Routing of Video/Audio Streams In Packet-Switched Networks

Ciro A. Noronha Jr.

Technical Report No. CSL-TR-94-653

December 1994

This report is the author's Ph.D. dissertation, which was completed under the advisorship of Professor Fouad A. Tobagi. This work was in part supported by NASA under grant NAGW-419, by NSF under grant NCR-9016032 and by Pacific Bell. Ciro Noronha was supported by a graduate scholarship from FAPESP from Sept/89 to Aug/93 under grant 89/1658.

Routing of Video/Audio Streams In Packet-Switched Networks

Ciro A. Noronha Jr.

Technical Report: CSL-TR-94-653

December 1994

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305

Abstract

The transport of multimedia streams in computer communication networks raises issues at all layers of the OSI model. At the physical layer, the main issue is one of providing the appropriate bandwidth at all levels of the network infrastructure, from the local area network to the campus and wide-area backbones. The data link layer must provide support for priorities to differentiate the various traffic types (data, video, audio, still images, etc.) and satisfy their specific requirements. The same requirements must be taken into account at the network layer, which is responsible for finding routes. The transport layer must provide new functions such as, for example, semi-reliability and multipoint transport. New functions at the session layer include connection control functions (call establishment and management), floor control functions and synchronization. This thesis considers some of the issues related to supporting multimedia streams at the network layer; in particular, the issue of routing algorithms appropriate for multimedia streams.

The traffic generated by multimedia applications has different requirements than those of traditional data traffic. These requirements pertain to: (i) bandwidth; multimedia streams use bandwidth for long periods of time in a continuous fashion, while data traffic is bursty; (ii) multipoint communications; multimedia streams are expected to make heavy use of multicasting, while data traffic uses it only occasionally; and (iii) low latency, required by interactive communications. The routing algorithms used in deployed networks are not able to take these new requirements into account; in fact, they find routes from a topological point of view only. Moreover, there is no algorithm in the literature that can find routes taking into account all the requirements described.

In this thesis, we devise a routing algorithm for multimedia streams that not only satisfies the requirements, but is efficient, i.e., optimizes the usage of network resources in order to maximize that amount of traffic it can carry. We show that the optimum multicast stream routing problem can be formulated as a linear integer programming problem. Since traditional solution methods scale poorly with the size of the problem, we propose an efficient solution technique for this problem, based on enhanced value-fixing rules to prune the search space for the integer solution, and a two-step decomposition, to speed-up the linear relaxation of the problem. We show that the proposed solution technique significantly decreases the time to compute the solution, when compared to traditional methods.

The optimum multicast stream routing problem is NP-complete, which means that the worst-case run time for the optimum solution increases exponentially with the size of the problem. Hence, it might not be suitable for implementation in large networks; it can, however, be used as a benchmark against which to compare the performance of simpler heuristic solutions. We use the optimum solution to characterize the performance of existing heuristic algorithms (Shortest-Path for minimum delay, and the KMB heuristic for minimum cost) under realistic network and traffic scenarios. The performance is measured both in terms of session blocking probability and in terms of run time. We found that, if latency constraints are not important, the heuristic algorithms give good performance. Otherwise, one has to use the optimum solution. Based on the evaluation, we also derive guidelines for upgrading the network capacity.

We also consider the problem of routing multimedia streams in a Wavelength-Division Multiplexing (WDM) optical network. The WDM network has an additional degree of freedom over traditional networks - its topology can be changed by the routing algorithm to create the routes as needed, by tuning optical receivers and/or transmitters to different wavelengths. We show that the optimum reconfiguration and routing problem can be formulated as a linear integer programming problem. Since the solution to this problem is complex, we propose a number of simpler heuristic algorithms, both for unicast and for multicast traffic. The heuristic algorithms make use of Dijkstra's Shortest Path algorithm to identify a transmitter/receiver pair to be tuned. We evaluate the performance of the heuristic algorithms under realistic traffic scenarios, and compare it with that of a fixed topology of equivalent size. We show that, under certain conditions, the WDM network can carry twice as much load as the fixed-topology network. We also consider the issue of physical multicasting: in a WDM network, if the receivers are tunable, many receivers can be tuned to the wavelength of a single transmitter, thus creating a multicast group. We investigate this property, and derive guidelines for its use.

Key Words and Phrases: multicast routing, audio and video streams, multimedia, linear programming, integer programming, decomposition, shortest path routing, minimum cost routing, WDM optical networks, simulated annealing, shortest path routing, minimum cost routing.

Acknowledgements

This thesis could not have been written without the help and support of a number of people. First of all, Professor Fouad Tobagi, my advisor. I applied to Stanford because I wanted to work with Fouad, and was fortunate that he accepted me. The most important lesson I learned from Fouad is one of critical thinking. I wish to thank also my associate advisor, Professor Leonid Kazovsky, from whom I learned what I know about optical technology, and the members of the Optical Communications Research Laboratory.

Of the many people who shared my voyage through the Ph.D. program, one deserves special mention: İsmail Dalgıç, who was always there when I needed someone to bounce some ideas off. Other students in the group also deserve to be mentioned; in chronological order, Jon Peha, Weijia Wang, Benoit Paul-Dubois-Taine, Fabio Chiussi, William Chien and Jerry Luk Pat.

I acknowledge FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) and ITA (Instituto Tecnológico de Aeronáutica) for the financial support during my first four years at Stanford.

From the personal side, I am forever indebted to my lovely wife, Tiyomari, and to my son, Rodrigo, for making my years at Stanford pleasant, and for the many times they got by without my presence. I would also like to thank my father, Cyro, and my mother, Maria Auxiliadora, for the support they gave me throughout my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Characteristics of Multimedia Traffic

One of the driving forces for networking in this decade is multimedia. The majority of multimedia applications are distributed in nature, and involve networking and communications. Examples of such applications are video-conferencing, video-on-demand and computer supported collaborative work. Current network infrastructures and protocols have been designed taking into account the characteristics and requirements of traditional data traffic. Multimedia traffic has different characteristics, and place new requirements in the networking infrastructure. These new requirements pertain to the following aspects:

Bandwidth - multimedia streams use relatively high bandwidth on a continuous basis for long periods of time, while data traffic is bursty, but the average bandwidth used is low. For example, a high-quality compressed video stream can use anywhere from 1.5 to 8 Mb/s for extended periods of time, while the average bandwidth used by typical data applications can be well below 1 Mb/s. In fact, it is common to find in practice 10 Mb/s Ethernet segments supporting 100 to 200 users. The aggregate bandwidth to serve a number of users running multimedia applications is much larger than that for the same number of data users.

Low Latency (on the order of 100-200 ms end-to-end), required for some applications (such as videoconferencing or collaborative computing) that provide interactive communications. Data applications typically do not have such strict latency constraints.

Multipoint Communications - it is expected that a significant fraction of the multimedia traffic will be multipoint. Examples are videoconferencing, one-way video distribution and collaborative computing. Data applications, on the other hand, typically make only occasional use of multicasting and thus can afford to use highly inefficient methods, such as broadcasting or sending one separate copy of the information to each destination.

Semi-Reliability - data applications require full reliability, which is provided at the transport layer by retransmissions. Multimedia applications, on the other hand, can tolerate some loss, but have a strict timing requirement. For example, an audio sample that is delayed in the network due to congestion and arrives at the receiver past the time it should have been played is useless and is as good as lost. The determination of the amount of tolerable loss is a complex task, and depends on the kind of information, encoding, and application requirements.

Integrated Services - multimedia represents the integration of several kinds of media, such as video, audio, text, still images, etc. Some of these kinds of media generate traffic similar to that of traditional data applications; for example, transferring a still image is similar to a file transfer. Other types of media, such as video and audio, have different characteristics. The network should provide mechanisms to differentiate between the various types of traffic, and provide to each one the level of services they require, in an integrated fashion.

In this thesis, we consider issues related to providing multimedia services at the network layer. To situate the problem, we describe the network infrastructure as it is today, identify its shortcomings when it comes to supporting multimedia applications, and then focus in the specific issues addressed by this thesis.

1.2 The Network Infrastructure

The network infrastructure comprises the first three layers of the OSI model (physical, data link, and network), and is composed of the physical media, switches, interfaces, routers and associated protocols. As shown in Figure 1.1, the network infrastructure is organized in a hierarchical fashion. At the lowest layer, one finds the subnetwork, responsible for interconnecting users at the workgroup level. The subnetworks are interconnected by the campus backbone, using routers. The same picture applies to the next level, in a larger scale: campus networks are interconnected by Wide-Area Networks (WANs), again using routers.

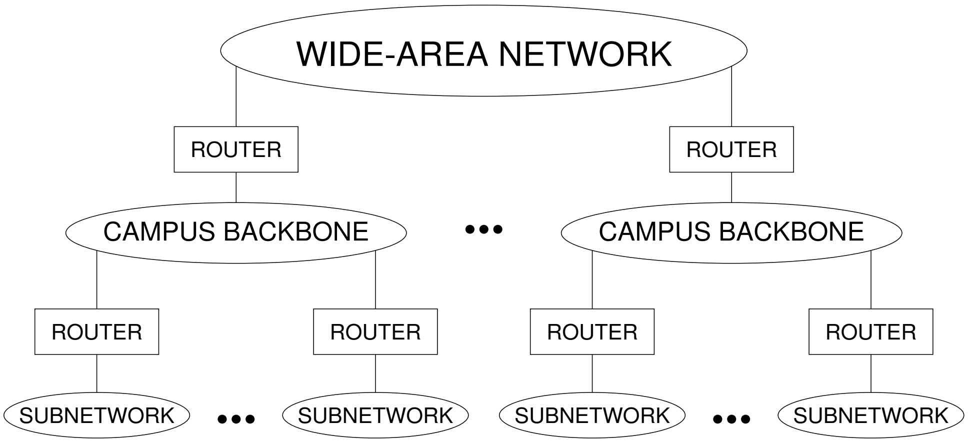


Figure 1.1: The Network Infrastructure

1.2.1 The Subnetwork

The function of the subnetwork is to connect the users in a workgroup, floor, or building. It is typically composed of one or more Local Area Network (LAN) segments, interconnected by *Bridges*. Bridges usually operate at the Medium Access Control (MAC) layer, and make the set of interconnected segments "look like" a single logical segment. Hosts in the network are typically not aware of their existence. Bridges monitor the traffic in the LAN segment to learn of the location of each host, and only forward traffic whose source and destination are in different segments. The network technologies used in the LAN segments are:

Ethernet (IEEE 802.3): It is the most commonly used LAN scheme, allowing multiple stations to share a single 10 Mb/s channel using the Carrier-Sense Multiple Access with Collision Detection scheme [1]. The original 10Base-5 standard specifies a coaxial cable interconnecting the stations in a bus topology; the maximum distance between two stations is 1.5 km (including repeaters). The more recent 10Base-T standard uses twisted pair to connect stations in a star topology. The center of the star is a multiport repeater, also known as a hub. The maximum distance between a station and the hub is 100 m.

Token Ring (IEEE 802.5): Stations are connected by point-to-point links and arranged in a ring. Access to this ring is controlled by a special bit pattern, called the *token.* The IEEE 802.5 specifies two possible bandwidths: 4 Mb/s and 16 Mb/s [2].

FDDI (ANSI X3T9.5): Another ring network, conceptually similar to the token ring, operating at 100 Mb/s [3].

The main issue at the subnetwork is to provide the appropriate level of bandwidth to the users. Other important issues are latency and integrated services. These issues can be handled as follows by the existing technology:

Bandwidth: In shared-channel LANs, the bandwidth of an individual segment is limited by the channel data rate (i.e., 10 Mb/s Ethernet, 16 Mb/s token ring). This limits the number of simultaneous users in that segment. For example, a 10 Mb/s ethernet cannot support more than 6 video streams at 1.5 Mb/s; as a matter of fact, other factors such as the multiple access scheme and the latency requirements might reduce this number to 4 or 5 streams [4]. If higher bandwidth is required, there are two alternatives: (i) increase the segment's bandwidth (by moving to another kind of LAN, e.g., from a 16 Mb/s token ring to a 100 Mb/s FDDI), or increase the number of segments, interconnecting them with bridges. However, there are limits to the extent this technique can be applied. These limits are illustrated in Figure 1.2 [5], where we plot the required number of bridged 10 Mb/s segments to achieve a given bandwidth. Note that, if the number of ports in the bridge is small, there achievable throughput is limited. For example, when 2-port bridges are employed, the upper bound in the bandwidth of any arrangement (under uniformly-addressed traffic) is twice the segment bandwidth. Multiport bridges, also known as *switching hubs*, are becoming a popular way of increasing the subnetwork bandwidth without having to change the user stations.

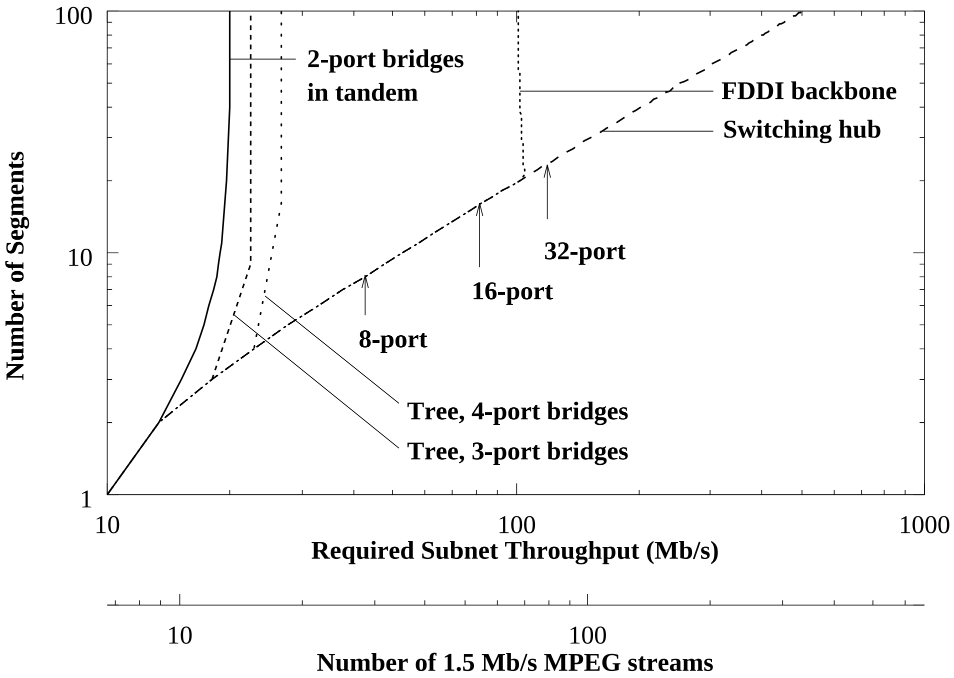


Figure 1.2: Number of segments required for bridged subnetworks, as a function of the aggregate bandwidth requirement

Latency: The CSMA/CD scheme used in Ethernet can lead to very large delays if the load in the network is high. To support multimedia in a CSMA/CD segment, the traffic load has to be limited. Token-passing ring networks are better in this aspect, since they provide bounded delays.

Integrated Services: The CSMA/CD scheme does not include a mechanism to differentiate between traffic types. Therefore, an Ethernet segment can only support a mixture of multimedia and data traffic to a very limited extent [4]. On the other hand, the ring networks have a priority scheme that can provide this function. Although most ring network adapters implement this function, it is typically not supported by the software; mechanisms to use the priority scheme still need to be defined.

A number of new technologies are emerging to support multimedia in the subnetwork. We have already mentioned one of them, switching hubs. Other emerging technologies include:

- 100 Mb/s Ethernet: Uses the same MAC protocol as standard Ethernet, with the data rate increased to 100 Mb/s. This standard is known as 100Base-T. Stations are connected via twisted pairs to a central hub (repeater), with a maximum cable length of 100 m. Due to the higher bandwidth, new physical layer specifications have been created [6, 7].

- Iso-Ethernet: Designed to provide isochronous service to the desktop. It overlays a 6.144 Mb/s synchronous channel, with ISDN signalling, on top of the traditional 10 Mb/s Ethernet. It uses the same wiring as the normal 10Base-T Ethernet, but with a more efficient line encoding. A special hub is needed, which can interface both with normal 10Base-T stations and Iso-Ethernet stations.

- 100VG-AnyLan: An alternative scheme to implement a 100 Mb/s Ethernet, but replacing the CSMA/CD access scheme. 100VG-AnyLan uses a conflict-free, centrally controlled, round-robin access scheme with two levels of priorities. It assumes a star topology, and the access is controlled by the hub.

1.2.2 The Campus Backbone and the Wide-Area Network

The campus backbone is responsible for interconnecting the subnetworks. Each subnetwork is connected to the backbone via a router. Campus backbones are interconnected together using Wide-Area Network links.

The same technologies used in the subnetwork can be employed in the campus backbone. If the traffic requirements are low, a single Ethernet, or a number of bridged Ethernets can do the job. Otherwise, a higher speed network can be used. Today, the main application of FDDI is as a campus backbone, and the Stanford University Network is a good example of this.

The as in the subnet, the main requirement at the backbone is one of providing bandwidth, and it is clear that a single LAN segment cannot go very far in that direction. Consider, for example, the Stanford Campus Network, where (December 1994) there are on the order of 20,000 hosts. If 10% of these hosts are sending 1.5 Mb/s video to destinations outside their subnetworks, an aggregate bandwidth of 3 Gb/s is required in the backbone. The trend today in providing this kind of bandwidth is through switches - the so-called *collapsed backbone*. These can be either large switching hubs or the new ATM switches [5].

1.3 State of the Art in the Network Layer

The primary function of the network infrastructure is to provide connectivity between the hosts. If the source and the destination are in the same subnetwork, no network layer functionality is needed - the packets are routed transparently based on the MAC address. If the subnetwork is made out of bridged segments (which, in the most general case, might be organized in a mesh topology), the bridges use the IEEE 802.1 protocol [8] to learn the topology and compute a single spanning tree for this topology. All the traffic is then routed over this spanning tree; links not in the spanning tree are kept in backup state, to be used only if a link in the spanning tree fails.

This scheme is clearly not scalable (because of the flat address space), and does not make full use of the available bandwidth in the infrastructure. Therefore, when interconnecting subnetworks via the campus backbone, or multiple campus backbones via the WAN, more functionality is needed. In particular, it desirable to utilize different sink trees[1] for different nodes. This service is provided by the network layer, using *Routers*.

Routers must, then, be able to forward packets they receive to their appropriate destinations. To do so, they must be able to somehow find the path from their location to all possible destinations, and forward the packets they receive to the correct output link. In case of a multicast packet, the routers are also responsible for replicating the packet as needed, so the data can reach all the intended destinations. A good way to route the packets is using the *Shortest Path* to the destination. One important property of the shortest path is that if the shortest path from node A to node B is via node C, then the path used from C to B is the shortest path between these two nodes. Due to this property, a router does not need to care about the sources of the packets it receives - it just needs to forward these packets through the shortest paths from its location to their destinations, regardless of where they came from.

In the shortest path computation, a numeric *label* is assigned to each link in the network, and the length of the path is the sum of the labels of the links in the path; the algorithm minimizes this length. In general, the link labels are statically configured by the network

[1] A *sink tree* for a given node is the tree formed by the paths from all nodes that that given node.

administrator. A common practice is to assign the value of "1" to all labels, which leads to routes with a minimum number of hops. For networks where there are different link data rates, the practice is to assign the link labels to values inversely proportional to the link data rates (e.g., a value of "1" for a 100 Mb/s FDDI, a value of "10" for a 10 Mb/s Ethernet). The following algorithms are available to compute the shortest path [9]:

Bellman-Ford: Iterates on the number of hops in the path. The network must not contain a loop with negative length. This algorithm is well suited for distributed implementation, and is widely used in practice due to this fact.

Dijkstra: Iterates on the path length. The network cannot have links with negative labels.

Floyd-Warshall: Iterates on the nodes in the path. As with Bellman-Ford, the network cannot have loops with negative length. This algorithm is well-suited for computing all the shortest paths in the network (i.e., the shortest path between every pair of nodes); however, it is not as efficient as the previous two when finding the path between a given pair of nodes.

The three algorithms above are equivalent; given the network and the source-destination pair, they will find the same path (or paths of equivalent lengths, if more than one shortest-path exist).

In a computer network, the routing algorithm has to be run in a distributed fashion by the routers. Since routers need to exchange information with each other, a *Routing Protocol* must exist to take care of this information exchange. In the most general case, the routers will exchange information about the network topology; with this information, each router will build a "picture" of the network topology, execute the shortest path algorithm to each of the possible destinations, and build its *Routing Table*, which gives the outgoing link to be used as a function of the requested destination. If the network topology changes, the routers re-run the algorithm for the new topology.

The most common routing protocols used in IP (Internet Protocol) networks are RIP and OSPF. RIP [10, 11] uses the Bellman-Ford algorithm. OSPF [12], uses Dijkstra's shortest path algorithm; the link labels used when routing are assigned by the network administrator, which makes it more flexible than RIP. Moreover, it converges faster to new routes if the network topology changes (e.g., when a link fails). Routers provide a rudimentary support for QoS by using the IP Type-of-Service (TOS) [13] field in the packet header; different routes are used for different values of the TOS field. We present below a detailed description of these two routing protocols.

1.3.1 The Routing Information Protocol (RIP)

The Routing Information Protocol (RIP) [10, 11] is classified as an "Interior Gateway Protocol" (IGP), i.e., it is intended for use in a network administered by a single entity with a reasonable degree of technical and administrative control - an "autonomous system". An example of an autonomous system would be a campus network. Other protocols, known as "Exterior Gateway Protocols" (EGP), are used to route between autonomous systems.

RIP finds the routes using the Bellman-Ford (also known as "distance vector") shortest-path algorithm. The protocol works as follows: each node in the network periodically advertises to its neighbors the distances to all the nodes in the network using the routes it knows about. To build its routing table, each node looks at the advertisements from its neighbors, and for each destination it adds the advertised distance to its distance to that neighbor, and selects the minimum over all the neighbors. It has been shown that, under fairly general conditions, the routes converge to the shortest paths in a finite time [9]. The link labels are statically configured by the network administrator. However, since the maximum path length is limited to 16 (to limit the time the network takes to converge to new routes if a link goes down), the link labels are almost always set to 1, and the routes minimize the number of hops.

A problem with this algorithm is that, if the network topology changes (due to a link failure, for example), the protocol takes some time to recover, and during this time, routing loops can occur. Techniques such as split horizon routing and triggered updates [10] can be used to accelerate convergence, but cannot completely eliminate these transitory routing loops.

1.3.2 The Open Shortest Path First (OSPF) Protocol

The Open Shortest Path First (OSPF) protocol is also an IGP, and was introduced to overcome some of the limitations of RIP, especially the speed of convergence when there are changes in the network topology. In a completely static network (topology remains fixed), both OSPF and RIP eventually will find the same (or equivalent) routes.

OSPF uses Dijkstra's shortest path algorithm (also known as "link state") to find the routes, which requires knowledge of the global network topology to operate. In other words, each router needs to have a "map" of the network in order to find the routes. To distribute this information, the protocol makes use of "flooding": periodically, each router sends route updates through all its outgoing links. Upon receiving an update from another node, a router immediately retransmits it in all its outgoing links, except the one in which it was received. Route updates contain sequence numbers, which allow routers to discard duplicates of the same update, as well as older updates that might still be floating in the network. The update message sent by each router contains the distances to each of its neighbors. By putting together the messages from each of the nodes in the network, a router can build a picture of its topology. Note that, while in RIP a router sends a copy of its *entire* routing table to its *neighbors only*, in OSPF it sends *the distance to its neighbors only* to the *whole network*.

OSPF is a much more complex protocol than RIP, with many more features (the RIP specification [10] has only 33 pages, while the OSPF specification [12] has 212). OSPF features include:

- Faster convergence in the case of changes in the network topology.
- Support for Type-Of-Service (TOS) IP routing (multiple routes with different priorities).
- Link labels can be set by the network administrator to any arbitrary 8-bit value; speed of convergence to new routes after a change in the network topology is not a function of the link labels.
- Support for simultaneous use of multiple routes of the same length.

