ABSTRACT

Title of Thesis:	Profile Based Topology Control and Routing in Wireless Optical Networks
Degree candidate:	Abhishek Kashyap
Degree and year:	Master of Science, 2004
Thesis directed by:	Professor Mark Shayman Department of Electrical and Computer Engineering

The problem of topology control and routing of bandwidth-guaranteed flows over wireless optical backbone networks is addressed. The input is a potential topology and a traffic profile. The constraints are that of limited interfaces at each node and the limited link bandwidth, and the objective is to maximize the throughput. The problem turns out to be NP-Hard.

A new framework for integrated topology control and routing is proposed. A simple heuristic is proposed, and efficient rollout algorithms are proposed which enhance the heuristic. The routing problem is formulated as a multi-commodity flow problem, and is used to enhance the rollout algorithms to achieve a higher throughput.

Another set of heuristics is proposed which use matching theory and multi-commodity flow formulation of routing to achieve the desired results. We enhance the heuristics to provide fairness to the ingress-egress pairs in terms of how much traffic we route for each of them.

Profile Based Topology Control and Routing in Wireless Optical Networks

by

Abhishek Kashyap

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
2004

Advisory Committee:

Professor Mark Shayman, Chair
Professor Samir Khuller
Professor Sennur Ulukus

DEDICATION

I dedicate this thesis to my family for their love and support.

ACKNOWLEDGMENTS

I am grateful to my advisor, Prof. Mark Shayman for his advice, support and encouragement in both academic and personal matters. I thank Prof. Samir Khuller for his advice and guidance on this work. I thank Prof. Mark Shayman, Prof. Samir Khuller and Prof. Sennur Ulukus for agreeing to serve on the thesis examination committee. My special thanks go to Mehdi Kalantari, Kwangil Lee, Tuna Guven and Fangting Sun for their valuable suggestions and feedback.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

World wide internet services, data communications, multimedia, virtual navigation and tele-medicine are demanding greatly increased bandwidth on wireless networks. Over the last few years, a number of approaches have been taken to meet the explosive traffic growth. They include efficient signal coding and modulation schemes, spatial processing using microwave phased array antennas, and the transfer to higher radio frequency for the carrier [1]. More recently, free-space optics is attracting great attention as an alternative to radio and wireline networks because of its attractive characteristics. Free-space optics technology is expected to deliver unprecedented bandwidth, massive carrier reuse, ultra-low inter-channel interference, low power consumption, and cost savings where electrical wires and optical fibers are too expensive to deploy and maintain. A key distinguishing feature of wireless optical networks is that the links are point-to-point rather than broadcast. Also, it has wide applicability from long range satellite to indoor wire-

less communications [1]. Therefore, wireless communication network design using free-space optics has become an important issue.

A suitable use of wireless optical links is in the backbone of hierarchical mobile ad-hoc networks (MANETs). Hierarchical MANETs have been proposed as a scalable extension to ad-hoc networks [12]. A hierarchical MANET consists of mobile devices (nodes) being divided into clusters. Each cluster has a cluster-head, and the cluster-heads form a backbone network. Within a cluster, each node can contact other nodes using RF links. For a node to contact another node in a different cluster, it needs to use the backbone network. Considering the large bandwidth requirements for the backbone links, and the need to provide Quality of Service (QoS) and traffic engineering for the traffic on the backbone links, more reliable and higher bandwidth links are required. Wireline links being too time-consuming and expensive to deploy, wireless optical links seem suitable at the backbone level of a hierarchical MANET.

The importance of providing QoS and traffic engineering to incoming traffic has become very important. Thus, backbone network design needs to incorporate QoS demands of the traffic flows. QoS requirements for delay and packet loss can be converted into bandwidth requirements [11]. Thus, it suffices to work with bandwidth guarantees. The network ingress/egress nodes are normally known, and the traffic profile (which is the aggregate demand between ingress-egress pairs in our case) can be measured over previous operation of the network or can be had from service level agreements (SLAs). This information can be used to provide

bandwidth guarantees to individual ingress/egress pairs, and for different traffic classes in the case of an MPLS network. The bandwidth reservations and routes calculated can be used for routing and admission control when the network is formed. Specifying explicit paths and bandwidth reservations on those paths allows the service providers to do traffic engineering on the incoming traffic. This protects the network from potential flooding by the ingress-egress pairs (which require more bandwidth at run-time than mentioned in the SLAs) by blocking their calls when they exceed the reserved bandwidth. A detailed description of this routing framework for wireline networks with fixed topology can be found in [3].

1.2 Related Work

Considerable amount of work has been done on topology control in wireless RF networks and wireline optical networks. We explain the differences between the existing work and our problem here.

The problem of topology control for wireless optical networks is different from that in wireless RF (radio frequency) networks since the links are point-to-point as opposed to broadcast. In wireless optical networks, each node has a limited number of transceivers, and hence can establish links with only a limited number of nodes within its transmission range. Thus, topology control is concerned with determining the neighbors with which to establish the limited number of possible links. In wireless networks, most research for topology control so far has focused on

RF networks (see e.g., [13, 14, 19, 20, 21, 22, 23]). In RF-wireless networks with isotropic antennas, topology control is closely related to power control. Power is controlled to reduce the transmission range to conserve power and decrease interference while providing adequate connectivity.

There are important differences between topology control for reconfigurable wireline optical networks and topology control for wireless optical networks. In the wireline case, transmission range (lightpath length) is not a major issue. Furthermore, if the optical layer has sufficient resources so the routing and wavelength assignment problem is always solvable, then whenever a source and destination both have available interfaces, a direct connection (one logical hop) can be established. In contrast, in the wireless case, unless the destination is within the transmission range of the source, a multihop connection is required. For these reasons, the many published results on logical topology design for wireline optical networks, [15, 16, 17, 18], are not directly applicable to free-space optical networks.

There has been recent work on topology control in wireless optical networks ([4], [5]): [5] does not take traffic into consideration while forming a network, while [4] considers only ring topologies.

1.3 Proposed Heuristics

The problem addressed is finding a topology and bandwidth reservations for a given potential topology (nodes and their potential neighbors (potential links))

and a given traffic profile. The total bandwidth reservations we are able to provide is called throughput here.

A new framework is proposed which solves the topology control and routing problems simultaneously while maximizing the throughput. A heuristic is proposed, and rollout algorithms are proposed which are guaranteed to work better than the heuristic. An extension of the rollout algorithms is proposed which allows the splitting of aggregate traffic between an ingress-egress pair as a commodity (routing problem is formulated as a multi-commodity flow problem).

Another heuristic is proposed which uses matching theory to come up with an initial topology based on link weights calculated using the traffic profile. The routing problem is then solved as a multi-commodity flow problem, and topology is sequentially changed so as to improve the throughput. An extension is proposed which tries to provide fairness to the traffic routed (or bandwidth reserved on the links) between different ingress-egress pairs.

1.4 Contribution

This work addresses the problem of topology design and routing of bandwidth-guaranteed flows in wireless optical networks taking traffic engineering into consideration. The area of research is new in itself, and there are no current algorithms to address this problem for a general network ([4] proposes some algorithms which only consider ring topologies). We prove the problem to be NP-Hard, and propose

some heuristics for finding sub-optimal solutions to the problem in a reasonable amount of time. The search space of the solution very large, thus making it infeasible to find optimal solutions even for small networks. Thus, the algorithms provided are expected to be useful for solving the problem addressed. The algorithms and the multi-commodity flow formulation of routing are extended to incorporate fairness for the traffic routed.

1.5 Thesis Organization

Chapter 2 gives the network model and provides a formal description of the problem along with the proof of NP-Hardness. Chapter 3 explains the rollout algorithms and the corresponding framework, along with simulation results and discussion. Chapter 4 discusses an extension to the rollout algorithms in which we allow the routing problem to be modelled as a multi-commodity flow problem. Chapter 5 discusses a matching theory based approach for solving the problem. It also proposes an extension which provides fairness to the traffic. Comparison results are presented at the end of Chapters 4 and 5. Chapter 6 concludes the thesis.

Chapter 2

Network Model and Problem Statement

2.1 Network Model

The network is modelled as a graph $G = (V, E)$, where V is the set of nodes and E the set of links between them (we call the links in the potential topology as the potential links). We consider wireless backbone networks in which each wireless node is equipped with point-to-point wireless optical interfaces. The term 'node' implicitly means "backbone node". Each node has the capability to perform routing. We assume that it does not move very frequently. We also assume that wireless links can be set up in any direction with all the nodes within transmission range, and take optical beam obscuration into consideration-i.e., some nodes within the transmission range may not be able to connect. Since the transmission distance is related to the power level of the node, the power level and thus the transmission range of each node can be different. The wireless links are unidirectional. If there is a pair of unidirectional links between two nodes, the link capacities may differ. The number of transmitters and receivers at each node is limited (which we call

an interface constraint), thereby restricting the number of nodes to which it can connect.

We have a traffic profile, which consists of the aggregate traffic demands between the sources and destinations. The traffic demand from node x to node y can be different from the traffic demand from node y to node x.

2.2 Problem Statement

Given a graph $G = (V, E)$, the problem addressed is to form a subgraph $G' = (V, E')$, such that the interface constraints are satisfied for all nodes in the set V (i.e., the degree of each node is bounded by the number of available interfaces), and the throughput is maximized considering the traffic profile. The algorithm forms this subgraph, which we call topology control and comes up with routes and bandwidth reservations for the ingress-egress pairs given in the traffic profile. The server should recompute the topology, routes and bandwidth reservations whenever either the traffic profile or the (backbone) node locations change significantly. We do not anticipate that this would be done more often than hourly. The nodes then use this information to perform routing and traffic engineering on incoming flows.

Theorem 1 *Given a graph $G = (V, E)$ and a traffic profile consisting of traffic demands between different vertices of G, the problem of finding a degree constrained subgraph $G' = (V, E')$ and routes for the traffic demands such that the total demand routed is maximized is NP-Hard.*

Proof: Consider a small amount of traffic between each pair of nodes in the network (small enough not to violate any link bandwidth constraints). The problem of maximizing the throughput reduces to finding a connected subgraph (satisfying the degree constraints) here. We can remove the extra edges and the problem reduces to finding a degree-constrained spanning tree, which is a known NP-Hard problem. A special case is where we have a degree constraint of 1 incoming and 1 outgoing edge on each node. In this case, the problem reduces to finding a hamiltonian cycle, which is known to be NP-Complete [7].

We also consider providing fairness to the ingress-egress pairs in terms of how much reservations we provide for each of them, in addition to maximizing the throughput.

