

Abstract

Title of thesis: TOPOLOGY RECONFIGURATION

WITH SUCCESSIVE APPROXIMATIONS

Degree Candidate: Eswaran Baskaran

Degree and Year: Master of Science, Fall 2006

Thesis directed by: Dr. Christopher Davis

Professor, Department of Electrical and Computer Engineering

University of Maryland

Recent technologies have enabled the formation of point-to-point directional wireless networks that are capable of dynamic changes in the network topology. The process of changing this topology in response to changes in available link capacities and load demands of various nodes is called topology control. One example of the type of communication network studied in this context is a Free Space Optical (FSO) network.

Topology control consists of computing new topologies to dynamically optimize the network under changing traffic conditions and then carrying out the reconfiguration process to achieve the target topology. This thesis considers the process of topology reconfiguration and use the packet drops that happen during this process as a cost metric for this process. It is shown that by implementing the topology reconfiguration as a series of smaller steps (successive approximation), the number of packets that are dropped during the reconfiguration are reduced. Using this knowledge, the topology computation algorithm can be refined to also minimize the reconfiguration cost along with the typical objective of minimizing congestion.

TOPOLOGY RECONFIGURATION WITH
SUCCESSIVE APPROXIMATIONS

by

Eswaran Baskaran

Thesis 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
Master of Science
2006

Advisory Committee:
Professor Christopher Davis, Chair
Professor Stuart Milner
Professor Mark Shayman

Acknowledgments

First and foremost, I would like to thank Dr. Christopher Davis (Professor, University of Maryland) and Dr. Stuart Milner (Research Professor, University of Maryland) for rendering valuable guidance and giving me an opportunity to work on an interesting problem. I also thank Dr. Mark Shayman (Professor, University of Maryland) for serving on the thesis committee. In particular, I thank Dr. Mehdi Kalantari (Research Scientist, University of Maryland) for his valuable insights into the research problem and for serving on the thesis committee. I cannot thank Aniket Desai and Jaime Llorca enough for the countless times their observations and help during our discussions have assisted in the progress of this research. It was truly a joy working with Aniket, Jaime and Shawn Ho in and out of the lab. I would like to acknowledge the contributions from Ashwin Swaminathan, Karthik Ravirajan and Kiran Kumar in helping me gain significant insights into the problem. Finally, I thank my roommates Deepak Sridharan, Rajesh Sathiyarayanan and Vidya Ramanan Ganesan and my almost-a-roommate Balaji Vasani for supporting me when I was writing the thesis.

Table of Contents

List of Tables	iv
List of Figures	iv
1 Introduction	1
1.1 FSO/RF Networks	2
1.2 Topology Control of FSO/RF Networks	4
1.3 Overview	7
2 Background and Related Work	9
2.1 Static Topology Design	11
2.1.1 Topology Control in RF Networks	11
2.2 Scalable Congestion Minimization Heuristics	17
2.3 Dynamic Topology Reconfiguration	21
2.3.1 Reconfiguration Cost	22
2.4 Summary	23
3 Successive Approximation	25
3.1 Motivation	25
3.2 Problem Model	28
3.3 Branch Exchange Sequence Generation	32
4 Topology Reconfiguration with Successive Approximations	35
4.1 Topology Reconfiguration Architecture	35
4.2 An Example	35
4.3 Problem Formulation	38
4.3.1 Reconfiguration Cost	40
4.3.2 Congestion Cost	41
4.4 Heuristics	43
5 Simulation Results	46
5.1 Simulation Methodology	46
5.2 Traffic Demand Generation	50
5.3 Simulation Setup and Verification	53
5.4 Rollout - Singlehop	58
5.5 Branch Exchange	68
6 Summary and Conclusions	75
A Pseudocode for Successive Approximations	77
Bibliography	81

List of Tables

5.1	Possible Ring topologies and the corresponding costs	57
5.2	Topology Recomputations - Averages from five simulation runs using Rollout heuristic	62
5.3	Topology Recomputations - Averages from five simulation runs using Branch Exchange heuristic	71

List of Figures

1.1	Timeline of the Topology Reconfiguration Process	2
1.2	Autonomous Topology Control Process	5
1.3	Topology Control Architecture	6
2.1	An example of a Branch Exchange	10
2.2	Topology Control in RF networks	12
3.1	Number of flows affected during Topology Reconfiguration	26
3.2	Bounds on the number of affected flows. $N=100$	27
3.3	Upper bound of the possible Gain. $N=100$	31
3.4	Upper bound of the possible G - Achievability $N=100$	34
4.1	Topology Control Architecture with Successive Approximations	36
4.2	Traffic Matrix and link loads in a 5-node network	36
4.3	Target Topology with (a) optimal congestion (b) sub-optimal congestion	37
5.1	Illustration of the DTCN with the Simulated Network	48
5.2	Total Number of Packets Sent in the Network	51
5.3	Packet drops due to Traffic imbalance	53

5.4	Dynamic Topology Control Node Process Model	54
5.5	Average Packet Drops with Rollout - Singlehop heuristic with and without Successive Approximation - Average from 5 simulation runs .	59
5.6	Average Reconfiguration Drops with Rollout - Singlehop heuristic with and without Successive Approximation - Average from 5 simulation runs	60
5.7	Reconfiguration Activity with (a) Rollout (singlehop) (b) Rollout (singlehop) with Successive Approximations - Traffic Pattern as in Figure 5.2	61
5.8	Average Congestion Drops with Rollout - Singlehop heuristic with and without Successive Approximation - Average from 5 simulation runs	64
5.9	Total Packets Sent and Average Packet Drops with traffic patterns (a) 1 and (b) 2 with the Rollout heuristic	65
5.10	Total Packets Sent and Average Packet Drops with traffic patterns (a) 3, (b) 4 and (c) 5 with the Rollout heuristic	66
5.11	Average Packet Drops with Branch Exchange heuristic with and without Successive Approximations - Average from 5 simulation runs . . .	68
5.12	Average Reconfiguration Drops with Branch Exchange heuristic with and without Successive Approximations - Average from 5 simulation runs	69
5.13	Average Congestion Drops with Branch Exchange heuristic with and without Successive Approximations - Average from 5 simulation runs	70
5.14	Total Packets Sent and Average Packet Drops with traffic patterns (a) 1, (b) 2 and (c) 3 with Branch Exchange heuristic	73
5.15	Total Packets Sent and Average Packet Drops with traffic patterns (a) 4 and (b) 5 with Branch Exchange heuristic	74

Chapter 1

Introduction

Wireless networks have an advantage over wireline networks in that they can be reconfigured to meet new requirements. Reconfiguration here means the logical - and physical - change in the network topology[1]. However, not all cases of topology reconfiguration in wireless networks are useful in the sense of meeting new network requirements. For example, in a network of mobile wireless nodes that move in an uncontrolled fashion, topology reconfiguration happens as a matter of fact and cannot be used as a tool for optimizing network performance. In a similar vein, there exist wireline networks in which topology reconfiguration is possible. An example involves Wavelength Division Multiplexing (WDM) networks in which different logical networks can be overlaid on top of a fixed physical network topology based on transmitter-receiver frequency tuning.

Reconfigurable networks also differ based on the order of link establishment/removal. In Mobile Ad-hoc Networks(MANETs) for example, the removal of unwanted links could happen after new links have been established. In the case of point-to-point directional wireless networks, like *Free Space Optical*(FSO) networks, the degree constraints imposed on the nodes imply that unwanted links have to be removed before new links can be created. Whenever this is this case, the network topology will be in a transient state after the removal of unwanted links and before the es-

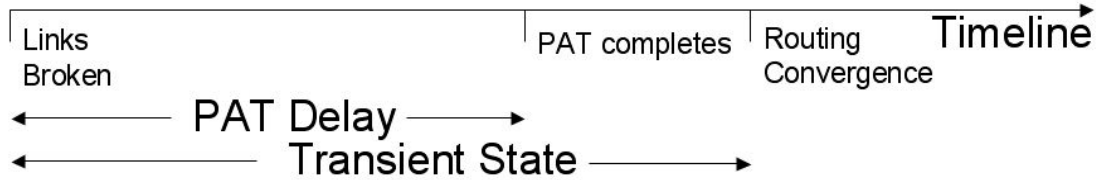


Figure 1.1: Timeline of the Topology Reconfiguration Process

establishment of new links. This transient state could exist for a small time interval, but it nevertheless exists and results in suboptimal network performance during this time. In this study, we are interested in such reconfigurable networks where a topology reconfiguration imposes a cost on the network performance.

1.1 FSO/RF Networks

FSO/RF networks are point-to-point broadband wireless networks that use combined or switchable optical and RF transceivers. In such communication networks, a laser beam can be accurately pointed to a receiver upto thousands of kms away. Reconfiguration in FSO/RF networks involves the physical re-alignment of the laser and RF transceiver assembly towards different nodes so that new links are established after the old links are destroyed. The new links are acquired after a process called Pointing, Acquisition and Tracking (PAT), The network is in a transient state during PAT - the time taken for PAT is called the PAT Delay.

Figure 1.1 shows the timeline of the topology reconfiguration process in a FSO/RF network. The transient state of the network ends when the routing protocol discovers the new topology and converges to the new routing table. However, as

we will see below, the new routing table can be predicted at the beginning of the topology reconfiguration and hence the additional time taken by the routing protocol convergence can be reduced to zero. This still leaves the network in a transient state during the PAT delay which is (almost) invariant to the number of links that are being created - as each link discovery is essentially independent of the establishment of other new links.

Another example of a network with a similar reconfiguration process is a WDM network. In a WDM network, network stations are physically connected to, and communicate over, a passive optical medium and each network station is equipped with a high speed user access port and a small number of transmitters and receivers that tap into the fiber transmission medium. The assignment of wavelengths to transmitter/receiver pairs defines a logical connection diagram among the stations embedded in the physical network topology. The multihop approach is used to route traffic from source to destination along the various links of the connection diagram (each link corresponding to one of the WDM channels), with each station providing traffic relay in addition to user access. Since all channels are wavelength multiplexed onto a common optical medium, the logical connection diagram and the topology of the medium are independent. The use of slowly tunable transmitters and/or receivers over a large fraction of the optical band allows the logical connectivity among network stations to be changed, independently of the fiber medium, in response to varying network conditions such as traffic patterns and station failures. The time taken for retuning such transmitters is in the order of 500 ms[6]. As can be inferred, the model of the reconfiguration process of the FSO/RF network outlined earlier

applies faithfully to the WDM networks as well[2] with the PAT delay of the FSO network being replaced by the retuning delay of the WDM network.

1.2 Topology Control of FSO/RF Networks

In such free space directional wireless networks, *Topology Control* is the rapid changing of the topology in response to changes in the atmospheric obscuration (effective link capacities) or demands at the various nodes. Atmospheric obscuration or occlusion in the path between a FSO/RF transmitter and the receiver causes the Bit Error Rate (BER) of that link to increase. This is analogous to a reduction in the effective data-rate of that link. This might cause congestion in some of the links. Alternatively, the traffic demand might change over time and could result in congestion in some of the links because of an imbalance in the traffic demand. The main challenge here is the dynamic, autonomous reconfiguration both in hardware and software in order to maximize communications availability and capacity in the network. Topology Control is the ability to optimize the network topology according to changing traffic demand, changing physical environment or both.

Topology control is a multilayer approach and involves tracking and acquisition of nodes, assessment of link-state information, collection and distribution of topology data, and the algorithmic solution of an optimal topology. A core component of the topology control process is the algorithmic decision making process by which a topology change is to be made. At the physical layer, a cost measure can be defined in terms of bit-error-rate and at the network layer the cost measure is typically

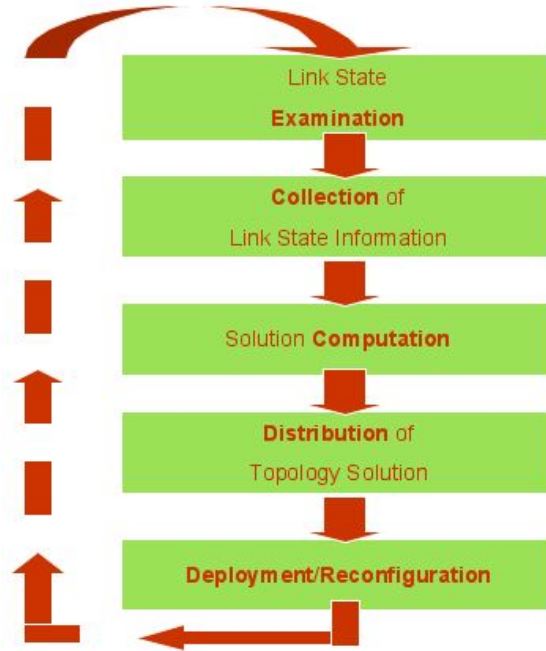


Figure 1.2: Autonomous Topology Control Process

congestion, flow-rate or end-to-end delay given the traffic demand matrix. The processes involved are depicted in figure 1.2(extracted from [33])

The solution computation, link assessment and topology dissemination are all executed by a Designated Topology Control Node (DTCN)[1] that interfaces with the TCP/IP stack as shown in the topology control architecture (figure 1.3). In short, the DTCN takes the current network topology as input (based on Link State Updates) and other relevant parameters like the traffic demand matrix and calculates the optimal network topology. This computation can be shown to be NP-complete and therefore several heuristics are used to compute the target network topology. We will show in our work that the heuristics previously proposed[19] have not taken the topology reconfiguration cost into account. Taking this cost into account will

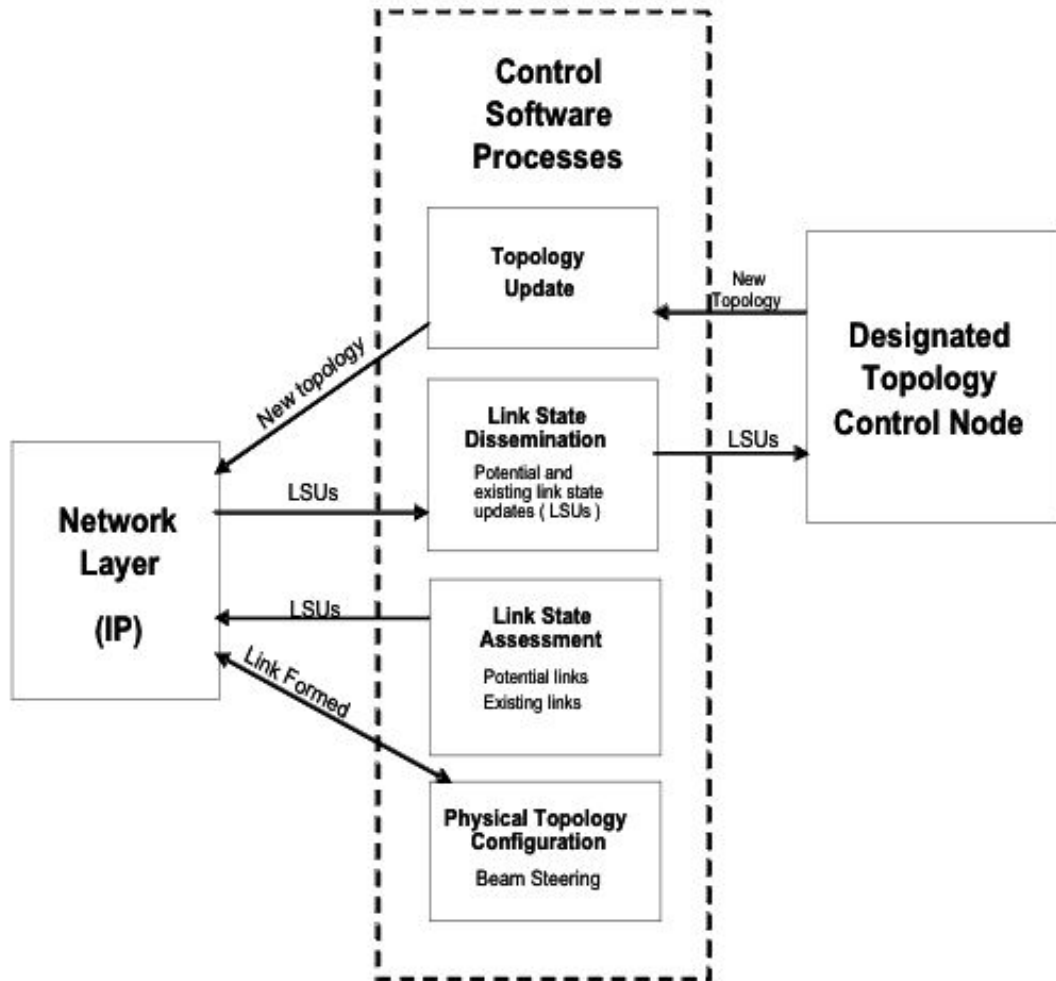


Figure 1.3: Topology Control Architecture

result in solutions superior to existing solutions. The setting we have in this work is a ring network - each node in the network has a degree of 2 and any topology reconfiguration removes and adds an equal number of links so that a new ring is formed at the end of the topology reconfiguration. In other words, we will develop methods to

1. characterize the reconfiguration cost and calculate the minimum possible reconfiguration cost in a setting and
2. incorporate this characterization into the heuristics that the DTCN uses to calculate the optimal network topology.

1.3 Overview

The overview of the rest of the chapters is as follows.

- Chapter 2 discusses the background of this problem and related work. We first look at the heuristics proposed for calculating the optimal topology for FSO/RF networks. Related problems in WDM networks are considered next and we show how the approach used there doesn't quite solve the problem for FSO/RF networks.
- Chapter 3 focusses on characterizing the topology reconfiguration cost and shows that the minimum reconfiguration cost is achieved when the target topology is achieved in successive approximations.
- Chapter 4 modifies the optimal topology computation problem by taking the

reconfiguration into account and proposes new heuristics based on this model.

- Chapter 5 describes the simulation methodology we used to evaluate the proposed heuristics and discusses the obtained results.
- Chapter 6 summarizes and concludes the report.

Chapter 2

Background and Related Work

An autonomous topology control scheme for a reconfigurable network should answer the following questions

1. What should be the target topology for the reconfiguration?
2. When to trigger the reconfiguration?
3. How to perform topology reconfiguration to achieve a target topology?

