):1380–1392, May 2021.

[94] J. Zhong and R. D. Yates. Timeliness in lossless block coding. In *IEEE DCC*, March 2016.

[95] P. Parag, A. Taghavi, and J. F. Chamberland. On real-time status updates over symbol erasure channels. In *IEEE WCNC*, March 2017.

[96] H. Sac, T. Bacinoglu, E. Uysal-Biyikoglu, and G. Durisi. Age-optimal channel coding blocklength for an M/G/1 queue with HARQ. In *IEEE SPAWC*, June 2018.

[97] P. Mayekar, P. Parag, and H. Tyagi. Optimal lossless source codes for timely updates. In *IEEE ISIT*, June 2018.

[98] J. Zhong, R. D. Yates, and E. Soljanin. Timely lossless source coding for randomly arriving symbols. In *IEEE ITW*, November 2018.

[99] M. Bastopcu, B. Buyukates, and S. Ulukus. Optimal selective encoding for timely updates. In *CISS*, March 2020.

[100] B. Buyukates, M. Bastopcu, and S. Ulukus. Optimal selective encoding for timely updates with empty symbol. In *IEEE ISIT*, June 2020.

[101] M. Bastopcu, B. Buyukates, and S. Ulukus. Selective encoding policies for maximizing information freshness. *IEEE Transactions on Communications*, 69(9):5714–5726, September 2021.

[102] M. Bastopcu and S. Ulukus. Partial updates: Losing information for freshness. In *IEEE ISIT*, June 2020.

[103] D. Ramirez, E. Erkip, and H. V. Poor. Age of information with finite horizon and partial updates. In *IEEE ICASSP*, pages 4965–4969, May 2020.

[104] M. A. Abd-Elmagid and H. S. Dhillon. Average peak age-of-information minimization in UAV-assisted IoT networks. *IEEE Transactions on Vehicular Technology*, 68(2):2003–2008, February 2019.

[105] J. Liu, X. Wang, and H. Dai. Age-optimal trajectory planning for UAV-assisted data collection. In *IEEE Infocom*, April 2018.

[106] M. A. Abd-Elmagid, A. Ferdowsi, H. S. Dhillon, and W. Saad. Deep reinforcement learning for minimizing age-of-information in UAV-assisted networks. In *IEEE Globecom*, December 2019.

[107] A. Alabbasi and V. Aggarwal. Joint information freshness and completion time optimization for vehicular networks. *IEEE Transactions on Services Computing*, pages 1–14, March 2020.

[108] E. T. Ceran, D. Gunduz, and A. Gyorgy. A reinforcement learning approach to age of information in multi-user networks. In *IEEE PIMRC*, September 2018.

[109] H. B. Beytur and E. Uysal-Biyikoglu. Age minimization of multiple flows using reinforcement learning. In *IEEE ICNC*, February 2019.

[110] M. A. Abd-Elmagid, H. S. Dhillon, and N. Pappas. A reinforcement learning framework for optimizing age of information in rf-powered communication systems. *IEEE Transactions on Communications*, 68(8):4747–4760, August 2020.

[111] J. Hu, H. Zhang, L. Song, R. Schober, and H. V. Poor. Cooperative internet of UAVs: Distributed trajectory design by multi-agent deep reinforcement learning. *IEEE Transactions on Communications*, 68(11):6807–6821, November 2020.

[112] S. Leng and A. Yener. Age of information minimization for wireless ad hoc networks: A deep reinforcement learning approach. In *IEEE Globecom*, December 2019.

[113] J. Zhong, E. Soljanin, and R. D. Yates. Status updates through multicast networks. In *Allerton Conference*, October 2017.

[114] A. W. Marshall and I. Olkin. *Life Distributions, Structure of Nonparametric, Semiparametric, and Parametric Families*. Springer, New York, 2007.

[115] B. Buyukates, A. Soysal, and S. Ulukus. Age of information in two-hop multicast networks. In *Asilomar Conference*, October 2018.

[116] A. Soysal and S. Ulukus. Age of information in G/G/1/1 systems. In *Asilomar Conference*, November 2019.

[117] M. Shaked and J. G. Shantikumar. *Stochastic Orders and Their Applications*. Springer-Verlag New York, 2007.

[118] S. Karlin and F. Proschan. Polya type distributions of convolutions. *The Annals of Mathematical Statistics*, 31(3):721–736, September 1960.

[119] B. Buyukates, A. Soysal, and S. Ulukus. Age of information in multihop multicast networks. *Journal of Communications and Networks*, 21(3):256–267, July 2019.