Chapter 3

Rollout Algorithms

3.1 Integrated Topology Control and Routing Framework

We propose a framework for finding the topology, routes and bandwidth reservations in an integrated way, so as to maximize the throughput while satisfying the interface and bandwidth constraints. Given a potential topology and traffic profile, we follow the following steps:

1. A demand is chosen based on some criteria and a locally optimal path (satisfying the interface constraints and bandwidth constraints) is computed for the demand. If none exists, the demand is rejected.
2. If the path includes potential links, then those links are marked as actual links.
3. The capacity of each link on the path in the existing topology is updated (decreased) to incorporate the bandwidth allocated to the demand routed.
4. The topology is updated by eliminating all the potential links that lead to the violation of interface constraints i.e., at all the nodes for which the

number of actual incoming (outgoing) links equals the number of interfaces, the incoming (outgoing) potential links incident on (going out of) those nodes are eliminated.

5. Steps 1, 2, 3 and 4 are repeated until all demands are either provisioned or rejected. This way, a topology is created from the potential topology and all the routes are computed for the demands given in the traffic profile (the ones we are able to route, the others are rejected).

Let us explain this approach of integrated topology control and routing with an example. In this example, we assume that each node has two interfaces available for establishing bidirectional links. The traffic profile is sorted in the order of decreasing demands, and demands are selected in that order. The link capacity of each link is assumed to be 10 units. We use constrained shortest-path routing for path selection, with the constraints being the limited interfaces and bandwidth. The weight of each link is assumed to be 1. Let the traffic demands be: $t_{38} = 6, t_{18} = 5, t_{45} = 3, t_{37} = 2$. We compute the shortest path for the first demand (first entry of the traffic matrix (profile)) t_{38} using the topology as shown in Figure 3.1. The shortest path for the traffic demand t_{38} is $3 - 5 - 8$. Figure 3.2 shows the potential topology after converting the potential links along the path $3 - 5 - 8$ to actual links and allocating the bandwidth for the demand. In Figure 3.2, the actual links are represented by thick lines and the potential links are represented by thin lines. As the number of available interfaces per node is two and node 5 uses those

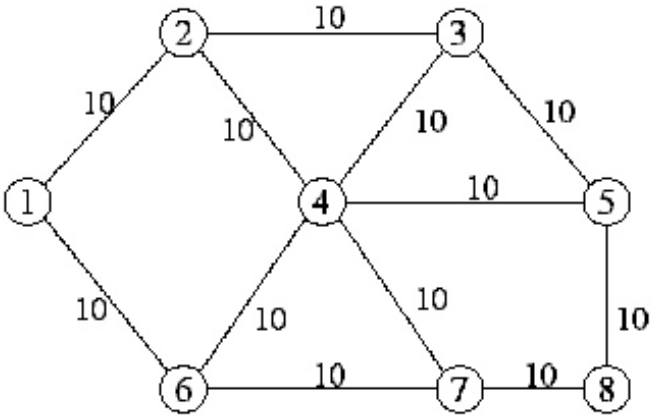

Figure 3.1: Potential Topology

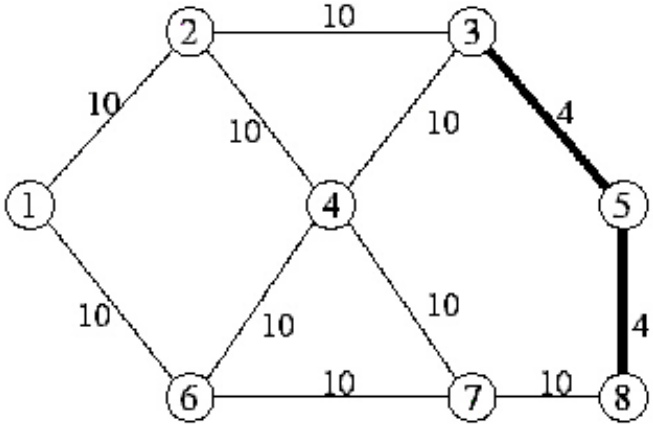

Figure 3.2: Topology after routing t_{38}

interfaces for links with node 3 and node 8, there are no more interfaces available for node 5 to establish a link with other nodes. Thus, the potential link between node 4 and 5 is eliminated, as can be seen by comparing Figures 3.1 and 3.2.

In the network of Figure 3.2, we find the shortest path $1-6-7-8$ for the demand t_{18} and the shortest path $4-3-5$ for t_{45}. Figure 3.3 shows the updated topology which reflects the routing of these demands. Now we compute the shortest

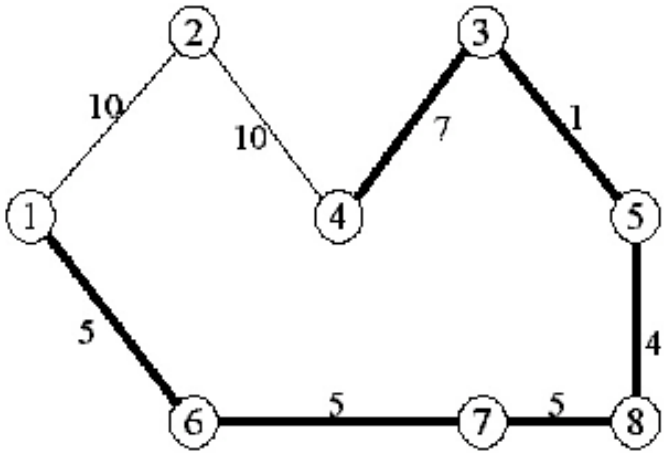

Figure 3.3: Topology after routing t_{38}, t_{18}, t_{45}

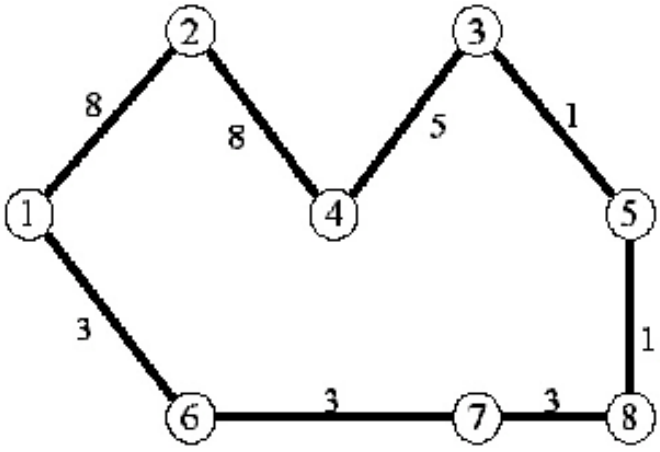

Figure 3.4: Topology after routing t_{38}, t_{18}, t_{45}, t_{37}

path for the demand t_{37} using the modified network, as shown in Figure 3.3. There are two paths available for t_{37}: $3-5-8-7$ and $3-4-2-1-6-7$. Since the available bandwidth along the path $3-5-8-7$ is 1, which is less than the demand, the path cannot be selected even though it is the shortest path in the network. So, the shortest path for the demand t_{37} is computed as $3-4-2-1-6-7$, and the network topology updated to get the final topology as shown in Figure 3.4.

3.1.1 Issues in Integrated Topology Control and Routing

The purpose of this integrated approach is to maximize the network throughput while routing demands sequentially. There are two key issues related to it. Let us consider them with the help of two example networks (potential topologies) shown in Figures 3.5 and 3.8. In these examples, the number of available interfaces at each node is two, and the links are assumed to be bidirectional for simplicity. Given the traffic matrix $\{t_{12}, t_{34}, t_{56}, t_{78}\}$, all demands being the same, consider the path provisioning and topology design for the network in Figure 3.5. When we provision a path for t_{12} first (shown by thick lines), we get the topology (thin lines indicate eliminated links) as shown in Figure 3.6 resulting in only one demand being provisioned. If we consider the other traffic demands first, then this demand cannot be provisioned but the other three demands can be provisioned, as shown in Figure 3.7. Thus, the topology of Figure 3.7 resulting from choosing the last three traffic demands before the first one gives a better throughput.

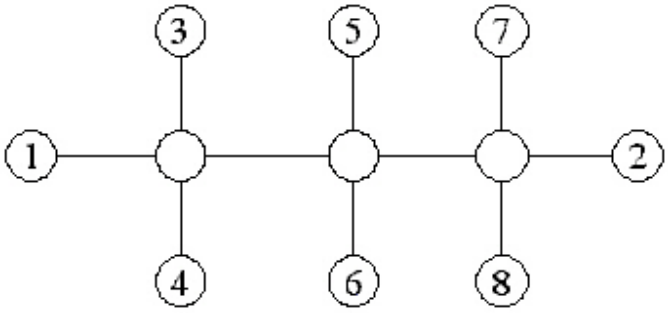

Figure 3.5: Potential Topology

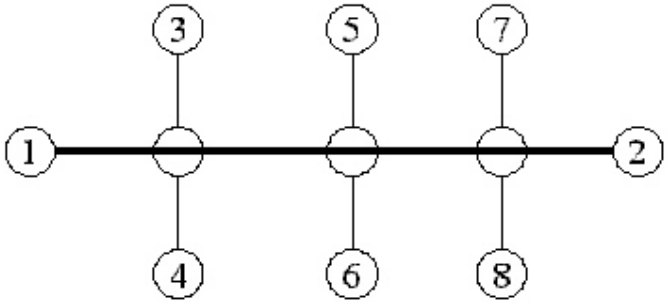

Figure 3.6: Path for t_{12}

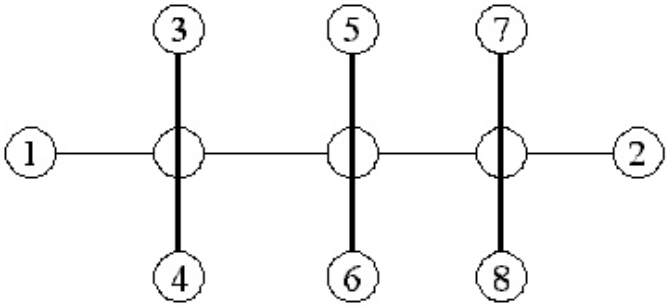

Figure 3.7: Paths for t_{34}, t_{56}, t_{78}

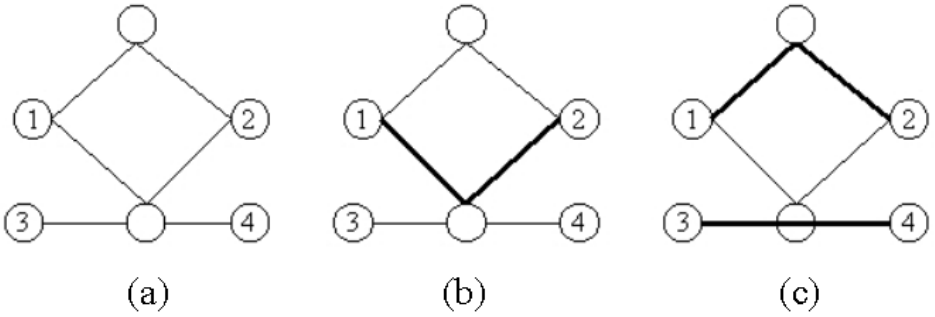

Figure 3.8: Topology Generation by using different shortest paths for a demand: (a) Example network, (b) Path for t_{12}, and (c) Paths for t_{12} and t_{34}.