Several previous studies([3], [4], [5]), have been carried out on the problem of designing an optimal topology given the traffic demand and the network conditions. This is called the static topology design problem, so called because while forming an optimal topology, calculations can be done offline because no parameters are dynamically varying. This problem is computationally intensive and several heuristics have been developed[19] in the literature to obtain the optimal topology in real-time.

Introducing dynamicity in the network operations entails solving problems (2) and (3). In particular, several previous studies([6], [7], [8]) have attempted to solve (1) and (2) together by trading off between the resource utilization and the disruption in the traffic caused by the reconfiguration. This introduces the notion of cost of reconfiguration and we will see how previous work has failed to take into account the differences between various schemes of reconfiguration to calculate the



Figure 2.1: An example of a Branch Exchange

reconfiguration cost. The scheme of reconfiguration is specified by problem (3) and an obvious method to reconfigure is to delete all unwanted links and create the new links simultaneously. In the case of FSO networks, this would mean all the relevant transceivers start their PAT operation simultaneously and in the case of WDM networks, this would mean all the transmitters and receivers are retuned at the same time. However, this operation would disrupt a major portion of the network. Another possibility - in the other extreme - is to carry out the reconfiguration by obtaining the sequence of exchanges of two links (called a branch-exchange) to migrate from the current topology to the target topology[2]. This scheme, again, suffers from a long transition period and hence, resources are inefficiently utilized for a long duration. An example of a branch exchange is given in 2.1 - the links between node pairs (1,2) and (3,4) are exchanged in this example.

2.1 Static Topology Design

The static topology design problem has involved different approaches in RF networks and in the point-to-point networks in which these problems first appeared.

2.1.1 Topology Control in RF Networks

The topology control schemes in RF networks primarily yield topologies with reduced interference between adjacent nodes. Since omni-directional antennas are usually used to transmit data, interference effects are prominent in RF ad-hoc networks. Thus controlling the transmitter power to a minimum level while maintaining a connected topology usually increases the throughput of the network. Hence almost all RF topology control algorithms focus on reducing the average degree of nodes in the network (See 2.2[9]). One important feature of RF networks is that as a node increases its power, it induces more edges in the graph (by connecting to nodes which are farther from it). However, while communicating to any specific node, it causes interference at all other nodes to which it is connected. The average degree of a node is the number of incident edges (or the number of nodes to which it is connected) on that node. Thus if the average node-degree is smaller, there will be less interference in the network. To reduce the average node degree, only the node power is reduced, which is why all RF topology control algorithms are power-control algorithms.

Studies that have been done to-date in RF topology control include Ramanathan et al. [10], Wattenhofer et al. [11], Li - Hou et al. [12] and V. Rodoplu

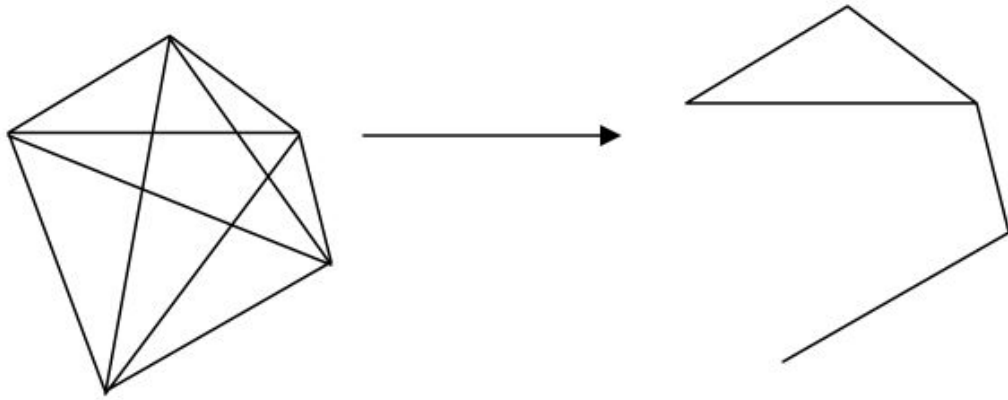


Figure 2.2: Topology Control in RF networks

et al. [13]. In Ramanathan et al.'s approach, a centralized algorithm is given to form a topology that solves the *CMP* (*Connected Min-Max Power*) problem. In their problem formulation, nodes are distributed in space and power assignments are done so that the maximum power spent by a node is minimized across all possible connected topologies. A connected topology is one where removing at least one link might cause a disconnection in the network topology. The algorithm used is a simple greedy algorithm [14] and uses a clustering mechanism on the list of node-pairs organized in non-decreasing order of physical distance. It was shown that this algorithm, after removing redundant edges, solves the CMP problem and hence minimizes the average node degree.

This earlier work was followed by some advanced versions [11] and [12] in which distributed algorithms were found to achieve objectives similar to [10]. The challenge in forming an efficient topology by distributed algorithms is that the topology must

be connected globally and be power-efficient at the same time, but it is assumed that the nodes only have local knowledge. The simplifying assumption that is used is that all nodes are considered to be in a 2D space. Under this assumption, Wattenhofer et al.([11]) showed that if a node finds a neighbor in every cone of the 2-D space, the topology will be connected. Li et al.'s approach [12] was to let each node build its own local spanning tree to build a minimum spanning tree in a distributed manner (A Spanning Tree topology is one where each node in the graph is connected to any other node by exactly one path). The main problems with all of these approaches is that there are no models to predict the throughput of a network based on the average node degree. In fact, reducing the average node-degree means average packet hop count increases and this could lead to reduced throughput. Also Gupta and Kumar have delineated certain fundamental limitations of RF networks[15]. They have shown that the throughput per node in these networks falls proportional to $\frac{1}{\sqrt{N}}$ (N being the total number of nodes). Due to the prominent interference effects in RF network, they do not perform as well as a base-station oriented architecture, as shown by Milner et al.([16]). In this base-station oriented architecture, all the nodes connect to the base stations in one hop and the base-stations are connected with a point-to-point wireless network, FSO connections providing one example.

Topology design of a point-to-point wireless network has occurred in different contexts in the literature before. One such example is the LAN design problem [17][18]. In this problem, the traffic demand matrix is assumed to be known. That is, the average traffic rate expressed in bps from one LAN to any other LAN is known *a priori*. The average end-to-end delay between any two LANs is constrained below a

specific value and the objective is to connect LANs with bridges so that the network of LANs and bridges form a Spanning Tree topology. The optimizing function is chosen to be a cost function of the bridges and this has to be minimized. This is a *nonlinear integer-programming* problem where the real-variables are average flows on the bridges and the integer variables are b_{kl} , which are either 1 or 0 ($b_{kl} = 1$ if a bridge is put between LANs k and l). Since interconnecting several LAN's is accomplished by laying out a spanning tree, routing is trivialized as there exists only one path between any two LANs. The main limitation of this approach is that the network is only minimally connected and any link failure results in network partition. To design a more robust topology, the problem has to be generalized to include topologies that are not just a spanning tree. A natural formulation of this problem follows the approach of Desai et al.([19]).

Variables:

- R_{sd} is the average flow-rate of traffic from source s to destination d .
- All links are bidirectional and maximum node degree is d .
- c_{ij} is the capacity of the potential link between nodes i and j .
- λ_{ij}^{sd} is the average flow-rate from source s to destination d on link (i, j) .
- b_{ij} is the decision variable - 1 if a link exists between i and j and 0 if it does not.

Assuming M/M/1 model for the queueing, the problem can be formulated as minimizing the end-to-end delay.

$$\min \sum_{all(i,j)} b_{ij} \left\{ \frac{c_{ij}}{c_{ij} - \sum_{(s,d)} \lambda_{ij}^{sd}} \right\} \quad (2.1)$$

Constraints:

$$\sum_j \lambda_{ij}^{sd} - \sum_j \lambda_{ji}^{sd} = R_{sd} \text{ if } s = i \quad (2.2)$$

$$= -R_{sd} \text{ if } d = i \quad (2.3)$$

$$= 0 \text{ otherwise} \quad (2.4)$$

$$\lambda_{ij}^{sd} < b_{ij} R_{sd} \forall i, j, s, d \quad (2.5)$$

$$\sum_{(s,d)} \lambda_{ij}^{sd} < c_{ij} \forall i, j \quad (2.6)$$

$$b_{ij} \in \{0, 1\} \quad (2.7)$$

$$\sum_j b_{ij} = d \quad (2.8)$$

$$\sum_{(i,j)} c_{ij} = c \quad (2.9)$$

The above problem is a Mixed Integer-Programming (MIP) problem. It can be observed that the number of flow-rate variables is $O(N^4)$. The number of integer variables is $O(N^2)$ where N is the number of nodes. This is because there are $O(N^2)$ possible links and $O(N^2)$ possible source-destination (SD) pairs; making the flow variables $O(N^4)$. It is also observed that the problem simultaneously attempts to solve routing (flow-rate variables λ) and topology design(integer variables b). This problem is computationally difficult to solve, and further distributed implementation of this approach is extremely difficult. Also, the node-degrees can be different in the optimal solution and the topology may not be regular. To make the solution computationally easier, the problem formulation can be modified to use shortest

hop routing. Even though this simplifies the problem, the problem remains NP-complete. The most computationally efficient heuristic is based on LP-relaxation and is described by Ramaswamy et al.[20].

The heuristics for *congestion* minimization problem in the literature can be broadly classified into 2 categories as *link insertion* and *link deletion*. In link insertion heuristics, the idea is to insert the links according to some sequence whenever interfaces are available. In link deletion heuristics, the starting point is a maximally connected graph, and links are removed according to some sequence until no node has more links attached to it than its available number of interfaces. This insertion-deletion philosophy is discussed by E. Leonardi et al [21]. The algorithms they have proposed are based on representing the topology as a bipartite graph and solving *1-maximal weight matching problem* repeatedly until all the nodes are unmatchable. These algorithms have a complexity of $O(N^4 \log N)$ and $O(N^5 \log N)$ respectively, which can be compelling for a large number of nodes. It is also pointed out that in the insertion case, the algorithm doesn't guarantee connectivity (and bi-connectivity) and in the deletion case, it doesn't guarantee feasibility because the node degree can be higher than the number of available interfaces. Both of these are drawbacks[21]. It is possible to modify these algorithms to reduce their computational complexity and guarantee biconnectivity and feasibility simultaneously. In this case, instead of adding multiple links in one round, links are added one by one. One classic example is the *Heuristic Logical (Topology) Design Algorithm (HLDA)*[20], which is extremely fast but yields poor quality solutions, since it only considers single hop flows. It is possible to modify it to take into consideration multihop flows as we will

see in the next section.

Heuristics can also be distinguished based on their degree of greediness. One approach to forming a topology is based on selecting a predetermined order of the source-destination (SD) flows and inserting links one by one. However it is possible to change the order of SD pairs to yield better quality solutions, by a technique called rollout [22]. However large computational complexity is a serious shortcoming of such a technique (We estimate that for ring topologies, it is at least $O(N^6)$). Finally, heuristics can be designed to make incremental changes in an already established topology as opposed to a complete reconfiguration of a topology. Narula-Tam[23] et al. described a simple algorithm based on branch exchange techniques.

2.2 Scalable Congestion Minimization Heuristics

The congestion minimization problem for ring topologies can be formulated as in eqn (2.1), except that the objective function to minimize is the congestion on the maximally congested link[19].

$$\min \max_{(i,j)} b_{ij} \sum_{(s,d)} \lambda_{ij}^{sd} \tag{2.10}$$

and the degree for each node is constrained as 2. Again, if no assumption is made about the routing scheme, this formulation attempts to find the optimal topology and optimal routing at the same time. Modifying the problem by selecting shortest path routing reduces the problem to the optimal linear ordering problem, which has proven to be NP-complete [24]. Desai et al.[19] evaluated several heuristics that minimize the congestion in terms of their scalability properties. These heuristics

were derived from the well-known multihop, rollout and branch exchange heuristics and were applied to minimize congestion in ring topologies.

- Single-hop:

The single-hop heuristic is designed to connect SD pairs with heavy traffic by a single hop. The single-hop flow(given below) is maximized with this heuristic.

$$\sum_i \sum_j b_{ij} R_{ij} \tag{2.11}$$

It was proven in [21] that matching theory can be used to solve this problem. However, as the solution could be a disconnected set of rings, the HLDA heuristic was proposed in [20]. The heuristic inserts links between source destination pairs beginning with the pair with the most traffic and working down the sorted list of source destination pairs - sorted according to the traffic between them. Though the complexity of the heuristic is less ($O(N^3)$), it is not very effective because multi-hop traffic is not considered.

- Multi-hop:

Desai et al. [19] developed a heuristic to take multihop traffic into account based on related development in [21]. In this heuristic, SD pairs are again ordered in decreasing order of traffic between them. In the single-hop heuristic if a direct connection between a SD pair is not possible, that entry is skipped and the next SD pair in the ordered list is considered. However in this heuristic, an attempt is made to create a multihop path from the source to the destination by adding a link across some other SD pair in the already partially connected

network. The computational complexity for this heuristic is again $O(N^3)$.

- Rollout:

Rollout heuristics were suggested in [22] to improve the performance of the congestion minimization heuristics described above. A description of the rollout heuristic is excerpted from [19] here.

1. Sort the SD pair list in the order of non-decreasing magnitude of traffic. Let this order be called as $\{SD[0], SD[1], \dots, SD[M-1]\}$, where M represents the number of SD pairs.
2. Create M topologies by SD-indexing. SD-indexing means that some SD index is fixed as the first index to be considered and the rest of the indices will be taken in the decreasing magnitude of traffic. Sub-steps a and b illustrate this point.
 - (a) Create a topology $T[0]$ by either single-hop or multi-hop heuristic using the order: $\{SD[0], SD[1], \dots, SD[M-1]\}$. Note that this topology is the same as the single-hop/multi-hop heuristic would obtain.
 - (b) Create $T[1]$ by using following order: $SD[1], SD[0], SD[2], \dots, SD[M-1]$ and so on.
3. Choose $SD[k]$ to be the first index, which finds topology $T[k]$ with minimum congestion. Add a link by single-hop/multi-hop heuristic to create a path between $s[k]$ (source) and $d[k]$ (destination).
4. Repeat steps 1-3 until the ring topology is completed. If the index $SD[j]$

cannot be used to add a link in the topology, pass to the next index.

The complexity of this algorithm was shown to be $O(N^5)$ after restricting the running of steps 1-3 to only a few times.

- Branch Exchange:

The branch exchange algorithm can be used to improve upon the congestion performance of an existing ring[23]. In case of ring topologies, it can be noted that if we exchange 2 non-node-sharing links with another pair, a new ring can be obtained. In a given ring, $\frac{N(N-3)}{2}$ such exchange sequences are possible to obtain a new ring. A simple branch exchange algorithm will create $\frac{N(N-3)}{2}$ new ring topologies from a given ring and pick the one with minimum congestion. It was shown that the computational complexity of this algorithm is $O(N^5)$, but its average running time was much less than that of rollout algorithms as subsequent branch exchanges do not result in significant improvement in performance.

It was shown that while *Rollout(multi-hop)* took the longest time to execute, it produced the most effective improvement in end-to-end delay among the heuristics evaluated. Generating a topology using the multihop heuristic and then running the branch exchange heuristic 3 times over the obtained topology (*Multihop + Branch Exchange (3 iterations)*) was found to have comparable performance to the rollout(multi-hop) heuristic but with significantly lesser running time.

2.3 Dynamic Topology Reconfiguration

Introducing dynamicity into the network that has to be autonomously reconfigured brings us the problems (2) and (3) mentioned at the beginning of this chapter - when do we trigger the reconfiguration and how do we achieve the reconfiguration. An example of a study that solves problem (2) is [25] where the time to reconfigure is determined when the improvement in resource utilization metric outweighs the reconfiguration cost. Several studies, particularly for WDM networks, have attempted to solve the problem of determining when to reconfigure together with the problem of determining the target topology. In [8] for example, a Mixed Integer Linear Programming (MILP) formulation for the target topology best suited to the changed traffic matrix and achievable by minimal disruption to the existing topology is presented. No tradeoff between the resource utilization and traffic disruption is given in this formulation, however. A more flexible model that constrains the number of changes from the existing virtual topology to a target virtual topology is presented in [26].

If the network state is considered to be the current topology and the traffic matrix, a change in the traffic matrix changes the state of the system and a decision has to be taken if the current topology has to be reconfigured to make it nearly optimal for the new traffic matrix. The set of all such decisions for all network states constitutes a reconfiguration policy [27]. If the future is completely predictable, then the reconfiguration decisions can be computed beforehand for all possible states based on the reconfiguration policy. For example, a topology that optimizes for

the sequence of traffic matrices instead of just the current traffic matrix can be calculated, totally avoiding the need for reconfiguration in the future. A Markov decision process formulation is used in [6] where each state is given by a two tuple (load balancing metric, cost of reconfiguration). The probability of transition from one state to another is calculated based on previous traffic patterns. The virtual topology can either be reconfigured to achieve near optimal load balancing or remain unchanged and take a transition to another state. A drawback with this approach is that the topology is designed to be optimal for the initial traffic matrix even if it is known that a reconfiguration will be not be carried out if the traffic matrix changes. The transition probabilities can also be learned when the network is in operation, but this would result in re-computing the reconfiguration decisions based on the updated transition probabilities. Sinha and Murthy[28] propose an information theoretic approach where the design of the topology is optimized for a set of traffic matrices based on either the fully predicted series of traffic matrices or the estimated series of traffic matrices. A sequence of such traffic matrices are clustered as a set and the topology design is optimized over this set. Again, the expectation that the entire series of traffic matrices will be known is an onerous requirement and neither is it easy to probabilistically predict the traffic matrices based on stochastic processes.