1.3.3 Multicasting

The simplest ways to implement multipoint communications are to send a separate and independent copy of the information from the source to each of the destinations, or to broadcast it to all nodes and have them filter the received data. These schemes are, of course, highly inefficient; they would make sense only if the need for multipoint communication arose only occasionally. If larger volumes of data are involved (such as with video streams), more efficient ways to do multicasting are needed.

The first step in providing more efficient multicasting is to make sure that: (i) at most one copy of the message is sent on any given link in the network, and (ii) the message is sent only to nodes that either are in the multicast group, or in the path from the source to the members of the multicast group [14]. To accomplish that, for each multicast message, the routers must know *who* the destinations are. One possibility is to list all the destinations with each message, but this might lead to inefficiency (because each datagram must contain the complete list of destinations) and scalability problems. Moreover, in some scenarios, such as TV broadcasting, the source might not even know who the destinations are. The solution, then, is to create *multicast addresses* to identify the multicast groups, and give the routers some mechanism to learn which hosts belong to each group. This way, the messages can be addressed to this "group address".

IP multicast [15] is a mechanism to convey group membership information to the routers. IP multicast routing is based on IP multicast addresses, or class D addresses (i.e., those with "1110" as their high-order four bits; from 224.0.0.0 to 239.255.255.255). A multicast packet has exactly the same format as a unicast packet; the only difference is the use of a multicast destination address. Multicast routers are required to know, for each multicast address in use, the locations of the current group members. The protocol used to "discover" the group members is called IGMP (Internet Group Management Protocol).

The basic operation of the IGMP protocol is:

- When a host decides to join a multicast group, it sends an IGMP "Host Membership Report" packet destined to the group address it wants to join. The router(s) directly attached to the network where the host resides listen to all multicast traffic and learn that a host in that network has joined that specific group. From this moment on, traffic addressed to that specific group should also be forwarded to the network where the host in question resides.

- Periodically, all the routers in the network send a "Host Membership Query" to the "all-hosts" group (multicast address 224.0.0.1; all the multicast-capable hosts in the network are required to join this group). Note that messages to the "all-hosts" group are never forwarded by the routers; they stay in the local subnetwork. Strictly speaking, the "all hosts" group means "all multicast-capable hosts in *this* subnetwork"; it so happens that the same address is used in all subnetworks. Each host is expected to answer the query with a "Host Membership Report" for all the groups it is a member. As before, the Host Membership Report is sent to the group address and thus received by all the members of the group in that network (but is not forwarded by the router). To prevent an explosion of answers, each host waits a random delay before sending its membership report. If, during this time, it receives a membership report from some other host in the same network, it cancels its own response; it is not needed anymore because the router already knows that there is at least one group member in that subnetwork.

- When a host decides to leave a group, it just stops answering the membership queries for that group. When a router no longer receives host membership responses on a particular group address, it knows that this group does not have any more members in that particular network, and it stops forwarding traffic addressed to this group to it.

It should be noted that the IP multicast RFC [15] does not specify any provision to translate names or services into multicast IP addresses, except for some "well-known" groups which have pre-assigned static addresses.

Once the routers know the location of the members of a group, they can find the multicast routes. A multicast route is a tree, whose root is at the source of the multicast, and reaching all the members of that multicast group. Currently, this tree is found by merging the unicast routes from the source to each of the destinations; as indicated above, these routes are found using shortest-path algorithms.

A multicast extension of the OSPF protocol, known as MOSPF (Multicast OSPF) [16], has been developed and is available in several commercial routers. In addition to the traditional IP unicast routing functions, MOSPF is responsible for communicating group membership information between routers. As with the unicast version (OSPF), the routes are found using Dijkstra's shortest-path algorithm, and then merged into the multicast tree.

1.4 Routing Multimedia Information

As we have seen in the previous section, routing in traditional data networks is done from a topological point of view, without taking into account the source requirements. The path is found using the shortest path algorithm, and the link labels which define the "length" of the path are statically configured by the network administrator, and are usually inversely proportional to the link data rates. In this section, we examine what is needed to satisfy the new multimedia requirements, indicate what has been done so far, and delimit the scope of this thesis.

1.4.1 The Bandwidth Requirement

A multimedia *stream* is a continuous flow of information (i.e., video frames or audio samples) that has to be delivered in a timely fashion. Some video/audio encoders produce constant bit-rate streams; others produce variable bit rate streams. However, even variable bit-rate streams are not as bursty as data traffic. A *bandwidth requirement* to support the stream can be defined both for constant and for variable bit rate streams. For constant bit-rate streams, the bandwidth requirement is just the data rate of the stream; for variable bit-rate streams, it is some statistical value, between the average and peak rates of the stream. This statistical value is computed based on the stream characteristics, and on an acceptable data loss (due to buffer overflows) and delay when this stream is multiplexed with other variable bit-rate streams. To properly support the streams, the network has to:

(i) Keep track of the current allocated bandwidth in each link.

(ii) When establishing a new stream, take into account the current network usage as the routes are computed.

In general, this means that the network should be able to use *multiple routes* between the same source and destination. If the available capacity in the shortest path has all been used up, the network should be able to use other (longer) paths to carry the additional load. Figure 1.3 is an example. The shortest path between nodes **S** and **D** is the direct connection (1 hop). If all links are 10 Mb/s links, then one can have at most 6 streams of 1.5 Mb/s between **S** and **D**. However, if the router is capable of using other (longer) routes, then 6 more streams from node **S** to node **D** can be routed through node **A**.

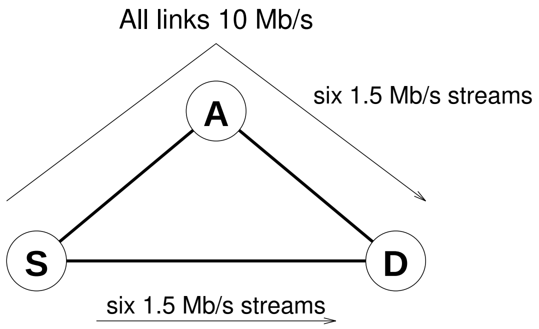


Figure 1.3: Multiple-route example

From a routing algorithm point of view, a simple modification of the shortest path algorithm can provide this additional functionality. Whenever an application generates a new stream, the network should recompute the shortest path to the destination. Before doing so, links with insufficient free bandwidth to support that stream are temporarily pruned from the network topology. If no path can be found, the stream is blocked. Note that this represents a change in the way routers operate. In traditional data networks, routers exchange information about the network topology; in a network supporting multimedia traffic, the routers will have to exchange additional information about the current usage of the links.

Another important issue is that of applications that generate *multiple-stream sessions*. One example is a video-conferencing session with P participants, where each conferee can see the other $P-1$ participants. Such a video-conference would be established with P simultaneous multicast streams. The shortest path algorithm is unable to compute multiple routes simultaneously - it has to be applied to each source-destination pair, sequentially. The problem of simultaneously routing multiple *unicast* streams corresponds to the well-known *multicommodity flow problem* [17], which is solvable by linear programming. An algorithm to route multiple *multicast* streams is given in this thesis.

Resource Reservation Protocols have been proposed to deal with the problem of allocating and reserving resources. Since current routers are unable to deal with reservations, the resource reservation protocols are, in principle, independent of routing. All they do is to take the existing routes and reserve resources along them (typically bandwidth). This separation between resource reservation and routing is clearly suboptimal (e.g., it does not allow for multiple routes between a pair of nodes), but it allows early deployment and testing of these protocols, which initially run in dedicated workstations. Eventually, as the routers evolve, the resource reservation protocols should become an integral part of routing, and be implemented at the routers.

Resource reservation protocols are an active research area. The two most important protocols for resource reservation are the Internet Stream Protocol, version 2 (ST-II) [18] and the Resource ReSerVation Protocol (RSVP) [19, 20]. These two protocols are described below.

1.4.1.1 The Internet Stream Protocol, version 2 (ST-II)

The ST-II protocol is an extension and enhancement of the original ST protocol proposed by Forgie [21]. It has two components: the ST Control Message Protocol (SCMP), which is a reliable transport for the protocol messages, and the ST protocol itself, which is an unreliable transport for the data. Figure 1.4 [18] shows the protocol relationships between ST-II, SCMP and the IP stack.

The functions of SCMP are as follows:

- create streams;
- refuse the creation of streams;
- delete a stream in whole or in part;

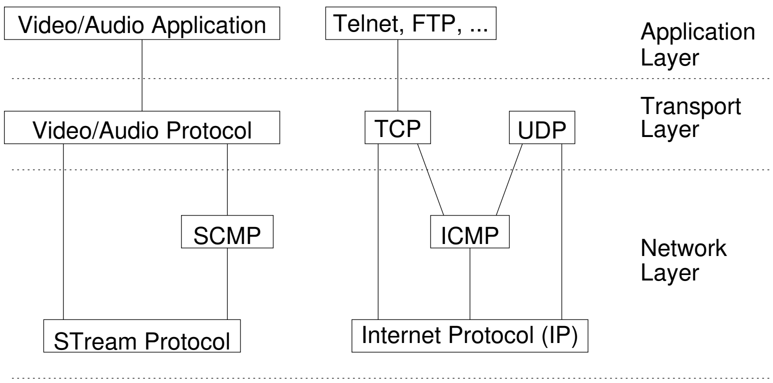


Figure 1.4: The ST-II protocol stack

- negotiate or change the stream's parameters; and
- recover from network failures by re-routing and tearing down old routes.

SCMP is a request-response protocol; reliability is provided by retransmissions after a suitable time-out.

The Stream Protocol (ST) is responsible for transferring user data from node to node. To expedite processing in the nodes, ST data packets do not carry complete addressing information: they have instead a Hop IDentifier (HID), similar to a virtual circuit number in a VC network, which is negotiated for each hop during the setup phase. ST includes a priority function to allow selective discarding of data packets if congestion happens, and an optional timestamp for each packet.

ST-II reservations originate from the source. The source sends an SCMP message with a *flow specification* ("flowspec"), which describes the stream requirements in terms of packet size, data rate, etc. For some of the parameters, such as data rate, the message contains both a desired value and a minimum value; intermediate nodes in the path can rewrite the desired value as long as it is above the minimum. Other flowspec parameters, such as delay, are not really source parameters: their values are measured along the path, and the received flowspec at the destination will contain the final value. These value pertains to the quality of service (QoS) of the path taken. The measurement process is as follows: the source initializes the flowspec field with the value of zero, and each intermediate node adds its delay to the current value in the flowspec, before forwarding it to the next node. An intermediate node in the path will refuse the connection if it does not have resources to provide the desired level of service. If the destination accepts the call, it returns the final flow specification to the source, to let it know the resulting QoS and traffic parameters. Parameters included in the ST-II flowspec are:

- Parameters describing the traffic at the source:
  - *Duty Factor:* estimated proportion of time that the requested bandwidth is in use;
  - *Precedence:* higher precedence streams are allowed to take resources previously committed to lower precedence streams;
  - *Recovery Time-out:* maximum amount of time the application is willing to wait for a system failure to be detected and corrected;
  - *Minimum Bandwidth:* minimum bandwidth required by the source;
- Parameters defining the quality of service expected by the source, which are not negotiable:
  - *Error Rate:* bit error rate;
  - *Reliability:* probability that a data packet will be dropped;
  - *Delay Limit:* maximum end-to-end delay;
- Negotiable parameters. The source supplies the desired and minimum values for each parameter; intermediate nodes are allowed to lower the desired value as long as it does not become less than the minimum.
  - *Protocol Data Unit (PDU) size.*
  - *PDU rate.*
- Parameters measured along the route: in the flowspec sent by the source, these values are set to zero. Each node in the path will add their current estimate of the parameter to the current value in the received flowspec, and forward the modified flowspec.

- *Accumulated Mean Delay.*
- *Accumulated Delay Variance.*

There are at least 4 operational implementations of ST-II, most notably the one at IBM Heidelberg [22]. A working group to revise ST-II was started at the IETF Amsterdam meeting.

1.4.1.2 The Resource ReSerVation Protocol (RSVP)

The motivation behind RSVP is to support heterogeneous receivers, with different reservations for the same stream. For example, consider a layered video stream, which can be seen as a stream where multiple video qualities are available. To describe the concept of layered video, let us assume first only two possible qualities: "basic" and "enhanced". In a layered stream, packets corresponding to the basic and enhanced qualities are tagged as such. A receiver wanting only the basic quality would drop or otherwise ignore the packets corresponding to the enhanced quality, and decode and display only the packets corresponding to the basic quality. A more capable receiver can use the packets corresponding to the enhanced quality to improve the received image; such a receiver would process all the packets. Of course, this concept can be extended to more than two layers; the more layers a receiver is able to retrieve and process, the better the quality. If a specific receiver is not able to process all the data in the stream, that data should not be delivered to it, thus saving in network bandwidth. Since each receiver knows its own capabilities, it is better to have receiver-oriented reservations in this scenario. The sender notifies the receiver(s) that it wishes to start communication, and the receiver(s) reserve the resources. This model of operation is also appropriate for scenarios where group membership is dynamic, i.e., receivers can join and leave a multicast transmission in progress; the sender is not required to know the identity of the receivers or their capabilities.

Unlike ST-II, RSVP [19] does not define a flow specification and does not have a data transfer component; it only transfers the reservations and keeps state (i.e., remembers who reserved what resources) at the intermediate nodes. The flow specification itself is seen as a block of data to be transferred when performing the reservation; RSVP only needs to be able to "compare" two flow specifications (to find which one requests more resources) and "merge" two or more flow specifications into a "larger" one. Of course, a RSVP implementation must have a well-defined flowspec, and implementations with different flowspecs will not interoperate. RSVP messages are sent as IP datagrams, and the routers keep "soft state", which is refreshed by periodic reservation messages. In the absence of the refresh messages, the routers delete the reservation after an appropriate timeout. The use of soft state in the routers allows the use of an unreliable transport (IP) for the RSVP messages; if a message is lost, it will be retransmitted anyway at a latter time. For a given multicast tree, the reservation refresh messages are merged as they proceed towards the sender, to avoid swamping it (and the upstream routers) with messages.

RSVP supports different reservation styles, to further optimize the usage of network resources. The following reservation styles have been defined:

Wildcard-Filter: Used when there are many senders that will communicate with a given receiver, but not simultaneously, as, for example, in an audio conference, where only one participant at a time can talk. A single "pipe" is created to carry the request. This is illustrated in Figure 1.5, where senders S_1 and S_2 will send a flow of f to receiver R; however, they cannot be active at the same time because only f is reserved from N to R. The name "wildcard" refers to the fact that the receiver can select *any* of the senders (but only one at at time).

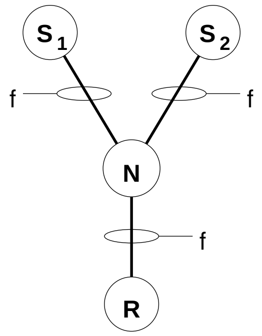


Figure 1.5: Example of wildcard-filter reservation style

Fixed-Filter: Receivers make separate reservations for different senders; all senders can be active at the same time, as, for example, in a video-conference, where a conferee can see all the other participants in the screen. Figure 1.6 illustrates the use of the fixed-filter reservation style. There, both S_1 and S_2 can be active at the same time, because a reservation was made for a flow of $2f$ in the link from N to R.

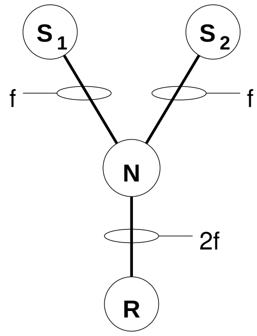


Figure 1.6: Example of fixed-filter reservation style

Dynamic-Filter: Receivers specify a set of N senders, sharing a common reservation; from this set of N senders, the receiver specifies a subset of at most K senders ($K \leq N$) which can use the reservation. This is shown in Figure 1.7, where only K senders out of $S_1, S_2, \ldots, S_N$ can be active simultaneously, because a reservation was made for a flow of Kf in the link from N to R.

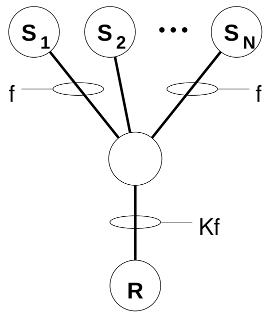


Figure 1.7: Example of dynamic-filter reservation style

RSVP is still being designed; first implementations and field trials are under way. Comparisons between ST-II and RSVP can be found in [23] and [24].

1.4.2 The Latency Requirement

Interactive communications require that the latency between the instant the information is generated at one end, and the instant it is delivered at the other end, to be no more than 100-200 ms. Ideally, this constraint should be taken into account when finding the routes. In practice, resource reservation protocols such as ST-II *measure* the latency *after* the route has been found, and accept it or not.

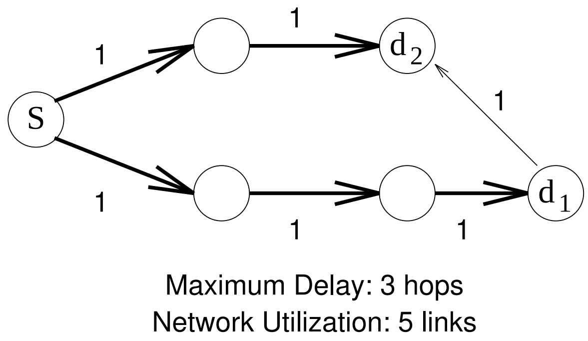
In general, a link can be seen as having two independent labels: a *delay* label, which depends on the propagation delay in the link and on its data rate, and a *cost* label, which measures how much (in $/bit, or in $/month) that link costs to operate. If routes are computed by the shortest path algorithm, and the link delays are used as labels in the algorithm, the routes correspond to the minimum possible delay; if these routes do not satisfy the latency requirements, then the requirements cannot be met. On the other hand, by using costs as link labels, one can find the "cheapest" routes, but no guarantee is offered on delay. What is needed is an algorithm that can find the cheapest routes that satisfy a certain latency constraint; the shortest path algorithm cannot provide this functionality. This thesis presents an algorithm that is able to compute minimum-cost routes subject to a latency constraint, for the more general (multicast) case.

1.4.3 The Multipoint Communications Requirement

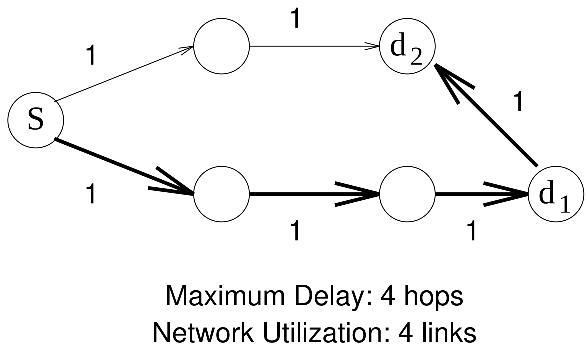
As indicated above, it is expected that multimedia applications will make extensive use of multipoint communications. Examples are multi-party videoconferencing, TV broadcasting, etc. Due to the high bandwidths involved, multicasting has to be as efficient as possible.

Currently, multicast routes are computed using the shortest-path algorithm. However, using shortest-path routes may lead to inefficient network utilization. This is illustrated in the routes of Figures 1.8(a) and (b). In both cases, node S wishes to send a multicast message to nodes d_1 and d_2. The multicast tree shown in Figure 1.8(a) corresponds to the shortest-path routes; notice that the maximum delay over all destinations is 3 hops (from S to d_1), but the multicast uses bandwidth in 5 links. The multicast tree in Figure 1.8(b) can deliver the data using bandwidth in only 4 links, but the maximum delay over all destinations increases to 4 hops (path from S to d_2).

The example of Figure 1.8 shows that, for multicast routes, there is a tradeoff between usage of network resources and delay (in this example, taken to be the number of hops). The routing algorithm implemented in current routers (shortest-path) is at one of the extremes of this tradeoff: minimum delay. Multimedia applications may specify a maximum tolerable delay; routers should be able to optimize the usage of network resources while meeting that delay constraint. From a routing algorithm point of view, finding the multicast tree that minimizes the usage of network resources corresponds to the well-known Minimum Cost Steiner Tree problem in graphs, for which a (large) number of heuristics and algorithms have been proposed [25]. As with the unicast case, a multicast routing algorithm which minimizes the cost while meeting a delay constraint is needed. This thesis proposes a multicast routing algorithm which is able to minimize any linear combination of cost and delay, while satisfying



(a) Shortest-Path routes



(b) Minimum network usage routes

Figure 1.8: Contrast between delay and network utilization

a delay constraint.

1.5 Dynamic Topology Change: The WDM Network

Due to its low attenuation and very high bandwidth, fiber has become the medium of choice for point-to-point links. By using Wavelength-Division Multiplexing, many channels can be created in the same fiber. If the optical signal from the various optical transmitters in the network is combined, and made available to the optical receivers, and if the optical transmitters and/or receivers are tunable, it is possible to build a network where point-to-point links can be dynamically created between nodes. In such a network, the routing algorithm has an additional degree of freedom: it can *modify* the network topology, as well as choose the routes. In other words, the routing algorithm is now allowed to actively change the network topology as a function of the traffic requirements.

Previous work in this field has focused in the following two extremes:

- Very fast reconfiguration: transceivers are tuned in essentially zero time, and communication is always single hop. This can be difficult to implement in practice, due to the need of very high-speed synchronization or some multiple-access scheme to resolve contention for the optical channel.

- Very slow reconfiguration: the network is reconfigured infrequently, in response to long-term changes in the traffic trends. Computing the optimum topology from the average traffic matrix becomes similar to traditional topological network design. Since optical devices can be tuned in the microsecond time range, this mode of operation does not make full use of the capabilities of the optical network.

In this thesis, we consider the problem of routing multimedia streams in a WDM network, where the tuning of transmitters and receivers happens at the stream arrival/departure time scale. This way, the synchronization problems associated with packet-by-packet tuning are avoided, while preserving the potential flexibility of the WDM network.

1.6 Outline of the Thesis

As indicated earlier in this chapter, the requirements underlying multimedia traffic are different from those underlying data traffic. The routing algorithms implemented in practice are not designed to take into account these requirements. Moreover, there is no algorithm available in the literature which is capable of simultaneously meeting all these requirements. In this thesis, we devise an optimum multicast routing algorithm, capable of meeting the requirements of bandwidth and latency for multimedia streams. Moreover, this algorithm is capable of optimizing the usage of network resources, thus maximizing the amount of load the network can support. The algorithm is presented in Chapter 2, where we show that the optimum multicast stream routing problem can be formulated as a linear integer programming problem. Since traditional solution methods for this problem do not scale well with the size of the problem, we present an efficient solution technique for this formulation, which uses decomposition to speed-up the linear relaxation of the problem, and improved value-fixing rules, to prune the search space for the integer solution. The speed-up due to these techniques is also evaluated in Chapter 2, by implementing the algorithm and measuring its run time as the various techniques are introduced.

The optimum multicast routing problem is known to be NP-complete (in fact, a simpler version of it, the minimum Steiner Tree problem in graphs, is already NP-complete [26]). Therefore, even with the improvements described in Chapter 2, the worst case run time of the optimum algorithm is still exponential with the size of the problem, which limits its usage to small networks. However, it can also be used as a benchmark against which to compare the performance of simpler heuristic algorithms. In Chapter 3, we use the optimum multicast routing algorithm developed in Chapter 2 to evaluate the performance of existing routing algorithms under realistic network and traffic scenarios. The previous work in evaluating routing algorithms has focused in computing the cost and/or delay of a single route on an empty network. We evaluate the performance of the algorithms in a dynamic scenario, where multimedia sessions arrive, are routed if enough resources are available, and leave the network after some period of time. In such a scenario, the primary performance measure of interest is not cost or delay, but rather the session blocking probability. Special care is taken to ensure that the evaluation scenarios are realistic, i.e., they reflect properties of deployed networks. As a result of this evaluation, we are able to compare the various algorithms under different scenarios, and derive guidelines for their usage. In particular, we find that the performance of the heuristic algorithms is close to the optimum if there are no latency constraints. We also investigate the issue of how to upgrade the capacity of a network.