Let us consider another example with potential topology as in Figure 3.8(a). We consider path provisioning for the sorted traffic demands $\{t_{12}, t_{34}\}$. There are two paths available for the demand t_{12}. If we choose a path for t_{12} as shown in Figure 3.8(b), then we cannot provide a path for t_{34} because of the interface constraint at an intermediate node. However, when we choose the other path as shown in Figure 3.8(c), both the traffic demands can be provisioned.

The above examples illustrate the importance of the two factors that affect the network throughput in our integrated algorithm: The sequence in which we route the demands given in the traffic matrix, and the selection of paths for routing the demands. These two factors affect the selection of links whose bandwidth will be used by the routed demand, and selection of links to be deleted due to interface constraints. So, these factors affect the future path computations and the output topology.

3.2 Rollout Algorithms for Topology Control and Routing

As mentioned in the section 3.1.1, the throughput of the network formed by our topology control and routing framework depends on the order in which the traffic demands are considered for link formation and routing, and the selection of the path for each demand. We start with reasonable heuristics for demand ordering and path selection and use the rollout technique, [8] to improve the heuristics to obtain potentially near-optimal solutions.

3.2.1 Basic Rollout Algorithm

Rollout is a general method for obtaining an improved policy for a Markov decision process starting with a base heuristic policy [8]. The rollout policy is a one step look-ahead policy, with the optimal cost-to-go approximated by the cost-to-go of the base policy. We use the specialization of rollout to discrete multistage deterministic optimization problems. Consider the problem of maximizing $G(u)$ over a finite set of feasible solutions U. Suppose each solution u consists of N components $u = (u_1, .., u_N)$. We can think of the process of solving this problem as a multistage decision problem in which we choose one component of the solution at a time. Suppose that we have a heuristic algorithm, the so-called "base heuristic", that given a partial solution $(u_1, .., u_n), (n < N)$, extends it to a complete solution $(u_1, .., u_N)$. Let $H(u_1, .., u_n) = G(u_1, .., u_N)$. In other words, the value of H on the partial solution is the value of G on the full solution resulting from application of

the base heuristic. The rollout algorithm R takes a partial solution $(u_1, .., u_{n-1})$ and extends it by one component to $R(u_1, .., u_{n-1}) = (u_1, .., u_n)$ where u_n is chosen to maximize $H(u_1, .., u_n)$. Thus, the rollout algorithm considers all admissible choices for the next component of the solution and chooses the one that leads to the largest value of the objective function if the remaining components are selected according to the base heuristic. It can be shown that under reasonable conditions, the rollout algorithm will produce a solution whose value is at least as great as the solution produced by the base heuristic. Note that the heuristic may be a greedy algorithm, but the rollout algorithms are not greedy as they make a decision based on the final expected value of the objective function, and not the increment to the value of the objective function at that decision step. The rollout algorithm typically achieves a substantial performance improvement over the base heuristic at the expense of extra computation that is equal to the computation time of the base heuristic times a factor that increases polynomially with the problem size.

3.2.2 Rollout Algorithms for Topology Control and Routing

In this section, we propose four different rollout algorithms: index rollout, route rollout, sequential rollout and integrated rollout. We start by explaining the base heuristic.

3.2.2.1 Base Heuristic

The base heuristic works as follows: Suppose that a partial topology has been obtained by choosing routes for n demands $(t_1, .., t_n)$ from the traffic matrix (demands indexed by source-destination pair IDs). The base heuristic routes the remaining demands in decreasing order of magnitude. (Routing demands in decreasing order of magnitude is known to be a useful heuristic for reconfigurable wireline optical networks [16, 18]). For each demand, it chooses a route using constrained shortest path first (CSPF), with constraints being that of interfaces and bandwidth. Thus, t_{n+1} is the largest remaining demand. The route chosen for this demand is a shortest unidirectional path in the partial topology satisfying the constraints. This means that every actual link in the path must have sufficient residual bandwidth for the demand; every potential link in the path must have an available transmitter at its head node and an available receiver at its tail node. If there is no feasible path, then the 'null' route is assigned–i.e., the demand is blocked. If there is a feasible path, the heuristic updates the topology by deleting the potential links which violate the interface constraints and decreasing the bandwidth of the links on that path (see Section 3.1 for description of this framework). Once t_{n+1} has been routed, the base heuristic routes the next largest demand t_{n+2} in the same way using the partial topology existing after t_{n+1} has been routed. The base heuristic algorithm continues in this way until all demands have been routed (or assigned null routes).

3.2.2.2 Index Rollout Algorithm

The example in Figures 3.5, 3.6 and 3.7 shows that the order in which traffic demands are routed plays an important role in determining the throughput of the resulting topology. Index rollout seeks to optimize this order. The index rollout algorithm works as follows: In the first step, the rollout algorithm uses CSPF to route the demand t_1 determined by the requirement that it maximize the total network throughput when the base heuristic is used to complete the topology starting with t_1. The base heuristic orders the remaining demands in decreasing order of magnitude, and routes them sequentially. For each demand, it chooses a route using constrained shortest path first (CSPF). The route chosen for each demand is a shortest unidirectional path in the partial topology satisfying the interface and bandwidth constraints. If there is no feasible path, then the 'null' route is assigned-i.e., the demand is blocked. After routing each demand, the base heuristic temporarily updates the topology to eliminate the links which violate the interface constraints, and decrease the residual bandwidth of the links on the path on which this demand is routed.

Now, suppose that the demands $(t_1, .., t_{n-1})$ have been routed in this order by the rollout algorithm. In the next step, the rollout algorithm uses CSPF to route the remaining demand t_n determined by the requirement that it maximize the total network throughput when the base heuristic is used to complete the topology starting with $(t_1, .., t_n)$. In other words, routing t_n next minimizes the sum of the

remaining demands that are blocked. After routing each demand, the index rollout updates the topology to eliminate the links that violate the interface constraints, and decrease the residual bandwidth of the links on the path on which this demand is routed.

3.2.2.3 Route Rollout Algorithm

The example in Figure 3.8 shows that the choice of path for each traffic demand plays an important role in determining the throughput of the resulting topology. Route rollout seeks to optimize the selection of path for each demand when the demands are considered in a fixed order. We consider the demands in decreasing order of magnitude. (Additional algorithms may be obtained by using different criteria to order the traffic demands; see Section 3.2.2.4 below.) Let $(t_1, .., t_N)$ be the ordered sequence of demands. The base heuristic works as follows: Suppose that a partial topology has been obtained by choosing routes $(p_1, .., p_n)$ for the first n demands $(t_1, .., t_n)$. The base heuristic routes the remaining demands $(t_{n+1}, .., t_N)$ sequentially using CSPF. The route rollout algorithm works as follows: Fix an integer $K > 1$. In the first step, the rollout algorithm considers at most K feasible shortest paths as candidates for the route p_1 for the demand t_1. For each potential choice of p_1 it uses the base heuristic to complete the topology by routing the remaining traffic demands. The rollout algorithm then selects for p_1 the candidate that results in the maximum total network throughput. Now, suppose that the demands $(t_1, .., t_{n-1})$ have been given routes $(p_1, .., p_{n-1})$ by the rollout algorithm.

In the next step, the rollout algorithm considers at most K feasible shortest paths as candidates for the route p_n for the demand t_n. For each potential choice of p_n it uses the base heuristic to complete the topology by routing the remaining traffic demands. The rollout algorithm then selects for p_n the candidate that results in the maximum total network throughput, and finalizes the links on this path along with updating the topology (according to the framework of Section 3.1). Note that if there is only one feasible shortest path for a traffic demand, the routing decision made by the rollout algorithm coincides with the decision made by the base heuristic. It might appear desirable to consider all feasible shortest paths as candidates for p_n. However, this is not possible since the problem of finding all such paths requires exponential time. Consequently, we limit the number of paths considered to K, where the upper bound K is chosen small enough to allow reasonable computation time given the size of the network.

3.2.2.4 Sequential Rollout Algorithm

Thus far, we have considered rollout algorithms either for the sequence of traffic demands or for path selection. Another possibility is to apply rollout in order to optimize both the sequence of traffic demands and the route path selection. This can be achieved by applying rollout algorithms sequentially. It means that we first apply the index rollout algorithm in order to optimize the sequence of traffic demands as explained in 3.2.2.2. Then we apply the route rollout algorithm described in 3.2.2.3 in order to optimize the path selections for the sequence of traffic

demands determined by the index rollout. The difference between the sequential rollout and route rollout algorithm is that the sequential rollout uses the sequence of traffic demands determined by index rollout while route rollout sequences the traffic demands in order of decreasing magnitude.

3.2.2.5 Integrated Rollout Algorithm

Instead of first choosing the sequence of traffic demands and then choosing the paths for the traffic demands, an alternative is to make those decisions at the same time. We call this the integrated rollout algorithm. In integrated rollout, each component of a solution is a pair (t_k, p_k) consisting of a traffic demand and its path. Thus, the algorithm seeks to optimize the sequence $((t_1, p_1), .., (t_N, p_N))$. The base heuristic takes a partial solution $((t_1, p_1), .., (t_n, p_n))$ and extends it to a complete solution by choosing the remaining traffic demands $(t_{n+1}, .., t_N)$ in order of decreasing magnitude and choosing paths $(p_{n+1}, .., p_N)$ (some of which may be null) for these traffic demands sequentially using CSPF. The integrated rollout algorithm works as follows: In the first step it considers all pairs (t_1, p_1) where t_1 is any of the traffic demands and p_1 is any one of a maximum of K feasible shortest paths for t_1. It selects the pair (t_1, p_1) that gives the maximum total network throughput when the base heuristic is used to extend it to a full topology. Now, if the rollout algorithm has produced the sequence $((t_1, p_1), .., (t_{n-1}, p_{n-1}))$, it considers pairs (t_n, p_n) where t_n is a remaining demand and p_n is any one of a maximum of K feasible shortest paths for t_n. It selects the pair (t_n, p_n) that

maximizes the total network throughput when the base heuristic is used to extend $((t_1, p_1), .., (t_n, p_n))$ to a full solution. After each decision step, the topology is updated according to the framework of Section 3.1.

At the first step, the throughput for rollouts is at least as large as that for heuristic as we form the whole topology according to the heuristic. The method of choosing the routes makes sure that the rollout algorithms work at least as good as the heuristic, as at each decision step, they always have the choice of going according to the heuristic which gives the throughput which was calculated at the previous step. Thus, the rollouts perform at least as well as the heuristic in terms of the throughput (the objective function).