2.3.1 Reconfiguration Cost

The success of all of the reconfiguration schemes discussed above are predicated on the accuracy of the reconfiguration cost estimates. [26], [27], [7] and [29] treat

the reconfiguration cost as the total number of lightpaths being added and removed during the reconfiguration. Though this simplistic model effectively compares the reconfiguration cost in migrating to two different virtual topologies, it fails to differentiate between the various possible reconfiguration schemes that can be used. For example, it might be easier and cheaper to migrate to a specific target topology (as compared to other target topologies) over a series of branch-exchanges[2] but the same target topology might have higher (as compared to other topologies) reconfiguration cost - measured in terms of packets lost - if the reconfiguration is done in one step. The packet drops that happen due to incorrect routing is taken to be the reconfiguration cost in [8] but again this fails to account for the variation in the reconfiguration schemes.

2.4 Summary

To summarize this section, we note that the static topology design for point-to-point networks can be applied to reconfigurable networks as well, but the problem formulation has to be considerably simplified to compute the solutions in real-time that are necessary for the dynamic environment of the reconfigurable networks. We hence have several heuristics that can be used to compute the optimal topology for a given traffic demand matrix. Dynamic topology reconfiguration involves deciding not just on the optimal topology but also the time for reconfiguration. This is necessarily a tradeoff between the expected performance gain and the reconfiguration cost, but most of the previous work assumes that the complete sequences

of traffic matrices are known or can be predicted. Also, the reconfiguration cost - an essential part of the tradeoff calculation - has to be estimated depending on the reconfiguration scheme in use. We will see further why successive approximations are a better way of implementing a target topology and because of that reason, why reconfiguration cost has to be calculated using successive approximations.

Chapter 3

Successive Approximation

Successive Approximation is, in its essence, a reconfiguration scheme to achieve a target topology starting from a current topology. To evaluate the quality of any reconfiguration scheme, we need a notion of the reconfiguration cost and the reconfiguration scheme should attempt to minimize this reconfiguration cost. If we consider a biconnected topology structure such as a ring as both initial and target topologies, any reconfiguration deletes (and creates) at least 2 links. This causes the network to be in a *transient* disconnected state for the duration of the *PAT Delay* as noted in the chapter 1. A natural measure of the reconfiguration cost is the packet drops that happen during this interval due to unavailability of paths.

3.1 Motivation

To see why the number of links that are deleted (and created) cannot act as a measure of the reconfiguration cost, we see in figure 3.1 a simple counterexample where a reconfiguration that deletes (and creates) more links actually results in fewer packet drops. We assume that nodes are connected in a ring, with traffic flowing between any two source destination pairs at the same rate. If the links depicted as gray lines in the network(s) are the links that are deleted during topolog reconfiguration, the number of traffic flows that will be affected during the reconfig-

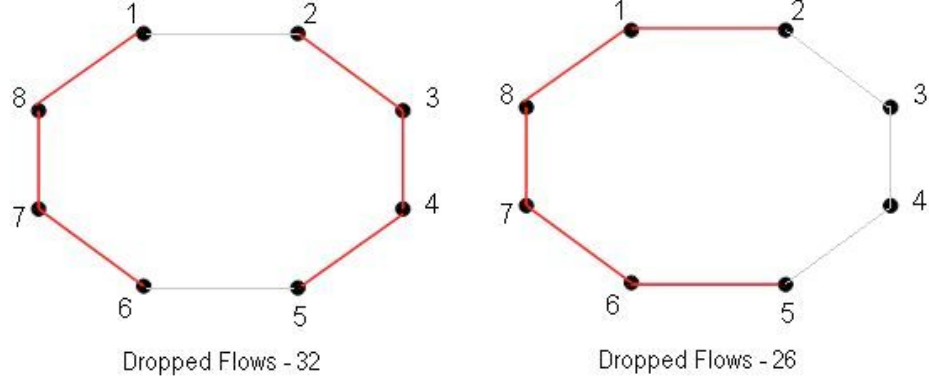


Figure 3.1: Number of flows affected during Topology Reconfiguration

uration can be calculated as shown in the figure. Because the topology is a ring, if p links are changed during reconfiguration, this creates p partitions in the topology in the transient state. As can be seen, the traffic loss during a TR for a fixed p will be minimized when the changes are in, in some sense, in a restricted area of the network.

If the number of nodes in the network is N , and if the k^{th} partition created during the TR process contains n_k nodes, the number of affected traffic flows is given by

$$N^2 - [n_1^2 + n_2^2 + \dots + n_p^2] \quad (3.1)$$

where $n_1 + n_2 + \dots + n_p = N$ by definition. Clearly, the maximum number of flows are affected when $n_1 = n_2 = \dots = n_p = \frac{N}{p}$ and the maximum number of affected flows in such a case is

$$\frac{p-1}{p} N^2 \quad (3.2)$$

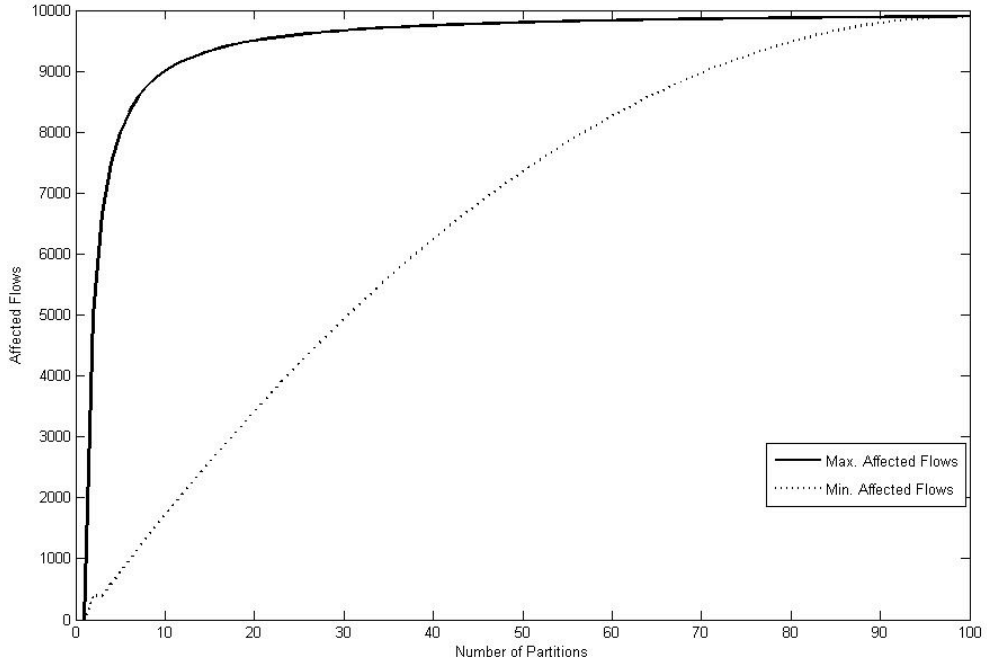


Figure 3.2: Bounds on the number of affected flows. $N=100$

Similarly, the minimum number of flows are affected when all partitions except one have only one node each. In this case, the minimum number of affected flows will be

$$(2N - p)(p - 1) \tag{3.3}$$

Figure 3.2 shows the maximum and minimum number of flows affected as the number of partitions vary from 2 to 100 in a 100 node network. Note that when the number of partitions is 2, if one of the partitions doesn't have a size of at least 2, it is not possible for the target topology to be different from the current topology. Because of this, when the number of partitions is 2, the minimum bound achieved is not $2N - 2$ (substituting $p = 2$ in eqn 3.3), but $4N - 8$. This situation does not

arise for other p 's because all but one partition can have size one and at least two unique topologies can result from such a disconnected network.

Clearly, figure 3.2 illustrates the wide variability within the possible TRs for a given number of link changes. In this example, a topology reconfiguration that results in 20 link changes could drop as much as 9500 flows or as little as 3200 flows. Similarly, a topology reconfiguration that results in 10 link changes could drop as few as 1500 flows. If the target topology differs from the initial topology in 20 links then 9500 flows could be dropped and if we could achieve the target topology instead in two steps where in each step only 10 links are changed (and each step drops only around 1500 flows), it is cheaper in terms of the reconfiguration cost to achieve the target topology in two steps. We generalize this problem below.

3.2 Problem Model

The current topology is given by graph $G_s = (V, E_s)$ and the target topology is specified by $G_d = (V, E_d)$ where V is the set of vertices and E_s and E_d are the set of edges in the current and target topologies respectively. The cost C of this TR is given by

$$f(V, E_s - (E_s - E_d), T) \tag{3.4}$$

Here $f(\cdot)$ is a function of the set of edges that are present in E_s and not deleted during the TR and T is the traffic matrix for the network. For a ring network and if the traffic is assumed to be uniform (that is, each node sends traffic at the same rate to every other node), the cost function is just a function of the number(p) and

size of each connected component in $E_s - (E_s - E_d)$.

$$C = f(p, \mathcal{N}) \tag{3.5}$$

where \mathcal{N} is the set containing the sizes of each connected component in $E_s - (E_s - E_d)$. This is known and can be computed directly from the current topology G_s and given target topology G_d , and this C will fall into one of the points in the vertical line corresponding to p in a graph similar to that of figure 3.2 (for appropriate N).

We aim to generate a sequence of graphs G_1, G_2, \dots, G_K such that

$$\sum_{i=0}^K f(p_i, \mathcal{N}_i) < f(p, \mathcal{N}) \tag{3.6}$$

and is minimized (p_i and \mathcal{N}_i are defined appropriately) with E_s being E_0 and E_d being E_{K+1} . In other words, the target topology is achieved through K intermediate topologies. We define the gain G

$$G = f(p, \mathcal{N}) - \sum_{i=0}^K f(p_i, \mathcal{N}_i) \tag{3.7}$$

and aim to maximize it. We have the constraints that

$$\sum_{i=1}^K p_i \geq p \tag{3.8}$$

$$1 < p_i \leq p \tag{3.9}$$

That is, the number of link changes in each step should sum up to at least p , the required number of link changes to reach the target topology G_d from G_s . And the number of link changes at each step is between 1 and p .

We analyze the bounds of this quantity G to gain an understanding of what kind of topology reconfigurations are required in each step to minimize the packets

dropped. The upper bound of G is achieved when $f(p, \mathcal{N})$ is the maximum *and* we can find the intermediate graphs such that $f(p_i, \mathcal{N}_i)$ is the minimum possible. That is,

$$f(p, \mathcal{N}) = \frac{p-1}{p} N^2 \quad (3.10)$$

$$f(p_i, \mathcal{N}_i) = (2N - p_i)(p_i - 1) \quad (3.11)$$

The gain then becomes,

$$G = N^2 - \frac{N^2}{p} - (2N - K) \sum_{i=1}^K p_i + \sum_{i=1}^K p_i^2 + 2NK \quad (3.12)$$

For every K , G is a convex function and therefore is maximized at the boundaries of p_i 's. The boundaries are (a) all p_i 's are p (b) all p_i 's except one are 2, with the boundary conditions satisfied. Which of these two boundary conditions maximize G depends on parameters like p, N and K . Obviously, if boundary condition (a) maximizes G , we can fix K to be one and this means we execute the topology reconfiguration in one step. We do not have any successive approximations in this case and the gain G is 0. We are more interested in the situation where condition (2) maximizes G . In that situation, G becomes a convex function of K and again, the maxima occur at the boundary conditions - K being 1 or as large as possible. For obvious reasons, we are not interested in K being 1. So, to maximize G , K must be as large as possible. But for $K > \frac{p}{2}$, G is always maximized by the boundary condition (a) for p_i . In conclusion, we see that the upper bound for G is maximized under the following conditions.

1. The TR process takes place in $\frac{p}{2}$ steps.

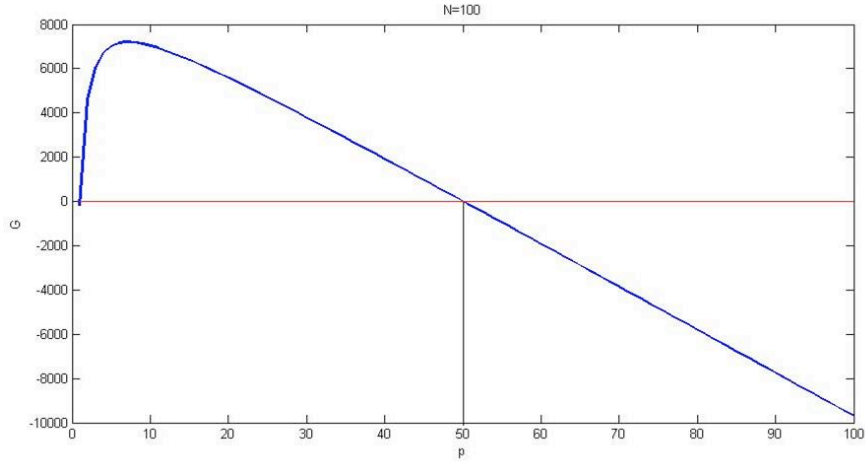


Figure 3.3: Upper bound of the possible Gain. $N=100$

2. At each step, two links are changed(branch-exchange), creating two links from the target topology and destroying two unwanted links from the current topology.
3. This branch-exchange has the lowest cost possible among all branch-exchanges.

We can compute the upper bound for G as,

$$G = N^2 - \frac{N^2}{p} - 2pN + 4p \quad (3.13)$$

Figure 3.3 plots the upper bound on the gain for a 100 node network. As we can see, for $p > \frac{N}{2}$ it is more expensive to carry out the reconfiguration in successive steps as against doing the reconfiguration in a single step. Based on these observations, we can propose an algorithm for generating the sequence of branch-exchanges that will achieve the target topology given a traffic matrix and the current topology.

3.3 Branch Exchange Sequence Generation

If there exists a sequence of branch exchanges that minimizes the reconfiguration cost - the packet drops during reconfiguration - then the order in which the branch exchanges are executed does not matter. This is because each branch exchange has to change the connectivity across 4 unique nodes to satisfy the requirement that the total number of branch exchanges required is $\frac{p}{2}$ for the optimal reconfiguration with successive approximations. This guarantees that the individual branch exchanges do not interfere with each other and therefore the order in which they are executed does not matter. At every step, from the list of such possible branch exchanges, we are free to choose any branch exchange. Selecting the branch exchange with the minimum reconfiguration cost is one strategy. We use the method outlined in [2] to generate the list of possible branch exchanges. This essentially consists of generating an auxiliary graph where nodes are created based on the difference matrix (obtained by taking the difference between the target matrix and the original matrix representations of the topology). The required branch exchanges can be identified by looking for cycles in this auxiliary graph with a length of 4. The steps of the algorithm for generating the branch exchange sequence is as follows.

1. Calculate the cost of the reconfiguration from the initial topology to the target topology (if executed in one step). This can be done by identifying the connected components in the topology when the network is in the transient state. The traffic flows between different components will be affected during this transient state, so the cost of reconfiguration can be calculated by adding

- up these traffic flows (obtained from the traffic matrix). This is the *start_cost*.
2. Generate the least cost branch exchange from the auxiliary graph as in [2].
and calculate the cost of this branch exchange as outlined above. This is the *br-ex_cost*.
 3. Obtain the topology that will result after the lowest cost branch exchange has been executed. Calculate the cost of reconfiguring the topology from this topology to the target topology in one step. We call this the *next_cost*.
 4. If $start_cost > br_ex_cost + next_cost$, execute this branch exchange. Otherwise, implement the target topology directly.
 5. If the target topology is still not achieved, repeat steps 2 – 5 with *next_cost* as the new *start_cost*.

Figure 3.4 shows the results of a simulation illustrating the achievability of the upper bound of G . Several target topologies were generated for each p and the branch-exchange sequences were generated using the above algorithm for each of the target topologies. The gain G was calculated based on an uniform traffic matrix and the maximum G is plotted for each p . The results show that the algorithm is successful in generating the optimal set of branch exchange sequences.

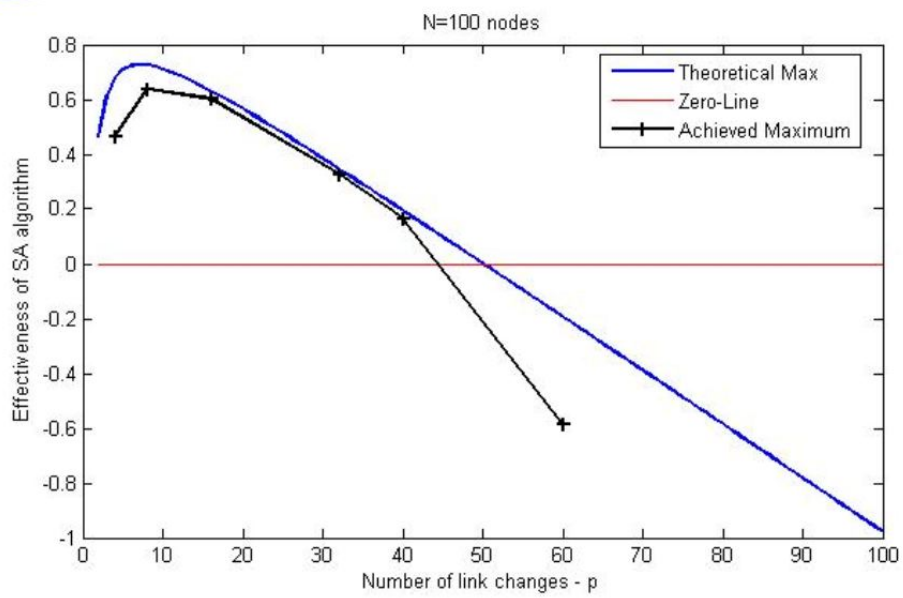


Figure 3.4: Upper bound of the possible G - Achievability $N=100$

Chapter 4

Topology Reconfiguration with Successive Approximations

4.1 Topology Reconfiguration Architecture

A dynamic topology control architecture incorporating the branch exchange sequence generation as discussed in the previous section is depicted in figure 4.1. The Dynamic Topology Control Node, as discussed in chapter 1(fig 1.3), collects the traffic flow-rate information and decides on the time to reconfigure and also calculates the target topology for this reconfiguration. This target topology computation can be done using a static topology design approach[19] and the resultant target topology can then be expected to minimize congestion - the load on the maximally loaded link in the network assuming shortest path routing. However, the solution computation algorithm in the DTCN can be modified to also take the reconfiguration cost into account to calculate the new target topology. The example in figures 4.2 and 4.3 illustrates the differences between the two approaches in calculating the target topology.