[120] S. K. Kaul and R. D. Yates. Age of information: Updates with priority. In *IEEE ISIT*, June 2018.

[121] E. Najm, R. Nasser, and E. Telatar. Content based status updates. In *IEEE ISIT*, June 2018.

[122] P. Gupta and P. R. Kumar. Capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[123] A. El Gamal, J. Mammen, B. Prabhakar, and D. Shah. Optimal throughput-delay scaling in wireless networks-part I: The fluid model. *IEEE Transactions on Information Theory*, 52(06):2568–2592, June 2006.

[124] M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Transactions on Networking*, 10(04):477–486, August 2002.

[125] M. J. Neely and E. Modiano. Capacity and delay trade-offs for ad hoc mobile networks. *IEEE Transactions on Information Theory*, 51(06):1917–1937, June 2005.

[126] G. Sharma, R. Mazumdar, and N. B. Shroff. Delay and capacity trade-offs in mobile ad hoc networks: A global perspective. In *IEEE Infocom*, April 2006.

[127] A. Ozgur, O. Leveque, and D. N. C. Tse. Hierarchical cooperation achieves optimal capacity scaling in ad hoc networks. *IEEE Transactions on Information Theory*, 53(10):3549–3572, September 2007.

[128] A. Ozgur and O. Leveque. Throughput-delay trade-off for hierarchical cooperation in ad hoc wireless networks. *IEEE Transactions on Information Theory*, 56(03):1369–1377, March 2010.

[129] B. Buyukates, A. Soysal, and S. Ulukus. Age of information in multicast networks with multiple update streams. In *Asilomar Conference*, November 2019.

[130] Z. Jiang, B. Krishnamachari, X. Zheng, S. Zhou, and Z. Niu. Timely status update in massive IoT systems: Decentralized scheduling for wireless uplinks. In *IEEE ISIT*, June 2018.

[131] H. A. David and H. N. Nagaraja. *Order Statistics*. Wiley, 2003.

[132] R. D. Yates and D. J. Goodman. *Probability and Stochastic Processes*. Wiley, 2014.

[133] J. Cho and H. Garcia-Molina. Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems*, 28(4):390–426, December 2003.

[134] A. Kolobov, Y. Peres, E. Lubetzky, and E. Horvitz. Optimal freshness crawl under politeness constraints. In *ACM SIGIR Conference*, July 2019.

[135] P. Kaswan, M. Bastopcu, and S. Ulukus. Freshness based cache updating in parallel relay networks. In *IEEE ISIT*, July 2021.

[136] M. Bastopcu, B. Buyukates, and S. Ulukus. Gossiping with binary freshness metric. In *IEEE Globecom*, December 2021.

[137] J. Zhong, R. D. Yates, and E. Soljanin. Two freshness metrics for local cache refresh. In *IEEE ISIT*, June 2018.

[138] A. Maatouk, S. Kriouile, M. Assaad, and A. Ephremides. The age of incorrect information: A new performance metric for status updates. *IEEE/ACM Transactions on Networking*, 28(5):2215–2228, July 2020.

[139] A. Maatouk, M. Assaad, and A. Ephremides. The age of incorrect information: an enabler of semantics-empowered communication. December 2020. Available on arXiv:2012.13214.

[140] C. Kam, S. Kompella, and A. Ephremides. Age of incorrect information for remote estimation of a binary markov source. In *IEEE Infocom*, July 2020.

[141] R. D. Yates. The age of gossip in networks. In *IEEE ISIT*, July 2021.

[142] B. Abolhassani, J. Tadrous, A. Eryilmaz, and E. Yeh. Fresh caching for dynamic content. In *IEEE Infocom*, May 2021.

[143] M. Bastopcu and S. Ulukus. Timely tracking of infection status of individuals in a population. In *IEEE Infocom*, May 2021.

[144] J. Zhong, R. D. Yates, and E. Soljanin. Multicast with prioritized delivery: How fresh is your data? In *IEEE SPAWC*, June 2018.

[145] B. Buyukates, A. Soysal, and S. Ulukus. Age of information scaling in large networks. In *IEEE ICC*, May 2019.

[146] B. Buyukates, A. Soysal, and S. Ulukus. Age of information scaling in large networks with hierarchical cooperation. In *IEEE Globecom*, December 2019.

[147] B. Buyukates, A. Soysal, and S. Ulukus. Scaling laws for age of information in wireless networks. *IEEE Transactions on Wireless Communications*, 20(4):2413–2427, April 2021.