3.3 Computational Complexity

Let the number of nodes in the network be N and the number of aggregate demands in the traffic matrix be M. We use a modified version of Dijkstra's shortest path algorithm, [9] as a heuristic for finding the shortest paths. It is modified to take care of the interface and bandwidth constraints while finding a shortest path. As the topology at any intermediate state of the algorithms is not expected to be sparse, so the process of finding a shortest path takes $O(N^2)$ time. The heuristic we use for sorting is sorting by decreasing order of traffic demands, which takes $O(MlogM)$ time for sorting M aggregate flows. This time is insignificant compared to the time taken by other components of the algorithms, so it does not show up in the

time complexity of any of our algorithms. The time complexity of the heuristic algorithm is $O(MN^2)$, as shortest paths are computed M number of times. If the set of source/destination nodes is fixed, then so is the number of aggregate demands. In this case, the complexity becomes $O(N^2)$.

The time complexity of the route rollout algorithm is $O(M^2N^2)$, as K is fixed. This complexity is due to the fact that at each decision step, $O(M)$ shortest paths are computed, and there are M decision steps in the algorithm. In the case of fixed M, the complexity is $O(N^2)$. The time complexity for the index rollout algorithm is $O(M^3N^2)$. At each decision step in the algorithm, $O(M^2)$ shortest paths are computed, and there are M decision steps in the algorithm resulting in the above complexity. This also reduces to $O(N^2)$ for fixed M. The complexity for the integrated rollout is also the same as the time is scaled by K which is a constant. The time complexity for the sequential rollout is the sum of the complexity for the index and route rollout algorithms i.e., $O(M^2N^2 + M^3N^2)$ which is the same as $O(M^3N^2)$. As in the previous cases, this also reduces to $O(N^2)$ for fixed M.

In the case where each node in the network can be a source or a destination, M scales as N^2 and the complexity of the heuristic algorithm becomes $O(N^4)$, while the route rollout algorithm takes $O(N^6)$, and the other three rollout algorithms take $O(N^8)$ time.

3.4 Simulation Results and Analysis

The simulations were done with two types of network data. The first set of simulations was done with a fixed number of sources and destinations in the network. The second set of simulations was done assuming any node can be a source or a destination node.

3.4.1 Simulation Set 1

The network was assumed to have the following parameters:

- Number of nodes in the network = 50. This represents a reasonably sized backbone network.
- Nodes are uniformly distributed, with each node having an average of 7.5 potential neighbors.
- Number of receive interfaces at each node = 3.
- Number of transmit interfaces at each node = 3.
- The transmission range of all nodes is assumed to be the same.
- Capacity of each link = 100 in each direction.
- Number of nodes capable of being a source/destination = 12.
- Number of source-destination pairs = 125, selected from among the nodes which can be sources or destinations. In this case, nearly all possible source-destination pairs are a part of the traffic matrix.
- Aggregate traffic between each pair: Uniformly distributed between 1 and

40 units.

- Number of Shortest Paths considered in Route Rollout, Sequential Rollout and Integrated Rollout, $K = 4$.
- Weight of each link for constrained shortest path computation = 1, thus making the shortest path as the constrained min-hop path.

The simulation was run 10 times and in each simulation, the network topology was formed starting with these parameters. The throughput (bandwidth reservations) and number of rejects (the demands which we could not route) were noted. Figure 3.9 shows the throughput for 5 of the 10 simulations, and Figure 3.10 shows the number of rejects for those 5 simulations. The simulations shown in these figures have been selected to show the general trend and the variation encountered in the results. Note that rejects and throughput are not directly related to each other–i.e., it is possible (but unlikely) to simultaneously achieve higher throughput and higher rate of rejection since the size of the demands is not constant.

As can be seen from the figures, all the four rollout algorithms work much better than the heuristic. The integrated rollout normally works the best among these, followed by the sequential rollout, index rollout, and route rollout, in that order. There are some exceptions to the general trend, as can be seen from simulations 4 and 5. In simulation 4, the index and sequential rollouts work better than the integrated rollout and in simulation 5, the route rollout works better than the index and sequential rollouts. As all the policies are suboptimal, none of the rollout

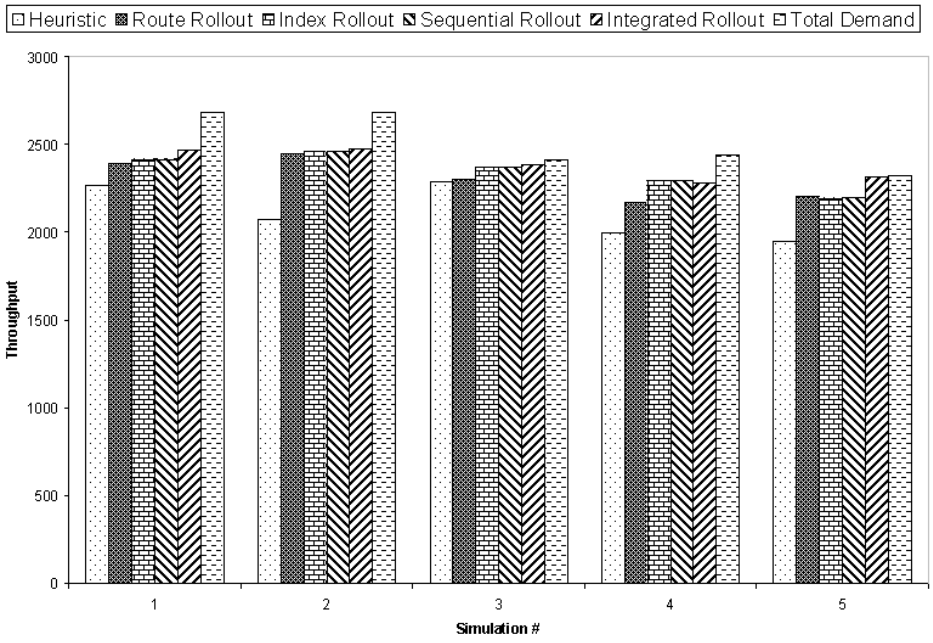

Figure 3.9: Throughput for Simulation Set 1

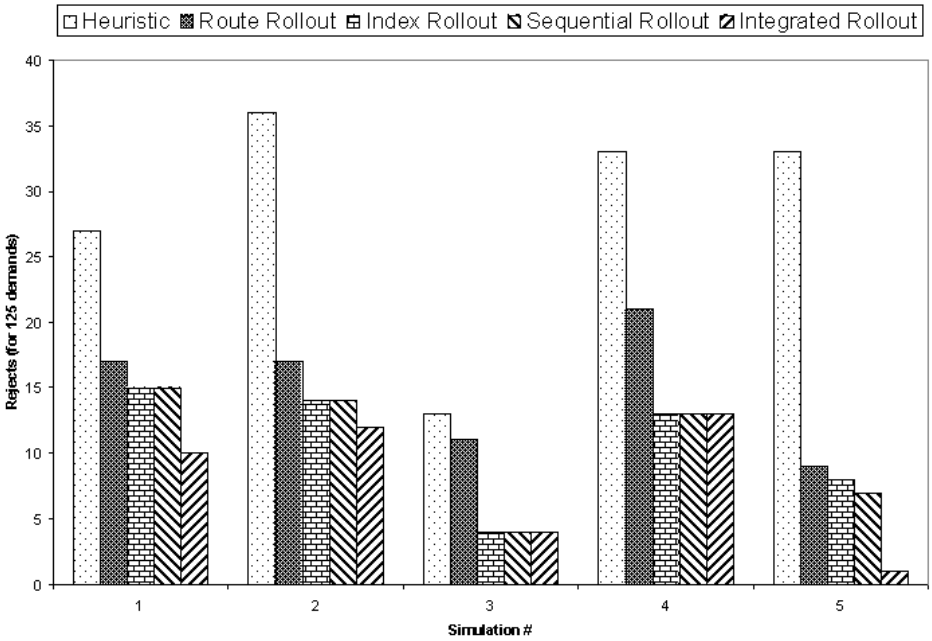

Figure 3.10: Rejects for Simulation Set 1

Table 3.1: Aggregate Results for Simulation Set 1

Policy	Throughput	Rejects
Heuristic	85.13%	20.72%
Route Rollout	92.18%	10.64%
Index Rollout	94.49%	7.12%
Sequential Rollout	94.51%	7.04%
Integrated Rollout	95.16%	6.24%

policies is guaranteed to perform better than the others as the decision at any stage of the algorithms is not optimal. This explains the results seen in simulations 4 and 5. Table 3.1 gives the average rejects over 125 aggregate demands (as a percentage of total demands) and the average throughput (as a percentage of total requested demand) over all simulations of this set.

As can be seen from Table 3.1, comparing with the heuristic in terms of throughput, the route rollout performs nearly 8.3% better, the index rollout performs 11% better, the sequential rollout performs 11% better and the integrated rollout performs 11.8% better. In terms of the number of rejects, the route rollout performs nearly 48.6% better, the index rollout performs 65.6% better, the sequential rollout performs 66% better and the integrated rollout performs 69.9% better than

the heuristic. So, generally the integrated rollout is expected to perform the best among these rollouts. Another observation from the results is that the index selection is more critical than the selection of routes from among multiple routes. This can be inferred from the fact that the index and sequential rollouts work much better than the route rollout while the integrated rollout does not work that much better than the index and sequential rollouts. This conclusion is further strengthened by the observation that index and sequential rollouts perform either the same or sequential rollout does slightly better than the index rollout; there is not a big margin between them, as can be seen from Table 3.1.

Regarding the connectivity of the network, the optimization of the throughput ensures with high probability that the source and destination nodes are all connected. If certain other nodes are not essential as transit nodes, it is possible that these nodes may be disconnected.

3.4.2 Simulation Set 2

This simulation set is for the case where all the nodes can be sources/destinations, and the network is more heavily loaded than in simulation set 1. The network was assumed to have the following parameters different from the simulation set 1:

- Number of nodes in the network = 20. This represents a reasonably sized backbone network.
- Nodes are uniformly distributed, with each node having an average of 6.5

potential neighbors.

- Any node can be a source or a destination.
- Number of source-destination pairs in the traffic matrix: between 135 and 170, selected uniformly from among all possible source-destination pairs (380 of them).
- Aggregate traffic between each pair: Uniformly distributed between 1 and 30 units.

Relative to the size of the network, the total demand is very large compared to the network in simulation set 1. The demand for simulation set 1 is around 2500 units for a network of size 50, while it is around 2000 units for a network of size 20 here.

The simulation was run 10 times and in each simulation, the network topology was formed starting with these parameters. Figure 3.11 shows the throughput for 5 of the 10 simulations, and Figure 3.11 shows the number of rejects for those 5 simulations. The simulations shown in these figures have been selected to show the general trend and the variation encountered in the results.

As can be seen from the figures, all four of the rollout algorithms work much better than the heuristic. The integrated rollout normally works the best among these, followed by the sequential and index rollouts, which work the same most of the time, followed by the route rollout. As in simulation set 1, there are instances when the index and sequential rollout work better than the integrated rollout.