In Chapter 4, we consider the routing of streams in optical Wavelength-Division Multiplexing networks. Due to the low attenuation of fiber, it is now widely used in point-to-point links for any distance over over 100-200 m and data rate over ≈ 45 Mb/s. By using multiple wavelengths in the fiber, one can create multiple channels; additionally, if the wavelengths are properly distributed and the network nodes are fitted with tunable optical transmitters and receivers, it is possible to build a network whose *logical topology* is independent from its *physical topology* and can be dynamically changed as a function of traffic. In traditional networks, such as the ones discussed in Chapters 2 and 3, the only degree of freedom allowed to the routing algorithm is the selection of paths; in a WDM network, however, an additional degree of freedom is available: the topology itself. In Chapter 4, we extend the linear programming formulation of Chapter 2 to WDM networks. Since the solution to this formulation is complex, we propose simpler heuristic algorithms, both for the unicast and for the multicast cases. The heuristic algorithms use Dijkstra's Shortest-Path algorithm to find a transmitter/receiver pair to be tuned.

In Chapter 4 we also present an evaluation of the proposed heuristic algorithms. For unicast traffic, we first derive an upper bound in the performance of any algorithm, and show that the performance of the heuristic is close to it, thus obviating the need to pursue the more complex optimum solution. We also compare the performance of the WDM network with that of a fixed-topology network, namely the ShuffleNet. We evaluate the performance of the WDM network in a dynamic environment, similar to that of Chapter 3, where multimedia sessions arrive, are routed if enough resources are available, stay in the network for some time, and terminate. We compare the performance of the WDM network to that of a centralized switch of equivalent complexity, and show that under certain conditions, the WDM network can outperform the centralized switch.

A WDM network can have either tunable transmitters or tunable receivers (or both). If the receivers are not tunable, physical communication has to be one-to-one; if two transmitters tune to the same wavelength, their signals collide; the receiver might be able to receive the signal from the transmitter with higher power, or it might not be able to receive at all. However, if the receivers are tunable, multiple receivers can be tuned to the same wavelength, creating a physical multicast group. Multicast traffic might potentially be able to make use of this additional degree of freedom. The heuristic routing algorithms proposed by us can make use of this property, if it is available. Therefore, when evaluating the performance of the WDM network under multicast traffic, we consider both WDM networks with tunable transmitters and WDM networks with tunable receivers. The evaluation is done under the same dynamic traffic scenario as in Chapter 3, and we are able to compare the various heuristic routing algorithms in the basis of their blocking probabilities. In particular, we find that the use of physical multicasting can *degrade* the performance of the network, if not correctly used. More specifically, there should be at least one receiver tuned to each optical transmitter in the network; a network with equal number of optical transmitters and receivers should not use physical multicasting.

Finally, in Chapter 5, we present our conclusions and list areas of possible future work.

Chapter 2

Optimum Routing of Multicast Streams

2.1 Introduction

In Chapter 1, we determined the need for new routing algorithms able to take into account the requirements of multimedia traffic. Moreover, due to the large amounts of data involved, the new routing algorithms must be efficient in allocating the network resources. The requirements for the routing algorithm are:

1. It should take into account the stream bandwidth requirements when finding the routes.
2. It should be able to minimize the cost of the routes found, while meeting the latency requirements of multimedia traffic.
3. It should be able to route multipoint traffic.
4. It should be able to route a session composed of multiple streams in a single pass, globally optimizing the routes for the various streams in the session.

In this chapter, we present such an algorithm, based on integer linear programming. We also propose an efficient solution technique and evaluate its speed-up over traditional methods.

In section 2.2, we describe the network and traffic models and give a formal problem definition for the multicast stream routing problem. In section 2.3, we summarize the existing multicast routing algorithms, and in section 2.4 we present an integer programming formulation for the optimum multicast stream routing problem. In section 2.5 we present an efficient solution technique, based on the branch-and-bound method, which has two parts: (i) an extension of the decomposition procedure, to speed-up the linear relaxation of the problem, and (ii) enhanced value-fixing rules, to prune the search space for the integer solution. This speed-up is characterized in section 2.6, where we also compare the run times of the optimum solution with those of the heuristic solutions. Finally, in section 2.7, we summarize our conclusions. The extension to the decomposition procedure is given in Appendix A.

2.2 The Problem Formulation

When an application requests a multicast session, the multicast routing algorithm is responsible for finding routes for each of its component streams; each route should have enough free bandwidth to support the stream, and should not exceed its latency constraint. If there are multiple routes that satisfy the requirements for a given stream, the routing algorithm will choose one so as to optimize a certain objective function. In this section, we define the traffic model and formulate the routing problem.

2.2.1 The Network Model

The network is seen as a collection of *nodes*, interconnected by *links* subject to a certain *topology*. In this thesis, we consider all links to be point-to-point and directed (i.e., information can flow in only one direction); later in this section we show how other kinds of topologies (such as shared-medium and WDM networks) can be accommodated under this assumption. Full-duplex connections between a pair of nodes correspond to two independent links, one in each direction; this is needed because multicasts are unidirectional, and the bandwidth is managed separately in each direction. Each link is characterized by the following parameters:

Capacity: Total link bandwidth, in bits/second.

Cost: Monetary cost of using the link, in \$/bit. In other words, the cost of routing a stream of r bits/second over a link costing C \$/bit is rC \$/second.

Delay: Each link has associated with it a certain delay D, which is the time between the instant a bit of information becomes ready to be transmitted at the origin of the link, and the instant it is received at the destination. The delay D has three components, as illustrated in Figure 2.1:

- The node processing delay, D_N. This is the time between the instant a packet is received at the node, and the instant it is queued for transmission in the appropriate output line. For a given node, this component of the delay is a constant. Since routing is usually hardware-assisted (it is only a table look-up), we neglect this component of the delay.
- The queueing delay, D_Q. This is the time between the instant a packet is submitted to the queue at an output link, and the instant the transmission of this packet is completed. For traditional data applications, it is common to assume an $M/M/1$ model for this queue; moreover, in general this delay is a function of the flow in the link. Due to its latency constraints and its requirement for guaranteed bandwidth, stream traffic has priority over data traffic (as illustrated in Figure 2.1). Moreover, stream traffic arrives in a regular fashion, and the packet sizes are usually constant (and, in any case, limited by the maximum packet size for the network in question). Therefore, if a given stream is using packets of size B bits and has been allocated a bandwidth of C_s, the queueing delay D_Q will be between B/C (empty link; no other streams or data using it) and B/C_s (link fully utilized). Since D_Q is bounded, for this thesis we assume that it is equal to a fixed value, independent of the link load. This value is taken to be the upper bound (B/C_s), since multimedia applications impose a limit on the *maximum* delay.
- The propagation delay, D_P. This is the time between the instant a bit of information is transmitted at the origin of the link, and the instant it is received at

the other end. This component of the delay is equal to the physical length of the link divided by the speed of light in that medium (typically 2/3 of the speed of light in vacuum for fiber and coaxial cable media).

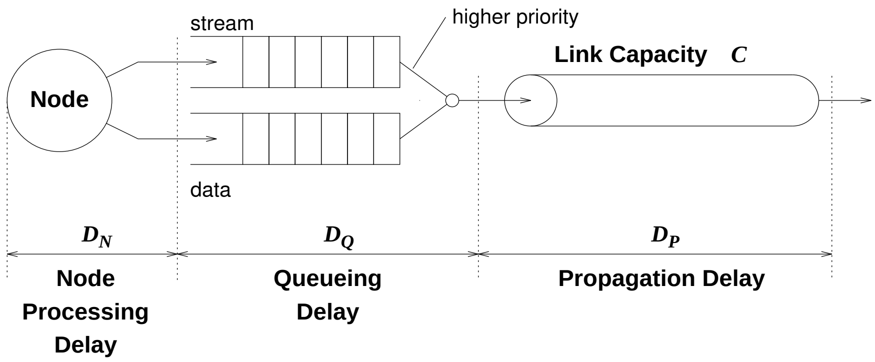


Figure 2.1: The components of the link delay

In this thesis, we consider that the link delay D is constant and independent of the flow in the link.

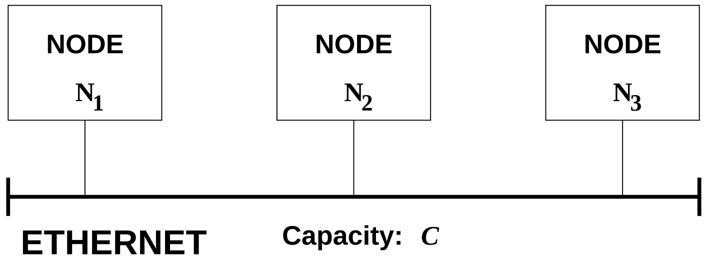
Formally, a network topology with N nodes and K links is described by:

Topology Matrix: Denoted by $\boldsymbol{A}$, is an $N \times K$ matrix where element (i, j) is 1 if node i is the origin of link j, -1 if node i is the destination of node j, and 0 if link j is not connected to node i.

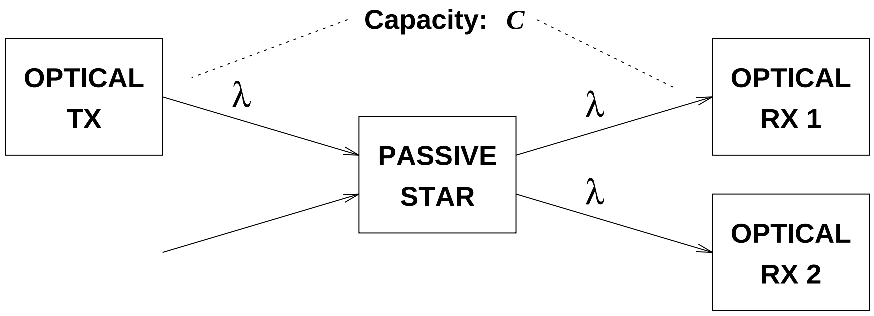
Link Parameters Vector: Denoted by $\boldsymbol{W}$, it is a vector with K triplets (V, C, D), where (V_i, C_i, D_i) are the capacity, cost and delay of link i.

We denote the network described by $\boldsymbol{A}$ and $\boldsymbol{W}$ by $G(\boldsymbol{A}, \boldsymbol{W})$.

The formulation presented here can also be used to describe network topologies which are capable of physical broadcast (shared-channel networks) or multicast. Two examples of such networks are shown in Figures 2.2(a) (a shared-medium network, such as an Ethernet, where a transmission from a node is heard by all other nodes, providing physical broadcast) and 2.2(b) (an optical network where two or more receivers can be tuned to the same wavelength λ, providing physical multicast).



(a) Bidirectional physical broadcast

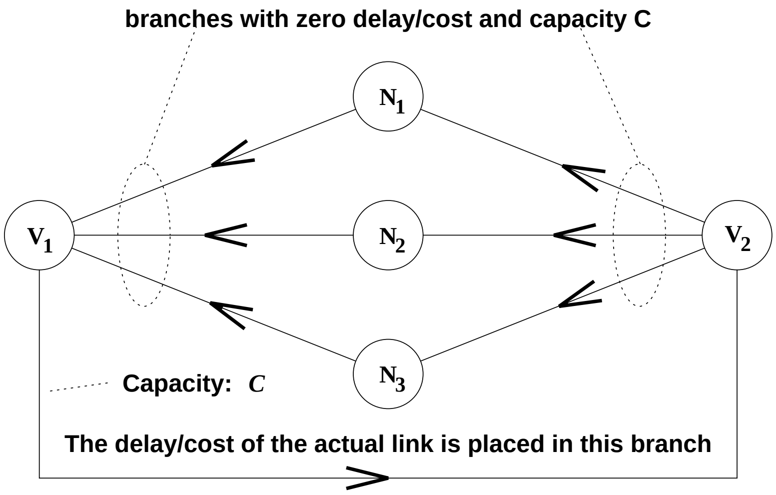


(b) Unidirectional physical multicast

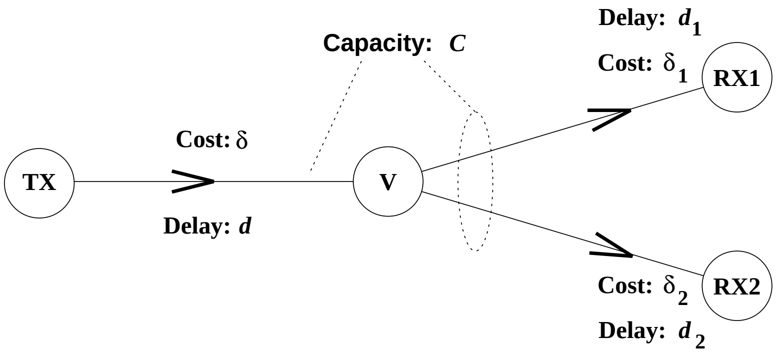
Figure 2.2: Physical multicast

The models for the networks of Figures 2.2 (a) and (b) are shown in Figure 2.3. For the bidirectional (broadcast) case (Figure 2.3(a)), two "virtual nodes" V_1 and V_2 are created. The access from the nodes to V_1 and from V_2 to the nodes is performed at zero cost and zero delay. The interconnection between V_1 and V_2 represents the shared channel, which imposes a limit on the capacity of the actual system. Since the capacity of the channel connecting V_1 to V_2 is C, the sum of the traffic generated by N_1, N_2 and N_3 is limited by C, as it is in the actual network. For the unidirectional (multicast) case (Figure 2.3(b)), only one "virtual node" (V) is needed, and the actual channel is represented by the link between the source and the virtual node. The delay and cost to reach the virtual node correspond to the delay and cost from the optical transmitter to the star. Likewise, the delay and cost from the virtual node to the optical receivers correspond to the delay and cost from the star coupler to the receivers.

Given a directed graph $G(\boldsymbol{A}, \boldsymbol{W})$, we define the following:



(a) Model for the bidirectional physical broadcast



(b) Model for the unidirectional physical multicast

Figure 2.3: Model for the physical multicast

Path: A path from node n_0 to node n_p is a sequence of links

$$P = \{(n_0, n_1), (n_1, n_2), \ldots, (n_{p-1}, n_p)\}$$

where the source node of each link is the same as the destination node of the preceding arc in the sequence, and $n_0, n_1, \ldots, n_p$ are all distinct nodes.

Chain: A chain is similar to a path except that the links are not necessarily directed from n_0 towards n_p.

Circuit: A circuit is a path from some node n_0 to some node n_p plus the link (n_p, n_o); i.e., the circuit is a closed path.

Cycle: A cycle is closed chain.

Weakly Connected Graph: A graph where there is at least one chain between any pair of nodes. This is what is usually known as a *connected* graph.

Strongly Connected Graph: A graph where there is at least one path between any pair of nodes.

Tree: A tree is a connected graph with no cycles.

Spanning Tree: A spanning tree, defined with respect to some graph G, is a connected subgraph of G with no cycles, which includes all nodes in G.

Directed Tree with root at n_o: A tree in which there is a path from some node n_0 (referred to as the *root* of the tree) to all the other nodes in the tree.

Multicast Path from node s to nodes $\{d_1, d_2, \ldots, d_n\}$: Tree formed by merging the paths from s to $d_1, d_2, \ldots, d_n$.

Cost of a Multicast Path: Sum of the costs of the links belonging to the multicast path.

Delay of a Path: Sum of the delays of the links in the path.

Delay of a Multicast Path: Maximum of the delays of the paths that compose the multicast path.

Note that for an undirected graph (i.e., a graph composed only of undirected or full-duplex links), there is no distinction between path and chain, circuit and cycle, and strongly connected/weakly connected. Moreover, any tree can be considered a directed tree with root at any of its nodes.

2.2.2 The Traffic Model

Multicast streams are offered to the network in *sessions*. A session is a group of multicasts which are logically related. One example is a video-conference, where each of the participants can see all other conferees; if there are P participants, a video-conference session would be composed of P multicasts, one from each of the participants to the other $P-1$ conferees. We will denote by T the number of multicasts in the session; each of these multicasts is characterized by:

Addressing Parameters: Source s_i and n_i destinations, denoted by $\{d_{i1}, \ldots, d_{in_i}\}$, $i = 1, \ldots, T$.

Bandwidth Requirement: The amount of bandwidth needed to carry the stream, in bits/second; it will be denoted by r_i, $i = 1, \ldots, T$.

Latency Constraint: The maximum delay acceptable between the source and any of the destinations in this multicast; it will be denoted by L_i, $i = 1, \ldots, T$.

2.2.3 Statement of the Problem

Formally, the multicast stream routing problem can be stated as: "Given the network and a session composed of T multicast streams, with multicast i, $i = 1, \ldots, T$, being characterized by its source s_i, its set of n_i destinations $\{d_{i1}, \ldots, d_{in_i}\}$, its maximum delay constraint L_i and its bandwidth requirement r_i, find a multicast path for each stream that satisfies its bandwidth and delay constraints, while minimizing a given linear combination of the costs and delays of the multicast paths."

2.3 Existing Routing Algorithms

As discussed in Chapter 1, current multicast routing algorithms have the following limitations: (i) they are able to route only a single multicast at a time; (ii) most do not directly take into account the bandwidth and latency constraints of the streams; and (iii) the structure of the algorithm defines the optimization criterion, which is either cost or delay. To compute the routes for a multiple-multicast session, these algorithms have to be applied sequentially to each multicast in the session, in some order (and the routes found will be function of the order used). For each multicast in the session, in order to take the bandwidth requirements into account, the network topology must be temporarily pruned of the links not having enough free bandwidth to support the stream, prior to routing it. Finally, after the route has been computed, its delay has to be checked against the latency constraint; if the constraint is not satisfied, the algorithm fails.

In this section, we describe the existing routing algorithms. We do not limit ourselves to algorithms implemented in current networks; we denote by *existing* not only the algorithms implemented in operational networks, but also the algorithms available in the literature.

Given the stream requirements (source, destination(s), bandwidth, latency, etc.), the routing algorithm is responsible for finding a route that satisfies these requirements. An objective function is defined to distinguish between multiple routes satisfying the requirements; for example, the routing algorithm might select the route with minimum delay, if multiple routes are available.

We use the following two independent criteria to classify the existing routing algorithms; both criteria have to do with the amount of information a routing algorithm is able to use when finding the routes:

- according to the *number of destinations routed simultaneously*:
  - Unicast routing algorithms, i.e., algorithms that can find a route between a source and a single destination.
  - Multicast routing algorithms, i.e., algorithms that can find routes between a source and multiple destinations.
- according to the *number of streams routed simultaneously*:
  - Single-stream routing algorithms, i.e., algorithms that can only route a single stream.
  - Multiple-stream routing algorithms, i.e., algorithms that can route multiple streams simultaneously.

Obviously, multicast routes can be found by executing a unicast routing algorithm multiple times, once for each destination, and merging the routes; a single-stream routing algorithm can also be executed multiple times to route a session with multiple streams. However, a *true* multicast routing algorithm can use the available knowledge of all the destinations it has to reach to potentially find a "better" route. A similar observation can be made for the multiple-stream algorithms.

2.3.1 Single-Stream Routing Algorithms

Without loss of generality, we can state that single-stream routing algorithms can always perform routing from a topological point of view, without needing to directly take into account the bandwidth of the stream. The reason is that bandwidth requirements can be trivially taken into account by pruning from the network graph any link whose available bandwidth is less than the bandwidth required by the stream. In this pruned graph, routing can happen without regard for the bandwidth.

2.3.1.1 Unicast Routing Algorithms

The most common single-stream unicast routing algorithms are the Shortest Path Algorithms. Each link is assigned a label, and the "length" of a path is the sum of the labels of the links in the path. Given a source and a destination, the algorithm will find the path with the smallest length between them. If the link labels are the link delays, the shortest path will be the minimum-delay path. If the link labels are the monetary costs of using the links, the shortest path will be the minimum-cost path. Three well-known algorithms to compute the shortest path [9] are the algorithms by Bellman-Ford, Dijkstra and Floyd.

Another issue with multimedia streams is one of latency. As indicated above, interactivity places a maximum latency constraint in the route. If the shortest path algorithm is executed with the link delays as link labels, the path found will be the minimum-delay route; if it does not satisfy the latency constraints, then these constraints cannot be satisfied. On the other hand, if one uses the link costs as labels for the shortest-path algorithm, the routes found are minimum-cost routes, without regard for latency. Ideally, what is needed is an algorithm that minimizes the cost subject to a latency constraint. One way to solve this problem is to compute the k shortest paths from the source to the destination (in cost), and, starting with the shortest, identify the first one to satisfy the latency constraint. This algorithm, however, can potentially take a long time to run (i.e., k can become very large, depending on the network). Another alternative is to use linear programming; this problem is a special case of the formulation presented later in this chapter.

2.3.1.2 Multicast Routing Algorithms

From an algorithmic point of view, in unicast routing there is no difference between minimizing cost or delay: in both cases, one just uses the shortest-path algorithm, with costs or delays as link labels. However, when it comes to multicast routing, minimizing delays and costs are problems that require different algorithms, because costs and delays are computed in a completely different way. In other words, the delays from the source to each of the destinations is computed independently, but the cost of a tree is an overall measure. In Chapter 1, we presented an example of this difference (Figure 1.8). In this section, we discuss in more detail the algorithms for minimum-delay and minimum-cost multicast routing.

Minimum-Delay Multicast Routing

Given a multicast tree, one can compute the delays from the source to each of the destinations. Since interactivity requirements place an upper bound in the delay from the source to any of the destinations, we define the *delay of the multicast tree* to be the maximum of the delays between the source and the destinations using that tree. The shortest-path algorithms presented in the previous section can be used to find minimum-delay multicast routes, by using the link labels as the link delays, computing the shortest path to each of the destinations, and merging these paths to form the multicast tree. This is an exact solution because the problem is decomposable - the delays are independent of each other. Note that this solution achieves the minimum delay to *all* the destinations, not only to the farthest.

Minimum-Cost Multicast Routing

The problem of finding the minimum-cost multicast tree given the source and the destinations corresponds to the well-known *Steiner Tree problem in graphs*, which is known to be NP-complete [26]. Many exact solutions (with exponential worst-case run times) and heuristics have been proposed to address this problem; see [25] for a comprehensive survey of the field. For exact solutions, mention should be made to the elegant algorithm proposed by Dreyfus and Wagner [27], the branch-and-bound solution by Shore et al [28], and the linear programming formulation by Beasley [29]. The most important heuristic solutions are the algorithms by Kou, Markowsky and Berman [30] (which we will refer to as KMB), Rayward-Smith [31] (which we will refer to as RS) and Takahashi and Matsuyama [32] (TM). The TM heuristics are based on Prim's minimum-cost spanning tree algorithm [9], while the RS heuristic is based on Kruskal's minimum-cost spanning tree algorithm [9]. A detailed description and a comparison of the KMB, TM and RS heuristics can be found in [31].

There is a difficulty in applying minimum-cost Steiner tree algorithms to multicast routing: since they are intended to solve *connectivity* problems, they assume *undirected* links. Given a graph composed of undirected links and a subset of its nodes, the algorithm will find the subgraph with lowest cost that still provides connectivity (i.e., there is at least one path in the subgraph between any pair of nodes in the subset under consideration). Because the links are undirected, this minimum-cost subgraph will be a tree[1]. However, when routing multicast streams, there is a very well-defined direction of flow, from the source to each of the destinations. Moreover, even if a link is physically full-duplex, the bandwidth in both directions is managed independently, so it has to be treated as a pair of directed links.

There are heuristics specifically designed for directed graphs (for example, see [25, 33, 34]); they are, however, much more complex than their undirected counterparts. An alternative to using these heuristics for multicast routing is to modify one of the well-known undirected heuristics to take into account the directionality of the links. While it is not clear how this would be done to the RS heuristic, it is relatively simple to modify the KMB and TM heuristics to operate in directed graphs. In the next section, we describe a modification to the KMB algorithm, which makes it capable of operating in directed graphs.

[1]If the links are directed, the solution to the minimum-cost connectivity problem is in general a forest of trees.

2.3.1.3 The Modified KMB Algorithm

As pointed out in the previous section, the KMB minimum-cost heuristic was proposed for undirected graphs. We describe here a modification to the KMB heuristic to make it capable of computing multicast trees in directed graphs.