4.2 An Example

Figure 4.2 shows the traffic matrix for a 5-node network and the current topology of the network. The traffic flow-rate between any two source-destination pair is

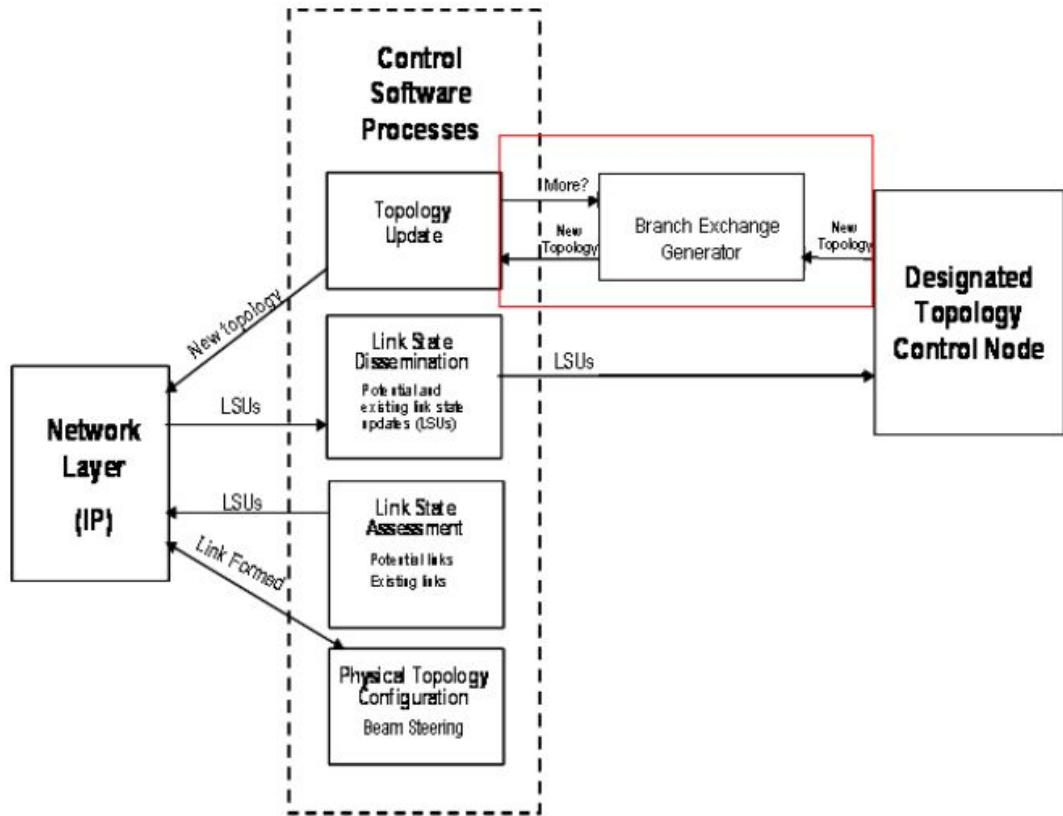


Figure 4.1: Topology Control Architecture with Successive Approximations

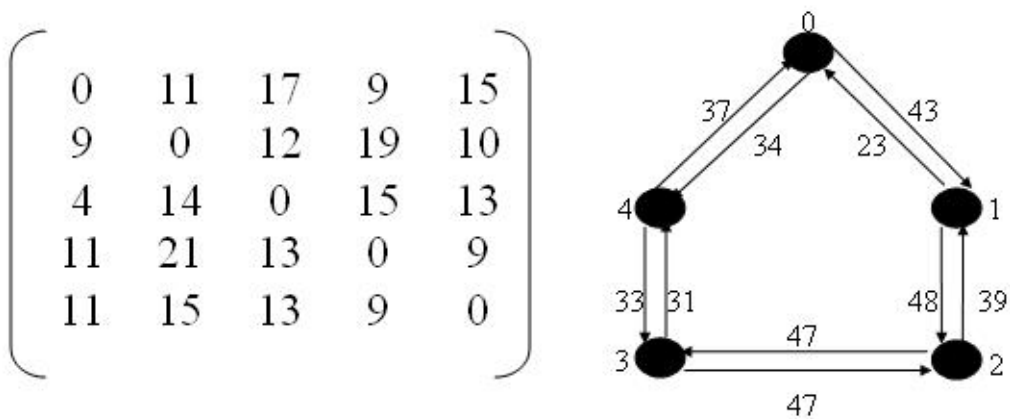


Figure 4.2: Traffic Matrix and link loads in a 5-node network

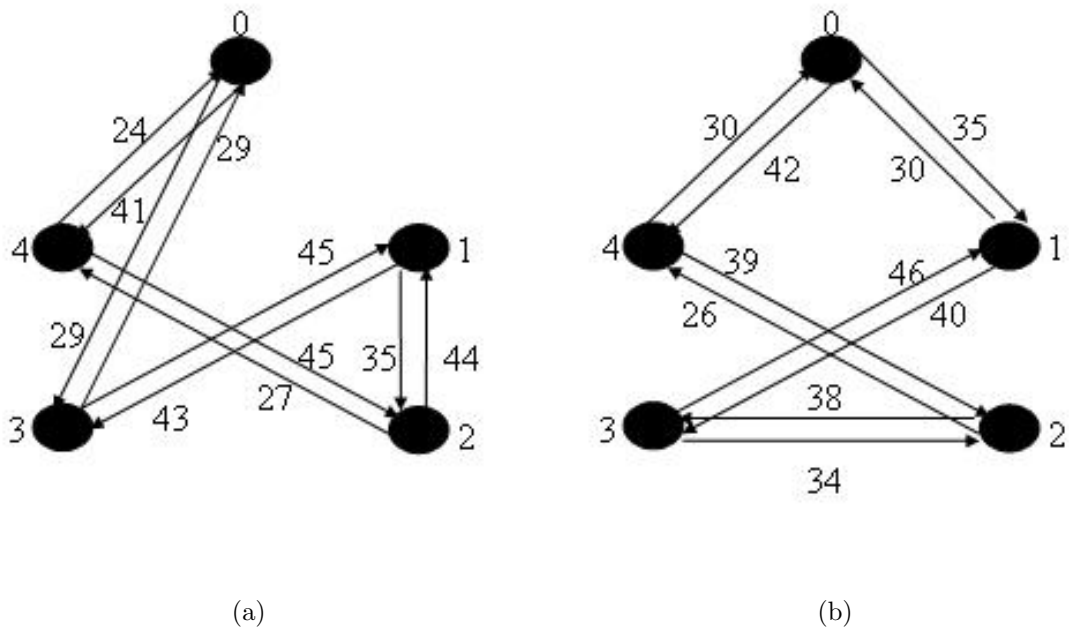


Figure 4.3: Target Topology with (a) optimal congestion (b) sub-optimal congestion given in packets per second and assuming shortest-path routing, the links are loaded with traffic as illustrated in the figure. As this is a 5 node network and shortest path routing is used, each link carries three end-to-end traffic flows. For example, for the from 4 to 0(37), the load on the link is the sum of the traffic flow from 4 to 0(11), 3 to 0(11) and 4 to 1(15). The *congestion* metric is the load on the most heavily loaded link in the network and in the current topology, the most heavily loaded link is the link 1 to 2 and the congestion measure is 48. For this traffic matrix, the optimal topology that minimizes congestion is given in 4.3(a) and for this topology, the maximally loaded links are links 4 to 2 and 3 to 1, both having a load of 45 each. However, implementing this target topology results in 3 partitions during the transition - nodes 0 and 4 is one connected components, nodes 1 and 2 form another connected component and node 3 is the third component. These components are

not connected to each other and any end to end traffic demand with the source and destination nodes being in different components will be dropped. The sum of such dropped traffic demands will be 198 packets per second for implementing this target topology. Figure 4.3(b) illustrates another topology that is not optimal with respect to congestion - the maximally loaded link is 3 to 1 with the maximum load being 46. This topology creates only 2 partitions during the implementation with a total of 151 packets per second being lost during the PAT delay. Though the topology in fig 4.3(a) has a lower congestion measure compared to the topology in fig 4.3(b) (45 versus 46), the latter topology can be expected to have fewer overall packet drops compared to the former depending on the time spent before another topology reconfiguration is carried out. It is also possible to avoid a topology reconfiguration if the expected congestion reduction does not justify the packet drops due to reconfiguration during the PAT delay.

4.3 Problem Formulation

The problem formulation that minimizes congestion cost *and* the reconfiguration cost in selecting a new target topology can be described as follows.

Variables:

- R_{sd} is the average rate of traffic from source s to destination d in packets per second.
- All links are bidirectional and maximum node degree is 2.
- c_{ij} is the capacity of the potential link between nodes i and j .

- λ_{ij}^{sd} is the average flow-rate from source s to destination d on link (i, j) .
- b_{ij} is the decision variable - 1 if a link exists between i and j and 0 if it does not.
- Current topology is given by graph $G_s = (V, E_s)$ and the target topology is given by the graph $G_d = (V, E_d)$.
- T is the traffic matrix with entries T_{sd} that specifies the traffic demand from source s to destination d .
- $f(V, E_s - (E_s - E_d), T)$ is the cost C of a topology reconfiguration from topology G_s to G_d specified in lost packets per second.
- T_{PAT} is the time taken for the pointing, acquisition and tracking process.
- T_{MTBR} is the mean time between reconfigurations.
- The reconfiguration is carried out in K steps with each intermediate topology specified by G_i .

Constraints:

$$\sum_j \lambda_{ij}^{sd} - \sum_j \lambda_{ji}^{sd} = R_{sd} \text{ if } s = i \quad (4.1)$$

$$= -R_{sd} \text{ if } d = i \quad (4.2)$$

$$= 0 \text{ otherwise} \quad (4.3)$$

$$\lambda_{ij}^{sd} < b_{ij} R_{sd} \forall i, j, s, d \quad (4.4)$$

$$\sum_{(s,d)} \lambda_{ij}^{sd} < c_{ij} \forall i, j \quad (4.5)$$

$$b_{ij} \in \{0, 1\} \quad (4.6)$$

$$\sum_j b_{ij} = d \quad (4.7)$$

$$\sum_{(i,j)} c_{ij} = c \quad (4.8)$$

$$\sum_{i=1}^K f(V, E_{i-1} - (E_{i-1} - E_i), T) < f(V, E_s - (E_s - E_d), T) \quad (4.9)$$

$$G_K = G_d \quad (4.10)$$

$$G_0 = G_s \quad (4.11)$$

The objective function to be minimized need to be obtained as a combination of the congestion measure and the packet drops that happen due to reconfiguration. The key challenge in combining the two costs is in expressing both quantities in the same units so that the costs can be added together.

4.3.1 Reconfiguration Cost

As chapter 3 describes, the minimum reconfiguration cost can be obtained by executing the reconfiguration in successive approximations and the minimum such reconfiguration cost can be obtained by generating the sequence of intermediate branch exchanges and the final reconfiguration by using the algorithm outlined in section 3.3. This reconfiguration cost is expressed as the lost packets per second.

$$PD_{G_s, G_d} = \min_{K, E_1, \dots, E_K} \sum_{i=1}^K f(V, E_{i-1} - (E_{i-1} - E_i), T) \quad (4.12)$$

where $E_0 = E_s$ and $E_K = E_d$ as shown in constraints 4.10 and 4.11. This quantity can be calculated directly as the outcome of the algorithm described in section 3.3 by calculating the sum of all *br-ex_costs* and the *start_cost* of the final step. The

reconfiguration cost can be directly expressed as lost packets by multiplying the lost packets per second with the PAT duration - T_{PAT} .

4.3.2 Congestion Cost

Congestion is the traffic flow-rate on the link with the maximum load in the network. This metric is a useful quantity for assessment because the packet drops in the network first happen in the most congested link. The total packet drops in the network are dominated by the packet drops in the most congested link because this link acts as a bottleneck in the network and reduces the load on the other links further.

$$\lambda_{max} = \max_{(i,j)} b_{ij} \sum_{(s,d)} \lambda_{ij}^{sd} \quad (4.13)$$

$$(I, J) = \arg \max_{(i,j)} b_{ij} \sum_{(s,d)} \lambda_{ij}^{sd} \quad (4.14)$$

An estimate of the expected packet drops on this link can be arrived by modeling the link as an M/M/1/1 queue. In the M/M/1/1 queue, the arrivals are modeled as a Poisson random process, the service times are exponential. There is one server and the system has capacity to hold just the packet in service in the queue. When a packet is in service, any other arriving packet is dropped on arrival. If there are no packets in service on a packet arrival, the packet enters service immediately. The blocking probability - the probability of dropping an arriving packet - is an important metric for this system. If λ is the arrival rate and the average service time is given by $\frac{1}{C}$ where C is the capacity in packets per second of the outgoing link, the

blocking probability P_b is given by the Erlang-B formula [30]

$$P_b = \frac{\lambda}{\lambda + C} \quad (4.15)$$

The M/M/1/1 model may not be an accurate model of the packet drops at the congested link because a queue of finite size might exist for the outgoing packets in the most congested link. In this scenario, a more appropriate model could be M/M/1/K, but with this queueing model, we have a measure of the end-to-end delay in addition to the blocking probability. However, as the reconfiguration cost is expressed in terms of packet drops, it is useful to express the congestion metric in terms of just expected packet drops. Using the M/M/1/1 model, the average packets lost per second can be calculated using the blocking probability. We can also assume a mean time between reconfigurations - the average time for which the network topology does not change - T_{MTBR} . The average number of packets lost due to congestion can be calculated from the mean time between reconfigurations and the blocking probability. It is likely that the chosen topology has a different flow-rate on the most congested link for a future traffic matrix. As the chosen topology is expected to be optimal with respect to the congestion metric, the changed traffic demand will result in higher congestion and therefore, a higher blocking probability. But information about the future traffic demands is not assumed to be available and by only using the mean time between reconfigurations, the calculated packet drops due to congestion is lower than actual number of packet drops due to congestion. From eqn 4.14, the packet drops due to congestion can be calculated as follows.

$$\frac{\lambda_{max}}{\lambda_{max} + C_{IJ}} \lambda_{max} \cdot T_{MTBR} \quad (4.16)$$

The objective function to be minimized can be obtained by combining the reconfiguration cost from 4.12 and 4.16

Objective function:

$$\min_{G_d} \left\{ \frac{\lambda_{max}}{\lambda_{max} + C_{IJ}} \lambda_{max} \cdot T_{MTBR} + PD_{G_s, G_d} \cdot T_{PAT} \right\} \quad (4.17)$$

4.4 Heuristics

The minimization problem in equation 4.17 is similar to the congestion minimization problem in eqn 2.10 in that it is an optimal linear ordering problem which is NP-complete [24]. To obtain heuristics to solve this problem, we start from well-known heuristics such as the single-hop heuristic, the single-hop rollout heuristic and the branch exchange heuristic[19]. The key difference between this minimization problem and the congestion minimization problem is that calculating the reconfiguration cost requires the current topology and the *complete* target topology. For the congestion minimization problem, some of the *link insertion* heuristics insert an edge in the network at each step minimizing a local objective function. Therefore, at each step, we only have a partial topology and this cannot be used to calculate the reconfiguration cost. A similar argument holds for the *link deletion* heuristics. Heuristics that evaluate complete topologies at each step can be modified to take reconfiguration cost into consideration when selecting the target topology as illustrated below.

- Rollout

The rollout method described in section 2.2 adapted to include reconfiguration cost has the following steps.

1. Sort the SD pair list in the order of non-decreasing magnitude of traffic. Let this order be called as $\{SD[0], SD[1], \dots, SD[M-1]\}$, where M represents the number of SD pairs.
2. Create M topologies by SD-indexing. SD-indexing means that some SD index is fixed as the first index to be considered and the rest of the indices will be taken in the decreasing magnitude of traffic. Sub-steps a and b illustrate this point.
 - (a) Create a topology $T[0]$ by either single-hop or multi-hop heuristic using the order: $\{SD[0], SD[1], \dots, SD[M-1]\}$. Note that this topology is the same as the single-hop/multi-hop heuristic would obtain.
 - (b) Create $T[1]$ by using following order: $SD[1], SD[0], SD[2], \dots, SD[M-1]$ and so on.
3. Choose $SD[k]$ to be the first index, which finds topology $T[k]$ with *minimum expected packet drops* - the combination of congestion cost and reconfiguration cost as outlined in eqn 4.17. Add a link by single-hop/multi-hop heuristic to create a path between $s[k]$ (source) and $d[k]$ (destination).
4. Repeat steps 1-3 until the ring topology is completed. If the index $SD[j]$ cannot be used to add a link in the topology, pass to the next index.

- Branch Exchange

The branch exchange algorithm can be used to improve upon the congestion performance of an existing ring[23]. In the case of ring topologies, it can be noted that if we exchange 2 non-node-sharing links with another pair, a new ring can be obtained. In a given ring, $\frac{N(N-3)}{2}$ such exchange sequences are possible to obtain a new ring. A simple branch exchange algorithm will create $\frac{N(N-3)}{2}$ new ring topologies from a given ring and pick the one with *minimum expected packet drops* - the combination of congestion cost and reconfiguration cost as outlined in eqn 4.17. This step can be repeated several times to obtain better topologies but with diminishing returns[19].

Chapter 5

Simulation Results

The topology reconfiguration heuristics that were developed in chapter 4 can be evaluated by comparing the topologies that are generated by these heuristics with the topologies that will be generated by the heuristics which do not take re-configuration cost into account. This comparison can be made with respect to end to end packet delay and dropped packets in a TCP/IP network.

5.1 Simulation Methodology

The TCP/IP network that is used for evaluating the topology reconfiguration heuristics is the FSO/RF network and the network operation is simulated using discrete event simulations in OPNET Modeler[31]. The main challenges in simulating the FSO/RF networks with topology control are

1. Interfacing the topology control processes with the TCP/IP stack.
2. Simulating point-to-point wireless links using the broadcast wireless links provided in OPNET Modeler.