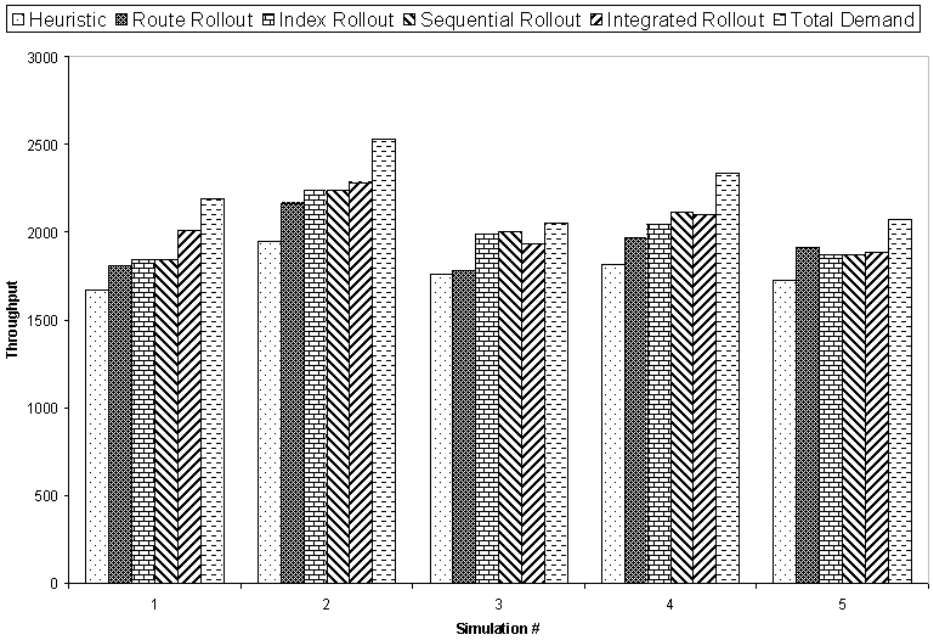

Figure 3.11: Throughput for Simulation Set 2

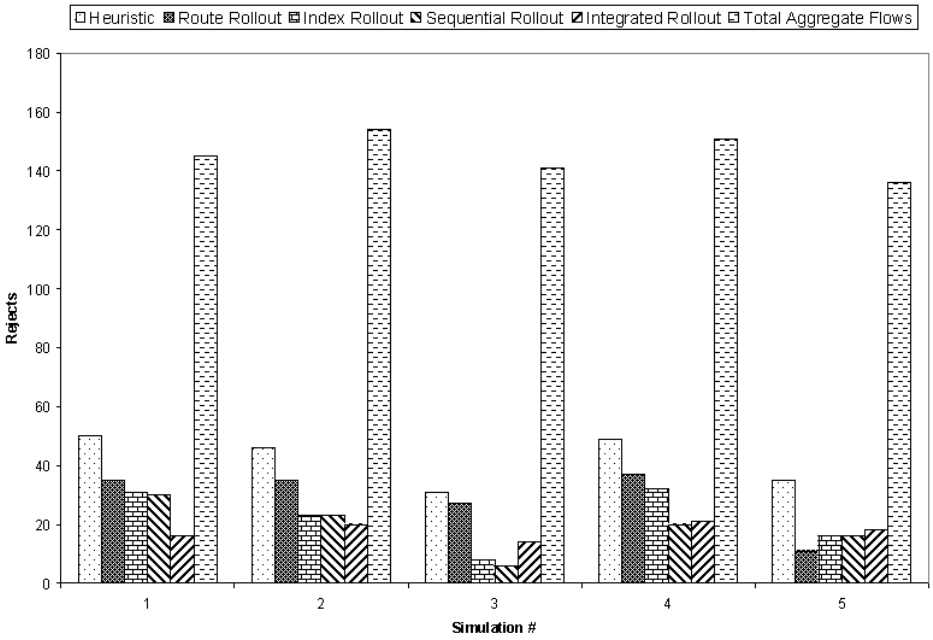

Figure 3.12: Rejects for Simulation Set 2

Table 3.2: Aggregate Results for Simulation Set 2

Policy	Throughput	Rejects
Heuristic	79.87%	30.56%
Route Rollout	86.59%	21.16%
Index Rollout	89.86%	14.44%
Sequential Rollout	90.25%	13.43%
Integrated Rollout	92.12%	10.61%

Table 3.2 gives the average rejects (as a percentage of total requested aggregate flows) and the average throughput (as a percentage of total requested demand) over all simulations of this set.

As can be seen from the table, comparing with the heuristic in terms of throughput, the route rollout performs nearly 8.4% better, the index rollout performs 12.5% better, the sequential rollout performs 13% better and the integrated rollout performs 15.3% better. In terms of the number of rejects, the route rollout performs nearly 30.8% better, the index rollout performs nearly 52.7% better, sequential rollout performs 56.1% better and the integrated rollout performs 65.3% better than the heuristic. In this case also, the network was connected for each simulation as the traffic matrix was comprehensive in terms of the nodes it covered.

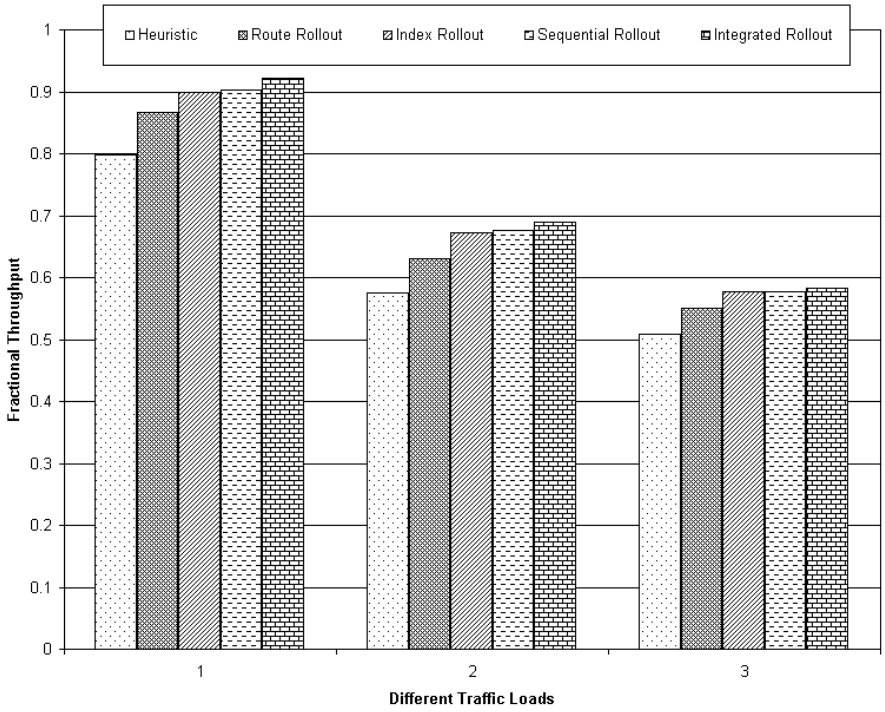

Figure 3.13: Throughput at different traffic loads

3.4.2.1 Comparison under Different Traffic Load Conditions

As has been proven, the rollout algorithms are guaranteed to work better than the heuristic. Changing the network parameters effectively changes the amount of load on the links. So, the algorithms were compared on the network of simulation set 2 under different traffic load conditions:

1. Traffic conditions of section 3.4.2.
2. 160 profile entries, each uniformly distributed between 1 and 40 units of traffic.
3. 160 profile entries, each uniformly distributed between 1 and 50 units of traffic.

Figure 3.13 shows the throughput for the algorithms under these conditions. As can be seen, the relative performance of the algorithms is similar under different traffic loads.

3.5 Online Routing and Admission Control

The topology is set-up, and the bandwidth reservation information and the route for each ingress-egress pair is given to the ingress node for that pair. Whenever a call (new request of traffic between an ingress-egress pair) arrives, the ingress router checks to see if there is enough bandwidth left from the bandwidth reserved for this pair. If there is bandwidth left, then the flow is routed through the path stored from the offline phase. We may have additional unreserved bandwidth on some links in the network (the bandwidth left unreserved), so in the case of reserved bandwidth being exhausted for an ingress-egress pair, the unreserved bandwidth is used (on a first-come-first-serve basis). If the call cannot be routed using the reserved bandwidth or the extra unreserved bandwidth, it is blocked.

3.5.1 Simulation Results and Analysis

We model the traffic as a collection of individual flows with Poisson arrival times with rate λ_i, exponential holding times (with mean T_i) and constant bit rate traffic (R_i) for each flow. The mean of the aggregate traffic demand for each ingress-egress pair (i) can be computed as $\lambda_i T_i R_i$. We generate the traffic profile using these mean

aggregate demands for each pair. The network was assumed to have the following parameters:

- Number of nodes in the network = 50.
- Nodes are uniformly distributed, with each node having an average of 7.5 potential neighbors.
- Number of receive interfaces at each node = 3.
- Number of transmit interfaces at each node = 3.
- The transmission range of all nodes is assumed to be the same.
- Capacity of each link = 100 in each direction.
- Number of nodes capable of being a source/destination = 12.
- Number of source-destination pairs = 100, selected from among the nodes which can be sources or destinations.
- Poisson Rate (λ_i): Uniformly distributed between 10 and 20 per unit time.
- Mean of Holding Time (T_i): Uniformly distributed between 1 and 2 units of time.
- Bit Rate of individual calls = 1 unit (same for all).
- Number of Shortest Paths considered in Route Rollout, Sequential Rollout and Integrated Rollout, $K = 4$.
- Weight of each link for constrained shortest path computation = 1, thus making the shortest path as the constrained min-hop path.

Table 3.3: Average Bandwidth Guarantees

Heuristic	Route Rollout	Index Rollout	Integrated Rollout
0.8782	0.9326	0.9582	0.9597

The simulation was run 10 times and in each simulation, the network topology was formed starting with these parameters. Table 3.1 shows the average fractional throughput (bandwidth guarantees/total demand) for the heuristic and the rollout algorithms.

As can be seen from Table 3.3, comparing with the heuristic in terms of throughput, the route rollout performs nearly 6.2% better, the index rollout performs 9.1% better and the integrated rollout performs 9.3% better. So, generally the integrated rollout is expected to perform the best among these rollouts. Another observation from the results is that the index selection is more critical than the selection of routes from among multiple routes. This can be inferred from the fact that the index rollout works much better than the route rollout while the integrated rollout does not work that much better than the index rollout.

The network was setup and Poisson traffic with exponential holding times and CBR rate (the parameters being the same as provided to offline phase) was generated and the network was run for 30 units of time for each of the 10 simulations. In each simulation, the traffic for evaluating the heuristic was the same as that for the rollout. Table 3.4 gives the average throughput (which is the same as call ac-

Table 3.4: Average Throughput

Heuristic	Route Rollout	Index Rollout	Integrated Rollout
0.7729	0.8232	0.8520	0.8542

ceptance rate as traffic is CBR with same rate for all pairs) for each of the policies. As can be seen, the relative performance is similar to the bandwidth guarantees we could achieve in the offline phase.