We first introduce a trivial modification to Prim's algorithm [9] to find a minimum-weight directed spanning tree, and then use it to obtain the modified KMB algorithm.

The Minimum Weight Directed Spanning Tree

PROBLEM: Given a directed graph $G(\boldsymbol{A}, \boldsymbol{W})$, where the link weights $\boldsymbol{W}$ are composed only of the link costs C, and a source node s_0, find the minimum-cost spanning tree T_M rooted at s_0, composed of paths from s_0 to all other nodes.

ALGORITHM:

Step 1: Initially, the tree T_M is empty. Add the node s_0 to T_M.

Step 2: Between all the links whose *source* is a node in T_M and whose *destination* is a node that is not in T_M, select the one with the smallest cost. Add this link and its destination node to T_M.

Step 3: If all nodes in the graph have been added to T_M, stop; otherwise, return to step 2.

The Modified KMB Algorithm

PROBLEM: Given a directed graph $G(\boldsymbol{A}, \boldsymbol{W})$, where the link weights $\boldsymbol{W}$ are composed only of the link costs C, a source node s and a set of n destination nodes $d_1, d_2, \ldots, d_n$, find the minimum cost directed subtree T_H, rooted at s, and composed of paths from s to $d_1, d_2, \ldots, d_n$.

ALGORITHM:

Step 1: Build an auxiliary directed graph G_1 as follows: the nodes in G_1 are s and $d_1, d_2, \ldots, d_n$. For every pair of nodes (n_i, n_j) in G_1, add a directed link in G_1

from n_i to n_j whose cost is equal to the cost of the shortest path (in cost) in the original graph G. Note that it is not necessary to build the links from $d_1, \ldots, d_n$ to s.

Step 2: Using the Minimum Weight Directed Spanning Tree algorithm described above, find the minimum spanning tree T_1 of G_1 with root in s.

Step 3: Construct a subgraph G_2 of G by replacing each link in T_1 by its corresponding shortest path in G.

Step 4: Find the minimum weight directed spanning tree T_2 of G_2 with root in s.

Step 5: Prune from T_2 any leaf nodes that are not in the $d_1, \ldots, d_n$ set; the resulting tree is the minimum cost T_H.

2.3.1.4 Using Single-Stream Algorithms to Route Multiple Streams

Single-stream algorithms can be applied sequentially to route a multiple-stream session. At each step, the network topology must be pruned of the links with insufficient free bandwidth to carry the stream being routed. In this section, we present a formal description of the complete algorithm, which is illustrated in Figure 2.4.

PROBLEM: Given a multicast session composed of T multicast streams, each stream having its source, set of destinations and bandwidth and latency requirements, and a network described by $G(\boldsymbol{A}, \boldsymbol{W})$, find multicast paths for each of the streams satisfying their requirements, optimizing a certain objective function.

ALGORITHM:

Step 1: Create a vector $\boldsymbol{U}$ to hold the current usage of each link. Initially set $U_j = 0, \quad j = 1, \ldots, K$. Make a list of not-yet-routed streams, initially containing all requests in the session.

Step 2: Take one request from the list of not-yet-routed streams; let us denote it by request i. For routing this stream, temporarily remove from the network topology all links in which $V_j < r_i + U_j, \quad j = 1, \ldots, K$ (i.e., all links that do not have enough free bandwidth to support this multicast).

Step 3: Route this request using the topology created in step 2, using the single-request routing algorithm (i.e., shortest path, minimum cost, etc).

Step 4: If the routing in step 3 was successful[2], and if the delay of the multicast satisfies the latency constraint, update the $\boldsymbol{U}$ vector as follows:

$$U_j \leftarrow U_j - r_i \qquad j \in \text{multicast path for stream } i$$

Otherwise, if no route was found or the route found does not satisfy this multicast's latency constraint, terminate; there is no solution to the routing problem.

Step 5: Remove request i from the list of not-yet-routed streams and record its route. If all streams have been processed, terminate; routes for all the components of the session have been found. Otherwise, return to step 2.

2.3.2 Multiple-Stream Routing Algorithms

When a number of streams have to be simultaneously routed, the bandwidth requirements cannot be taken into account by simply pruning the network graph of the links with insufficient bandwidth. Unless the available bandwidth in each link is greater or equal to the sum of the bandwidths requested by all the streams, the problem of optimally routing N streams *cannot* be divided into N single-stream routing problems - the link capacity constraints create dependencies between the single stream routing problems. One heuristic solution is to route the streams sequentially in a given order, at each step subtracting the bandwidth used by the previous routes from the available link bandwidth. An optimum algorithm, however, must consider all the streams simultaneously.

[2]The routing may fail if the network becomes disconnected when removing links in step 2.

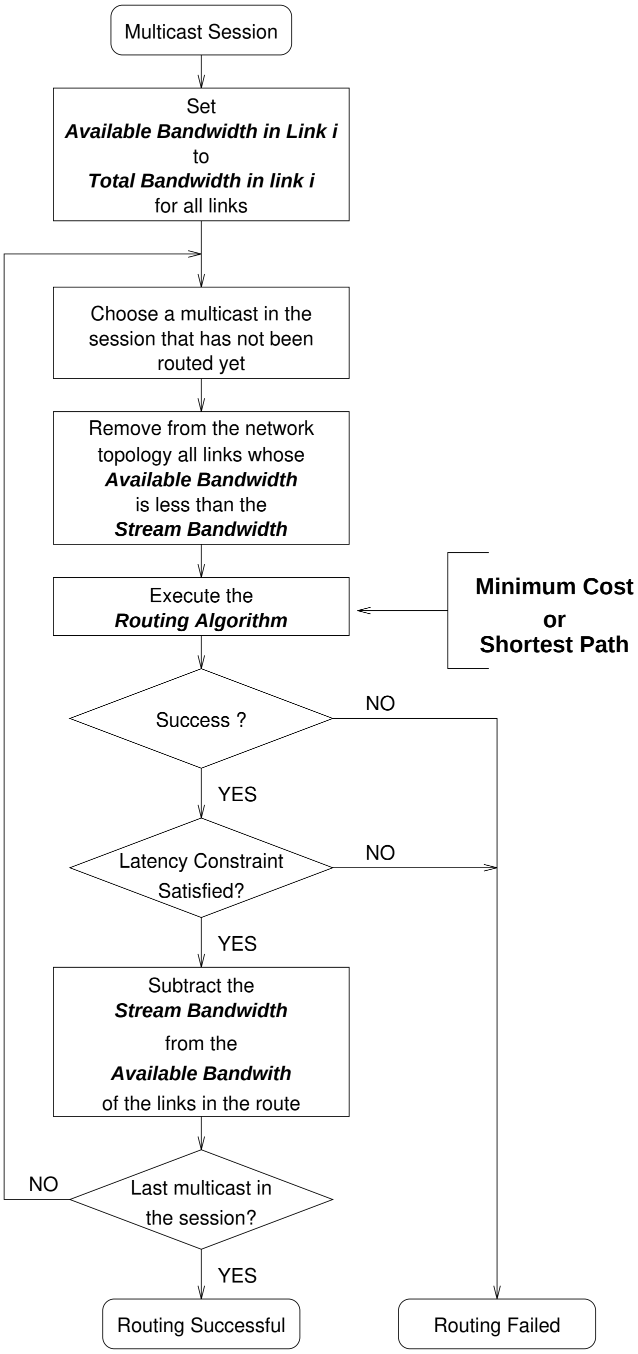


Figure 2.4: General Multicast Stream Routing Algorithm

2.3.2.1 Unicast Routing Algorithms

The problem of routing N unicast streams, each characterized by a source s_i, a destination d_i and a bandwidth requirement r_i in a network $G(\boldsymbol{A}, \boldsymbol{V})$, while minimizing the average path length (which can be either the average delay or the cost, depending on the link labels used in the optimization), corresponds to the well-known multicommodity flow problem. If the link labels are independent of the flow in the link, the multicommodity flow problem can be written as a linear programming problem, which can be solved efficiently by decomposition [17]. The problem of routing N streams is decomposed into N single-stream routing subproblems, and the solution algorithm iterates between a master problem and these subproblems until the optimum is found. Note that each of the single-stream routing subproblems corresponds to a shortest-path problem, with link labels set iteratively by the master problem. The shortest-path algorithms listed in section 2.3.1.1, however, cannot be used to solve them because of the possibility of loops with negative total length; they have to be solved by linear programming methods such as network simplex.

One important difference between the single- and multiple-stream routing algorithms is that the optimum solution to the latter can involve flow bifurcation, i.e., the route for a given stream might include several paths between the source and the destination, each carrying a fraction of the flow. A packet-switched network can support flow bifurcation by sending some of the packets generated by the source through each of the multiple routes; however, other considerations such as re-sequencing and synchronization might impose the restriction that all the flow belonging to a certain stream use a single route. In this case, a set of *integer constraints* is added to the linear-programming problem, making it an integer programming problem. Standard techniques such as the branch-and-bound method [35] can be used to solve the integer problem.

2.3.2.2 Multicast Routing Algorithms

There are no multiple-stream multicast routing algorithms in the literature.

2.3.3 Summary of the Existing Algorithms

Table 2.1 summarizes the existing routing algorithms. Existing multicast routing algorithms have the following shortcomings:

- There is no routing algorithm capable of simultaneously routing a number of multicast streams.

Table 2.1: Summary of Existing Routing Algorithms

	Unicast	**Multicast**
Single Stream	Shortest Path	Shortest Path Minimum Cost
Multiple Streams	Multicommodity Flow (Simplex)	None available

- The optimization criterion, in most cases, is defined by the structure of the algorithm (i.e., minimum cost, minimum delay, etc). Kumar and Jaffe [36] proposed a way of trading off cost and delay in a single multicast by performing a minimum-cost route, and then replacing some of the higher-delay routes with shortest-path routes. No algorithm, however, is able to provide a continuous tradeoff between delay and cost.

- Most algorithms do not directly take into account a latency constraint. Minimum-delay algorithms (shortest path), in general, do not need to do that because they try to achieve the minimum possible delay; if they cannot satisfy a latency constraint, it is likely that it cannot be satisfied[3]. On the other hand, minimum-cost algorithms tend to produce higher delays. Kompella et al [37] proposed a variation of the KMB heuristic which is able to find low-cost trees subject to a delay constraint. However, their algorithm is applicable only to undirected graphs, and thus unsuitable for use in a real network where the bandwidth is managed independently in both directions.

2.4 An Integer Programming Formulation for the Optimum Multicast Routing

In this section, we show that the multicast routing problem defined in section 2.2.3 can be written as an integer programming problem, or a linear programming problem if flow

[3]For a single multicast session, if the shortest path cannot satisfy a latency constraint, it really cannot be satisfied; for a multiple multicast session, this is not true in general.

bifurcation is allowed.

Definitions:

N	:	Number of nodes in the network.
K	:	Number of (directed) links in the network.
$\boldsymbol{A}$	:	$N \times K$ network topology matrix; $A_{ij} = 1$ if node i is the source of link j, $A_{ij} = -1$ if node i is the destination of link j, and $A_{ij} = 0$ if link j is not connected to node i.
$\boldsymbol{C}$	:	$1 \times K$ cost vector.
$\boldsymbol{D}$	:	$1 \times K$ delay vector.
$\boldsymbol{V}$	:	$K \times 1$ available capacity vector.
T	:	Number of multicast streams.
s_i	:	Source node for multicast i
n_i	:	Number of destinations for multicast i
$\{d_{ik}\}$	:	Set of destinations for multicast i, $k = 1, \ldots, n_i$
r_i	:	Bandwidth requirement for multicast i
$\boldsymbol{X}^i$	:	$K \times n_i$ multicast routing matrix for multicast stream i. $X^i_{jk} = 1$ if link j is used in the multicast path for stream i to reach destination d_{ik}, otherwise $X^i_{jk} = 0$, $k = 1, \ldots, n_i$.
Y^i	:	$K \times 1$ multicast path vector for stream i. $Y^i_j = 1$ if link j is in the multicast path for stream i, otherwise $Y^i_j = 0$.
M_i	:	Delay for multicast request i.
L_i	:	Latency constraint for multicast request i.
$\boldsymbol{B}^i$	:	$N \times n_i$ source-destination matrix for multicast stream i; $B^i_{jk} = 1$ if $j = s_i$, $B^i_{jk} = -1$ if $j = d_{ik}$, and $B^i_{jk} = 0$ otherwise, $k = 1, \ldots, n_i$.
β_c	:	Weight of the cost in the optimization.
β_d	:	Weight of the delay in the optimization.

The optimum routing problem can be formulated as follows:

GIVEN: $\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{V}, N, K, T, \boldsymbol{B}^i, r_i, \boldsymbol{L}, \beta_c, \beta_d$

MINIMIZE:

$$\sum_{i=1}^{T} r_i \left(\beta_c \boldsymbol{C}\boldsymbol{Y}^i + \beta_d M_i\right) \tag{2.1}$$

WITH RESPECT TO: $\boldsymbol{X}^i, \boldsymbol{Y}^i, M_i, \quad i = 1, \ldots, T$

UNDER CONSTRAINTS:

1. For every stream, there must be a path from its source to each of its destinations. This is equivalent to writing a set of flow conservation equations for routing one unit of flow from the source to each of the destinations:

$$\boldsymbol{A}\boldsymbol{X}^i = \boldsymbol{B}^i \quad i = 1, \ldots, T; \tag{2.2}$$

2. If a link is in the path from the source to any of the destinations, then it must be included in the multicast path.

$$X^i_{jk} \leq Y^i_j, \quad k = 1, \ldots, n_i, \quad j = 1, \ldots, K, \quad i = 1, \ldots, T; \tag{2.3}$$

3. The delay for a multicast is the delay to the farthest destination:

$$M_i - \sum_{j=1}^{K} D_j X^i_{jk} \geq 0, \quad k = 1, \ldots, n_i, \quad i = 1, \ldots, T; \tag{2.4}$$

4. There is a maximum delay constraint for each of the multicast streams:

$$M_i \leq L_i, \quad i = 1, \ldots, T; \tag{2.5}$$

5. The total flow through a link cannot exceed its bandwidth:

$$\sum_{i=1}^{T} r_i \boldsymbol{Y}^i \leq \boldsymbol{V}; \tag{2.6}$$

6. No bifurcation of stream flow of data; a single path is taken from the source to each of the destinations.

$$\boldsymbol{X}, \boldsymbol{Y} \quad \text{are binary.} \tag{2.7}$$

Constraints (2.1) to (2.6) define a linear programming problem, which can be solved by the simplex method. When constraint (2.7) is included, the problem becomes an integer programming problem[4]. It should be noted that, if the session is composed only of unicast streams and if the delay constraints are absent, this problem reduces to the traditional multicommodity flow problem.

Different choices of β_c and β_d in equation (2.1) lead to different optimizations. In this thesis, we consider the following cases:

- $\beta_d = 0, \beta_c > 0$: Cost minimization.
- $\beta_c \gg \beta_d > 0$: Cost minimization, with delay as a secondary objective.
- $\beta_d \gg \beta_c > 0$: Delay minimization, with cost as a secondary objective.

The case $\beta_c = 0, \beta_d > 0$, which would correspond to delay minimization without regard to cost, is not considered because: (i) for a single multicast, this problem can be solved optimally using the shortest path algorithm; and (ii) constraints (2.2) to (2.7) do not guarantee that there will be no loops in the route[5]; when $\beta_c = 0$, additional equations would have to be added to the constraint set to avoid routing loops (or the final solution would have to be pruned, using, for example, Prim's algorithm). Therefore, there is no advantage in using this objective function. On the other hand, for single-multicast sessions, the multicast path found by setting $\beta_d \gg \beta_c > 0$ (minimize delay, with cost as a secondary objective) will have the same delay as a multicast path found using the shortest path algorithm. However, this only means that the routes used in both cases to reach the farthest destination will be the same (or have the same delay). The shortest path algorithm will also minimize the delay to the other destinations; the integer programming solution, on the other hand, is able to make use of the fact that the only constraint on the routes to the other destinations is that their delay should not exceed the delay to the farthest destination to further minimize the cost of the multicast, while still achieving the minimum delay.

[4]In a packet-switched network, it is conceivable that bifurcation of flow could be allowed. In this case, constraint (2.7) does not apply, and the problem is no longer NP-complete.

[5]This is not needed when $\beta_c > 0$ because the cost minimization will guarantee that loops do not exist.

It should be noted that this integer programming formulation solves the optimum *static* routing problem; in other words, given the network and the session, it will find the best routes (as defined by the objective function) without any regard for the future sessions; the success of the routing computation (i.e., the feasibility of the problem) is independent both of the particular objective function used and of the costs and delays of the links. In a dynamic scenario, where sessions arrive, are routed (or blocked), exist in the network for a certain time (if accepted), and terminate, the routing algorithm is executed for each session arriving, and decides whether it can be accepted or not, and if accepted, which routes to use. The optimum routing algorithm described here takes the best decision for each session, but there is **no** guarantee that the sequence of decisions is optimal in any sense. A true optimum routing algorithm for this scenario should try to optimize also the sequence of decisions, based on the network topology and on the traffic statistics, which is a much more complex problem.

2.5 Solution of the Optimum Multicast Routing Problem

In this section, we present an efficient solution technique for the integer programming problem presented in section 2.4. The technique is based on the well-known branch-and-bound method [35], which has two phases: (i) the linear relaxation (where the integer constraints are relaxed, and the problem is solved as a linear problem) and (ii) the branch and bound phase, where the values of variables with integer constraints are fixed either to zero or to one. In this section, we present enhancements to speed-up both phases.

2.5.1 Solution to the Linear Relaxation

In this section, we provide a solution to the linear relaxation of the integer programming problem presented in section 2.4. In this phase we disregard the integer constraints and solve the problem defined by equations (2.1) to (2.6), which generally yield non-integer X^i_{jk}.

Although the linear relaxation could be solved using the traditional simplex method, it is possible to apply an extension of the decomposition procedure [17] to obtain a solution more efficiently. In other words, it is possible to decompose the problem of routing T multicast streams into T single-multicast routing problems. Moreover, each of the single-multicast routing problems can be further decomposed into n_i unicast routing problems, which can be efficiently solved by methods such as network simplex [17]. In Appendix A we show the extension to the decomposition equations presented in [17] used in the solution of the optimum multicast routing problem.

First Decomposition

In this section, we show the decomposition of the problem of routing T multicast streams into T single-multicast routing subproblems. To accomplish this, we observe that equations (2.2), (2.3), (2.4) and (2.5) apply to each multicast in isolation, while equation (2.6) is the only connection between flows belonging to different multicasts. Rewriting those equations, and using positive slack variables to turn inequalities into equalities, we find:

GENERAL CONSTRAINTS (valid for $i = 1, \ldots, T$):

$$\begin{aligned}
\boldsymbol{A}\boldsymbol{X}^i &= \boldsymbol{B}^i \\
X^i_{jk} - Y^i_j + Z^i_{jk} &= 0, \quad k = 1, \ldots, n_i, \quad j = 1, \ldots, K \\
Y^i_j + S_{Yj} &= 1, \quad j = 1, \ldots, K \\
\sum_{j=1}^{K} D_j X^i_{jk} - M_i + S^i_{ak} &= 0, \quad k = 1, \ldots, n_i \\
M_i + S_{Li} &= L_i
\end{aligned}$$

(Z^i_{jk}, S_{Y_j}, S^i_{ak} and S_{L_i} are positive slack variables)

COMPLICATING CONSTRAINTS:

$$\sum_{i=1}^{T} r_i \boldsymbol{Y}^i + \boldsymbol{S} = \boldsymbol{V}$$

Let $\boldsymbol{I}_K$ denote the $K \times K$ identity matrix; we define:

$$
\mathcal{X}_i = \left[\begin{array}{c}
\boldsymbol{X}_1^i \\
\boldsymbol{X}_2^i \\
\vdots \\
\boldsymbol{X}_{n_i}^i \\ \hline
\boldsymbol{Y}^i \\ \hline
M_i \\ \hline
\boldsymbol{Z}_1^i \\
\vdots \\
\boldsymbol{Z}_{ni}^i \\ \hline
\boldsymbol{S}_Y \\ \hline
\boldsymbol{S}_a^i \\ \hline
S_{Li}
\end{array}\right] \tag{2.8}
$$

$$
\mathcal{A}_i = \left[\begin{array}{cccc|c|c|c|c|c|c}
\boldsymbol{A} & 0 & \cdots & 0 & & & & & & \\
0 & \boldsymbol{A} & \cdots & 0 & & & & & & \\
\vdots & \vdots & & \vdots & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \cdots & \boldsymbol{A} & & & & & & \\ \hline
\boldsymbol{I}_K & 0 & \cdots & 0 & -\boldsymbol{I}_K & & & & & \\
0 & \boldsymbol{I}_K & \cdots & 0 & -\boldsymbol{I}_K & & & & & \\
\vdots & \vdots & & \vdots & \vdots & 0 & \boldsymbol{I}_{Kn_i} & 0 & 0 & 0 \\
0 & 0 & \cdots & \boldsymbol{I}_K & -\boldsymbol{I}_K & & & & & \\ \hline
\multicolumn{4}{c|}{0} & \boldsymbol{I}_K & 0 & 0 & \boldsymbol{I}_K & 0 & 0 \\ \hline
\boldsymbol{D} & 0 & \cdots & 0 & & -1 & & & & \\
0 & \boldsymbol{D} & \cdots & 0 & & -1 & & & & \\
\vdots & \vdots & & \vdots & 0 & \vdots & 0 & 0 & \boldsymbol{I}_{n_i} & 0 \\
0 & 0 & \cdots & \boldsymbol{D} & & -1 & & & & \\ \hline
\multicolumn{4}{c|}{0} & 0 & 1 & 0 & 0 & 0 & 1
\end{array}\right] \tag{2.9}
$$

$$\mathcal{B}_i = \left[\begin{array}{c} \boldsymbol{B}^i_1 \\ \boldsymbol{B}^i_2 \\ \vdots \\ \boldsymbol{B}^i_{n_i} \\ \hline 0 \\ \hline \mathbf{1} \\ \hline 0 \\ \hline L_i \end{array}\right] \tag{2.10}$$

$$\mathcal{C}_i = \left[\begin{array}{c|c|c|c|c|c|c} 0 & r_i\beta_c\boldsymbol{C} & r_i\beta_d & 0 & 0 & 0 & 0 \end{array}\right] \tag{2.11}$$

$$\mathcal{D}_i = \left[\begin{array}{c|c|c|c|c|c|c} 0 & r_i\boldsymbol{I}_K & 0 & 0 & 0 & 0 & 0 \end{array}\right] \tag{2.12}$$

We can re-write the multicast optimization problem as:

MINIMIZE:

$$\sum_{i=1}^{T} \mathcal{C}_i \mathcal{X}_i$$

WITH RESPECT TO:

$$\mathcal{X}_i$$

SUBJECT TO:

$$\begin{aligned} \mathcal{A}_i \mathcal{X}_i &= \mathcal{B}_i \\ \sum_{i=1}^{T} \mathcal{D}_i \mathcal{X}_i + \boldsymbol{S} &= \boldsymbol{V} \end{aligned}$$

which is the same formulation as the one presented in equations (A.1), (A.2) and (A.3) in Appendix A, with $\boldsymbol{E} = 0$ and $\boldsymbol{F} = 0$. The condition $0 \leq \boldsymbol{\mathcal{X}}_i \leq \boldsymbol{U}_i$ is satisfied because $\boldsymbol{X}^i$, $\boldsymbol{Y}^i$ and $\boldsymbol{Z}^i$ are binary, and M_i and $\boldsymbol{S}_a$ are limited by the maximum delay in the network. Each of the subproblems corresponds exactly to the problem of routing a single multicast request over an empty network, with arbitrary link weights.

Second Decomposition

In the previous section, we showed how the general multicast problem can be decomposed into T subproblems, each one corresponding to one multicast request, and a master problem. In this section, we will look into these multicast subproblems in detail, and show how they further decompose into n_i unicast routing problems.