[148] A. R. Teel, A. Subbaraman, and A. Sferlazza. Stability analysis for stochastic hybrid systems: A survey. *Automatica*, 50(10):2435–2456, October 2014.

[149] J. Hespanha. Modelling and analysis of stochastic hybrid systems. *IEEE Proceedings-Control Theory and Applications*, 153(5):520–535, January 2007.

[150] O. Dogan and N. Akar. The multi-source preemptive M/PH/1/1 queue with packet errors: Exact distribution of the age of information and its peak. July 2020. Available on arXiv:2007.11656.

[151] M. Moltafet, M. Leinonen, and M. Codreanu. Average AoI in multi-source systems with source-aware packet management. *IEEE Transactions on Communications*, 69(2):1121–1133, November 2020.

[152] Q. Kuang, J. Gong, X. Chen, and X. Ma. Age-of-information for computation-intensive messages in mobile edge computing. January 2019. Available on arXiv: 1901.01854.

[153] J. Gong, Q. Kuang, X. Chen, and X. Ma. Reducing age-of-information for computation-intensive messages via packet replacement. In *IEEE WCSP*, October 2019.

[154] A. Arafa, R. D. Yates, and H. V. Poor. Timely cloud computing: preemption and waiting. In *IEEE Allerton*, May 2019.

[155] X. Song, X. Qin, Y. Tao, B. Liu, and P. Zhang. Age based task scheduling and computation offloading in mobile-edge computing systems. In *IEEE WCNCW*, April 2019.

[156] J. Zhong, W. Zhang, R. D. Yates, A. Garnaev, and Y. Zhang. Age-aware scheduling for asynchronous arriving jobs in edge applications. In *IEEE Infocom*, April 2019.

[157] P. Zou, O. Ozel, and S. Subramaniam. Trading off computation with transmission in status update systems. In *IEEE PIMRC*, September 2019.

[158] A. Arafa, K. Banawan, K. G. Seddik, and H. V. Poor. On timely channel coding with hybrid ARQ. In *IEEE Globecom*, December 2019.

[159] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran. Speeding up distributed machine learning using codes. *IEEE Transactions on Information Theory*, 64(3):1514–1529, March 2018.

[160] R. Tandon, A. Lei, G. Dimakis, and N. Karampatziakis. Gradient coding: Avoiding stragglers in distributed learning. In *ICML*, August 2017.

[161] A. B. Das, L. Tang, and A. Ramamoorthy. C^3LES: Codes for coded computation that leverage stragglers. In *IEEE ITW*, November 2018.

[162] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover. On the optimal recovery threshold of coded matrix multiplication. *IEEE Transactions on Information Theory*, 66(1):278–301, July 2019.

[163] N. Ferdinand and S. C. Draper. Hierarchical coded computation. In *IEEE ISIT*, June 2018.

[164] M. Mohammadi Amiri and D. Gunduz. Computation scheduling for distributed machine learning with straggling workers. In *IEEE ICASSP*, May 2019.

[165] E. Ozfatura, D. Gunduz, and S. Ulukus. Speeding up distributed gradient descent by utilizing non-persistent stragglers. In *IEEE ISIT*, July 2019.

[166] E. Ozfatura, D. Gunduz, and S. Ulukus. Gradient coding with clustering and multi-message communication. In *IEEE Data Science Workshop*, June 2019.

[167] E. Ozfatura, S. Ulukus, and D. Gunduz. Distributed gradient descent with coded partial gradient computations. In *IEEE ICASSP*, May 2019.

[168] M. F. Aktas and E. Soljanin. Straggler mitigation at scale. *IEEE/ACM Transactions on Networking*, 27(6):2266–2279, December 2019.

[169] C. S. Yang, R. Pedarsani, and A. S. Avestimehr. Timely coded computing. In *IEEE ISIT*, July 2019.

[170] K. R. Duffy and S. Shneer. MDS coding is better than replication for job completion times. *Operations Research Letters*, 49(1):91–95, January 2021.

[171] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, April 2017.

[172] J. Feng, C. Rong, F. Sun, D. Guo, and Y. Li. PMF: a privacy-preserving human mobility prediction framework via federated learning. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1):10–21, March 2020.

[173] G. Damaskinos, R. Guerraoui, A. M. Kermarrec, V. Nitu, R. Patra, and F. Taiani. FLEET: Online federated learning via staleness awareness and performance prediction. June 2020. Available on arXiv: 2006.07273.

[174] L. Li, Y. Fan, M. Tse, and K.-Y. Lin. A review of applications in federated learning. *Elsevier Computers & Industrial Engineering*, 149:1–15, November 2020.