Chapter 4

Extension of Rollout Algorithms

In the framework we proposed in Chapter 3, the whole aggregate demand between an ingress-egress pair is routed through a single path accommodating the whole demand, or it is not routed at all. If we allow the splitting of a demand over multiple paths and try to route as much as possible (rather than routing whole or zero), we expect to get a better throughput. In this chapter, we give a linear formulation of the problem of routing the traffic profile over a fixed topology, and describe how we can use that along with the rollout algorithms. We provide some simulation results at the end of this chapter.

4.1 Routing as Multi-commodity Flow Problem

We set up the problem of routing a given traffic profile over a computed topology for maximizing the throughput as a linear multi-commodity flow problem [3]. We treat each aggregate demand (profile entry) as a commodity which we can split among multiple paths. Let there be M commodities (the value of each commodity

is $profile(i)$), N nodes and L links in the network. We add a dummy link (infinite cost, infinite capacity) between the source and destination of each commodity to achieve feasibility (thus, there are M such links). Let $x_i(l)$ be the amount of commodity i routed through link l. Let $cost(l)$ represent the cost of each link, which is 1 for an actual link for our objective of maximizing the throughput. Let the set of incoming and outgoing links at node j be denoted by in_j and out_j respectively. Let $source_i$ and $dest_i$ represent the source and destination of profile i. Equation 4.1 achieves the objective of maximizing the throughput as the algorithm tries to route on the actual links due to large cost of the dummy links. Along with maximizing the throughput, the objective function also minimizes the weighted hop count (the number of links used for each path) for the value of throughput it achieves. Equation 4.2 represents the bandwidth constraints. Equations 4.3 and 4.4 represent the flow conservation laws at transit nodes and source nodes for each commodity respectively. The traffic that goes over the dummy links is the traffic that is not routed in the actual network.

$$minimize \quad \sum_{l=1}^{L+M} (cost(l) \sum_{i=1}^{M} x_i(l)) \tag{4.1}$$

$$\sum_{i=1}^{M} x_i(l) \leq capacity(l) \;\; \forall l \in \{1, .., L\} \tag{4.2}$$

$$\sum_{l \in in_j} x_i(l) \quad = \sum_{l \in out_j} x_i(l) \; \forall j \in \{1, .., N\} - \{source_i, dest_i\}, \forall i \in \{1, .., M\} \quad (4.3)$$

$$\sum_{l \in out_j} x_i(l) \quad - \sum_{l \in in_j} x_i(l) \quad = \quad profile(i), \; j \quad = \quad source_i, \forall i \quad \in \quad \{1, .., M\} \quad (4.4)$$

4.2 Extended Rollout Algorithms

We can extend the heuristic and all the rollout algorithms to include the multi-commodity flow formulation as described. We calculate the topology and bandwidth reservations as we did before, but we no longer use the bandwidth reservations calculated. Instead, we fix the topology we get at the end of the rollout (or heuristic) algorithms and solve the multi-commodity flow problem for the traffic profile over that topology. This gives the routes and bandwidth reservations. This is guaranteed to give a throughput at least as high as the throughput we get at the end of the rollout algorithms (i.e., without splitting the traffic).

4.3 Simulation Results and Discussion

The network model is kept the same as in Section 3.4, but the traffic demands have been increased to allow for potential improvement in throughput.

Table 4.1: Average Throughput for Rollout Algorithms

Heuristic	Route Rollout	Index Rollout	Integrated Rollout
0.7867	0.8457	0.8690	0.8932

Table 4.2: Average Throughput for Rollout Algorithms with Traffic Splitting

Heuristic	Route Rollout	Index Rollout	Integrated Rollout
0.8935	0.9222	0.9247	0.9400

4.3.1 Simulation Set 1

We have the same network model as the 50 node network of Section 3.4.1. The traffic is now uniformly distributed between 10 and 40 units, with 10 nodes capable of being a source or destination and there are 90 source-destination pairs among them.

Tables 4.1 and 4.2 show the average fractional throughput for the rollout algorithms and their heuristic for the case where we do not split the traffic and the case where we split the traffic as a commodity respectively. As can be seen, splitting the traffic increases the throughput considerably: 13.6% for the heuristic, 9% for the route rollout, 6.4% for the index rollout and 5.2% for the integrated rollout. We do not show the results for sequential rollout as it does not improve much on index rollout.

Table 4.3: Average Throughput for Rollout Algorithms

Heuristic	Route Rollout	Index Rollout	Integrated Rollout
0.5748	0.6313	0.6732	0.6892

Table 4.4: Average Throughput for Rollout Algorithms with Traffic Splitting

Heuristic	Route Rollout	Index Rollout	Integrated Rollout
0.7090	0.7223	0.7335	0.7550

4.3.2 Simulation Set 2

For the 20 node network of Section 3.4.2, the traffic is now uniformly distributed between 1 and 40 units, with 160 source-destination pairs.

Tables 4.3 and 4.4 show the average fractional throughput for the rollout algorithms and their heuristic for the case where we do not split the traffic and the case where we split the traffic as a commodity respectively. As can be seen, splitting the traffic increases the throughput considerably: 23.3% for the heuristic, 14.4% for the route rollout, 9% for the index rollout and 9.5% for the integrated rollout. We do not show the results for sequential rollout as it does not improve much on index rollout.

There are two primary reasons for the improvement in throughput:

1. The multi-commodity flow algorithm finds the routes and bandwidth reser-

vations to maximize the throughput over the same topology that is established after running the rollout algorithms. So, it is guaranteed to work at least as well as the corresponding rollout algorithm.

2. The rollout algorithms (and heuristic) either routes the whole aggregate demand of an ingress-egress pair or do not route it at all, which is not the case when we solve the multi-commodity flow problem and allow the traffic to split.

4.4 Comparison with Single Path Rollout Algorithms

There are advantages and disadvantages of splitting the traffic. If we do not allow splitting the traffic, and use a single path per ingress-egress pair, then the ingress router has to store less routing information per pair (instead, it is for a single path per ingress-egress pair), and the bandwidth reservation information does not have to be stored for each link for each ingress-egress pair, as is the case in which we split the traffic between different paths. Also, the decision of selecting a route is much easier the case of single path per ingress-egress pair with same bandwidth reservation on all the links of that path.

Splitting the traffic has the advantage that it results in higher throughput in terms of the total bandwidth reserved. Also, the network is not tolerant to link/node failure in the case of having reservations on only one path per ingress-egress pair. So, having bandwidth reservations on multiple paths makes the net-

work more tolerant to link/node failures.

Chapter 5

Heuristics using Matching Theory

We propose some heuristics for topology control and routing which use matching theory, [10] and the formulation of routing as a multi-commodity flow problem. We provide an extension to the algorithms to incorporate fairness in the topology control and routing decisions, where fairness is measured in terms of the fraction of traffic demand reserved for each traffic profile entry. We then present the simulation results to compare these heuristics with the rollout algorithms.

5.1 Matching Theory

Given an undirected graph $G = (V, E)$, a matching is a subgraph $G' = (V, E')$ such that $degree(v) <= 1 \, \forall \, v \in V$ and $E' \subset E$. The vertices having a degree of 1 in G' are called to be matched. We describe different types of matchings based on the constraints they satisfy.

- *Maximum Cardinality Matching*: Given an undirected graph $G = (V, E)$, a maximum cardinality matching $G' = (V, E')$ is one in which the number of

matched vertices is maximum among all possible matchings (i.e., the number of edges is maximum).

- *Maximum Weight Matching*: Given a graph $G = (V, E)$, with edge weights $w(e) \forall e \in E$, a maximum weight matching is one which gives a matching (G') with sum of weights of all edges in G' the maximum among all possible matchings.
- *Perfect Matching*: Given an undirected graph $G = (V, E)$, a perfect matching $G' = (V, E')$ is a matching satisfying the condition $degree(v) = 1 \forall v \in V$ i.e., all vertices in G' are matched.

5.2 Application of Matching Theory to Topology Control

The basic algorithm is outlined below and explained in the following sections.

1. Weight the links in the initial graph (potential topology) to favor the links which are expected to have a higher traffic flow.
2. Map the graph to one which can be given as an input to a maximum weight matching problem and solve the matching problem to get a maximum weight subgraph which satisfies the interface constraints.
3. Solve the multi-commodity flow problem over this topology.
4. Modify the topology sequentially, solving the multi-commodity flow problem each time, and finally, keep the topology which gives the maximum throughput.

5.2.1 Giving Initial Weight to Edges

We propose three strategies of weighting the edges:

1. *Uniform Weighted Matching (UWM)*: The way we map the problem to maximum weight matching problem, if we give the same weight to all the links in the network, the output topology will have the maximum number of links while satisfying the degree constraints (as we try to maximize the weight during matching algorithm, so same weight to all edges would result in maximizing the number of edges).
2. *Traffic Weighted Matching (TWM)*: As our objective is maximizing the throughput, so it is better to give extra weight to edges which are expected to carry more traffic. We explain this strategy below:
   - Give a weight of 1 to all edges.
   - Find K shortest paths (maximum) of same length for each traffic profile (over the potential topology).
   - Each time a link comes in a path, add to its weight $profile(i)/numberSP$, where $profile(i)$ is the value of the profile entry, and $numberSP$ the number of shortest paths we found for that profile entry.
3. *Flow Weighted Matching (FWM)*: Another possibility is to give weights according to the number of paths a link comes on irrespective of the amount of traffic. In this case, we add 1 to the weight of a link in step 3 above.

We work with shortest paths as the multi-commodity flow formulation will prefer shorter paths for a commodity because it minimizes the cost function of Equation 4.1 (to maximize the throughput).

5.2.2 Mapping to Maximum Weight Matching

Given a graph $G = (V, E)$ (which corresponds to the potential network) with edge weights as explained in section 5.2.1, we form a graph $G' = (V', E')$ and give it as an input to the maximum weight matching problem. Let the number of transmit and receive interfaces (i.e., input and output degree constraints) at each vertex be Δ. Figure 5.1 shows an example potential topology with $\Delta = 2$. The steps of the mapping are explained below.