The subproblem to be solved in the first decomposition is:

MAXIMIZE:

$$(\omega \mathcal{D}_i - \mathcal{C}_i)\mathcal{X}_i + \alpha_i \tag{2.13}$$

WITH RESPECT TO:

$$\omega$$

SUBJECT TO:

$$\mathcal{A}_i \mathcal{X}_i = \mathcal{B}_i \tag{2.14}$$

Introducing (2.8), (2.11) and (2.12) in the objective function (2.13), it becomes:

$$(\omega \mathcal{D}_i - \mathcal{C}_i)\mathcal{X}_i = r_i \left[(\omega - \beta_c \boldsymbol{C}) \boldsymbol{Y}^i - \beta_d \boldsymbol{M} \right] \tag{2.15}$$

We can now rewrite the problem defined by equations (2.14) and (2.15) into the format presented in Appendix A; most of the matrices involved can be identified by inspection from equations (2.8) to (2.12):

$$\overline{\mathcal{X}}_j = \boldsymbol{X}^i_j$$

$$\overline{\mathcal{A}}_j = \boldsymbol{A}$$

$$\overline{\mathcal{B}}_j = \boldsymbol{B}^i_j$$

$$\overline{\mathcal{C}}_j = 0$$

$$\overline{\mathcal{Y}} = \left[\begin{array}{c} \boldsymbol{Y}^i \\ \hline M_i \end{array}\right]$$

$$\overline{\mathcal{E}} = \left[\begin{array}{c|c} -r_i(\boldsymbol{\omega} - \beta_c \boldsymbol{C}) & r_i \beta_d \end{array}\right]$$

$$\overline{\mathcal{D}}_j = \left.\left[\begin{array}{l} 0 \\ \vdots \\ \boldsymbol{I}_K \quad \text{(position } Kj\text{)} \\ \vdots \\ 0 \\ \hline 0 \\ \hline 0 \\ \vdots \\ D_i \quad \text{(row } n_iK+K+j\text{)} \\ \vdots \\ 0 \\ \hline 0 \end{array}\right]\right\} n_iK + K + n_i + 1$$

$$\overline{\mathcal{F}} = \left.\left[\begin{array}{c|c} -\boldsymbol{I}_K & \\ \vdots & 0 \\ -\boldsymbol{I}_K & \\ \hline \boldsymbol{I}_K & 0 \\ \hline & -1 \\ & -1 \\ 0 & \vdots \\ & -1 \\ \hline 0 & 1 \end{array}\right]\right\} n_iK + K + n_i + 1$$

$$\overline{\mathcal{S}} = \begin{bmatrix} \boldsymbol{Z}_1^i \\ \vdots \\ \boldsymbol{Z}_{n_i}^i \\ \boldsymbol{S}_Y \\ \boldsymbol{S}_a^i \\ S_{Li} \end{bmatrix}$$

$$\overline{\mathcal{V}} = \left[\begin{array}{c} 0_{n_iK\times 1} \\ \hline \mathbf{1}_{K\times 1} \\ \hline 0_{n_i\times 1} \\ \hline L_i \end{array}\right]$$

The problem can be expressed as follows:

MINIMIZE:

$$\sum_{j=1}^{n_i} \overline{\mathcal{C}}_j \overline{\mathcal{X}}_j + \overline{\mathcal{E}\mathcal{Y}} = \overline{\mathcal{E}\mathcal{Y}} \tag{2.16}$$

WITH RESPECT TO:

$$\overline{\mathcal{X}}_j, \overline{\mathcal{Y}}$$

SUBJECT TO:

$$\begin{aligned} \overline{\mathcal{A}}_j \overline{\mathcal{X}}_j &= \overline{\mathcal{B}}_j \qquad j = 1, \cdots, n_i & (2.17) \\ \sum_{j=1}^{n_i} \overline{\mathcal{D}}_j \overline{\mathcal{X}}_j + \overline{\mathcal{F}\mathcal{Y}} + \overline{\mathcal{S}} &= \overline{\mathcal{V}} & (2.18) \end{aligned}$$

Summary of the Decomposition Procedure

We have shown that the problem of routing T multicast streams can be first divided into T subproblems, each one representing the routing of a single multicast request to its n_i destinations. This problem can be further decomposed into n_i unicast routing problems, from the source to each of the destinations of that stream. In other words, the original problem of routing the T multicast streams is decomposed into $\sum_{i=1}^{T} n_i$ unicast routing problems, as illustrated in Figure 2.5. In this section we summarize the full algorithm.

The complete solution algorithm is:

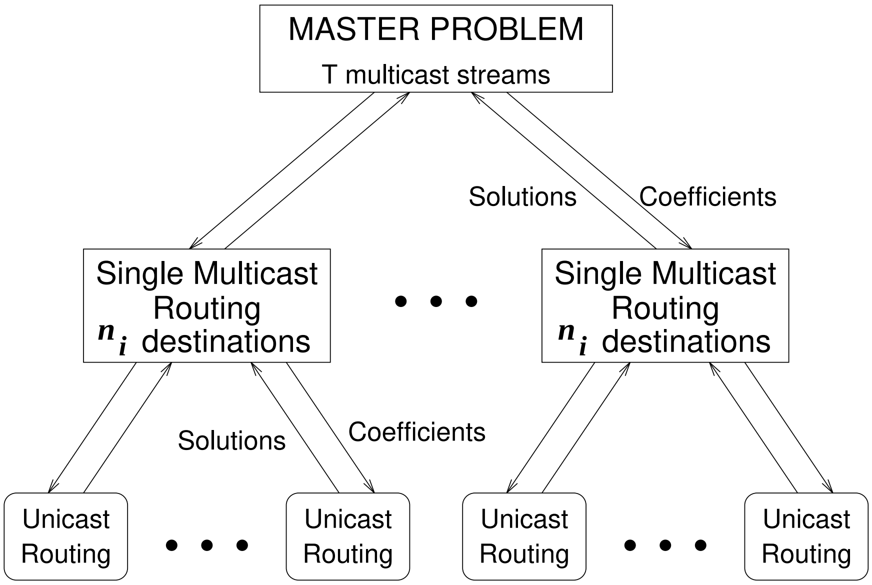


Figure 2.5: Illustration of the decomposition algorithm

INITIALIZATION STEP:

Start with a feasible initial solution. The unicast subproblems can be initialized by just computing the shortest path (or any path) between each source and its destinations; the initial solutions to the multicast subproblems are directly computed from the solution to the unicast subproblems by using equation (2.18). The initial solution to the master problem can be found by the two-phase method [17, 35]. The revised simplex array (see equation A.14) is then built.

MAIN STEP

- Select the entering variable according to one of the rules indicated below. If no variable can enter, then stop - the optimum has been reached. The rules to identify the entering variable are:

  - If $\omega_k > 0$, then S_k can enter the basis, $k = 1, \cdots, K$
  - If $r_i\left[(\boldsymbol{\omega} - \beta_c \boldsymbol{C})\boldsymbol{Y}^i - \beta_d M_i\right] + \alpha_i > 0$, then the λ_{ij} corresponding to this solution can enter the basis. The solution itself is found by applying the decomposition principle again, as follows:

* Select the entering variable according to the rules below. If no entering variable can be found, then the solution is at hand. The entering variable is identified as follows ($[\overline{\boldsymbol{\omega}} \quad \overline{\boldsymbol{\alpha}}]$ are the dual variables for this subproblem):
  · If $\overline{\omega}_k > 0$, then $\overline{\mathcal{S}}_k$ can enter the basis, $k = 1, \ldots, (n_i+1)(K+1)$.
  · If $\overline{\omega}_{n_i*K+k} - \sum_{l=0}^{n_i-1} \overline{\omega}_{l*K+k} + r_i(\omega_k - \beta_c C_k) > 0$ then $\boldsymbol{Y}_k^i$ can enter the basis, $k = 1, \ldots, K$.
  · If $\overline{\omega}_{n_i(K+1)+1} - \sum_{l=1}^{n_i} \overline{\omega}_{l+n_iK} - r_i\beta_d > 0$, then M_i can enter the basis.
  · If $\begin{bmatrix} \overline{\omega}_{1+(j-1)K} + D_1\overline{\omega}_{n_iK+K+j} & \cdots & \overline{\omega}_{jK} + D_K\overline{\omega}_{n_iK+K+j} \end{bmatrix} \boldsymbol{X}_j^i + \overline{\alpha}_j > 0$ then the $\overline{\lambda_{ij}}$ corresponding to this solution can enter the basis, $j = 1, \ldots, n_i$. The solution can be found by using network simplex (see [17]), by maximizing this objective function subject to $\boldsymbol{A}\boldsymbol{X}_j^i = \boldsymbol{B}_j^i$.
* For each entering variable, pivot as follows:
  · If $\overline{\mathcal{S}}_k$ enters the basis, the column to pivot is $\boldsymbol{B}^{-1}\begin{bmatrix} \boldsymbol{e}_k \\ 0 \end{bmatrix}$
  · If $\boldsymbol{Y}_k^i$ enters the basis, the column to pivot is $\boldsymbol{B}^{-1}\left[\begin{array}{c} -\boldsymbol{e}_k \\ \vdots \\ -\boldsymbol{e}_k \\ \hline \boldsymbol{e}_k \\ \hline 0 \end{array}\right]$
  · If M_i enters the basis, the column to pivot is $\boldsymbol{B}^{-1}\begin{bmatrix} 0 & [n_iK\times 1] \\ 0 & [K\times 1] \\ -1 & [n_i\times 1] \\ 1 & [1\times 1] \\ 0 & [n_i\times 1] \end{bmatrix}$

· If $\overline{\lambda_{ij}}$ enters, the column to pivot is $B^{-1}\left[\begin{array}{c} 0 \\ \vdots \\ \boldsymbol{X}_j^i \\ \vdots \\ 0 \\ \hline 0 \\ \hline \left.\begin{array}{c} 0 \\ \vdots \\ \sum_{k=1}^{K} D_k X_{kj}^i \\ \vdots \\ 0 \end{array}\right\} n_i \times 1 \\ \hline 0 \\ \hline \boldsymbol{e}_j \end{array}\right]$ (the summation appears in row $n_i K + K + j$)

- For each entering variable, pivot as follows:
  - If S_k will enter the basis, the column to pivot is $B^{-1}\begin{bmatrix} \boldsymbol{e}_k \\ 0 \end{bmatrix}$
  - If λ_{ij} will enter the basis, then the column to pivot is $B^{-1}\begin{bmatrix} r_i \boldsymbol{Y}^i \\ \boldsymbol{e}_i \end{bmatrix}$, where $\boldsymbol{Y}^i$ corresponds to the extreme point associated with λ_{ij}

2.5.2 Solution to the Integer Programming Problem

The general integer programming problem can be solved by the "branch-and-bound" method. The solution to the linear relaxation, as described in section 2.5.1, is a necessary step to the final solution. In this section, we will describe only the additions to the branch-and-bound method to prune the search space, thus making it run faster.

A phase in the branch-and-bound method consists of setting binary variables either to 0 or to 1. We can use our knowledge of the structure of the problem to further reduce the number of free variables in this phase [38]. The variables with integer constraints are $\boldsymbol{X}$ and

$\boldsymbol{Y}$; however, if $\boldsymbol{X}$ is integer, the constraint set and the objective function will automatically ensure that $\boldsymbol{Y}$ is also integer. Therefore, when fixing values, we can consider only the components of $\boldsymbol{X}$. The following rules can be applied to reduce the number of free variables:

At initialization: Since the links leading to the source of a multicast will never carry any flow for that multicast, one can set:

$$X^i_{jk} = 0 \qquad i = 1, \ldots, T; \quad k = 1, \ldots, n_i; \quad j : A_{s_i j} = -1$$

where s_i is the source of multicast i. By the same token, links originating from multicast destinations cannot be in the path to that destination:

$$X^i_{jk} = 0 \qquad i = 1, \ldots, T; \quad j, k : A_{d_{ik} j} = 1$$

When X^i_{kj} is set to 0: This means that request i, on its way to its k^{th} destination, will not use link j. Denoting by $\mathcal{N}$ the node from which link j originates, and assuming that l links originate from $\mathcal{N}$, if $\mathcal{N}$ is the origin of request i (or if one of the incoming links to $\mathcal{N}$ has been set to 1 as a result of previous value-fixing) and if the X^i_{kj} corresponding to $l-2$ branches out of the remaining $l-1$ branches have been fixed to 0, then the last remaining X^i_{kj} out of $\mathcal{N}$ can be set to 1.

When X^i_{kj} is set to 1: This means that request i, on its way to its k^{th} destination, will use link j. Denoting by $\mathcal{N}$ the node from which link j originates, and assuming that l links originate from $\mathcal{N}$, the X^i_{kj} corresponding to the remaining $l-1$ links must be all set to 0. Set also $Y^i_j = 1$, as per equation (2.3).

Capacity Constraints: At any given step in the value-fixing process, there are some X^i_{jk} that have been set to 1, setting the corresponding Y^i_j also to 1. This means that a certain amount of bandwidth (given by r_i) has been reserved on that link already. Therefore, streams requiring more than the free bandwidth on that link should not use it. Formally:

$$\text{Define:} \quad U_j = V_j - \sum_{i:Y^i_j \text{ is set to 1}} r_i \qquad j = 1, \ldots, K$$

$$\text{If } r_i > U_i, \text{ set } X^i_{jk} = 0, \quad k = 1, \ldots, n_i, \quad i : Y^i_j \text{ is not set}$$

Note that these rules can be applied recursively, i.e., if setting a variable to 1 implies setting another variable to 0 (or to 1), the corresponding rule can also be invoked. Moreover, if a conflict arises (e.g., the rule being used calls for fixing a certain variable to 0 when it is already fixed to 1, or vice-versa), then it is not necessary to solve the linear relaxation for that particular case; it can be immediately marked as infeasible.

2.6 Run Time Evaluation

In this section, we give a numerical evaluation of the average run times of the algorithm presented in this chapter, and compare them with run times for the heuristic solutions as described in section 2.3. The algorithms were implemented in a DEC 5000/240 workstation in C, and compiled with the highest level of optimization available. Each data point reported in this section corresponds to computing a number of routes on an empty network, and averaging the result. In all cases, the 95% confidence interval was also obtained and found to be small relatively to the measured value; it is not plotted for clarity.
The numerical evaluation is organized as follows:

Step 1: Obtain the run times for the linear relaxation of the integer programming problem presented in section 2.4, both for the traditional simplex method and for the decomposition shown in section 2.5.1, to characterize the improvement gained when the original problem is decomposed into subproblems.

Step 2: Obtain the run times for the integer programming problem, with and without the pruning rules of section 2.5.2, to characterize the speed-up when they are employed.

Step 3: Obtain the run times for the heuristic algorithms of section 2.3 (shortest path and minimum cost) and compare with the linear programming approach.

2.6.1 Evaluation Scenarios

The evaluation scenarios have two components: (i) the *network* scenario, and (ii) the *traffic* scenario. This section describes the choices made for these two components in the evaluation. For the network scenario, we chose the following:

Topology: A simplified version of the NSFNet T3 backbone, with 12 nodes and 15 full-duplex links, shown in Figure 2.6. We also considered topologies with 6 nodes and 8 full-duplex links (half of the size of the NSFNet), and 6 nodes and 15 full-duplex links (a completely-connected topology), generated at random.

Link Costs: Link costs have been set to 1 for all topologies.

Link Delays: For the NSFNet, the link delays have been set equal to the propagation delays, as indicated in Figure 2.6, where the link delays (in milliseconds) are shown next to the links. For the 6-node topologies, the link delays were generated at random between 0 and 18 ms, which correspond to the propagation delays if the nodes were spread over an area roughly equivalent to that of the United States.

Link Capacities: All links are assumed to have the same capacity.

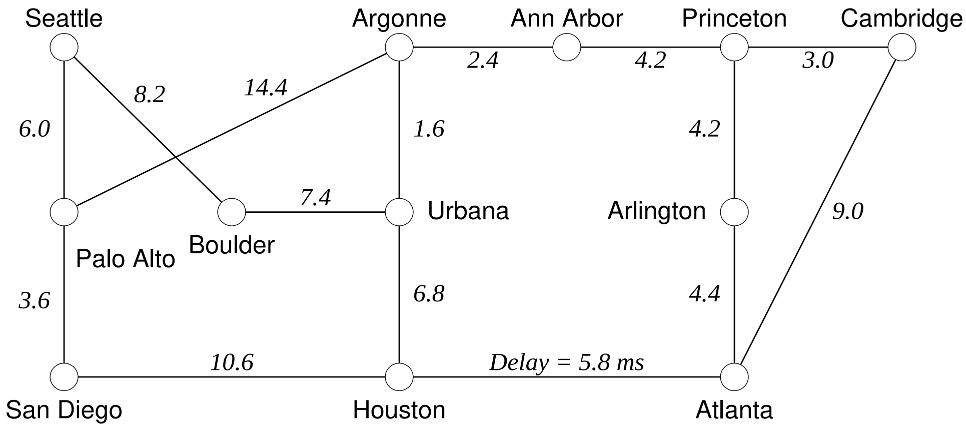


Figure 2.6: The NSFNet T3 backbone (simplified)

For the traffic scenario, we chose the following:

Number of Multicasts per Session: between 1 and 4.

Number of Destinations: variable, although all the multicasts in a given session have the same number of destinations.

Addressing: sources and destinations are chosen at random, uniformly between all the nodes in the network.

Stream bandwidth: the bandwidth requirement for all streams in a given session is constant. In this evaluation, for single-multicast sessions, the bandwidth is irrelevant as long as it is less than the link capacity. For 2-multicast sessions, we set the bandwidth requirement to 60% of the link bandwidth, and for 4-multicast sessions, to 30%. These particular values are chosen so as to make the problem *not* naturally decomposable; i.e., no single link in the network can support all the streams.

Latency Constraint: no latency constraint was imposed.

We considered the following objective functions for the optimum multicast routing algorithm:

- minimum cost ($\beta_c = 1$, $\beta_d = 0$ in equation (2.1)); this solution will be referred to as "Cost" in the discussion;

- minimum cost, with delay as a secondary objective ($\beta_c \gg \beta_d > 0$); this solution will be referred to as "Cost/delay"; and

- minimum delay, with cost as a secondary objective ($\beta_d \gg \beta_c > 0$); this will be referred to as "Delay/cost".

Note that for single-multicast sessions, the multicast path found by the "Delay/Cost" solution will have the same delay as a multicast path found using the shortest path algorithm. However, this only means that the routes used in both cases to reach the *farthest* destination will be the same or equivalent. The shortest path algorithm will also minimize the delay to the other destinations; the integer programming solution, on the other hand, is able to make use of the fact that the only constraint on the routes to the other destinations is that their delay should not exceed the delay to the farthest destination to further minimize the cost of the multicast.

2.6.2 Run Time Evaluation of The Optimum Multicast Routing Algorithm

In this section, we evaluate the speed-up gained by the use of decomposition (as compared to the traditional simplex) and by the enhanced value-fixing rules, when computing the optimum solution for the multicast stream routing problem.

Speed-Up due to Decomposition

The run time for the linear relaxation of the routing problem is a function of its size; more specifically: (i) number of links in the topology; (ii) number of destinations in the multicast; and (iii) number of multicasts in the session.

We applied both the traditional simplex method and the decomposition shown in section 2.5.1 to the scenarios described in section 2.6.1; the general conclusion is that decomposition *significantly* reduces the time needed to compute the optimum. The difference in performance is a function of the size of the problem and the objective function, going all the way from *increase* in run time by a factor of 2 (observed at 6 nodes, 15 links, single multicast, minimum cost) when using decomposition, to an improvement by a factor of 100 or more; for example, to compute the routes for a 4-multicast session in the NSFNet, each multicast having 5 destinations, the traditional simplex method took 542 seconds, while using decomposition the time was reduced to 4.8 seconds.

In Figure 2.7, we plot the run time as a function of the number of destinations in the multicast, for 2-multicast sessions in the NSFNet, using the various objective functions. For this scenario, an improvement of 10 times is observed. Figure 2.7 also shows that the run time for the linear relaxation is largely independent of the objective function when using decomposition. Figure 2.8 shows a case where decomposition actually *increases* the computation time, namely minimizing cost in a 6-node, 15-link network for a single-session multicast. The reason for the increase is that this is a small, strongly-connected network, where the paths are simple to find; therefore, the overhead of the decomposition procedure is not compensated by the reduction in execution time. The figure also shows that if the objective is minimizing delay, then the run time is reduced by using decomposition. For sessions of more than one multicast, decomposition always reduces the run time.

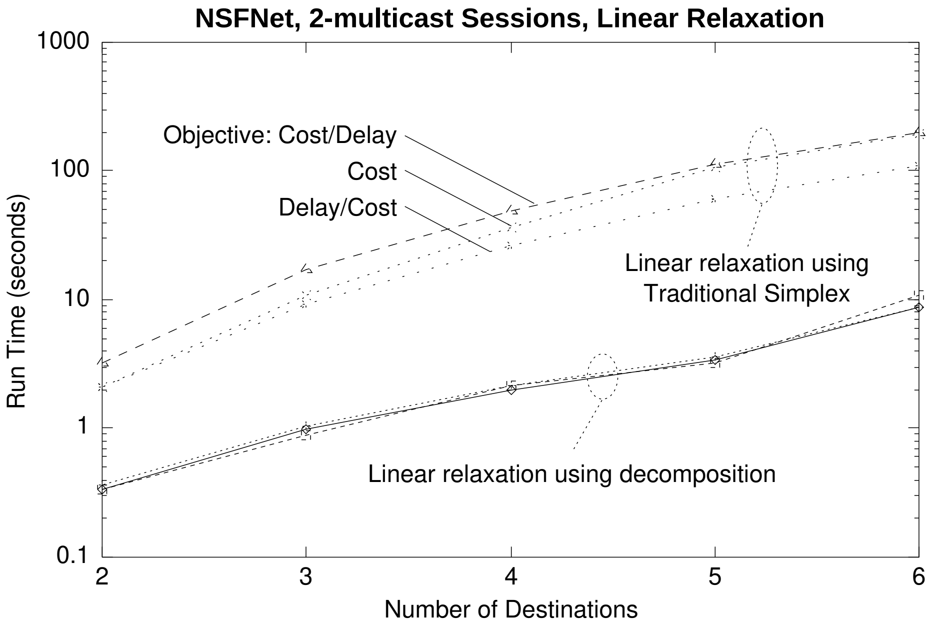


Figure 2.7: Observed run times for the NSFNet topology, 2-multicast sessions, linear relaxation only, 10 routes/point

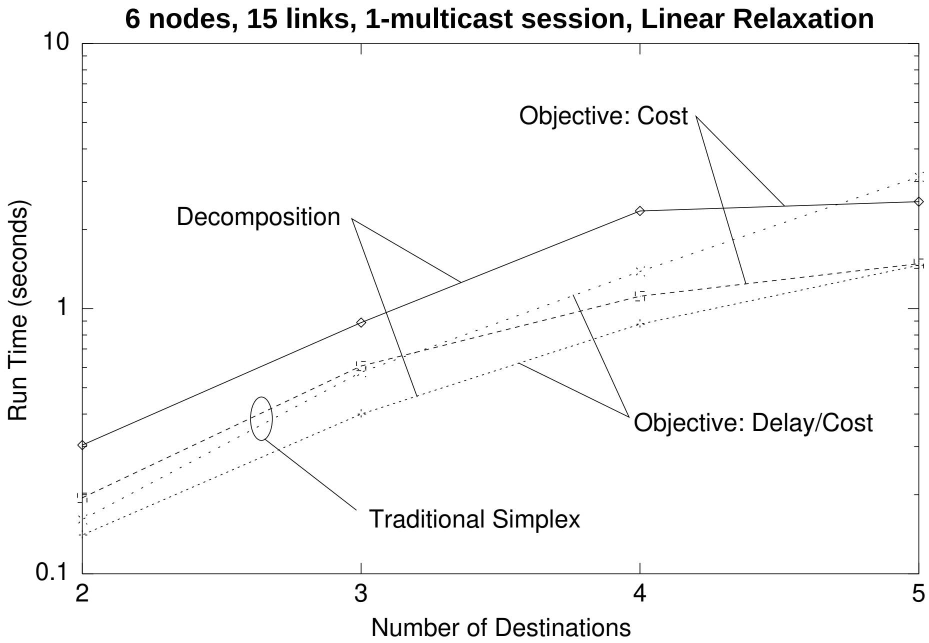


Figure 2.8: Observed run times for a 6-node, 15-link topology, 1-multicast sessions, linear relaxation only, 200 routes/point

Speed-Up due to the Pruning of the Search Space

We evaluated the reduction in the search space of the integer problem resulting from the value-fixing rules described in section 2.5.2. In all cases, the linear relaxations of the original problem were solved using decomposition. The main result is that, in general, the impact of the value-fixing (pruning) rules is much less dramatic than the effect of decomposition. Figure 2.9 shows the run times for 2-multicast sessions in the NSFNet, as a function of the number of destinations. The figure shows that while there is little advantage if the objective is to minimize cost, there is a large advantage (2 to 4 times) if the objective is to minimize delay. Similar comments can be made for the other scenarios.