[175] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Federated learning with non-iid data. June 2018. Available on arXiv:1806.00582.

[176] T. Nishio and R. Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *IEEE ICC*, May 2019.

[177] M. Mohammadi Amiri and D. Gunduz. Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air. In *IEEE ISIT*, July 2019.

[178] L. P. Barnes, H. A. Inan, B. Isik, and A. Ozgur. rTop-k: a statistical estimation approach to distributed SGD. *IEEE Journal on Selected Areas in Information Theory*, 1(3):897–907, November 2020.

[179] S. Dhakal, S. Prakash, Y. Yona, S. Talwar, and N. Himayat. Coded federated learning. In *IEEE Globecom*, December 2019.

[180] M. M. Amiri, D. Gunduz, S. R. Kulkarni, and H. V. Poor. Convergence of update aware device scheduling for federated learning at the wireless edge. *IEEE Transactions on Wireless Communications*, 20(6):3643–3658, June 2021.

[181] W. T. Chang and R. Tandon. Communication efficient federated learning over multiple access channels. January 2020. Available on arXiv:2001.08737.

[182] B. Buyukates and S. Ulukus. Timely distributed computation with stragglers. *IEEE Transactions on Communications*, 68(9):5273–5282, September 2020.

[183] E. Ozfatura, B. Buyukates, D. Gunduz, and S. Ulukus. Age-based coded computation for bias reduction in distributed learning. In *IEEE Globecom*, December 2020.

[184] H. H. Yang, A. Arafa, T. Q. S. Quek, and H. V. Poor. Age-based scheduling policy for federated learning in mobile edge networks. In *IEEE ICASSP*, May 2020.

[185] J. Konecny, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, December 2016.

[186] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the parameter server. In *USENIX Conference on Operating Systems Design and Implementation*, October 2014.

[187] R. K. Maity, A. Singh Rawat, and A. Mazumdar. Robust gradient descent via moment encoding and LDPC codes. In *IEEE ISIT*, July 2019.

[188] S. Li, S. M. M. Kalan, Q. Yu, M. Soltanolkotabi, and A. S. Avestimehr. Polynomially coded regression: Optimal straggler mitigation via data encoding. January 2018. Available on arXiv: 1805.09934.

[189] Q. Yu, M. Maddah-Ali, and A. S. Avestimehr. Polynomial codes: an optimal design for high-dimensional coded matrix multiplication. In *NIPS*, December 2017.

[190] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr. Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding. In *IEEE ISIT*, June 2018.

[191] S. Dutta, Z. Bai, H. Jeong, T. M. Low, and P. Grover. A unified coded deep neural network training strategy based on generalized polydot codes. In *IEEE ISIT*, June 2018.

[192] H. Park, K. Lee, J. Sohn, C. Suh, and J. Moon. Hierarchical coding for distributed computing. In *IEEE ISIT*, June 2018.

[193] S. Kiani, N. Ferdinand, and S. C. Draper. Exploitation of stragglers in coded computation. In *IEEE ISIT*, June 2018.

[194] A. Mallick, M. Chaudhari, and G. Joshi. Fast and efficient distributed matrix-vector multiplication using rateless fountain codes. In *IEEE ICASSP*, May 2019.

[195] Y. Yang, M. Interlandi, P. Grover, S. Kar, S. Amizadeh, and M. Weimer. Coded elastic computing. In *IEEE ISIT*, July 2019.

[196] H. Park and J. Moon. Irregular product coded computation for high-dimensional matrix multiplication. In *IEEE ISIT*, July 2019.

[197] R. Bitar, Y. Xing, Y. Keshtkarjahromi, V. Dasari, S. El Rouayheb, and H. Seferoglu. Private and rateless adaptive coded matrix-vector multiplication. *EURASIP Journal on Wireless Communications and Networking*, 2021(1):15, January 2021.

[198] Y. Sun, J. Zhao, and D. Gunduz. Heterogeneous coded computation across heterogeneous workers. In *IEEE Globecom*, December 2019.

[199] B. Hasircioglu, J. Gomez-Vilardebo, and D. Gunduz. Bivariate polynomial coding for exploiting stragglers in heterogeneous coded computing systems. January 2020. Available on arXiv:2001.07227.

[200] M. Ye and E. Abbe. Communication-computation efficient gradient coding. In *ICML*, July 2018.

[201] W. Halbawi, N. Azizan, F. Salehi, and B. Hassibi. Improving distributed gradient descent using Reed-Solomon codes. In *IEEE ISIT*, June 2018.