The topology control processes such as link state assessment, link state dissemination, target topology generation using the topology reconfiguration heuristics and target topology dissemination are all implemented as a process module which

represents the Dynamic Topology Control Node (DTCN). These topology control processes are interfaced with the TCP/IP stack through the routing protocol - chosen to be Open Shortest Path First (OSPF) in this case. The DTCN uses the link state advertisement (LSA) mechanism of the OSPF protocol in obtaining the link state information and for disseminating the target topology. It is also possible for the DTCN to precompute the routing table for a future target topology and disseminate the information along with the topology control information to the nodes[32]. However, for the purposes of comparing the performance of the topologies generated by the heuristics before and after incorporating reconfiguration cost, the time taken for the routing table convergence after a topology change does not matter because it remains the same in both cases. The DTCN can also be configured with the heuristic that is to be used for the target topology generation and if successive approximations have to be enabled during the reconfiguration. As the reconfiguration cost is calculated using successive approximations, enabling successive approximations modifies the selected heuristic to also take the reconfiguration cost into account when calculating the target topology. All the heuristics were evaluated in networks with 10 nodes. Figure 5.1 shows a network consisting of 10 nodes along with the DTCN, labeled as manager in the figure along with the configuration options for the DTCN.

There are two approaches to simulating point-to-point wireless links in OPNET Modeler [32]. In the first approach, wireline links are used to simulate each possible wireless link in the network and a failure/recovery mechanism can be used to fail and recover appropriate links to obtain any required topology based on the

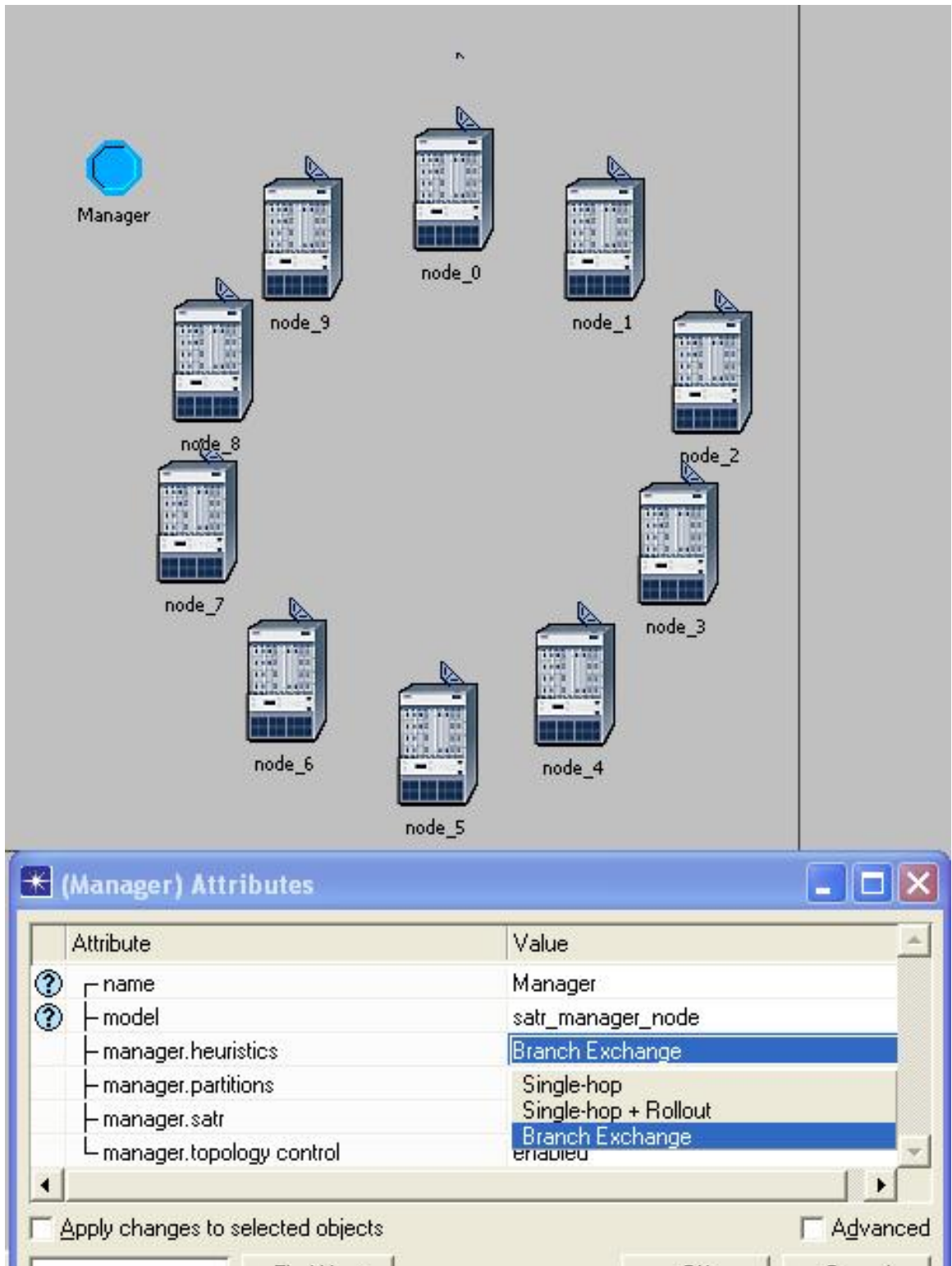


Figure 5.1: Illustration of the DTCN with the Simulated Network

network nodes. The disadvantage with this approach is that the number of links required in the network increases as $O(N^2)$ if N is the number of nodes in the network because it is possible for a link to exist between any two nodes. This model does not scale with increasing number of nodes in the network and therefore it is imperative to use wireless links in OPNET to model the point-to-point wireless links in the FSO/RF network. However, the OPNET wireless links are omnidirectional and cannot be used directly to model point-to-point wireless links. A solution to this problem is to use dynamic receiver groups to derive point-to-point wireless links from broadcast wireless links.

Dynamic receiver groups are provided by the OPNET Modeler mainly as an optimizing mechanism. The simulation being a discrete event simulation, any packet transmitted out of a wireless link has to be replicated for every possible recipient in the simulated network so that wireless parameters like gain, signal to noise ratio and bit error rate can be calculated for each possible recipient independently. However, this results in highly inefficient simulations when the number of nodes in the simulated wireless network increases. As an optimizing mechanism, it is useful to limit this packet replication to a set of recipient nodes if it is known ahead of time that some recipients cannot receive packets from particular senders during the course of the simulation. These are called *receiver groups* and if the receiver groups for each sender can be modified during the course of the simulations, these are called *dynamic receiver groups*. To emulate a point-to-point link in a FSO/RF network, we can configure the dynamic receiver groups to include only the neighbors for each wireless transceiver based on the network topology at any given time. When a topol-

ogy has to be reconfigured, the new links in the topology can be obtained simply by updating the dynamic receiver groups of the senders with the appropriate new neighbors. A similar operation can be carried out to remove links in the network. In addition, the routing protocol - OSPF in our example - has to be notified of the update to the topology so that the new topology can be discovered using the link state update mechanism.

5.2 Traffic Demand Generation

The key requirement for the traffic matrix generation mechanism is to dynamically update the traffic matrix over time so that the network topology becomes non-optimal resulting in the triggering of a topology reconfiguration. We start with a uniform traffic matrix - every node sends traffic to every other node in the network at the same rate - and vary the entries in the traffic matrix uniformly. That is, each entry in the traffic matrix indicates the traffic flow-rate demanded from the source to the destination and this value is increased or decreased at each unit time with equal probability. For a N node ring topology, if shortest-hop routing strategy is used (which will be the case with OSPF if all interface costs are the same), each link carries $\frac{(N+1)(N-1)}{8}$ end to end (SD) flows in either direction (for odd values of N). If we set l to be the link loading factor and C is the capacity of the link, each end-to-end flow has a traffic rate of

$$\frac{8lC}{(N+1)(N-1)} \tag{5.1}$$

The simulations were carried out with wireless links of capacity 1 Mbps and the

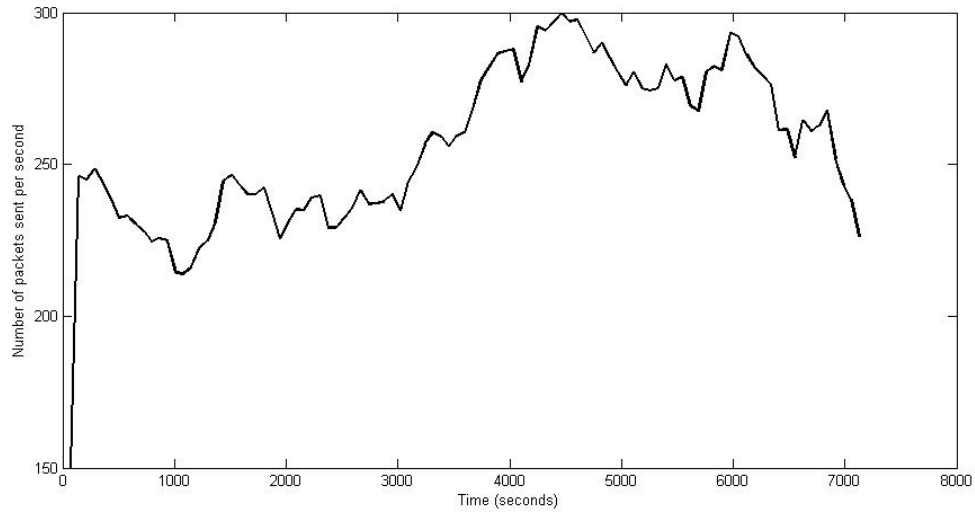


Figure 5.2: Total Number of Packets Sent in the Network

link loading factor was set at 50% corresponding to a moderately loaded network. The above formula was used to calculate the traffic rate of each end to end flow and an exponential traffic generator is used to generate the traffic at the calculated rate. After every 10 seconds, each entry in the traffic matrix is either increased or decreased by 10% of the original traffic rate for the end to end flow with equal probability. In increasing or decreasing the entries in the traffic matrix, a minimum value of 0 packets per second is an obvious lower limit. There is no limit on the maximum value for the traffic rate for an end-to-end flow but everytime the lower limit is hit on any traffic matrix entry, the next increase is skipped as well so that the total traffic remains constant. As each entry is increased or decreased with the same value and with equal probability, the total number of packets sent per second can be modeled as a random walk. This implies that the expectation of the total

number of packets per second is a constant but the variance increases over time. Also, the traffic load on the links become non-uniform over time and this creates an imbalance in the traffic distribution in the network. Some of the links are loaded close to or more than the capacity of that link due to this imbalance in the traffic demand. This results in congestion in that link and is observed as packet drops at the output queue of the corresponding wireless transmitter. Figure 5.2 shows the total number of packets sent per second in the network during one simulation run. With this traffic pattern, the packet drops that happen due to the imbalance in the traffic demand can be seen in figure 5.3. During this simulated two hours, an average of about 40 packets per second were lost due to congestion with the peak occurring around 4000 seconds with close to 100 packets per second lost. This peak coincides with the increase in the number of packets sent at that time to 300 packets per second as can be seen from figure 5.2.

The packet drops due to congestion increases over time as the topology becomes suboptimal with respect to the traffic demand matrix. To update the network with optimal topologies, a new topology is computed whenever one or more of the links have 100% or more link utilization. When this occurs, a topology computation is triggered and this phase uses one of the heuristics to calculate the new target topology. The execute phase implements this target topology directly or by using successive approximations based on the configuration. This process model for the DTCN can be seen in figure 5.4.

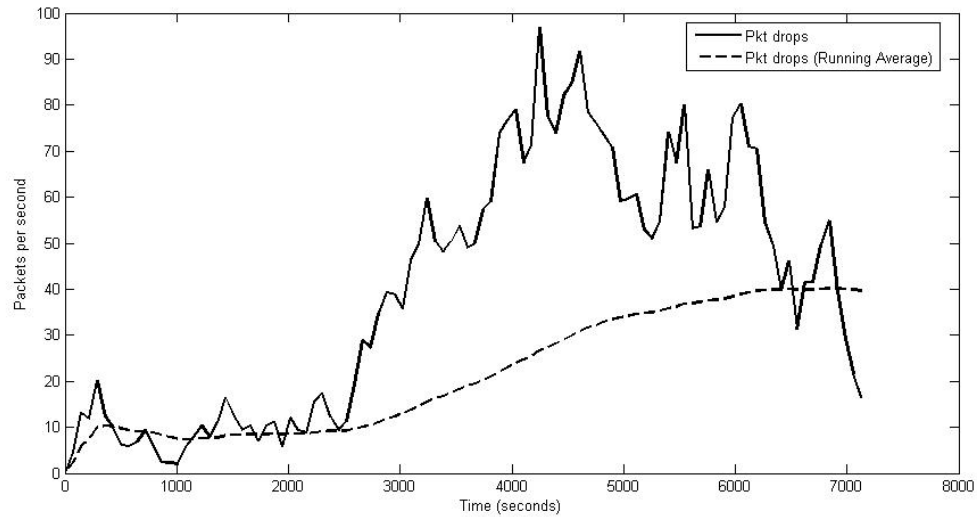


Figure 5.3: Packet drops due to Traffic imbalance

5.3 Simulation Setup and Verification

The rollout heuristic(with singlehop as the base heuristic) and the branch exchange heuristic along with the successive approximations module were all implemented as C functions in OPNET Modeler. Appendix A includes the pseudo code for the successive approximations module. All simulations were run in a Intel Pentium 4 3.60 GHz CPU with 1GB of RAM running Windows XP. Each simulation used one of the two traffic patterns from figure 5.2. As noted above, the heuristic to be used in calculating the target topologies can be configured as a parameter to the process model (figure 5.4). For each heuristic, the end to end packet drops were calculated as the sum of reconfiguration drops and congestion drops.

- Reconfiguration Drops:

When a topology reconfiguration is executed by the DTCN, the network is

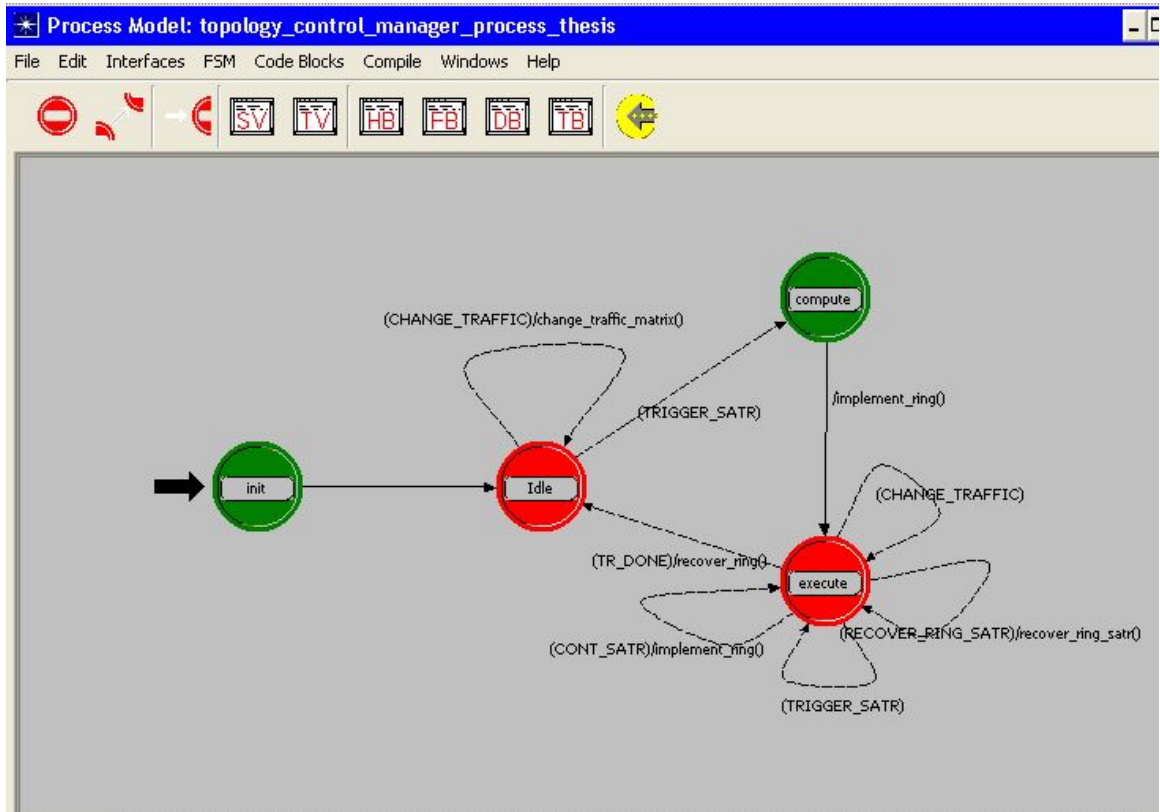


Figure 5.4: Dynamic Topology Control Node Process Model

partitioned and each node loses the route to some of the destinations. The packets that are transmitted for such destinations will be dropped by the IP layer after failing to find the next hop for this destination. Such drops are called reconfiguration drops and they are measured as packet drops at the IP layer.

- Congestion Drops:

The packet drops that happen when the network is in normal operation, but when the traffic demand on some links are very high relative to its capacity, are called congestion drops. These drops occur at all times as against the reconfiguration drops that happen only when a topology reconfiguration is executed. The IP layer routes all the packets correctly but these drops happen because the output queues of the wireless transmitters are of finite size.

The simulation was then repeated for each heuristic using the same traffic pattern with successive approximations enabled and the total packet drops were measured using the same method as above. The packet drops thus measured in the two cases can be compared to evaluate the effect of enabling successive approximations on each heuristic. For all the simulations, T_{PAT} was chosen to be 2 seconds and T_{MTBR} was 10 seconds. T_{MTBR} was chosen to be 10 seconds because this is the minimum possible time between reconfigurations as in these simulations, the traffic matrix changes every 10 seconds. Choosing the minimum possible time for T_{MTBR} ensures that the reconfiguration cost contributes the maximum possible to the total cost (equation 4.17) and this is useful in studying the impact of successive

approximations.