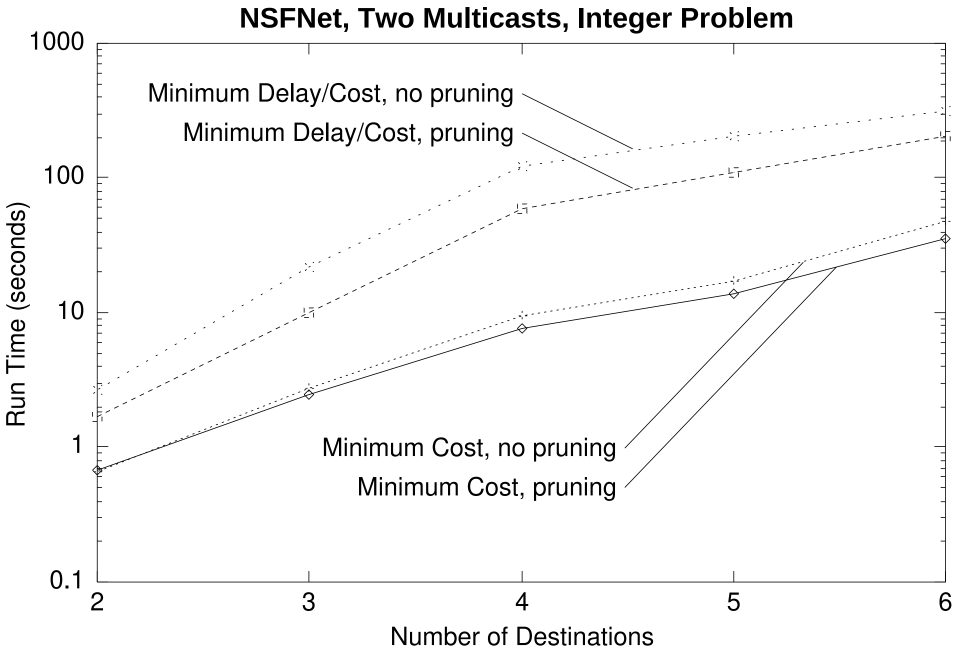


Figure 2.9: Observed run times for the NSFNet topology, 2-multicast sessions, integer problem, 100 routes/point

2.6.3 Run Times for the Heuristic Algorithms

In this section, we evaluate the run times for the minimum-cost and shortest-path algorithms, as described in section 2.3. The general conclusion is that these algorithms run much faster than the optimum presented in this chapter. However, when routing multiple multicast sessions, these algorithms may indicate that the problem has no feasible solution, when in reality it does have a solution. Of course, if a solution exists, the optimum will always find it. The run times for the KMB heuristic and the shortest-path algorithm are shown

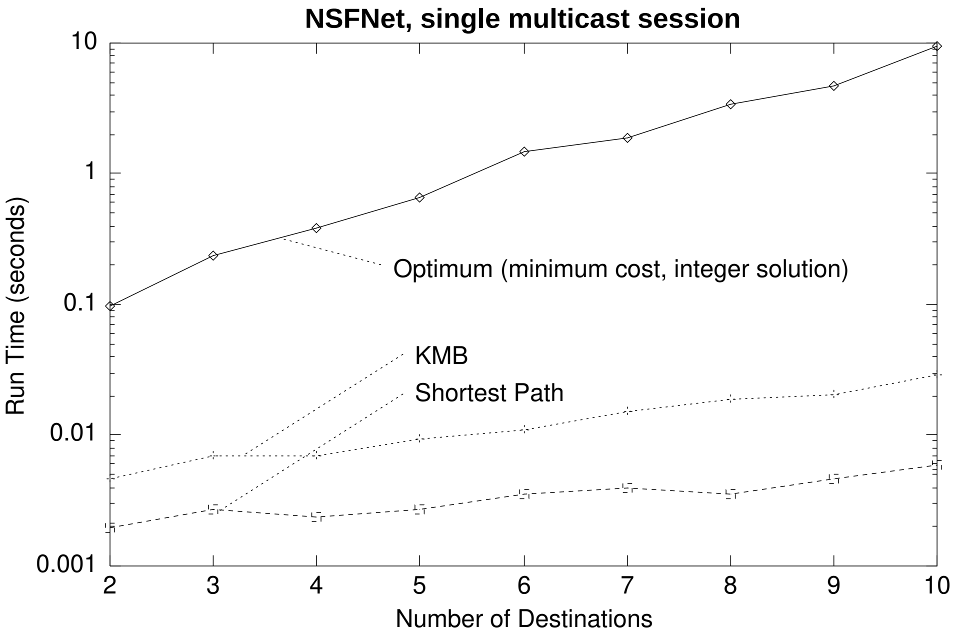


Figure 2.10: Observed run times for the NSFNet topology, single-multicast sessions, 10 routes/point

in Figure 2.10, for single-multicast sessions over the NSFNet; the heuristics are one to two orders of magnitude faster than the optimum.

Table 2.2: Success Probability for a 6-node, 8-link network under 2-multicast sessions

# Destinations	**KMB**	**Shortest Path**	**Optimum**
2	0.95	0.95	1.0
3	0.90	0.80	
4	0.75	0.70	
5	0.60	0.50	

Table 2.2 shows the fraction of successful routes for 2-multicast sessions, routed through a 6-node, 8-link network, for the different methods. In all cases, there was at least one solution, and the optimum algorithm always found it; however, the heuristics were not always able to find the routes. For example, for 5-destination multicasts, the shortest path algorithm failed in 50% of the cases, and the KMB heuristic failed in 40%. However, it should be stressed that these figures are typical of scenarios where the stream bandwidth requirements are a

large fraction of the link bandwidth; if they are not, then the fraction of successful routes for the heuristic algorithms is close to the optimum.

2.7 Conclusions

In this chapter, we showed that the optimum multicast routing problem for multimedia streams can be formulated as an integer programming problem, and proposed an efficient solution technique. We have shown that the proposed techniques vastly reduce the run times when compared with traditional methods. We also compared the run times of the optimum solution with those of well-known heuristic algorithms (modified to operate in the multicast stream environment). The run time for the heuristics is one to two orders of magnitude less than that for the optimum. However, when routing a multiple-multicast session, they might fail to find any solution in some cases where a solution does exist. In the next chapter, we present a complete evaluation of the optimum multicast routing algorithm under realistic conditions, and compare its performance to the heuristic algorithms.

For large networks, the optimum solution presented here may not be practical; its main use is as benchmark for other multicast routing algorithms.

Chapter 3

Performance Evaluation of Multicast Routing Algorithms For Multimedia Streams

3.1 Introduction

The optimum multicast routing problem presented in the previous chapter is known to be NP-complete; thus, the worst case run time of algorithm presented there increases exponentially with the size of the problem. Still, one important use of the optimum algorithm is as a benchmark for other algorithms.

In this chapter, we present a performance evaluation of routing algorithms for multicast streams. Most of the previous work so far in performance evaluation of routing algorithms has focused in computing the cost and/or delay of a single route in an empty network. However, in real networks, multimedia sessions are generated, routed, stay in the network for a period of time, and terminate. The fundamental performance measure in this case is the probability that the session is blocked - i.e., the probability that the routing algorithm cannot find the necessary resources to accept that session. This performance measure cannot be inferred from cost and delay alone. Therefore, in this chapter, we present an evaluation of the existing routing algorithms under this dynamic traffic scenario, and compare them with regard to their blocking probability. Another important factor in the evaluation is the network scenario - it must be "realistic". Routing algorithms should be evaluated over a large number of network topologies. Ideally, the topologies used in the evaluation should correspond to deployed networks; however, the sample space there is rather limited. Therefore, one needs to use randomly-generated topologies, but care must be exercised in making sure that these topologies have the same properties as those of existing networks. Prior to the evaluation, we investigate what properties make a network topology "realistic" for the purposes of this investigation, and limit the evaluation to these topologies.

As a result of the evaluation, we present a number of guidelines on the use of the routing algorithms considered. We also provide guidelines on what is be best way to upgrade the capacity of the network.

This chapter is organize as follows: in section 3.2, we discuss the previous work in evaluating multicast routing algorithms, which has been mostly limited to considering a single multicast in an empty network, where the bandwidth constraints do not come into play. In section 3.3, we present a performance evaluation of the optimum multicast routing algorithm and of the existing algorithms, both from a performance point of view and a cost (measured by the run times of the various algorithms) point of view, under realistic network and traffic scenarios. Finally, in section 3.4, we present our conclusions.

The algorithms evaluated in this chapter are:

Existing Algorithms: Implemented as described in Chapter 2, section 2.3. They can be subdivided into:

- Shortest Path Algorithms: can be used with either the link delays or the link costs as labels. In this chapter, we denote by **SP/delay** the shortest path algorithm with link delays as the labels, and by **SP/cost** with link costs.
- Minimum Cost Algorithms: for the evaluation, we have used the KMB heuristic, modified for directed graphs as described in section 2.3. This algorithm will be denoted by **KMB**.

The Optimum Multicast Routing Algorithm: As pointed out in Chapter 2, the optimum multicast routing algorithm is parameterized by the relative weights of the multicast costs and delays in the objective function. For the evaluation, we chose the following combinations:

- Minimum cost, which will be denoted by **Optimum/cost**.
- Minimum cost, with delay as a secondary objective; will be denoted by **Optimum/cost/delay**.
- Minimum delay, with cost as a secondary objective; denoted by **Optimum/delay/cost**.

3.2 Previous Work in Evaluating Multicast Routing Algorithms

The algorithms described in section 2.3 have been studied in the context of multicast routing by a number of researchers. In most cases, they studied the routing of a single multicast in an empty network, and the performance measures were the multicast costs and delays.

Kumar and Jaffe [36] compared a number of minimum delay and minimum cost algorithms when the link cost and delay weights are the same and derived analytical bounds in cost/delay under that assumption. They also proposed a general algorithm that is able to "trade off" cost and delay, by initially performing a minimum cost routing, and then replacing individual paths with excessive delay with the shortest path; the trade-off is controlled by the number of paths that are replaced. They evaluated numerically costs, delays and run times for routing a single multicast on an empty network, using the algorithms discussed in their paper. For the evaluation, they used both an earlier version of the NSFNet topology, and randomly-generated topologies of different sizes and node degrees[1]. The random topologies were generated by starting with a ring and adding links (uniformly) at random, until the desired degree is reached. The link labels (used both for cost and delay) were set to unity in some cases, and chosen at random between $\{1, 2, 3, 4\}$ in others. Their main conclusions were:

[1]Ratio between the number of links and the number of nodes.

- In general, algorithms that minimize cost take about one order of magnitude more time to run than algorithms that minimize delay.

- The differences in cost/delay between the algorithms evaluated are in the order of 30-40%.

- Results for the NSFNet and for random topologies of the same size are similar.

Waxman [39] studied the problem of dynamically adding and removing destinations on a single multicast that had been already routed in an empty network. He studied the RS (Rayward-Smith) and KMB heuristics, together with a dynamic algorithm he proposed to add the routes to the new destinations joining the multicast. He used randomly-generated network topologies that are supposed to "resemble" actual networks. The algorithm to generate random topologies resembling actual networks proposed by Waxman is based on the observation that, in actual networks, links are more likely to exist between nodes that are located closely together than between nodes that are far apart. To generate the topologies, the nodes are first distributed at random over a rectangular grid. For every pair of nodes (u, v) introduce a link with probability:

$$P(\{u, v\}) = \beta \exp\left[\frac{-d(u, v)}{L\alpha}\right] \tag{3.1}$$

where α and β are parameters in the range $(0, 1]$, $d(u, v)$ is the Euclidean distance between u and v and L is the maximum distance between two nodes. The parameter β controls the degree of the graph, and the parameter α the density of "short" links in relation to "long" links. Note that this method does not guarantee that the network generated is connected; if this property is desired, the final network must be checked for connectivity, and discarded if it fails the test (Waxman makes no mention to this point in his paper). In all cases, the link labels (costs) were set to the distance between the nodes. Waxman found that, on average, the RS and KMB heuristics find solutions which are very close to the optimum minimum cost, and the worst case cost (observed) is about 35% higher than the minimum. When the destination set changes, if one is not allowed to re-route the existing multicast, the cost of the new multicast (using his routing algorithm) can be up to 2-4 times the minimum in the observed worst case.

Doar and Leslie [40] also studied the problem of adding and removing destinations to an existing multicast. They used Waxman's method for generating the network topology, but scaled the link placement probability in equation (3.1) by $k\overline{d}/N$, where N is the number of nodes, $\overline{d}$ is the average degree of a node, and k is an experimental factor found to be about 0.25, to keep the node degree constant as the network size is changed. They evaluated the use of the shortest path algorithm to add the routes to the nodes that have joined the multicast, and compared the resulting cost to the minimum cost for that multicast (obtained by applying the KMB algorithm). They found that the difference in cost can be as large as 2-3 times for networks of 50 nodes.

Leung and Yum [41] proposed a number of minimum-cost heuristic algorithms, and compared their performance with the RS algorithm for routing a single multicast over an empty network. Their algorithms are applicable to directed graphs as well, although this aspect is not explored in their paper. One of their algorithms is actually a variation on the basic TM (Takahashi-Matsuyama) algorithm (and was also given by Chow [42]). They evaluated the algorithms using two networks: one is an early version of the ARPANET and the other is a single random random topology; the main conclusion is that there is very little difference between the cost achieved by the heuristics and the optimum.

Chow [42] explicitly studies the case of single multicasts in directed graphs, and proposes a minimum cost heuristic algorithm for the case where not all nodes are capable of multicasting. He uses the exact minimum cost algorithm proposed by Dreyfus and Wagner [27] to compute the optimum solution, and compares the cost of this solution with the cost obtained by his heuristics. He finds that there is very little difference between the costs of the optimum solution and the heuristic solution, but the optimum takes much more time to run.

Ammar et al [43] studied the minimum cost routing under a variety of additional constraints using a non-linear integer programming formulation. They used a 16-node irregular network, and reported the costs for the multicasts under various scenarios.

Kompella et al [37] studied the case of minimum-cost routing under maximum delay constraints. They assumed both unit and random link costs, random (integer) link delays, bidirectional links, and gave a variation of the KMB heuristic to compute a sub-optimal solution in the case where the latency constraints belong to a discrete set of values. The worst-case run time of their algorithm is exponential. They evaluated their algorithm using random networks; the only constraint imposed in the network topology was that at least one solution to the routing problem under consideration must exist. Their traffic model was a single multicast being routed over an empty network. The main results were: (i) the cost of their heuristic is about 10% higher than the cost of the optimum solution (obtained by exhaustive search); (ii) the cost of using the shortest path algorithm in the same scenarios is about 70-80% higher than the optimum. These results show little dependency with the values of the link costs.

McKinley and Liu [45] considered the problem of routing a single multicast in a network composed of a mesh of buses, where each bus is a shared channel providing broadcast to all nodes connected to it. They propose a number of algorithms for routing in this scenario, and compare their costs (measured in number of buses used to establish a multicast). They also give an exact algorithm when the topology is a regular grid.

Jiang [46] considered the problem of establishing video-conferences. A conference with P participants is established as P multicasts, from each conferee to all other members of the conference. He proposed variations to the KMB and RS algorithms to take into account the link bandwidths. Links are bidirectional, with the bandwidth available in both directions. For the evaluation, he used Waxman's algorithm to generate the network graph, and assigned the capacities of 10, 30 and 100 Mb/s to the links at random, with probabilities 0.6, 0.3 and 0.1 respectively. He assumed that 75% of the capacity of each link was reserved for stream traffic, and that each stream used 3 Mb/s. He reported on the session blocking probability for single video-conference sessions over empty networks, for several different variations of the RS and KMB heuristics.

In summary, most of the previous work in this field has focused on proposing and evaluating the performance of minimum-cost heuristic algorithms for routing a single multicast, usually on bidirectional graphs generated at random (see table 3.1). None of them has evaluated the performance of the routing algorithm in a realistic scenario, where streams are established, exist for a period of time, and terminate. In such a scenario, the blocking probability (i.e., the probability that a session arrives and cannot be routed) is the basic performance measure, instead of just cost or delay.

Table 3.1: Summary of the previous work in evaluation of multicast routing algorithms

Paper	**Objective Function**	**Direction of Flow**	**Number of Requests**	**Algorithms**	**Network Topologies**
Kumar and Jaffe [44, 36]	Cost	Full-Duplex	Single	Steiner tree heuristics; Shortest Path	ARPANET; random
Waxman [39]	Cost	Full-Duplex	Single	Minimum Steiner tree (KMB, RS)	Random
Ammar et al [43]	Cost	Full-Duplex	Single	Heuristic based on simplex	16-node irregular topology
McKinley and Liu [45]	Cost	Full-Duplex	Single	Extension to KMB; set-covering	Mesh of buses; grid of buses
Jiang [46]	Cost	Full-Duplex	Single Conferences	Extensions to KMB and RS	Random
Leung and Yum [41]	Cost	Full-Duplex	Single	Heuristics for the minimum Steiner tree	ARPANET; single random network
Chow [42]	Cost	Half-Duplex	Single	Optimal; Takahashi Matsuyama heuristic	Specific regular topologies
Kompella et al [37]	Cost with Delay Limit	Full-Duplex	Single	Extension to the KMB heuristic	Random
Doar and Leslie [40]	Cost	Full-Duplex	Single	Shortest-Path, KMB	Random

3.3 Performance Evaluation of the algorithms

In this section, we present an evaluation of the various multicast routing algorithms. The evaluation has two parts: (i) a *performance* evaluation, where the results of the different algorithms are compared, and (ii) a *cost* evaluation, where the average run times of the algorithms are characterized with respect to the network size, and compared. We chose to measure "cost" as average run time because it indicates how "expensive", in terms of processing time, it is to find a routing solution. The motivation for this choice is that, if algorithm A finds a solution that is 10% better than that of algorithm B but takes twice the computation time to do it, the better solution might not be worth the added expense. An implementor might choose the worse algorithm, because it might be more cheaply implemented.

The first step is to compare the cost and delay results when routing a single-multicast session in an empty network, as it was done in much of the previous work in the area. The purpose of this step is to compare the cost/delay characteristics of the heuristic algorithms with the optimum, when the cost and delay of each link are independent measures.

The next step in the evaluation is determining the effect of the network topology; it is our objective to evaluate the algorithms under *realistic* conditions. Therefore, we determine what properties make a network "realistic". We then characterize the algorithms in a baseline case, using the "realistic" topologies, and show how variations in this baseline case affect the results. We also investigate how the performance changes as the bandwidth of the network is upgraded, and what is the best way to increase the capacity of a network. Finally, we characterize the cost of the different algorithms, as measured by their average run times. As in the previous chapter, we obtained the results by simulation. The 95% confidence intervals were also obtained and found to be small compared to the measured value.

3.3.1 The Evaluation Scenarios

The evaluation scenarios have two parts: (i) the traffic model, and (ii) the network scenario. In this section, we describe both parts of the scenario.

The Traffic Model

We consider that all multicasts in a session arrive and depart simultaneously. The session arrivals form a Poisson process, with rate λ, and the session duration is exponentially distributed, with rate μ. We assume that sources and destinations are uniformly distributed over the network, and that the set of destinations is fixed for the duration of the session (i.e., no destinations join or leave the multicast while it is in progress). In some cases, we consider the problem of routing a single multicast session in an empty network; this would correspond to a very small λ/μ. The following kinds of sessions are considered:

- *Single-Multicast Sessions:* Each session is composed of a single multicast, whose number of destinations is chosen at random, uniformly between 1 and n_{max}; the value of n_{max} is chosen depending on the number of nodes in the network under evaluation.

- *Videoconference Sessions:* Each session has P multicasts and corresponds to a videoconference with P participants. P is chosen at random between 2 and 4.

We consider that all streams in a session have the same bandwidth requirement; the exact value depends on the particular evaluation scenario being considered. We also assume that blocked sessions are lost, and our primary performance measure is the overall session blocking probability. Given the traffic characteristics, we define the **network capacity** for a certain blocking probability as the load (λ/μ) for which that blocking probability is reached.

The Network Model

The network model is characterized by the following parameters:

Size: Number of nodes (N) and links (K) in the network.

Topology: Interconnection pattern between the nodes and links. All topologies considered in this study are composed of full-duplex links.

Link Parameters: Link costs, delays and capacities.

For this evaluation, we assume that the capacities of all links in the network are equal, and therefore can be normalized to 1. Moreover, we set all the link costs to 1; this way, the cost of a multicast is proportional to its usage of network resources.

For the network topology, we assume: (i) topologies drawn from existing networks, to evaluate the performance of the algorithms under realistic scenarios; and (ii) topologies generated at random, to test the algorithms over a broad range of topologies. For topologies drawn from existing networks, the link delays are set to the propagation delays in the nodes. For random topologies, the nodes are distributed at random over a rectangle, and the link delays are set to the cartesian distance between the endpoints of the link. For this evaluation, we consider the nodes placed in a rectangle with dimensions 15 ms by 10 ms, roughly equivalent to the dimensions of the United States. Additionally, we restrict our attention to random topologies which are strongly connected[2].

We consider the following kinds of random topologies:

Completely Random Topologies: Nodes are interconnected at random.

Random Topologies, Short Links: In "actual" networks, links are more likely to exist between nodes that are "closely located" than between nodes that are "far apart". In this kind of topology, links are more likely to connect nodes that are close.

Two-Connected Topologies: Have at least *two* paths between any pair of nodes. Existing networks are usually two-connected.

The algorithms used to generate these random topologies are described in Appendix B.

3.3.2 Evaluation of Costs and Delays for a Single Multicast

In this section, we evaluate the costs and delays of the various algorithms for a single-multicast session being routed over an empty network. This is what was done in much of the previous work in this field; therefore, we limit ourselves to a single network scenario, and perform the evaluation only to compare the cost/delay results of the optimum multicast routing algorithm with the heuristics.

For the network scenario, we chose the simplified version of the NSFNet T3 backbone used in the previous chapter, and shown in Figure 3.1. The numbers next to the links represent the propagation delays in milliseconds over the link. We set all the link costs to 1, which makes the multicast cost equivalent to the usage of network bandwidth.

Figure 3.2 shows the average cost for a single multicast, in hops, as a function of the number of destinations, for the various multicast routing algorithms. We observe that, as expected (and reported by others), the cost obtained by the KMB algorithm is very close to the optimum. The costs for the routes computed by the algorithms which minimize delay are 0.5 to 1 hop higher than those of the optimum, and the difference increases with increasing

[2] A strongly connected network has at least one path between any pair of nodes.