[202] J. Zhang and O. Simeone. LAGC: Lazily aggregated gradient coding for straggler-tolerant and communication-efficient distributed learning. *IEEE Transactions on Neural Networks and Learning Systems*, 32(3):962–974, March 2021.

[203] S. Kadhe, O. O. Koyluoglu, and K. Ramchandran. Gradient coding based on block designs for mitigating adversarial stragglers. In *IEEE ISIT*, July 2019.

[204] H. Wang, S. Guo, B. Tang, R. Li, and C. Li. Heterogeneity-aware gradient coding for straggler tolerance. In *IEEE ICDCS*, July 2019.

[205] L. Tauz and L. Dolecek. Multi-message gradient coding for utilizing non-persistent stragglers. In *Asilomar Conference*, November 2019.

[206] R. Bitar, M. Wootters, and S. El Rouayheb. Stochastic gradient coding for straggler mitigation in distributed learning. *IEEE Journal on Selected Areas in Information Theory*, 1(1):277–291, May 2020.

[207] N. Charalambides, M. Pilanci, and A. O. Hero. Weighted gradient coding with leverage score sampling. In *IEEE ICASSP*, May 2020.

[208] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis. Gradient coding from cyclic MDS codes and expander graphs. *IEEE Transactions on Information Theory*, 66(12):7475–7489, December 2020.

[209] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz. Revisiting distributed synchronous SGD. April 2016. Available on arXiv:1604.00981.

[210] S. Li, S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi. Near-optimal straggler mitigation for distributed gradient methods. In *IEEE IPDPS*, May 2018.

[211] S. Dutta, G. Joshi, S. Ghosh, P. Dube, and P. Nagpurkar. Slow and stale gradients can win the race: Error-runtime trade-offs in distributed SGD. In *AISTATS*, April 2018.

[212] N. Ferdinand and S. C. Draper. Anytime stochastic gradient descent: A time to hear from all the workers. In *Allerton Conference*, October 2018.

[213] A. Behrouzi-Far and E. Soljanin. On the effect of task-to-worker assignment in distributed computing systems with stragglers. In *Allerton Conference*, October 2018.

[214] M. Mohammadi Amiri and D. Gunduz. Computation scheduling for distributed machine learning with straggling workers. *IEEE Transactions on Signal Processing*, 67(24):6270–6284, December 2019.

[215] E. Ozfatura, S. Ulukus, and D. Gunduz. Straggler-aware distributed learning: Communication computation latency trade-off. *Entropy, Special Issue on the Interplay Between Storage, Computing, and Communications from an Information-Theoretic Perspective*, 22(5):544, May 2020.

[216] E. Ozfatura, S. Ulukus, and D. Gunduz. Coded distributed computing with partial recovery. July 2020. *IEEE Transactions on Information Theory,* to appear. Available on arXiv:2007.02191.

[217] P. Jiang and G. Agrawal. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In *NIPS*, December 2018.

[218] B. Buyukates and S. Ulukus. Age of information with Gilbert-Elliot servers and samplers. In *CISS*, March 2020.

[219] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms.* Springer, 2008.

[220] M. Costa, S. Valentin, and A. Ephremides. On the age of channel information for a finite-state Markov model. In *IEEE ICC*, June 2015.

[221] G. D. Nguyen, S. Kompella, C. Kam, and J. E. Wiselthier. Information freshness over a Markov channel: The effect of channel state information. *Ad Hoc Networks*, 86:63–71, April 2019.

[222] D. Ploger, M. Segata, R. Lo Cigno, and A. Timm-Giel. Markov-modulated models to estimate the age of information in cooperative driving. In *IEEE VNC*, December 2019.

[223] B. Buyukates, M. Bastopcu, and S. Ulukus. Age of gossip in networks with community structure. In *IEEE SPAWC*, September 2021.

[224] B. Buyukates, M. Bastopcu, and S. Ulukus. Version age of information in clustered gossip networks. September 2021. Available on arXiv:2109.08669.

[225] B. Buyukates and S. Ulukus. Timely updates in distributed computation systems with stragglers. In *Asilomar Conference*, November 2020.

[226] B. Buyukates and S. Ulukus. Timely communication in federated learning. In *IEEE Infocom*, May 2021.

[227] B. Buyukates, E. Ozfatura, S. Ulukus, and D. Gunduz. Gradient coding with dynamic clustering for straggler mitigation. In *IEEE ICC*, June 2021.

[228] B. Buyukates, E. Ozfatura, S. Ulukus, and D. Gunduz. Gradient coding with dynamic clustering for straggler-tolerant distributed learning. August 2021. Available on arXiv: 2103.01206.