The implementation of the heuristics were verified using the example traffic matrix and the current topology shown in fig 4.2. There are 12 possible rings that can be constructed out of 5 nodes. Table 5.1 lists all possible ring topologies in the first column (when the nodes are labeled 0, 1, 2, 3 and 4) and their congestion, reconfiguration and total costs. MATLAB was used to generate all possible rings that can be constructed with 5 nodes and for each ring, the congestion measure was calculated as the maximum link load and the reconfiguration cost was calculated assuming 012340 as the current topology. The final column shows the total cost calculated based on equation 4.17 that combines the reconfiguration cost and the congestion metric with the assumption that T_{PAT} is 2 seconds[33] and T_{MTBR} is 10 seconds. From the table, it can be seen that the optimal topology with respect to congestion is *023140* and the optimal topology with respect to the total cost is *012340*. That is, if total cost is considered, the most optimal topology is the current topology itself and the optimal topology control decision will be to not reconfigure at all. To verify the implementation of the heuristics, the topologies computed by the heuristics can be compared with the optimal topologies identified above. The target topology computed using the rollout(singlehop) heuristic was *031240*. This topology had a congestion measure of 45, only a little worse than 44 - the congestion measure of the optimal topology. The branch exchange heuristic computed *021340* as the target topology and this topology too has a congestion measure of 45. As discussed earlier, enabling successive approximations in the simulation process model modifies the heuristics to minimize total cost. After enabling successive approximations,

Rings	Maximum Load (pkts/s)	Reconfiguration drops (pkts/s)	Expected pkt drops
012340	48	0	480
012430	49	165	820
013420	47	212	894
014230	47	202	874
014320	47	158	786
021340	49	160	810
021430	49	206	902
023140	44	196	832
024130	45	250	950
031240	45	198	846
032140	51	130	770
013240	46	151	762

Table 5.1: Possible Ring topologies and the corresponding costs

both heuristics computed *012340* as the target topology which is consistent with the observation that the current topology is the optimal one when total cost is considered. These observations verify that these heuristics do minimize congestion cost but minimize total cost when successive approximations are enabled.

5.4 Rollout - Singlehop

During each simulation run, a 2 hour operation of a 10 node network that uses the rollout(singlehop) heuristic for calculating target topologies was simulated. Five such simulation runs were made with each run differing in the way the traffic demand matrix is changed. For each run, the total packet drops were measured as the sum of reconfiguration drops and congestion drops. The average packet drops were calculated as a simple average of the results from the five simulation runs. Figure 5.5 shows the running average of the average total packet drops when the rollout heuristic is used with and without successive approximations. The basic observation that can be made is that enabling successive approximations reduces the overall packet drops. To verify this assertion statistically, a cubic polynomial function can be used to fit both the running average curves in the graph in figure 5.5. The leading coefficient for the cubic polynomial was $1.8059e-10$ and $5.1472e-11$ respectively for the rollout and the rollout with SA average packet drop curves (error variances were 0.26 and 0.21 respectively). The coefficients of the cubic terms - the leading coefficients - of the two curves determine if the two curves converge or diverge or if there are no differences between them. In this case, it can be seen that the coefficient for the cubic term in the first function is larger than for the second one and this means the first function will have larger values than the second function and this difference will increase with time. In other words, the two functions diverge with time and as the sum of the error variances is smaller than the difference between the two functions, the average packet drops without successive approximations are

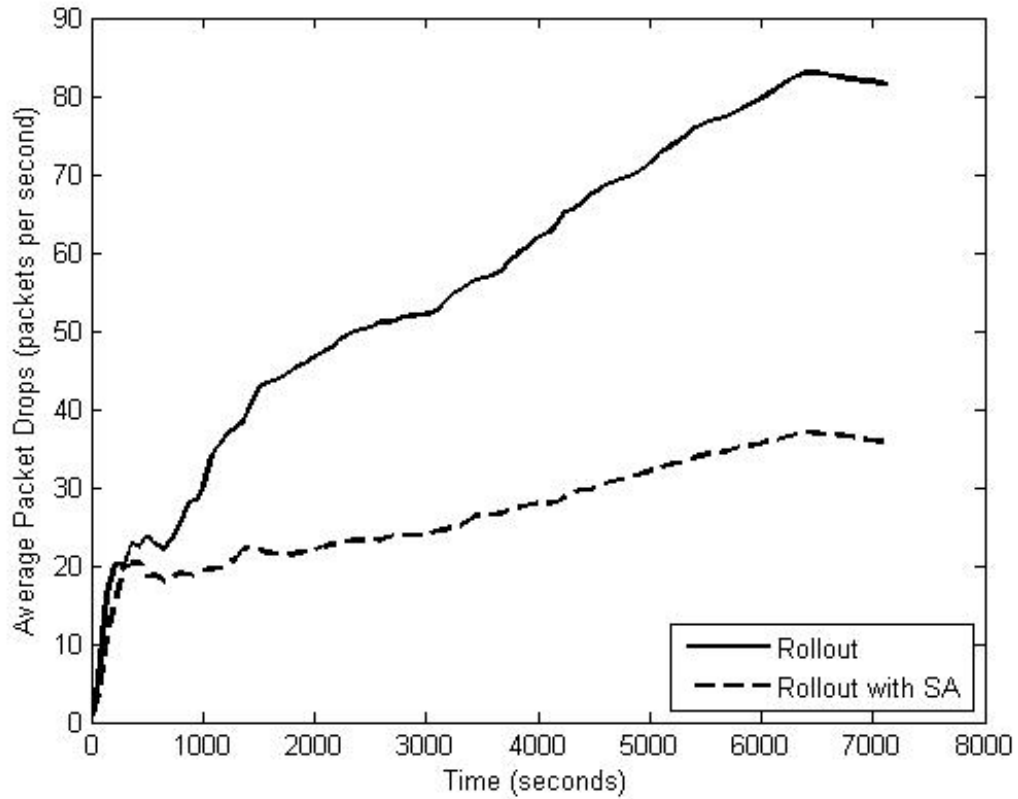


Figure 5.5: Average Packet Drops with Rollout - Singlehop heuristic with and without Successive Approximation - Average from 5 simulation runs

statistically higher than the average packet drops with successive approximations when the rollout(singlehop) heuristic is used.

- Reconfiguration Drops:

The reconfiguration drops - the packet drops measured at the IP layer - was averaged over the five simulation runs and figure 5.6 shows the running average of the average reconfiguration drops. Clearly, modifying the rollout heuristic to incorporate successive approximations reduces the packet drops that happen during reconfiguration as expected. This reduction in packet drops is

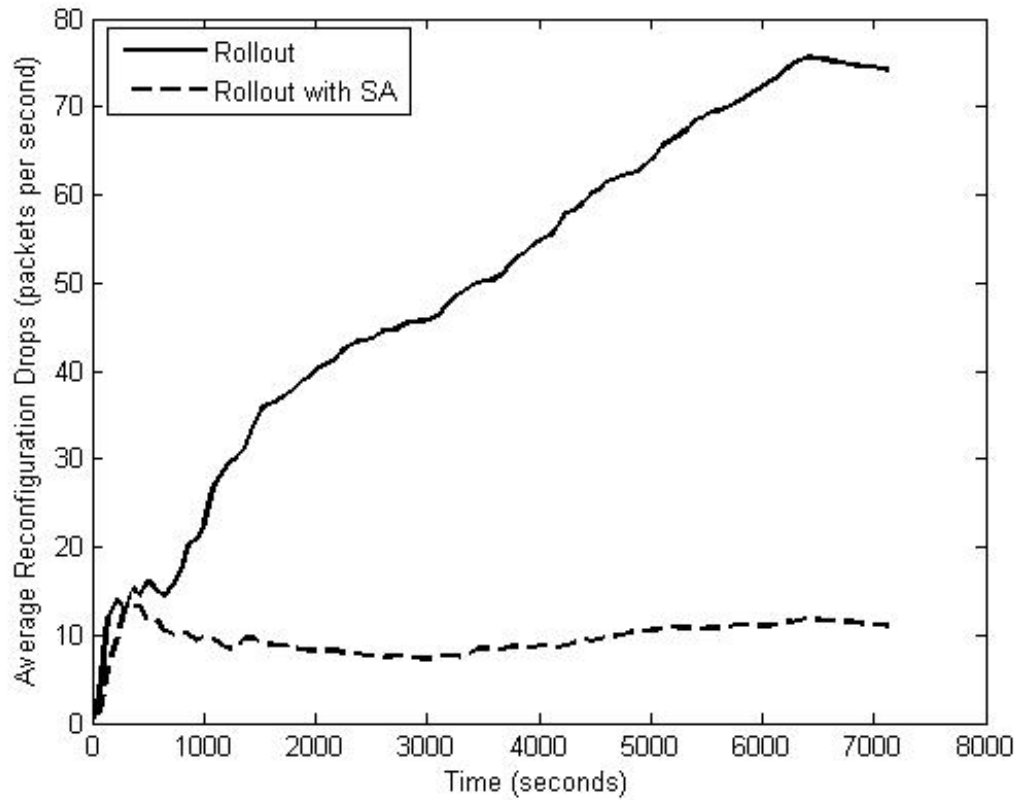


Figure 5.6: Average Reconfiguration Drops with Rollout - Singlehop heuristic with and without Successive Approximation - Average from 5 simulation runs

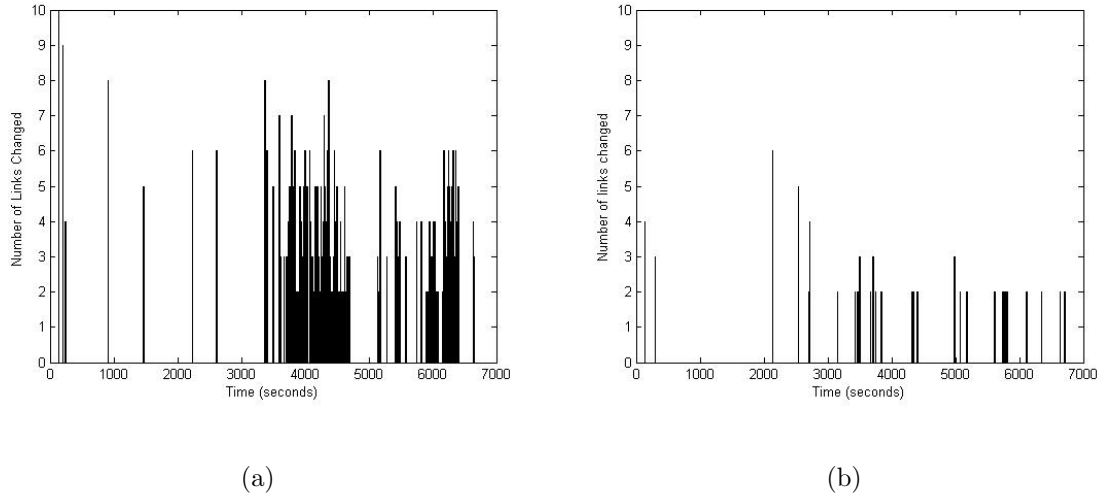


Figure 5.7: Reconfiguration Activity with (a) Rollout (singlehop) (b) Rollout (singlehop) with Successive Approximations - Traffic Pattern as in Figure 5.2

primarily due to a reduction in the number of topology reconfigurations in the network. A dynamic view of the topology reconfigurations in the network is seen in figure 5.7 where the number of links changed during a topology reconfiguration is plotted across time for one of the five simulation runs. The total packets sent for this traffic pattern is as in figure 5.2 and figure 5.7(a) shows the reconfiguration activity with the rollout(singlehop) heuristic and figure 5.7(b) shows the reconfiguration activity with successive approximations enabled. The increase in reconfiguration activity after about 3000 seconds when successive approximations is not enabled (figure 5.7(a)) is absent when successive approximations are enabled (figure 5.7(b)). As the number of reconfigurations are lower with SA, the drops that happen due to reconfiguration is lower.

The average number of topology recomputations and topology reconfigurations

	Top. Computations	Top. Reconfigurations	Links Changed
Rollout	236.6	217	552
Rollout with SA	269.8	42.8	108

Table 5.2: Topology Recomputations - Averages from five simulation runs using Rollout heuristic

for the five 2-hour simulations are tabulated in table 5.2. The average number of topology reconfigurations is 42.8 when successive approximations is enabled while that number is 217 when SA is disabled. In other words, on an average, a topology reconfiguration is implemented in the network every 168 seconds in the network with the rollout heuristic when SA is enabled whereas a topology reconfiguration is implemented in the network every 33 seconds when the rollout heuristic is used without SA (A note of caution is that these numbers cannot be directly applied in a practical scenario because the traffic matrix in the simulations are changed every 10 second which are not necessarily the case in a practical setting). This reduction in the number of reconfigurations happens in spite of an increase in the number of times a topology computation is triggered. As topology computations are triggered whenever one or more of the link utilizations reach 100%, this means the network operates with more maximum link load when SA is enabled. This contributes directly to more packet drops due to congestion.

- Congestion Drops: Figure 5.8 shows the average over 5 simulations of these

congestion drops that are measured as layer 2 packet drops in the simulations. The average packet drops actually increase when successive approximations are used because the optimizing function is the combination of reconfiguration drops and congestion drops when successive approximations are used (eqn 4.17). If successive approximations are not used, the objective function minimizes the congestion cost and any topology other than the one computed by the congestion minimization heuristic is expected to result in increased congestion cost - expressed in this case as packet drops. In other words, the network topology is not reconfigured as frequently when SA is enabled as it would otherwise have been. This keeps the network topology suboptimal with respect to the traffic matrix for a longer duration and this results in the increase in congestion drops.

However, as shown in figure 5.5, the total of the reconfiguration packet drops and congestion drops is reduced when successive approximations are used. This happens because the decrease in reconfiguration drops compensates for the increase in congestion drops and reduces the total further. It was also observed that the impact of using successive approximations on the end-to-end delay is to increase it slightly. The average end-to-end delay was about 0.35 seconds and this increased to about 0.4 seconds when successive approximations were used.

The total packet drops - with and without SA - from the individual simulation runs are shown in figures 5.9 and 5.10. For each run, the total number of packets sent and the running average of total packet drops is shown. The decrease in the number

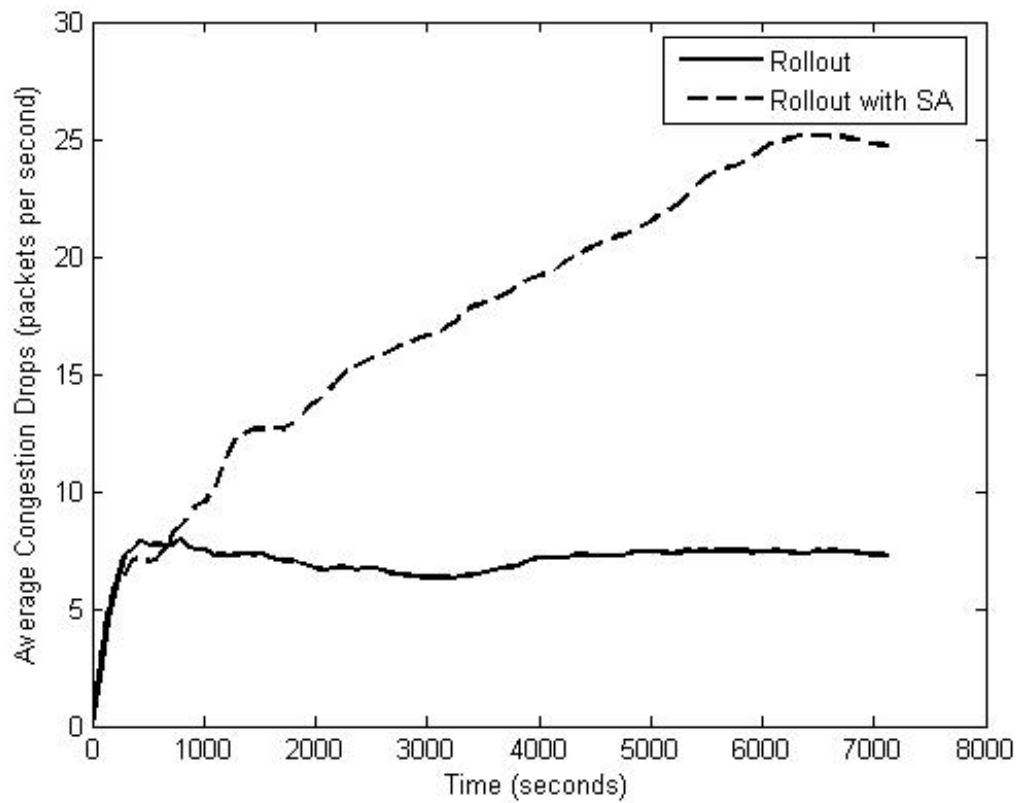
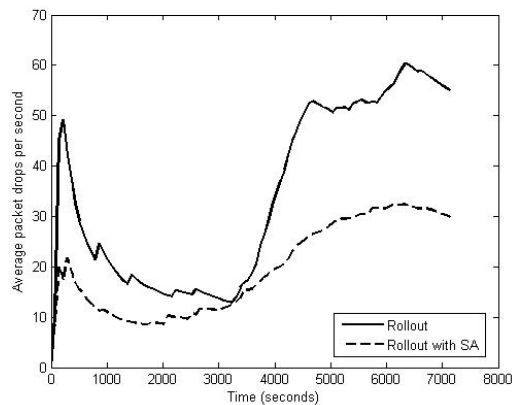
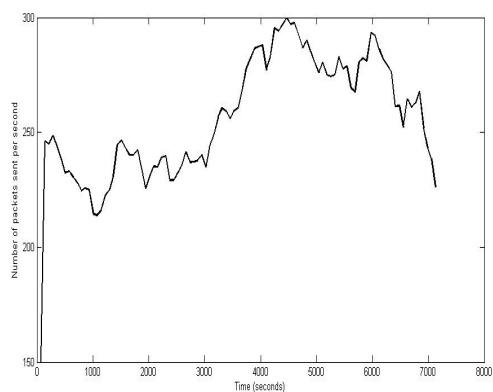
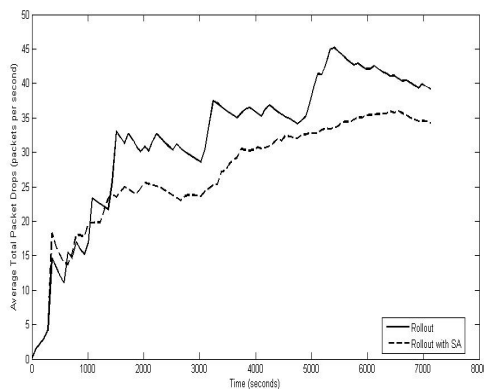
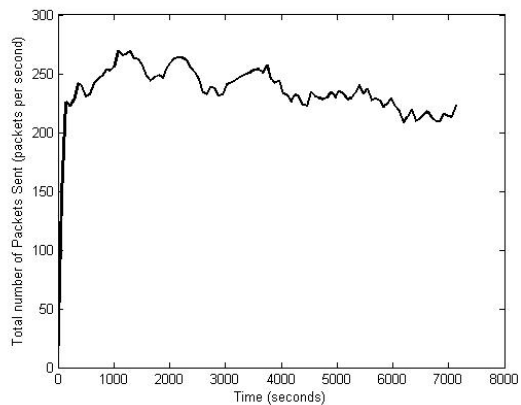