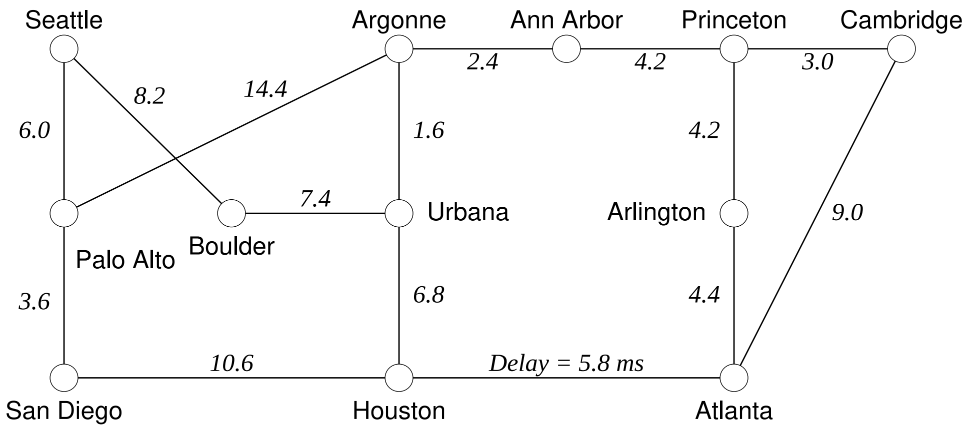


Figure 3.1: The NSFNet T3 backbone (simplified)

number of destinations. Figure 3.3 shows the average delays as a function of the number of destinations in the same scenario; we see that, when one compares the different solutions, the small improvement in cost of the minimum-cost algorithms over the minimum-delay algorithms is "paid for" with a larger (in relative terms) difference in delay. For example, for a 9-destination multicast, the difference in cost between the shortest-path and the KMB algorithms is about 1 hop for a total cost of 9 hops, or about 11%, while the difference in delay is 9 ms for a total delay of 23 ms, or 39%.

It should be stressed that the cost/delay results cannot be directly used to predict the network performance on a dynamic environment, where sessions compete for the resources. In general, all one can say is that "low cost" is a desirable property, because lower-cost routes will use less network resources (if the costs are set correctly) and thus reduce the probability that an incoming session is blocked, but at the price of a higher delay. It is still necessary to numerically assess the effect of the routing algorithms in such environments, in terms of session blocking probability and network capacity, which are the important measures of interest.

3.3.3 Effect of the Network Topology

One of our goals is to evaluate the algorithms under *realistic* network scenarios. Existing networks are usually two-connected, and, as observed by Waxman, links are more likely to exist between "close" nodes than between nodes that are "far apart" (i.e., they are "biased"

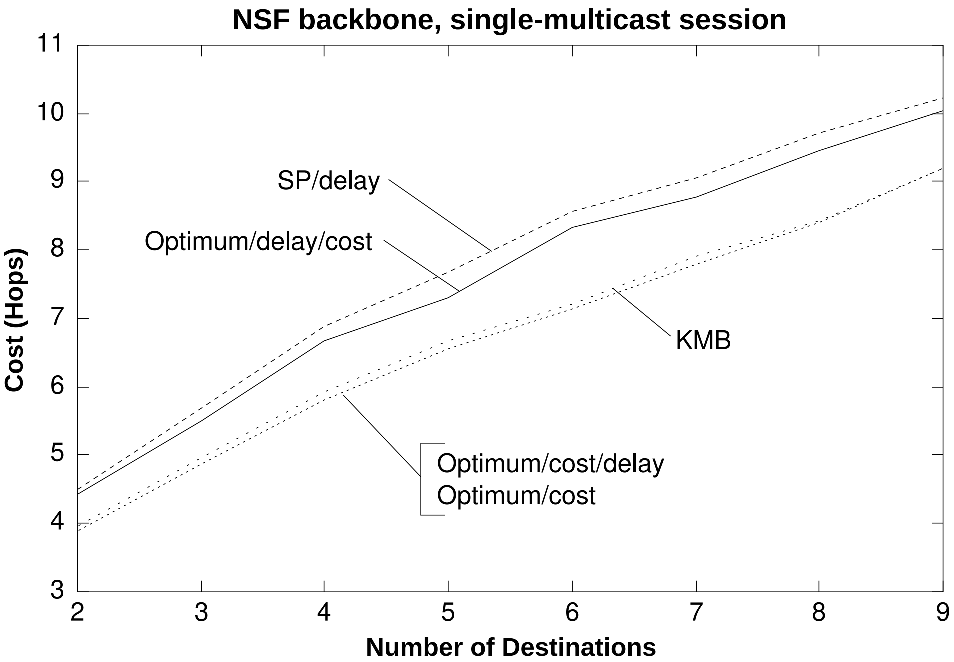


Figure 3.2: Cost of the multicast as a function of the number of destinations in the NSFNet T3 backbone, 100 routes/point

towards short links). In order to evaluate the effect of the kind of topology in the results, we considered first the problem of routing a single multicast session in an empty network. The evaluation scenarios were:

Network: (i) a simplified version of the NSFNet T3 backbone, shown in Figure 3.1; (ii) two-connected topologies, generated at random; (iii) random topologies, biased towards short links; and (iv) completely random topologies. The random topologies are of the same size as the NSFNet (12 nodes, 15 full-duplex links). All the links have the same capacity, the node positions are generated at random in an area similar to that of the United States, and the link delays are set proportional to the distances. All topologies are at least strongly-connected.

Traffic: One multicast session, composed of 5 multicasts, each with 5 destinations. The bandwidth of each stream was generated at random, using a bimodal distribution, with varying average.

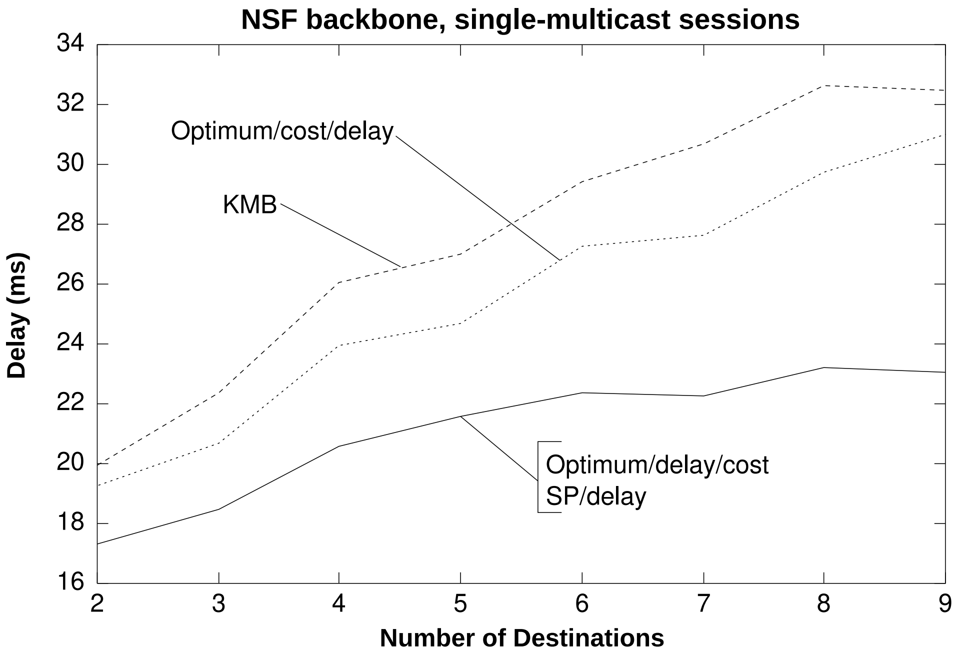


Figure 3.3: Delay of the multicast as a function of the number of destinations in the NSFNet T3 backbone, 100 routes/point

Experiment: Start with an empty network; for a given average stream bandwidth, try to route the session and record the number of cases where routing was successful, as a function of the average stream bandwidth.

Routing Algorithm: Optimum routing algorithm (the objective function is irrelevant, as we are dealing with a single session and recording only the number of successful routes).

The results of this simulation are shown in Figure 3.4, and indicate that the performance is much better for the two-connected networks (both existing - NSFNet - and generated at random). The plot also confirms that what is important when generating a random topology that "resembles" an actual topology is to make it two-connected, and **not** bias it towards short links; the performance of the routing algorithm in the NSFNet and in the random two-connected networks is essentially the same. Two-connected networks have higher performance due to the larger number of independent paths; networks that do not have this property will have a link that, if congested, "divides" the network into two disconnected subnetworks, causing all subsequent sessions with members in both subnetworks to be blocked.

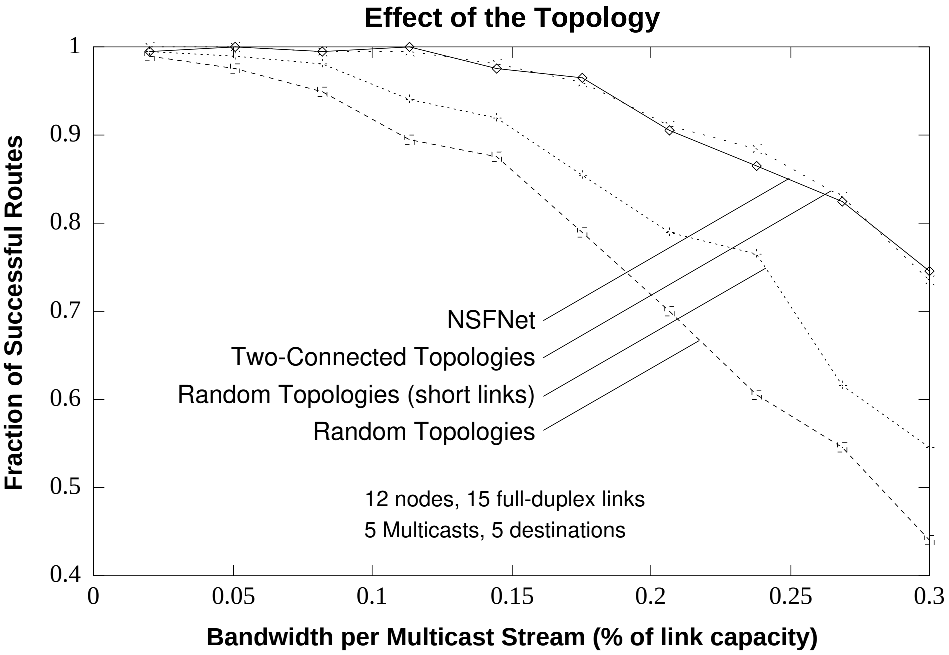


Figure 3.4: Fraction of successful routes for different kinds of topologies, using the optimum routing algorithm, 200 routes/point

In this kind of network, the performance is essentially dictated by the network itself, not by the particular routing algorithm used; our simulations have confirmed this observation. It is interesting to note that Kumar and Jaffe [36] evaluated the cost and delay of several multicast routing algorithms, using both an early version of the ARPANET (a precursor of the NSFNet) and random topologies with the same number of nodes, and seem surprised to find that, for a given routing algorithm, the performance was essentially the same in both scenarios. The ARPANET was designed to be two-connected for reliability purposes; the algorithm used by Kumar and Jaffe to generate the random topologies, described in [44], also generates two connected networks by construction, since it starts with a ring. Our result explains their observation; random topologies and existing topologies of the same size will yield similar results *provided that they are two-connected*; otherwise, the results will be very different, as indicated by Figure 3.4.

To confirm these results in a more general dynamic environment, where sessions come and go, we generated at random ten topologies, all with 12 nodes and 15 full-duplex links, which are shown in Figure 3.5. Half of the topologies were completely random (topologies 1

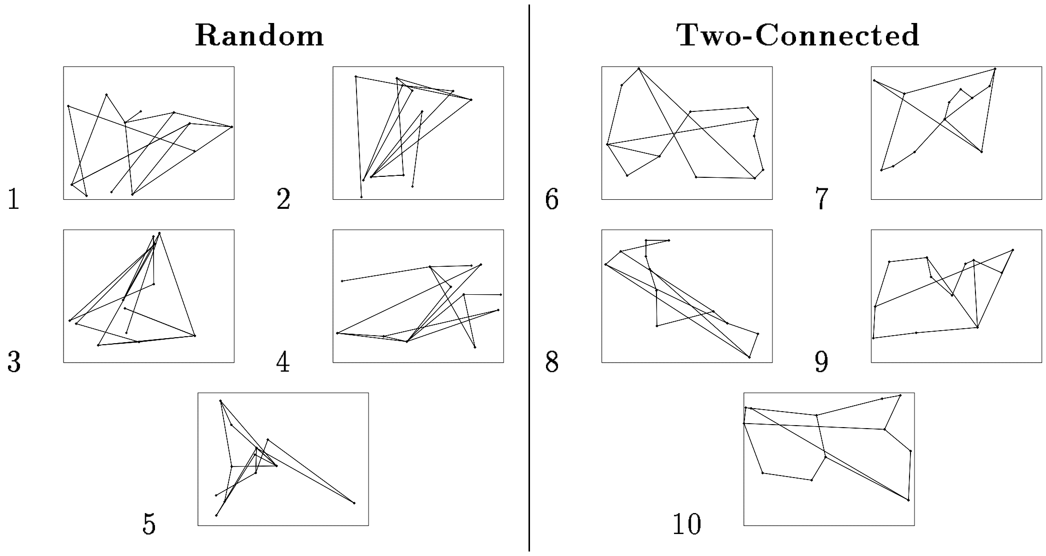


Figure 3.5: Random network topologies used in the evaluation

to 5), and the other half (6 to 10) was composed of two-connected topologies. We obtained the blocking probability in each of the networks, for each of the algorithms. We also repeated the same process for the NSFNet T3 backbone (also with 12 nodes and 15 links), shown in Figure 3.1. The traffic was composed of single-multicast sessions, with a random number of destinations between 1 and 10 and exponential duration and interarrival times. The session blocking probability results for each of the topologies, using the Optimum/cost routing algorithm, are shown in Figure 3.6; similar results were observed for the other algorithms. The main observation is that the blocking probability is much higher in the completely random topologies, and confirms our conclusions from the single-session evaluation.

In the remainder of this chapter, we consider only two-connected networks.

3.3.4 The Baseline Case

In this section, we present a baseline case, and characterize its performance. We then discuss how the results change as the traffic scenario changes from the baseline case. The baseline case corresponds to two-connected topologies, with 12 nodes and 15 links (same size as the NSFNet). Traffic is composed of single-multicast sessions, with a random number of destinations between 1 and 10. The sessions arrive according to a Poisson process, and stay in the network for an exponential amount of time. Each stream requires 10% of the link

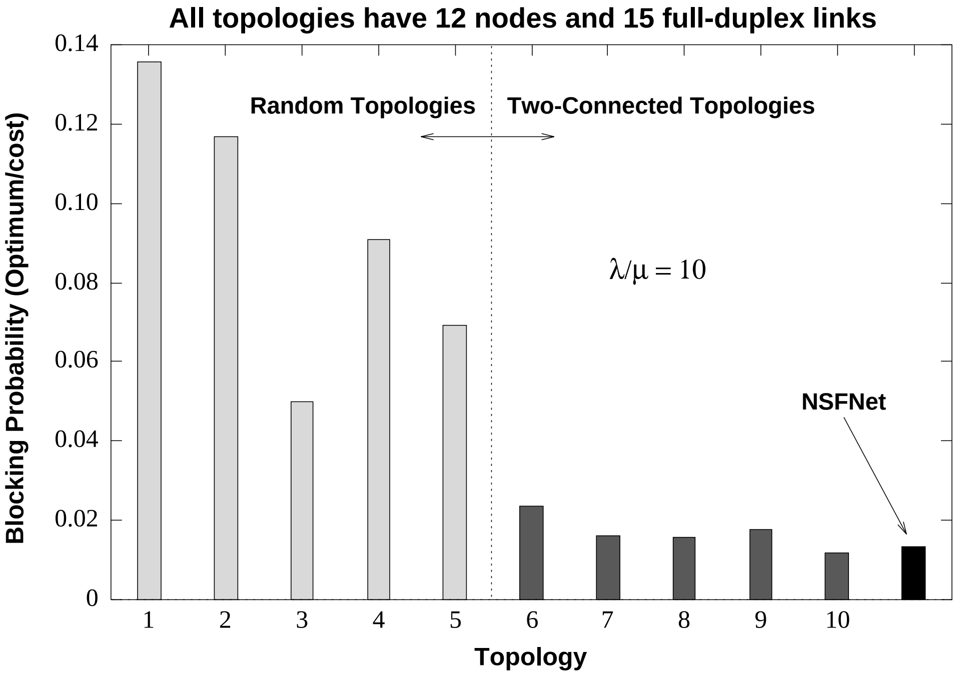


Figure 3.6: Blocking probability in random topologies and two-connected topologies, 5,000 routes/point

bandwidth and there is no latency constraint.

In Figure 3.7 we plot the overall blocking probability, averaged over a large number of two-connected topologies, as a function of the offered load (λ/μ), for all algorithms. The figure indicates that, as expected, the blocking probability for the cost-based algorithms (Optimum/cost, Optimum/cost/delay, KMB) is lower than that for the delay-based algorithms (Optimum/delay, SP/cost, SP/delay). At 1% blocking, the network capacity for this traffic scenario is about 17 for the cost-based algorithms, and 13 for the delay-based algorithms. At 10% blocking, the values are 25 and 22 respectively. We repeated the same runs for the NSFNet T3 backbone and found similar results (with a more marked difference between the two groups of algorithms). The results for the NSFNet are shown in Figure 3.8.

When multicasts with different numbers of destinations co-exist in a network, we expect that the blocking probability be higher for the multicasts with higher number of destinations. In Figure 3.9 we show the blocking probability as a function of the number of destinations, at different load values, for the Optimum/cost algorithm. Figure 3.9 shows that, for low loads, the blocking probability is a weak function of the number of destinations. Even at high

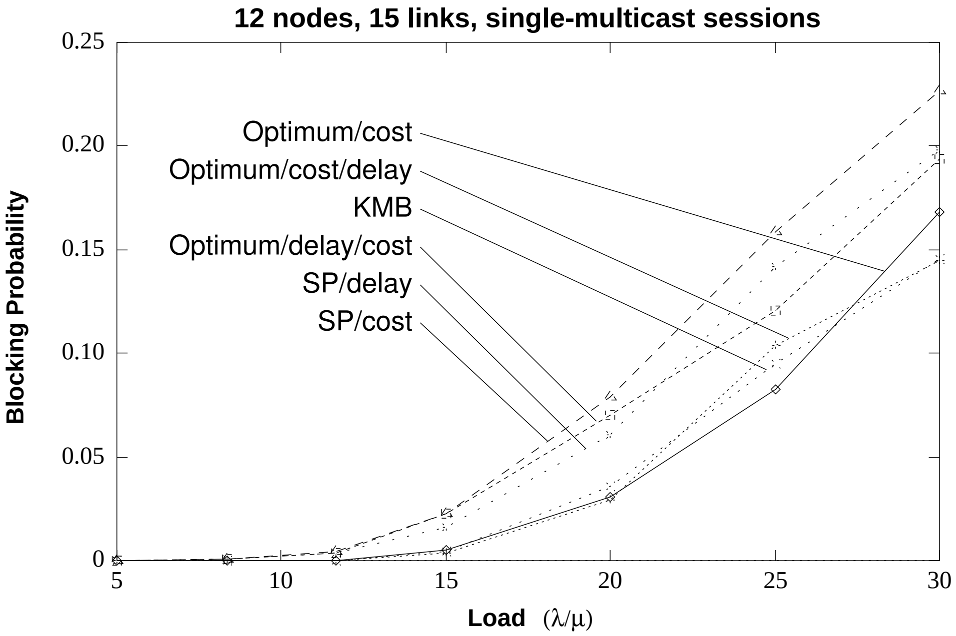


Figure 3.7: Blocking probability for two-connected topologies with 12 nodes and 15 links, single-multicast sessions, 5,000 routes/point

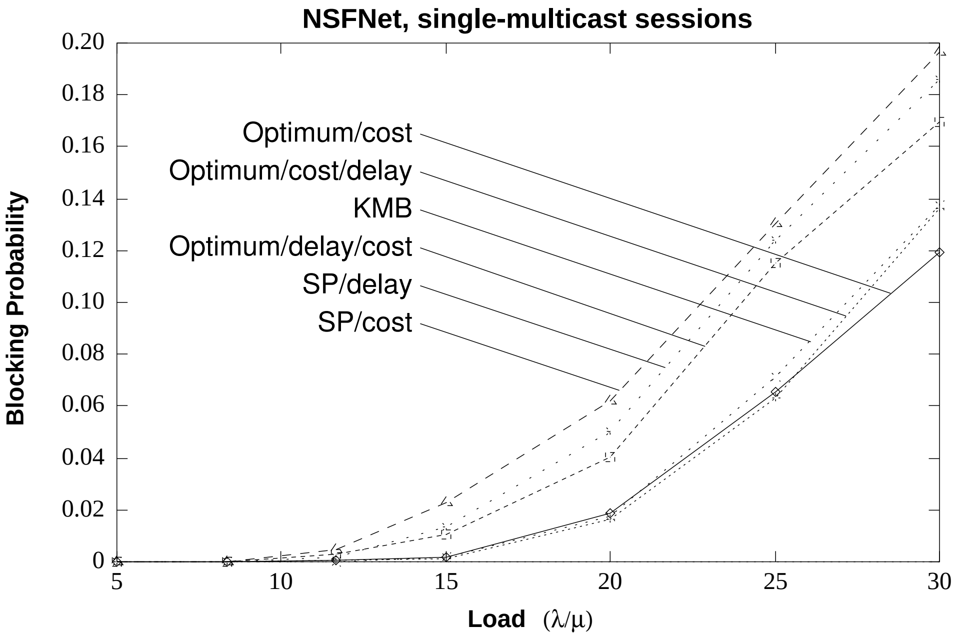


Figure 3.8: Blocking probability for the NSFNet, single-multicast sessions, 5,000 routes/point

loads, the ratio between the blocking probability for 10-destination multicasts and unicasts (one destination) is in the range of 2-3. The plots for the other algorithms are similar.

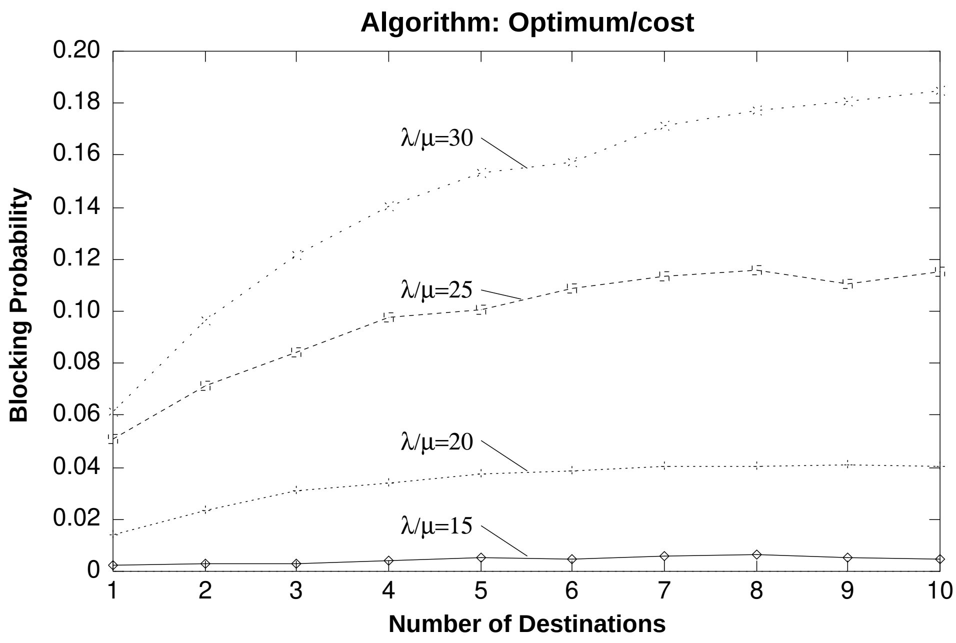


Figure 3.9: Blocking probability as a function of the number of destinations in the multicast, algorithm Optimum/cost, 10,000 routes/point

3.3.5 Introducing a Delay Constraint

The objective of this set of runs was to determine the effect of a constraint in the blocking probability of the various algorithms. For these runs, we chose the NSFNet T3 backbone, under single multicast sessions, with the stream bandwidth fixed at 10%. The diameter[3] of this network in delay is 28 ms (shortest path from Seattle to Arlington).

From the delay histograms for the the runs where no delay constraint was imposed, we observed that there were no successful routes with delay higher than 80 ms. Therefore, any constraint equal to or higher than 80 ms would have no effect in the results. We imposed a constraint of 40 ms, which is a reasonable value considering that what is being discussed here is the component of the delay in the wide-area network; in an audio/video communication, there are other components to be considered, such as the delays due to the encoders/decoders and the delays in the local networks where the sources and destinations are attached [4]. In Figure 3.10 we plot the the blocking probability as a function of the load under the 40 ms constraint. Table 3.2 shows the fraction of the routes in the unconstrained case which would

[3]Maximum shortest path over all pairs of nodes.

not satisfy the 40 ms latency requirement.