- For each vertex, we form 2Δ vertices in graph G' (Δ of them correspond to the transmit interfaces, and Δ to receive interfaces).
- For each edge between two vertices in G, we form two vertices in G' (we call these edge vertices), and add an edge between them (of weight 0), as well as between one of them and the transmit interface vertices (all Δ of them) of the vertex it was going out of (with weight equal to the weight of the corresponding edge in G). Also add edges between the other vertex (in G') of this edge (from G) and the receive interface vertices of the vertex it was incident on (with weight equal to the weight of the corresponding edge in G). The resulting graph for this example is shown in Figure 5.2 (the edges

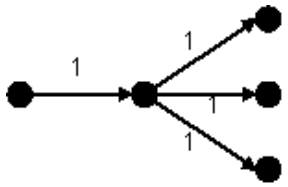

Figure 5.1: Potential Topology, $\Delta = 2$

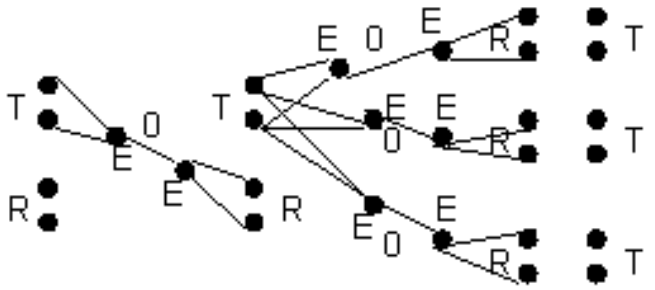

Figure 5.2: Modified graph from potential topology

shown without weights have a weight of 1 here). T, R and E refer to the vertices corresponding to transmit interfaces, receive interfaces and edges in G respectively.

- Add a clique with $2\Delta N$ vertices to the graph G', with each edge in the clique having weight 0. Add zero weight edges between each of these vertices and each of the vertices in G' which correspond to the transmit and receive interfaces of the vertices in graph G.
- Sum the weight of all the edges in G' and add a weight more than that to the weight of all edges in G'. We do this to make sure we get perfect matching using a maximum weight matching algorithm (we explain this in detail in Section 5.2.2.1).

Figure 5.3: Vertices are connected

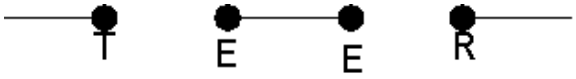

Figure 5.4: Vertices are not connected

We solve the maximum weight matching problem on G', and deduce the resulting topology based on the result we get. We show the rules for mapping from the output of matching to the graph representing the topology: Figure 5.3 shows the scenario (a subgraph from the output of matching algorithm) when two vertices will have an edge between them in the final topology. Figure 5.4 shows the case when two vertices which had an edge between them in G will not have the edge in the resulting topology (as they are not connected to the edge vertices corresponding to the edge between them). We detect such subgraphs in the output graph from matching algorithm (the output graph will be composed of such subgraphs only), and use these rules to make the resulting topology.

We added a clique to the graph G' to avoid the scenario in which we cannot deduce the topology from the result of the matching algorithm. Figure 5.5 shows

Figure 5.5: Connection between vertices undecidable

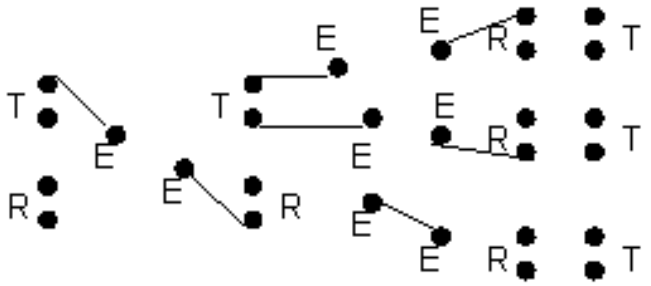

Figure 5.6: Result of Matching Algorithm

the case when this has happened between two vertices. In this case, one of the vertices connects to the edge vertex in G', while the other vertex does not connect with the corresponding edge vertex. This can be avoided by having perfect matching (matching in which all vertices have a match), and for achieving that we add a clique of size $2\Delta N$ and connect each of them to all the vertices in G' which correspond to transmit and receive interfaces of vertices in G. A perfect matching would result in the output to be as in Figure 5.4 by connecting the unmatched vertices (corresponding to transmit and receive interfaces in G) to the dummy edges, and connecting the edge vertices together. By adding $2\Delta N$ dummy vertices, we accommodate the worst case in which all the vertices will be disconnected in the final topology.

Figure 5.6 shows the output (minus the clique vertices and corresponding edges) of the matching algorithm and Figure 5.7 shows the final topology we deduce from the output of matching (shown in Figure 5.6) for the example network of Figure 5.1.

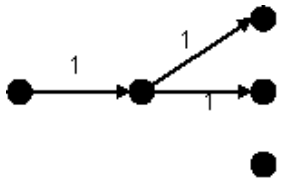

Figure 5.7: Final Topology

5.2.2.1 Perfect Matching using Maximum Weight Matching Algorithm

Given a graph $G = (V, E)$, for which a perfect matching $G' = (V, E')$ exists, we want to get a perfect matching using a maximum weight matching algorithm. The way we form the input to the matching algorithm, a perfect matching always exists and its weight is the maximum among all possible matchings (as the extra edges we add have a weight 0). We want to make sure that among the matchings of maximum weight, the output is a perfect matching. We do that by summing the weights of all the edges, and adding that to the weight of all edges. The reason for doing this is the following:

Let the maximum weight matching have the weight w, and there be a couple of unmatched vertices. So, in the worst case, matching those vertices and changing the matching of other vertices so as to get a perfect matching can lead to the reduction of the net weight by w. If we make sure the weight of this new edge is more than w, then the weight of this new matching will be more than the matching we got before. This new matching will have more edges than the previous matching as it is a perfect matching. So, we sum the weights of all edges in the graph, and add a weight more than that sum to the weights of all the edges. This makes sure

that the situation of Figure 5.5 does not arise, as it would rather match the two vertices corresponding to edges with each other and the dummy vertices to the vertices corresponding to the transmitter and receiver to get higher weight and a perfect matching. Thus, we get a maximum weight perfect matching, and the weight of the output topology (concluded from the output of matching algorithm) would still be the same as we added the same weight to all edges.

5.2.3 Topology Change Strategy

We solve the multi-commodity flow problem (as explained in Section 4.1) on the topology we get from the algorithm explained in section 5.2.2. We sequentially change this topology and solve the multi-commodity flow problem on the resulting topologies to get an improvement in the throughput. The algorithm for changing the topology is as explained below:

1. Make a list of the profile entries for which we could route less than $x\%$ of the demand (where x is chosen by the user depending on the current throughput levels, network size and processing power of the computing unit - we keep it fixed for our simulations), in decreasing order of demands.
2. For the first entry in this list, find (maximum) K shortest paths each in the potential topology and the current topology. Form the first path which is present in the potential topology, but not in the existing topology by deleting the least loaded links (with traffic as given by the result of multi-

commodity flow problem) at each of the interface-starved nodes in the current topology on this path. If all the paths are the same then repeat this step for the next entry in the list.

3. Solve the multi-commodity flow problem for this changed topology. If the throughput is more than the throughput in the current topology then change the current topology to this topology and update the list by deleting the entries for which we have routed more than $x\%$ on this topology. If the throughput is less than before, then let the current topology remain the same and start with step 2 for the next entry in the list.

We keep the topology we have at the end of this procedure.

5.3 Incorporating Fairness

In all the algorithms we have seen so far, we do not consider fairness, i.e., trying to route at least a certain fraction of each profile entry while maximizing the throughput. We address this problem by making changes to the multi-commodity flow formulation. We use two different weighting strategies for comparison:

- *Fairness1*: Use *Traffic Weighted Matching*.
- *Fairness2*: Use *Flow Weighted Matching*. This strategy is expected to be more fair than *Fairness1* as it weights the links according to the number of flows expected to pass through it rather than to the amount of traffic.

5.3.1 Extended Multi-commodity Flow Formulation

The primary change from the formulation explained before is the addition of the fairness constraint of Equation 5.1. Here, we sum the outgoing and incoming flows only over the actual links (and not over the (infinite capacity, infinite cost) links we added between the source and destination, as the traffic flowing over them is what we could not route). We also change the flow conservation law at intermediate nodes (as in Equation 4.3) to include only the actual links adjacent to those nodes (Equation 5.2). Here, ain_j denotes the actual incoming links at node j, and $aout_j$ denotes the actual outgoing links at j. This is done to make sure the traffic for a profile entry is not routed through the extra links between some nodes other than the source and destination of this profile entry (otherwise that will happen as the algorithm tries to force the condition of Equation 5.1).

$$\sum_{l \in aout_j} x_i(l) - \sum_{l \in ain_j} x_i(l) \ \geq \ y * profile(i),\ j \ = \ source_i, \forall i \ \in \ \{1, .., M\} \quad (5.1)$$

$$\sum_{l \in ain_j} x_i(l) \ = \sum_{l \in aout_j} x_i(l), \forall j \in \{1, .., N\} - \{source_i, dest_i\}, \forall i \in \{1, .., M\} \quad (5.2)$$

This formulation will give an infeasible result if it is not able to route at least $y\%$ of each profile entry. We follow the following procedure to get the reservations for the topology we get from the matching algorithm:

- Start at $y = 0.95$ and solve the changed multi-commodity flow problem.
- If it returns a feasible result, keep the current topology and these reservations

and exit. Else, decrease y by 0.05, and solve the extended multi-commodity flow problem again.

If we get the answer at $y = 0$, then the answer is the same as what we get in the algorithm explained in Section 5.2 (without sequential topology change).

5.4 Computational Complexity

Let the number of nodes in the network be N, and the number of flows be M. The computational complexity of the proposed algorithms is explained by breaking them into the following steps:

1. Weighting: It calculates shortest paths (takes $O(N^2)$ time for each path calculation) for $O(M)$ flows. So, this takes $O(MN^2)$.
2. Maximum Weight Matching: There are N vertices, $O(N^2)$ edges in the original graph G. In the graph G' (input to matching), we create $O(N+N^2)$ vertices from the vertices and edges of graph G, and add $O(N)$ vertices which form a clique. So, number of vertices in G' is $N' = O(N^2)$. In G', there are $O(N^2)$ edges between the vertices corresponding to edges and vertices in G. There are $O(N^2)$ edges in the clique which we add. There are edges between all vertices of the clique ($O(N)$ vertices) and all $O(N)$ vertices which correspond to vertices in G. So, number of edges in G', $E' = O(N^2)$. Maximum weight matching takes $O(E'N'logN')$ time and thus matching takes $O(N^4logN)$ time.

3. Multi-commodity flow algorithm is a linear program, so it takes $O(n^{0.5})$, where n is the number of variables, which is $O(MN^2)$ (one variable for each commodity and each link) in our case. So, it takes $O(M^{0.5}N)$ time.
4. In sequential topology change, we find shortest paths for all profile entries (worst case) and run the multi-commodity flow algorithm those many times, so it takes $O(MN^2 + M^{1.5}N)$ time for this step.

The maximum order of M is $O(N^2)$, and so the maximum weight matching step determines the order of the proposed algorithms. Thus, the algorithms take $O(N^4 logN)$ time, with N being the number of nodes in the network. This complexity is much lower than that for route rollout ($O(N^6)$) and index and integrated rollout algorithms ($O(N^8)$). The heuristic used by these rollout algorithms takes $O(N^4)$ time.