Figure 5.8: Average Congestion Drops with Rollout - Singlehop heuristic with and without Successive Approximation - Average from 5 simulation runs

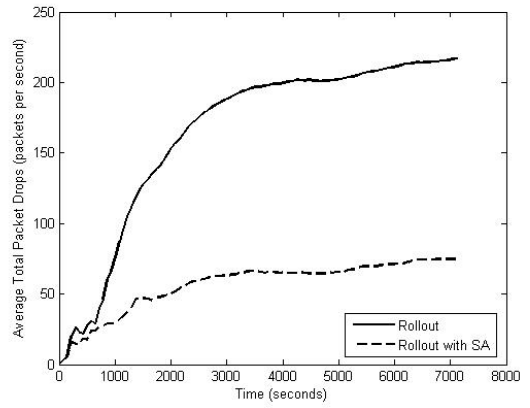
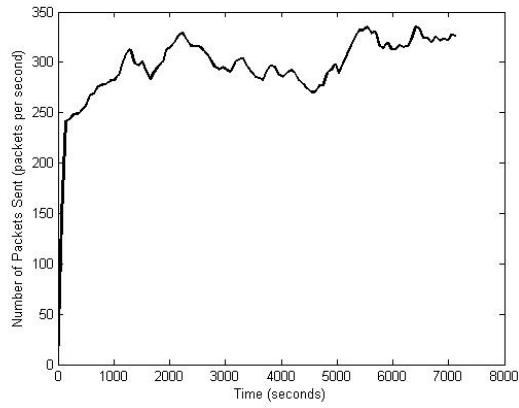


(a)

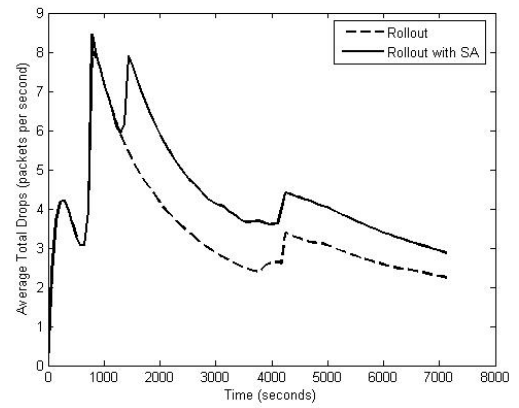
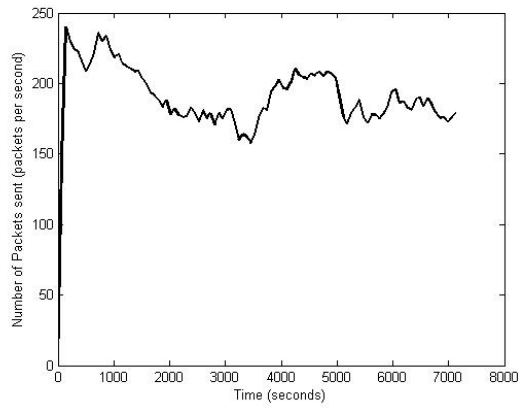


(b)

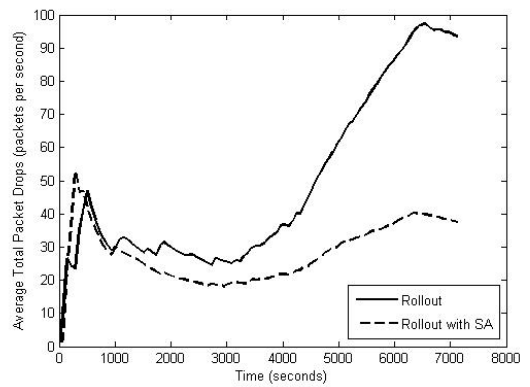
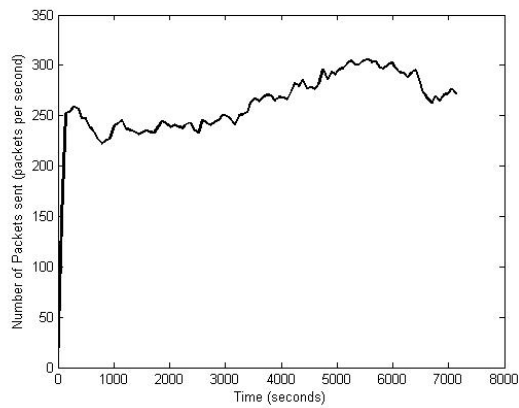
Figure 5.9: Total Packets Sent and Average Packet Drops with traffic patterns (a) 1 and (b) 2 with the Rollout heuristic



(a)



(b)



(c)

Figure 5.10: Total Packets Sent and Average Packet Drops with traffic patterns (a) 3, (b) 4 and (c) 5 with the Rollout heuristic

of packet drops by using SA is marked in the simulation runs where the load on the network is higher. For example, figure 5.10(b) shows the results from a simulation run in which the packet drops are an order of magnitude smaller than the packet drops in the simulation runs shown in figures 5.10(a) or 5.10(c). This difference can be explained by the fact that the traffic matrix is initialized to load each link to 50% of the capacity and then each entry of the traffic matrix is increased or decreased by 10% every 10 seconds. For every random seed, the traffic matrix evolves in a unique manner so that in some cases, the link utilization for some links reaches 100% resulting in a large number of packet drops. If the load on the links are lower than 100%, the packet drops will also be very low. Higher loads in the network result in more number of topology reconfigurations when the rollout heuristic is used. When successive approximations are enabled, the packet drops due to reconfigurations are reduced and this is reflected in the fewer packet drops. In other words, the impact of enabling successive reconfigurations can be seen when there are a large number of topology reconfigurations and this happens in a heavily loaded network than in a lightly loaded one.

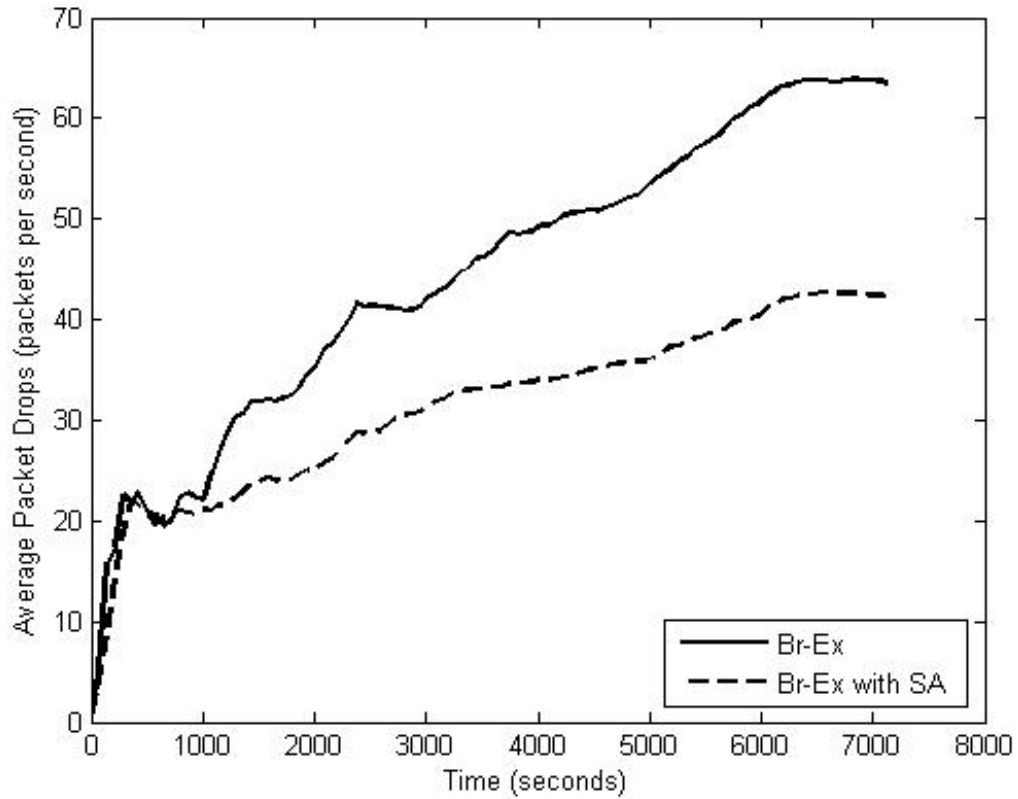


Figure 5.11: Average Packet Drops with Branch Exchange heuristic with and without Successive Approximations - Average from 5 simulation runs

5.5 Branch Exchange

The branch exchange heuristic outlined in section 4.4 was used to identify the optimal topology that differs from the current topology by one branch exchange. This operation is repeated 4 times to obtain the target topology for topology re-configuration. A simulation run, as outlined in the previous section, consists of simulating the network for a duration of 2 hours while the traffic demand matrix is changed every 10 seconds as described in section 5.2. Five different simulation

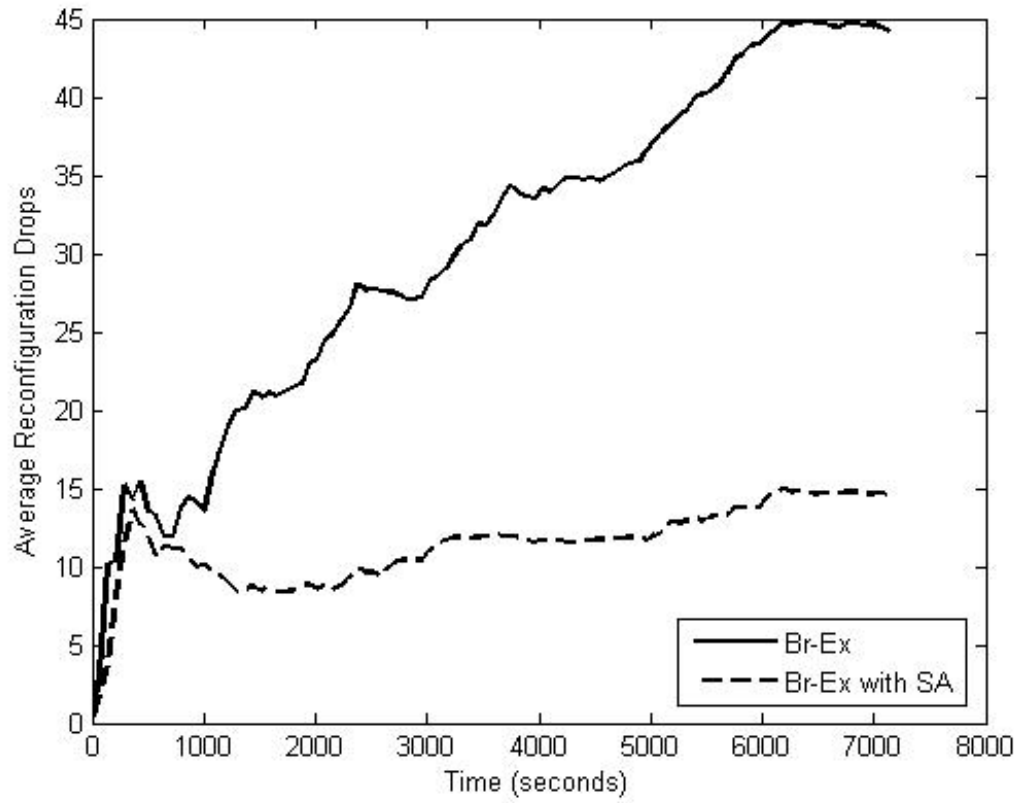


Figure 5.12: Average Reconfiguration Drops with Branch Exchange heuristic with and without Successive Approximations - Average from 5 simulation runs

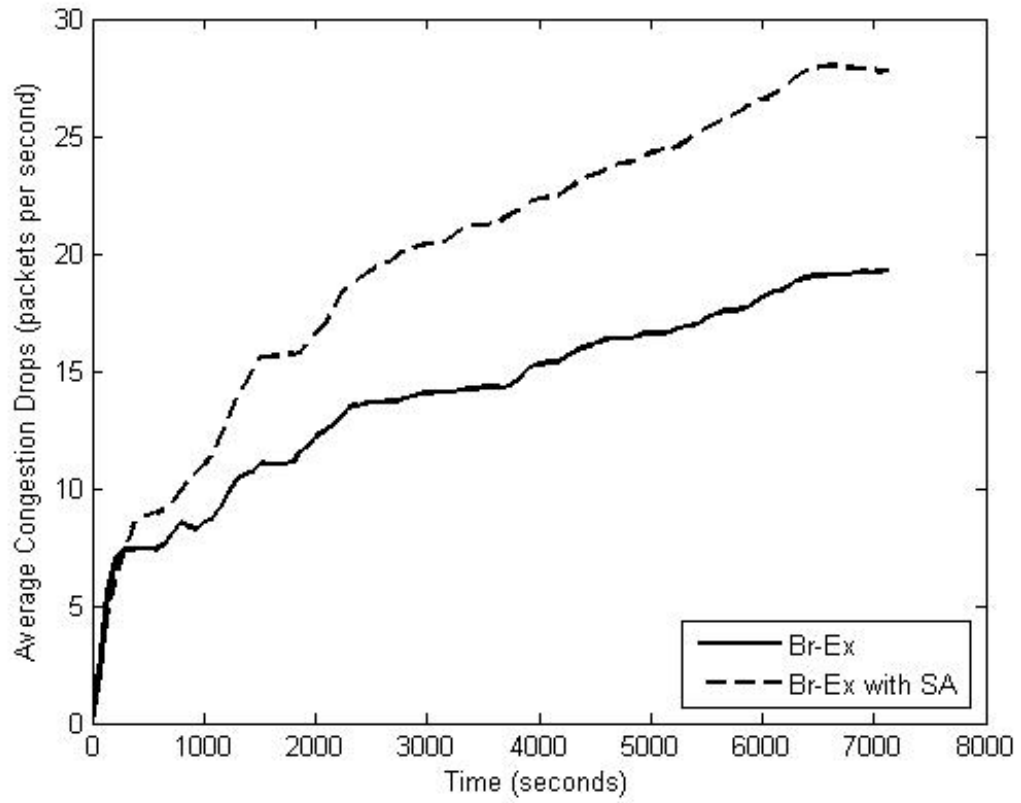


Figure 5.13: Average Congestion Drops with Branch Exchange heuristic with and without Successive Approximations - Average from 5 simulation runs

	Top. Computations	Top. Reconfigurations	Links Changed
Br-Ex	335.2	178.6	471.2
Br-Ex with SA	372.8	43	154.5

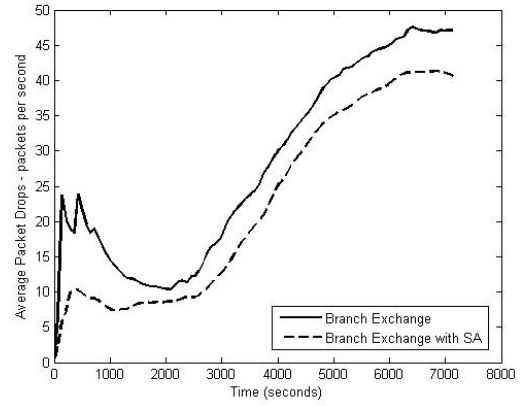
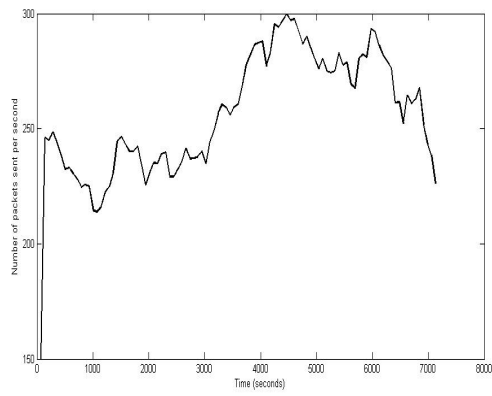
Table 5.3: Topology Recomputations - Averages from five simulation runs using Branch Exchange heuristic

runs were made with each run differing in the pattern of traffic demand changes. The running average of total packet drops, averaged over the five simulation runs, is shown in figure 5.11. The figures 5.12 and 5.13 show the running average of the average reconfiguration drops and the average congestion drops respectively. As noted in the previous section, while the congestion drops increase when successive approximations are enabled, the reconfiguration drops decrease and this results in a lower total packet drops. Fitting the running average of the packet drops to a cubic polynomial function, the leading coefficients obtained are $1.5081e-10$ and $1.3479e-10$ respectively for the branch exchange heuristic and the branch exchange with successive approximations heuristic (with error variances 0.21 and 0.20). Clearly, the packet drops are shown to be statistically lower with successive approximations enabled because of the lower leading coefficient.