5.5 Simulation Results and Discussion

5.5.1 Simulation Set 1

The network used for simulations was assumed to have the following parameters:

- Number of nodes in the network = 20.
- Nodes are uniformly distributed, with each node having an average of 6.5 potential neighbors.
- Number of receive interfaces at each node = 3.

- Number of transmit interfaces at each node = 3.
- The transmission range of all nodes is assumed to be the same.
- Capacity of each link = 100 in each direction.
- Number of source-destination pairs = 160, chosen randomly.
- Aggregate traffic between each pair: Uniformly distributed between 1 and 40 units.
- Threshold, x, for including a profile in the sequential topology change list = 0.2.
- Number of shortest paths considered in Route Rollout and Integrated Rollout = 4.
- Number of shortest paths considered for initial weighting and sequential topology change, K = 3.
- Weight of each link for constrained shortest path computation = 1, thus making the shortest path as the constrained min-hop path.

The simulation was run on a different random network and random profile 10 times and in each simulation, the network topology was formed starting with these parameters. The matching algorithm used was an implementation of H. Gabow's N-cubed weighted matching algorithm [24].

Table 5.1 shows the average throughput for the extended rollout algorithms (same as in Table 4.4). Table 5.2 shows the average throughput for the matching algorithms. As we can see, the algorithm with *Traffic Weighted Matching (TWM)*

Table 5.1: Average Throughput for the Extended Rollout Algorithms

Heuristic	Route Rollout	Index Rollout	Integrated Rollout
0.7090	0.7223	0.7335	0.7550

Table 5.2: Average Throughput for Matching Heuristics

UWM	FWM	TWM	Fairness1	Fairness2
0.7178	0.7606	0.7720	0.7080	0.6719

works the best, followed by *Flow Weighted Matching (FWM)*, *Uniform Weighted Matching (UWM)* and the fairness schemes. *Traffic Weighted Matching* works 8.88% better than the heuristic for rollout and 2.25% better than the integrated rollout. Thus, the best matching algorithm works better than the extended rollout algorithms, and has a lower time complexity.

Table 5.3 shows the average throughput we get in the proposed algorithms without using the sequential topology change strategy. As can be seen by comparing Tables 5.2 and 5.3, the improvement in throughput is the maximum in *Uniform Weighted Matching* followed by *Flow Weighted Matching* and *Traffic Weighted Matching*. This, along with the results of Table 5.2, shows that *Traffic Weighted Matching* works the best among these three strategies.

Table 5.3: Average Throughput for Matching Heuristics without Sequential Topology Change

UWM	FWM	TWM
0.6650	0.7294	0.7535

Table 5.4: Average Minimum Routed for Fairness schemes

Fairness1	Fairness2
0.425	0.53

The metric we use to evaluate fairness is the minimum of the fraction of the demand routed for each profile entry. Table 5.4 shows the average of this parameter over the simulations. *Fairness2* is more fair (0.53) than *Fairness1* (0.425) as expected, though at the cost of throughput, as can be seen from Table 5.2. This metric turns out to be zero for all other algorithms proposed here.

5.5.2 Simulation Set 2

The network used for simulations was assumed to have the following parameters:

- Number of nodes in the network = 50.
- Nodes are uniformly distributed, with each node having an average of 7.5 potential neighbors.

- Number of receive interfaces at each node = 3.
- Number of transmit interfaces at each node = 3.
- The transmission range of all nodes is assumed to be the same.
- Capacity of each link = 100 in each direction.
- Number of nodes capable of being a source/destination = 12.
- Number of source-destination pairs = 90.
- Aggregate traffic between each pair: Uniformly distributed between 10 and 40 units.
- Threshold, x, for including a profile in the sequential topology change list = 20%.
- Number of shortest paths considered in Route Rollout and Integrated Rollout = 4.
- Number of shortest paths considered for initial weighting and sequential topology change, K = 3.
- Weight of each link for constrained shortest path computation = 1, thus making the shortest path as the constrained min-hop path.

The simulation was run on a different random network and random profile 10 times and in each simulation, the network topology was formed starting with these parameters. The matching algorithm used was an implementation of H. Gabow's N-cubed weighted matching algorithm [24].

Table 5.5: Average Throughput for Matching Heuristics

UWM	FWM	TWM
0.9045	0.8749	0.8797

Table 5.5 shows the average throughput for the matching algorithms. As we can see, the algorithm with *Uniform Weighted Matching (UWM)* works the best, followed by *Traffic Weighted Matching (TWM)* and *Fair Weighted Matching (FWM)*. Table 5.7 shows the average throughput for the extended rollout algorithms (same as in Table 4.2). As can be seen, all the extended rollout algorithms (along with the extended heuristic) work better than the matching-based algorithms in this kind of network. Thus, for a network with less number of ingress-egress pairs compared to the size of the network, extended rollout algorithms work better than the heuristics based on matching theory.

Table 5.6 shows the average throughput we get in the proposed algorithms without using the sequential topology change strategy. As can be seen by comparing Tables 5.5 and 5.6, the improvement in throughput is the maximum in *Uniform Weighted Matching* followed by *Flow Weighted Matching* and *Traffic Weighted Matching*.

Table 5.6: Average Throughput for Matching Heuristics without Sequential Topology Change

UWM	FWM	TWM
0.8436	0.8698	0.8742

Table 5.7: Average Throughput for the Extended Rollout Algorithms

Heuristic	Route Rollout	Index Rollout	Integrated Rollout
0.8935	0.9222	0.9247	0.9400

Chapter 6

Conclusion

The problem of profile based topology control and routing of bandwidth-guaranteed flows for maximizing throughput is addressed. The problem is proved to be NP-Hard. A framework which integrates topology control and routing decision is proposed, and efficient rollout algorithms are proposed to achieve good results. The routing problem is formulated as a multi-commodity flow problem, and the rollout algorithms extended to achieve better throughput.

This work also proposes a new set of heuristics which use matching theory to solve the problem. The matching theory heuristics are shown to work better than the extended rollout algorithms in a network in which all nodes can be ingress/egress, and worse in the case where only a few are ingress/egress nodes. The matching theory algorithm has a much lower time complexity than the rollout algorithms. The matching theory based heuristics and the multi-commodity flow algorithm are extended to achieve fairness among the different ingress-egress pairs.

BIBLIOGRAPHY

[1] N. A. Riza, "Reconfigurable Optical Wireless", *LEOS '99* IEEE , vol.1 , pp.70-71, 8-11 Nov. 1999.

[2] Z. Yaqoob, N. A. Riza, "Smart Free-Space Optical Interconnects and Communication Links using Agile WDM Transmitters", *2001 Digest of the LEOS Summer Topical Meetings* , 30 July-1 Aug. 2001.

[3] S. Suri, M. Waldvogel, D. Bauer, P. R. Warkhede, "Profile-Based Routing and Traffic Engineering", *Computer Communications*, vol. 24(4), pp. 351-365, March 2003.

[4] A. Desai, "Dynamic Topology Control of Free Space Optical Networks", M.S. Thesis, Department of Electrical Engineering, University of Maryland, 2003.

[5] P. C. Gurumohan, J. Hui, "Topology Design for Free Space Optical Networks", *Proc. ICCCN 2003.*

[6] Abhishek Kashyap, Kwangil Lee, Mark Shayman, "Rollout Algorithms for Integrated Topology Control and Routing in Wireless Optical Backbone Net-

works", Technical Report, Institute for Systems Research, University of Maryland, 2003.

[7] M. Garey and D. Johnson, "Computers and Intractability: A Guide to the theory of NP-Completeness", Freeman and Company, 1979.

[8] D. P. Bertsekas, "Dynamic Programming and Optimal Control", vol. 1, Athena Scientific, 2000.

[9] Edsger W. Dijkstra, "A note on two problems in connection with graphs", *Numerische Mathematik*, vol. 1, pp. 269-271, 1959.

[10] L. Lovász and M. D. Plummer, "Matching Theory", North-Holland, 1986.

[11] R. Guerin, H. Ahmadi, M. Naghshineh, "Equivalent Bandwidth and its application to bandwidth allocation in high-speed networks", *IEEE Journal on Selected Areas in Communications*, vol. 9(7), pp. 968-981, September 1991.

[12] K. Xu, X. Hong, Mario Gerla, "An Ad hoc Network with Mobile Backbones", *Proc. IEEE ICC 2002*.

[13] R. Ramanathan and R. Rosales-Hain, "Topology Control of Multihop Wireless Networks using Transmit Power Adjustment", *IEEE Infocom 2000*, vol. 2, pp. 404-413, 26-30 March 2000.

[14] Z. Huang, C-C. Shen, C. Srisathapornphat and C. Jaikaeo, "Topology Control for Ad Hoc Networks with Directional Antennas", *ICCCN 2002*, pp. 16-21, Miami, Florida, October 2002.

[15] D. Banerjee, B. Mukherjee, "Wavelength-routed Optical Networks: Linear Formulation, Resource Budgeting Tradeoffs and a Reconfiguration Study", *IEEE/ACM Trans. Networking*, vol. 8, pp. 598-607, Oct. 2000.

[16] K. H. Liu, C. Liu, J. L. Pastor, A. Roy, J. Y. Wei, "Performance and Testbed Study of Topology Reconfiguration in IP over WDM", *IEEE Transactions on Communications*, in press, 2002.

[17] E. Leonardi, M. Mellia, M. A. Marsan, "Algorithms for the Logical Topology Design in WDM All-Optical Networks", *Optical Networks Magazine*, pp. 35-46, Jan. 2000.

[18] R. Ramaswami, K. N. Sivarajan, "Design of Logical Topologies for Wavelength-Routed optical Networks", *IEEE JSAC*, pp. 840-851, Jun. 1996.

[19] L. Li, J. Halpern, P. Bahl, Y-M. Wang and R. Wattenhofer, "Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks", *ACM Symposium on Principles of Distributed Computing*, 2001.

[20] V. Rodoplu and T. Meng, "Minimum Energy Mobile Wireless Networks", *IEEE International Conference on Communication*, vol. 3, pp. 1633-1639, 7-11 June 1998.

[21] R. Wattenhofer, L. Li, P. Bahl and Y-M. Wang, "Distributed Topology Control for Wireless Ad-hoc Networks", *IEEE Infocom 2001*, pp. 1388-1397.

[22] S. Ruhrup, C. Schindelhauer, K. Volbert and M. Grunewald, "Performance of Distributed Algorithms for Topology Control in Wireless Networks", *International Parallel and Distributed Processing Symposium*, 2003.

[23] L. Bao and J. J. Garcia-Luna-Aceves, "Topology Management in Ad Hoc Networks", *ACM Mobihoc*, pp. 129-140, Annapolis, Maryland, June 2003.

[24] H. Gabow, "Implementation of Algorithms for Maximum Matching on Nonbipartite Graphs", Ph.D. thesis, Stanford University, 1973.