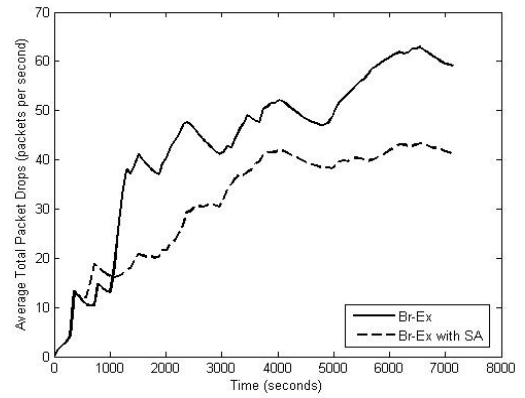
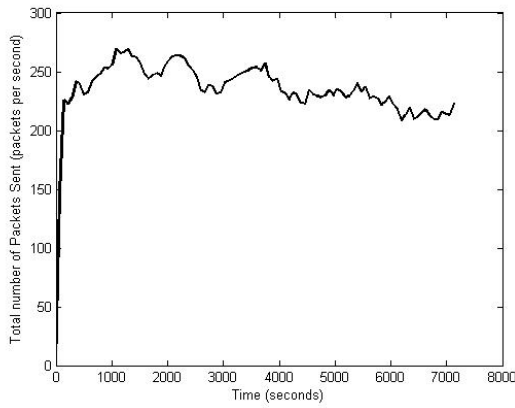
Table 5.3 shows the decrease in the number of topology reconfigurations that is achieved when successive approximations is used with the branch exchange heuristic. The average number of topology reconfigurations is 43 when successive approximations is enabled while that number is 178.6 when SA is disabled. In other words, on

an average, a topology reconfiguration is implemented in the network every 168 seconds in the network with the branch exchange heuristic when SA is enabled whereas a topology reconfiguration is implemented in the network every 40 seconds when the branch exchange heuristic is used without SA (Again, a note of caution is that these numbers cannot be directly applied in a practical scenario because the traffic matrix in the simulations are changed every 10 second which are not necessarily the case in a practical setting). This decrease in the number of reconfigurations happens in spite of an increase in the number of times a topology computation is triggered.

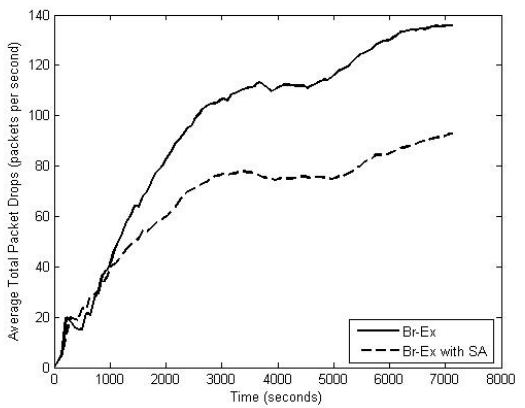
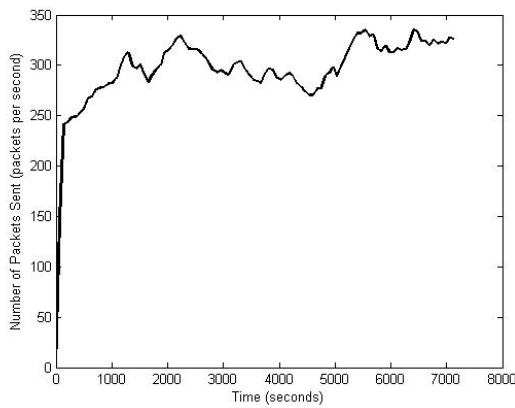
The total packet drops - with and without SA - from the individual simulation runs are shown in figures 5.14 and 5.15. For each run, the total number of packets sent for each traffic pattern is shown along with the running average of the packet drops. The packet drops for the different simulation runs differ by an order of magnitude or more, similar to the packet drops when the rollout heuristic was used. For every simulation run, however, the average packet drops are decreased when successive approximations are enabled and it was also shown statistically that enabling SA results in fewer packet drops based on the averages from the individual simulation runs.



(a)

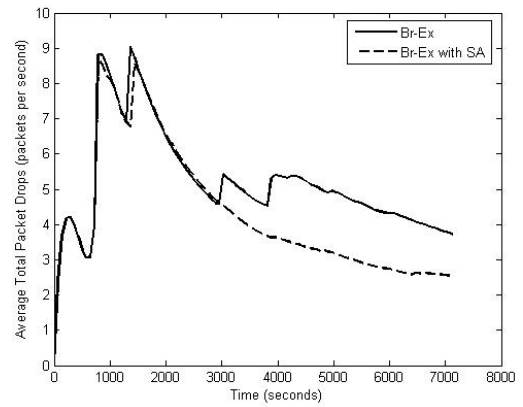
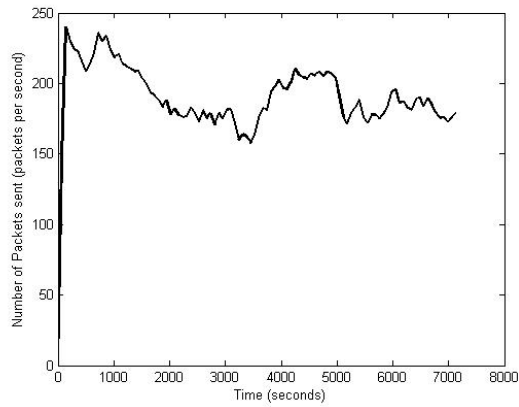


(b)

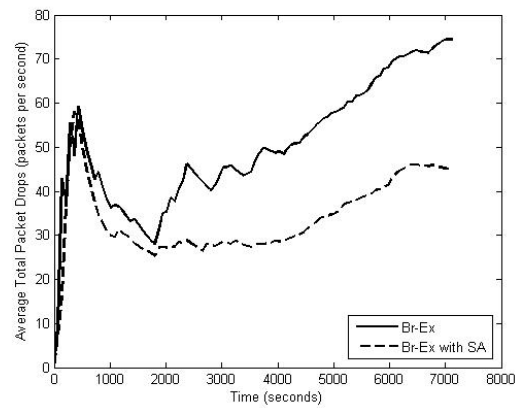
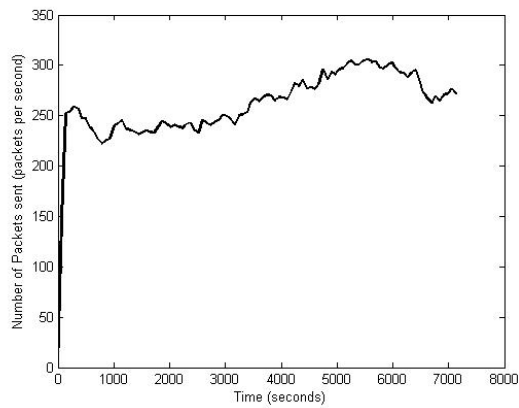


(c)

Figure 5.14: Total Packets Sent and Average Packet Drops with traffic patterns (a) 1, (b) 2 and (c) 3 with Branch Exchange heuristic



(a)



(b)

Figure 5.15: Total Packets Sent and Average Packet Drops with traffic patterns (a) 4 and (b) 5 with Branch Exchange heuristic

Chapter 6

Summary and Conclusions

The twofold goals of this research were to a) understand how a topology reconfiguration can be implemented with minimum cost and b) to further refine the topology control process so that the computed target topologies lend themselves to easier reconfiguration. A natural metric for the reconfiguration cost is the packet drops that happen during a reconfiguration. Intuitively, these packet drops are minimized when the changes that happen in the network topology are localized. This observation leads to the proof in section 3.2 that the minimum reconfiguration cost is achieved when the topology reconfiguration is replaced with sequences of branch exchanges that exchanges two edges in the graph in each step. Section 3.3 describes the algorithm that can be used for generating these branch exchange sequences. The reduction in packet drops when this algorithm is used can be upper bounded in the case of a uniform traffic matrix and simulations verify that this upper bound can be achieved by the proposed algorithm(fig 3.4).

In a dynamic setting, the changes in traffic demand trigger topology recomputations that minimize the congestion cost. As the example in fig 4.2 and fig 4.3 shows this computed target topology could have a high reconfiguration cost. Another target topology that has a much lower reconfiguration cost but only a slightly worse congestion metric could exist. An objective function that minimizes the over-

all cost is needed where the cost has to be appropriately defined. The expected packet drops are chosen as the cost metric and section 4.3.2 shows how the congestion metric were converted to the expected packet drops. Assuming a PAT duration and a mean time between reconfigurations, the reconfiguration cost and congestion cost can be combined to obtain the objective function(eqn 4.17).

The heuristics that are typically used for congestion minimization can be modified to reflect the different minimization objectives. As examples, the roll-out(singlehop) heuristic and the branch exchange heuristics were chosen and the effect of using successive approximations on these heuristics were studied using simulations of a FSO network operation under a time-varying traffic demand. The entries of the traffic matrix are increased or decreased with equal value and probability over time. This imbalance in the traffic demand loads some of the links to their capacity and this triggers a topology recomputation. The packet drops that happen in this situation were observed for the various heuristics and the results are discussed in sections 5.4 and 5.5. It was observed that total packet drops are decreased when successive approximations are used and this happens in spite of the increase in packet drops due to congestion. Based on the observed packet drops in five simulation runs, it was statistically shown that using successive approximations decreases the total packet drops. More simulation runs and statistical analysis will help in evaluating the effectiveness of successive approximations with more precision. It was also observed that using successive approximations resulted in fewer topology reconfigurations and hence, a more stable network.

Chapter A

Pseudocode for Successive Approximations

```
/*
 * Input - Current and Target ring represented as a list of edges
 *
 *          Traffic matrix
 * Returns - Minimum possible reconfiguration cost
 * Also generates branch exchange sequence to achieve target topology
 */
static float calculate_satr_cost(int **c_ring, int **n_ring, float **tr_matrix)
{
    //Represent the topologies in matrix form
    convert c_ring to orig_matrix;
    convert n_ring to target_matrix;
    disconnected_matrix = orig_matrix && target_matrix;

    start_cost = calculate_reconfiguration_cost(disconnected_matrix, tr_matrix);
    while(1)
    {
        brex_cost = get_br_exchange(orig_matrix, target_matrix,
                                    disconnected_matrix, tr_matrix);
```

```

//orig_matrix is updated with the branch exchange
disconnected_matrix = orig_matrix && target_matrix;
next_cost = calculate_reconfiguration_cost(disconnected_matrix, tr_matrix)
if (start_cost > brex_cost + next_cost)
{
    start_cost = next_cost;
    total_cost += brex_cost;
}
else
{
    total_cost = start_cost + total_cost;
    break;
}
}
return total_cost;
}

/*
* Input - The transient topology represented in matrix form.
*         Traffic demand matrix.
* Returns - the total packets per second that will be dropped during this reconfi
*/
static float calculate_reconfiguration_cost(int **c_matrix, float **tr_matrix)

```

```

{
    cost_graph = init_graph();
    insert_nodes(cost_graph);
    insert_edges(cost_graph, c_matrix);

    cost = 0;
    for (i=0; i<no_nodes; i++)
        for (j=0; j<no_nodes; j++)
            {
                if (!path_exists(i,j))
                    cost = cost + tr_matrix[i][j];
            }
    return(cost);
}

/*
 * Input - Current and Target topology in matrix representation
 *          Traffic demands matrix
 * Output - The branch exchange with minimum reconfiguration cost
 */
static float
get_br_exchange(int **orig_matrix, int **target_matrix, int **top_matrix, float *
{

```

```

top_matrix = orig_matrix - target_matrix;

//Generate Auxiliary Graph - A node whenever top_matrix[i][j] is 1 or -1
aux_graph = init_graph();

insert_nodes(aux_graph, top_matrix);

//Insert edges from -1 to 1 in the same column and from 1 to -1 in same row
insert_edges(aux_graph, top_matrix);

//From cycles of size 4 in aux_graph, select cycle with minimum reconfigurati
for all cycles in aux_graph
{
    top_matrix = get_cycle(aux_graph);

    cost = calculate_reconfiguration_cost(top_matrix, tr_matrix);

    if (min_cost > cost)
    {
        min_cost = cost;

        min_br_exchange = top_matrix;
    }
}

top_matrix = min_br_exchange;

return(min_cost);
}

```

Bibliography

- [1] C.C.Davis, I.I.Smolyaninov, and S.D.Milner, “Flexible optical high data rate wireless links and networks,” *IEEE Commun. Mag.*, pp. 51–57, Mar. 2003.
- [2] J.-F. P.Labourdette, G. W.Hart, and A. S.Acampora, “Branch-exchange sequences for reconfiguration of lightwave networks,” *IEEE Trans. Commun.*, vol. 42, pp. 2822–2832, Oct. 1994.
- [3] A. Desai, J. Llorca, and S. D. Milner, “Autonomous reconfiguration of backbones in free space optical networks,” *IEEE MILCOM*, 2004.
- [4] J. Llorca, A. Desai, and S. D. Milner, “Obscuration minimization in dynamic free space optical networks through topology control,” *IEEE MILCOM*, 2004.
- [5] J. Llorca, A. Desai, U. Vishkin, C. Davis, and S. Milner, “Reconfigurable optical wireless sensor networks,” *Proc. of SPIE Remote Sensing*, Sept. 2003.
- [6] I. Baldine and G. N.Rouskas, “Traffic adaptive wdm networks: A study of reconfiguration issues,” *J. Lightwave Technol.*, pp. 433–455, Apr. 2001.
- [7] D. Banerjee and B. Mukherjee, “Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study,” *IEEE/ACM Trans. Networking*, pp. 598–607, Oct. 2000.
- [8] G. N.Rouska and M. H.Ammar, “Dynamic reconfiguration in multihop wdm networks,” *J. High Speed Networks*, vol. 4, no. 2, pp. 221–238, 1995.
- [9] A. Desai, “Dynamic topology control of free space optical networks,” *Master’s Degree Thesis, University of Maryland*, 2003.
- [10] R. Ramanathan and R. Rosales-Hain, “Topology control of multihop wireless network using transmit power adjustment,” *Proc. of INFOCOM*, pp. 404–413, Mar. 2000.
- [11] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang, “Distributed topology control for power efficient operation in multihop wireless ad hoc networks,” *Proc. of INFOCOM*, July 2001.
- [12] N. Li, J. C. Hou, and L. Sha, “Design and analysis of an mst-based topology control algorithm,” *Proc. of INFOCOM*, 2003.
- [13] V. Rodoplu and T. H. Meng, “Minimum energy mobile wireless networks,” *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1333 – 1344, Aug. 1999.
- [14] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. McGraw Hill, 2000.

- [15] P. Gupta and P. Kumar, “The capacity of wireless networks,” *IEEE Trans. Inform. Theory*, vol. 46(2), pp. 388–404, Mar. 2000.
- [16] S. D. Milner, S. Thakkar, K. Chandrashekar, and W.-L. Chun, “Survivability and quality of service in mobile wireless networking,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7(2), pp. 69–79, 2003.
- [17] P. Fetterolf and G. Anandalingam, “Optimizing interconnections of local area networks: An approach using simulated annealing,” *ORSA-Journal on Computing*, vol. 3(4), pp. 275 – 287, 1991.
- [18] —, “Optimal design of lan-wan internetworks: An approach using simulated annealing,” *Annals of Operations Research*, vol. 36, pp. 275 – 298, 1992.
- [19] S. Milner and A. Desai, “Autonomous reconfiguration in free-space optical sensor networks,” *IEEE J. Select. Areas Commun.*, vol. 23, pp. 1556–1563, Aug. 2005.
- [20] R. Ramaswami and K. N. Sivarajan, “Design of logical topologies for wavelength routed optical networks,” *IEEE J. Select. Areas Commun.*, vol. 14(5), pp. 840–851, June 1996.
- [21] E. Leonardi, M. Mellia, and M. A. Marsan, “Algorithms for the logical topology design in wdm all optical networks,” *Optical Networks Magazine*, pp. 35–46, Jan. 2000.
- [22] A. Kashyap, K. Lee, and M. Shayman, “Rollout algorithms for integrated topology control and routing in wireless optical backbone networks,” *Technical Report, Institute of Systems Research, University of Maryland*, 2006.
- [23] A. Narula-Tam and E. Modiano, “Dynamic load balancing in wdm packet networks with and without wavelength constraints,” *IEEE J. Select. Areas Commun.*, vol. 18(10), pp. 1972–1979, Oct. 2000.
- [24] D. Bienstock and O. Gunluk, “Computational experience with a difficult mixed-integer multicommodity flow problem,” *Mathematical Programming*, vol. 68, pp. 213 – 237, 1995.
- [25] X. Yang and B. Ramamurthy, “An analytical model for virtual topology reconfiguration in optical networks and a case study,” *Proc. IEEE ICCCN*, pp. 302–308, Oct. 2002.
- [26] B. Ramamurthy and A. Ramakrishnan, “Virtual topology reconfiguration of wavelength routed optical networks,” *Proc. IEEE GLOBECOM*, pp. 1269–1275, Nov. 2000.
- [27] I. Baldine and G. N. Rouskas, “Dynamic load balancing in broadcast wdm networks with tuning latencies,” *Proc. IEEE INFOCOM*, pp. 78–85, Mar. 1998.

- [28] S. Sinha and C. S. R. Murthy, "Information theoretic approach to traffic adaptive wdm networks," *IEEE/ACM Trans. Networking*, vol. 13(4), Aug. 2005.
- [29] A. Gencata and B. Mukherjee, "Virtual-topology adaptation for wdm mesh networks under dynamic traffic," *Proc. IEEE INFOCOM*, pp. 48–56, June 2002.
- [30] D. Bertsekas and R. Gallager, *Data Networks*. Prentice Hall India, 1992.
- [31] OPNET, "Modeler," *www.opnet.com*.
- [32] A. Desai, E. Baskaran, and S. Milner, "Modeling and simulation of point to point broadband wireless networks," *OPNETWORK*, Dec 2005.
- [33] T. Ho, S. Trisno, I. Smolyaninov, S. Milner, and C. Davis, "Studies of pointing, acquisition and tracking of agile optical wireless transceivers for free-space optical communication networks," *Proc. SPIE Conference, Remote Sensing, Barcelona, Spain*, pp. 142–158, Sept. 2003.