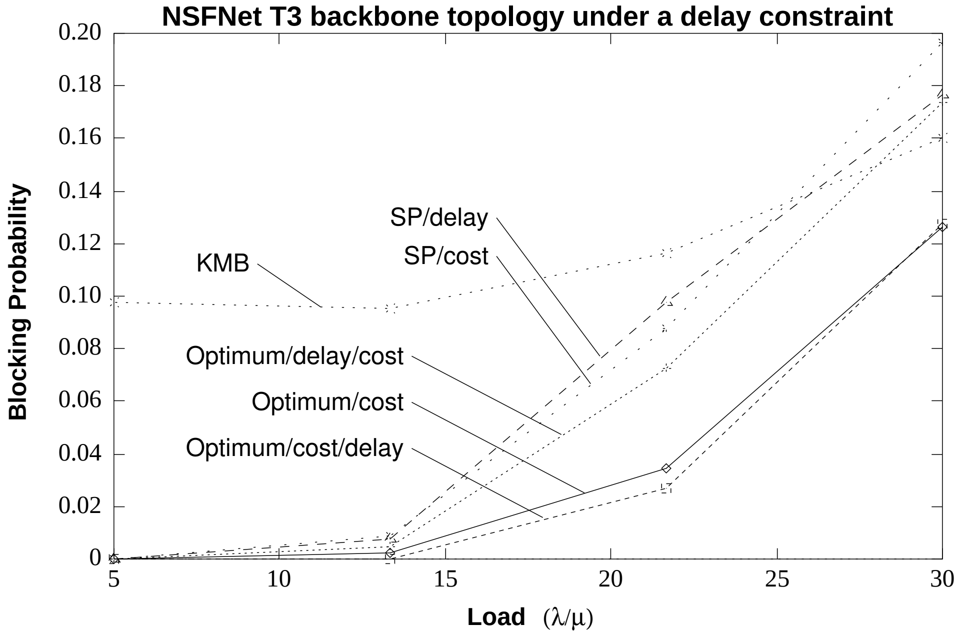


Figure 3.10: Simulation results for the NSFNet T3 backbone, under delay-constrained traffic, 5,000 routes/point

Table 3.2: Fraction of routes that do not satisfy the 40 ms latency in the unconstrained case

Algorithm	**Fraction**
Optimum/cost	16%
KMB	9%
Optimum/cost/delay	2.6%
SP/cost	$\approx$ 0%
Optimum/delay/cost	$\approx$ 0%
SP/delay	$\approx$ 0%

Figure 3.10 indicates that, as hinted by the numbers in table 3.2, the delay-based algorithms are essentially unaffected by this constraint. The optimum algorithms take the delay constraint into account when computing the routes; they are also not affected because they can usually identify alternate routes satisfying the constraint. In the case of the KMB algorithm, however, there is a large effect since its objective is cost, not delay. Even at low loads,

the blocking is high because the algorithm is forced to reject sessions whose routes would exceed the latency constraint. The bigger the number of destinations in the multicast, the larger the effect: while at low load the KMB algorithm is capable to accommodate almost all unicasts under the 40 ms constraint, the blocking probability is over 20% for 10-destination multicasts. This is illustrated in Figure 3.11, where we plot the blocking probability as a function of the number of destinations when $\lambda/\mu = 5$. Under that load, the blocking for all algorithms except the KMB is zero.

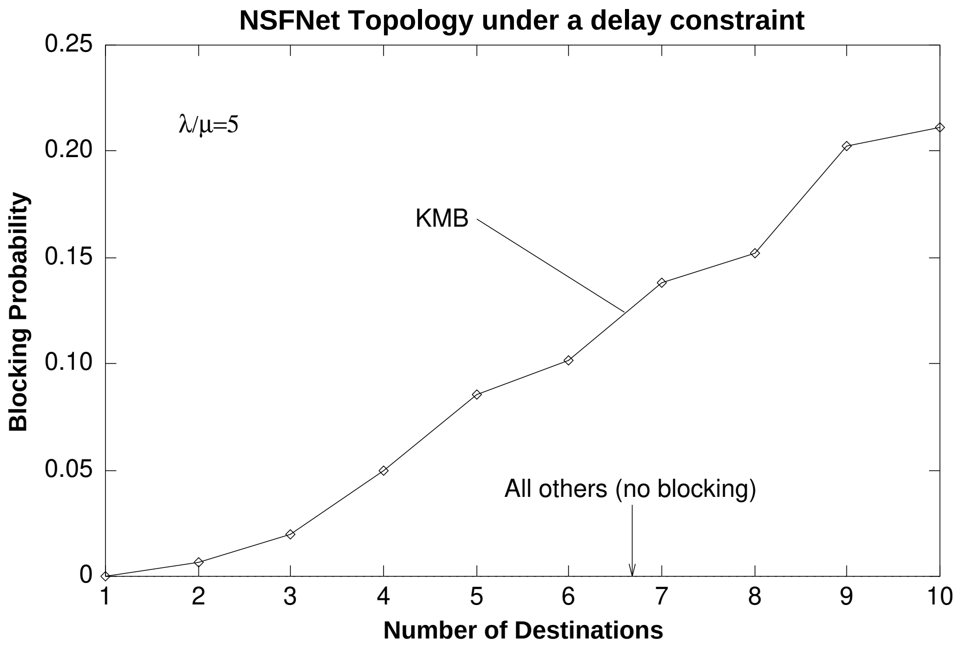


Figure 3.11: Blocking probability as a function of the number of destinations under low load, in the presence of a delay constraint

As the delay constraint is decreased from 40 ms, the blocking probability “plateau” observed at low loads for the KMB algorithm (see Figure 3.10) will significantly increase (a constraint of 30 ms would move it to about 30%, and 20 ms to over 70%). As the constraint is made tighter, the optimum algorithms will tend to use the shortest paths in delay, regardless of their objective functions. Of course, if the constraint is set to a value lower than the network diameter, the blocking will be high for all algorithms.

3.3.6 Upgrading the Network

In this session, we consider the problem of adding bandwidth to the network. We consider networks with a fixed number of nodes and a fixed session arrival rate, and increase the network capacity by adding links to it, and observing the corresponding decrease in the blocking probability.

Figure 3.12 shows the blocking probability for two-connected 6-node networks, when the number of full-duplex links varies from 6 (ring topology) to 15 (fully-connected topology). The figure indicates that the blocking probabilities for the cost-based algorithms (Optimum/cost, Optimum/cost/delay and KMB) are lower than those for the delay-based algorithms (SP/delay, SP/cost and Optimum/delay/cost). The curve representing the relation between the blocking probability and the number of links is concave, and has two distinct regions: (i) the high-blocking region, where an increase in the number of links results in an essentially linear decrease in blocking probability, and (ii) a low-blocking region, where the network is capable of carrying almost all the offered traffic, and addition of links has little effect.

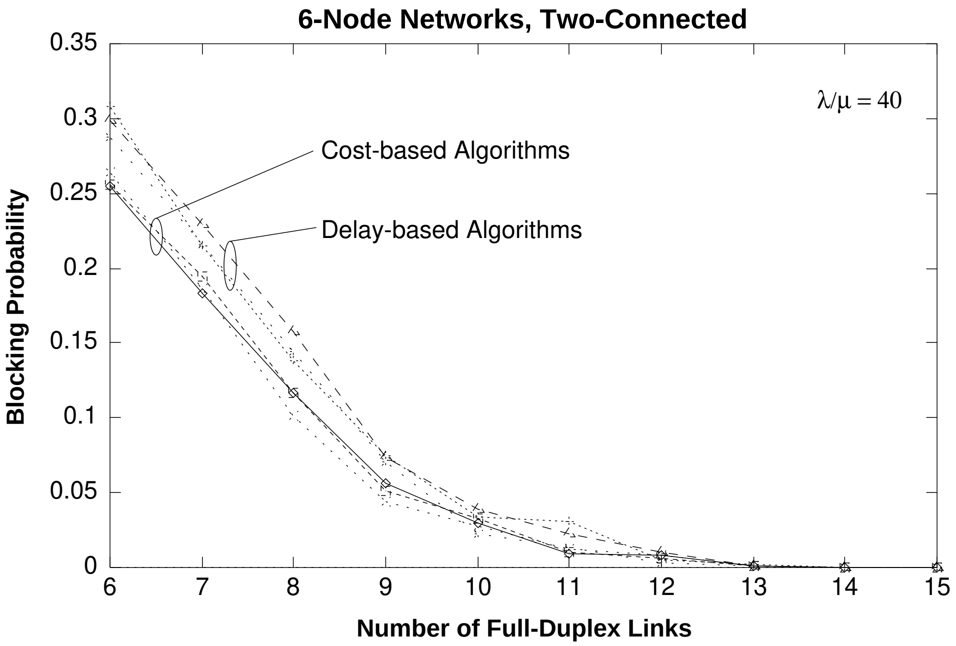


Figure 3.12: 6-node networks, varying number of links under constant session arrival rate; number of destinations varies from 1 to 4, 15,000 routes/point

Figures 3.13 and 3.14 show the blocking probability for 12-node networks, with varying number of links (the NSFNet corresponds to the point where $K = 15$ in the plots). The plot in Figure 3.13 shows the blocking probability when the number of destinations is uniformly distributed between 1 and 4, and the plot in Figure 3.14 shows the blocking probability when the number of destinations is between 1 and 10. Both curves show the high-blocking region, where an increase in the number of links produces an approximately linear decrease in the blocking probability. Additionally, Figure 3.13 shows the low-blocking region.

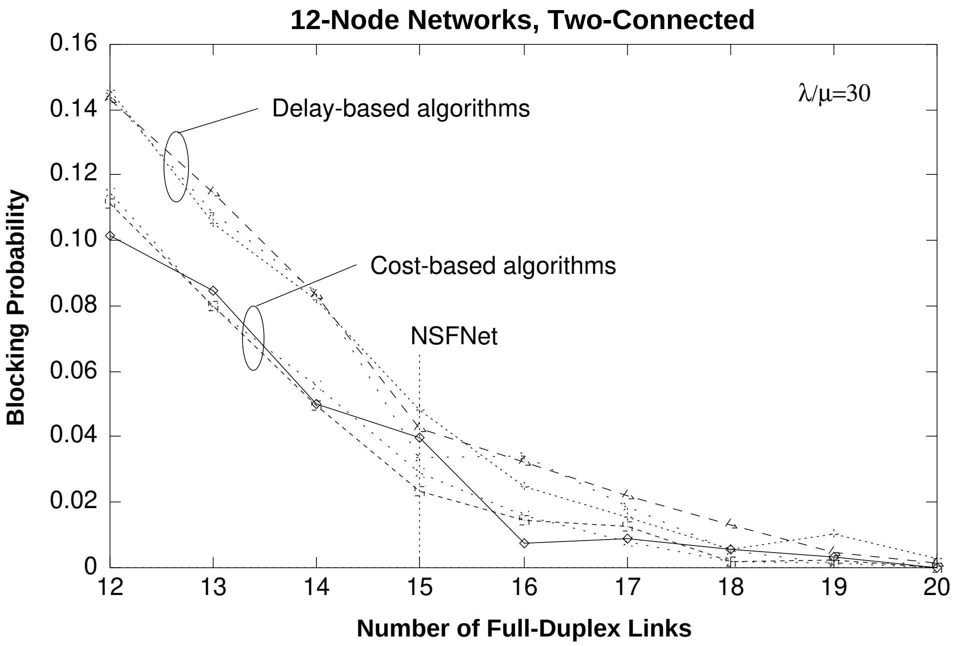


Figure 3.13: 12-node networks, varying number of links, constant session arrival rate; number of destinations between 1 and 4 (10,000 routes/point)

Figure 3.15 shows the blocking probability for a large (50-node) network, when the number of full-duplex links varies from 50 to 300. In this case, we considered only the heuristic algorithms as the run time for the optimum would be extremely large. Again, we see a small advantage for the cost-based algorithm (KMB) over the delay-based ones.

Finally, we considered the following question: given a network, is it better to add bandwidth by adding links (as done in the plots in Figures 3.12 to 3.15), or to increase the bandwidth of the existing links? To answer this question, we considered again a 50-node network, under single multicast sessions, with 50 full-duplex links. Since we focus on two-

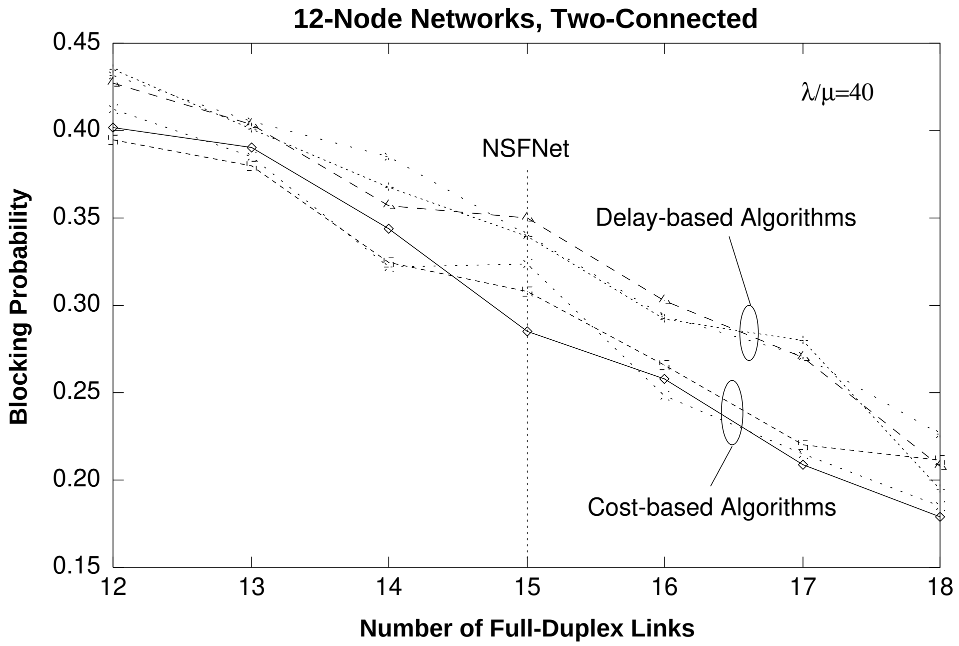


Figure 3.14: 12-node networks, varying number of links, constant session arrival rate; number of destinations between 1 and 10 (20,000 routes/point)

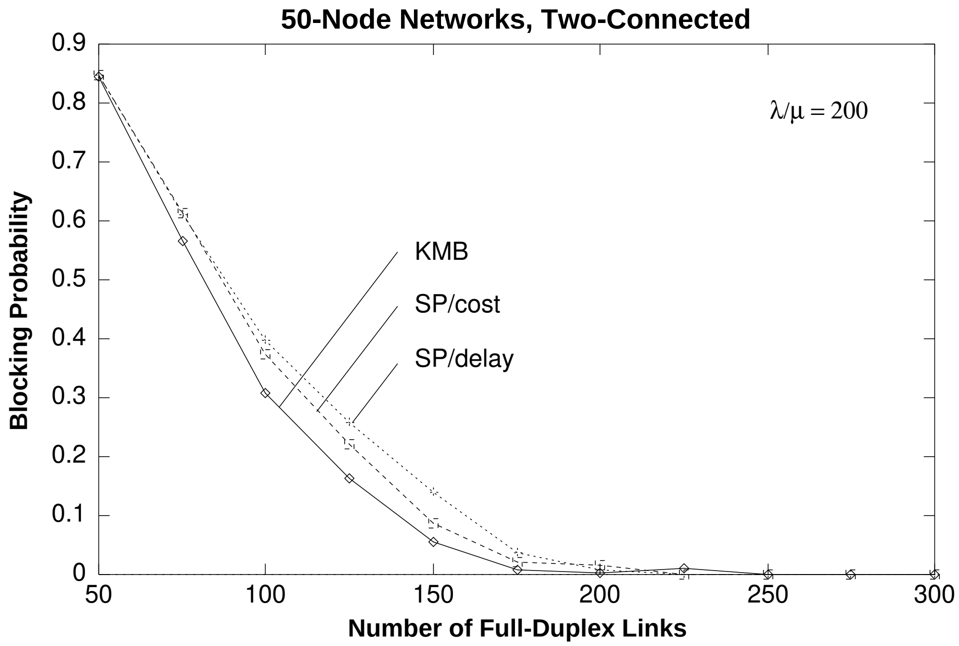


Figure 3.15: 50-node networks, varying number of links, constant session arrival rate; number of destinations between 1 and 10 (15,000 routes/point)

connected networks, the topology of this network is a ring. Using the KMB algorithm to compute the routes, we obtained the blocking probability for the case where links are added to the network (same as Figure 3.15), and for the case where the bandwidth of the existing links is increased, and the topology preserved. Note that in both cases we are adding the same amount of bandwidth to the network; what changes is *where* this bandwidth is added. The results can be seen in Figure 3.16; it is clear that, from a blocking probability point of view, when upgrading the available bandwidth in the network, it is far better to do it by adding new links than by increasing the bandwidth of existing links. This happens because as links are added, not only the capacity increases, but the average path length in the network decreases, further reducing the blocking probability. In practice, however, adding additional links to a network may be more expensive than increasing the bandwidth of the existing links.

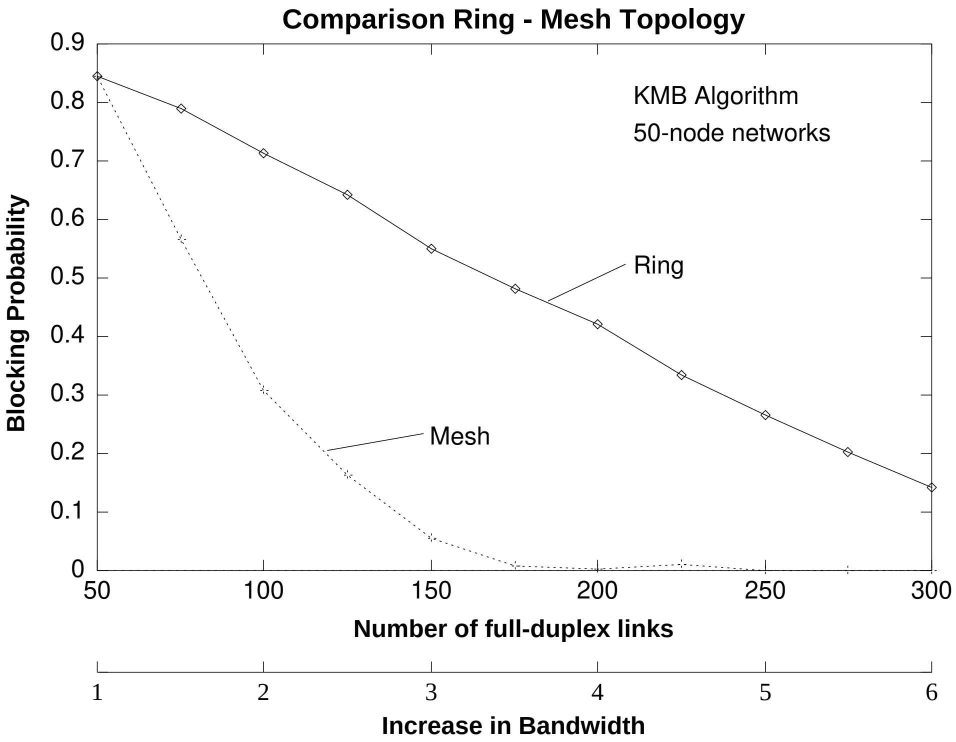


Figure 3.16: 50-node networks, comparison between ring and mesh topologies

3.3.7 Other Scenarios

In this section, we investigate other variations of the baseline case, namely video-conferencing sessions, non-unit cost and other bandwidth distributions for the traffic.

Videoconferencing Sessions

We considered multiple-multicast (video-conferencing) sessions. A video conferencing session with P participants is composed of P multicast streams, from each of the participants to the other $P-1$ conferees. The stream bandwidth was fixed at 10% of the link capacity. The network scenarios were: (i) 12 nodes, 15 links; (ii) 6 nodes, 8 links; and (iii) 6 nodes, 12 links. The main observation, valid for all scenarios, is that there is very little difference in blocking probability between all the algorithms, although the cost-based algorithms still have a small advantage. This is due to the fact that a session is blocked if any of its components is blocked; therefore, the blocking probability is a much "coarser" measure of performance for multiple-multicast sessions than for single-multicast sessions. Moreover, since in the cases evaluated the number of multicasts in the session is small and each stream requests a small fraction of the link bandwidth, the problem is in most cases naturally decomposable (i.e., there is no coupling between the routes of the streams in the session) and there should be little difference between the optimum solution (which takes all streams into account simultaneously when computing routes) and the solution found by the heuristic algorithms (which considers each stream in isolation). Note that if the stream bandwidth is a significant fraction of the link bandwidth, this is not true anymore, as shown in Chapter 2, and there will be a large difference between the optimum and the heuristics. The simulation results for the NSFNet under videoconferencing traffic (number of participants uniformly distributed between 2 and 4, stream bandwidth set to 10% of the link capacity) are shown in Figure 3.17.

Another observation is that, except at very low loads, the blocking probability is now highly dependent on the number of participants in the conference for all the algorithms. For example, under the Optimum/cost/delay algorithm, the blocking probability at $\lambda/\mu = 10$ is about 5% for conferences with 2 participants, while it reaches 22% for conferences with 4 participants, for networks of the size of the NSFNet.

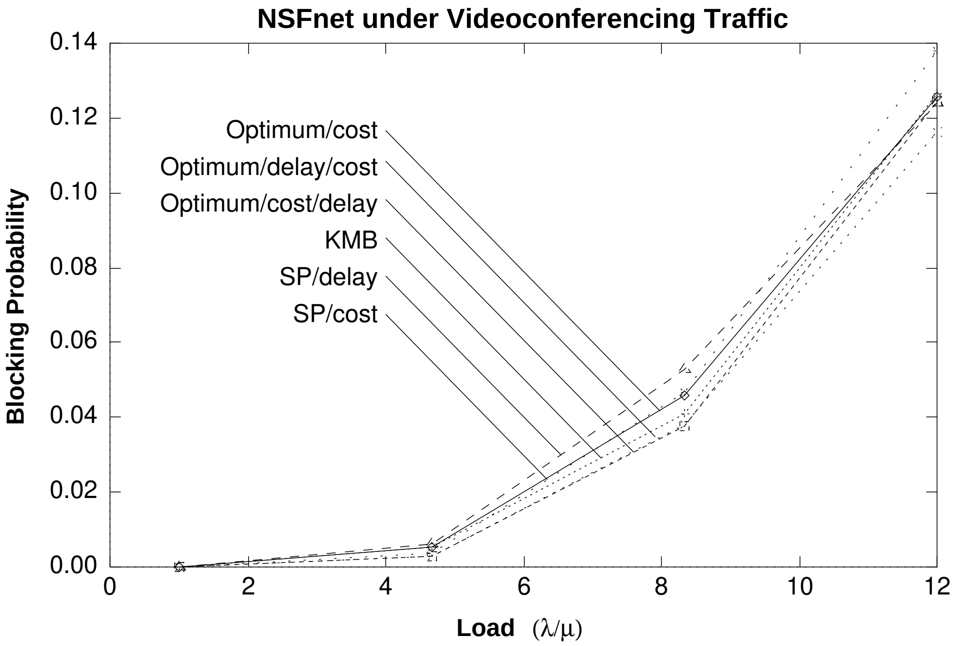


Figure 3.17: Blocking probability for the NSFNet under videoconference traffic (up to 4 destinations, 10% of the link bandwidth), 5,000 sessions/point

Non-Unit Costs

In this section, we investigate the effect of non-unit costs. We repeated the simulation for the following three scenarios, using the NSFNet topology:

- Unit link costs;
- Link costs generated at random, uniformly between 0 and 1; and
- Link costs set to the link lengths (i.e., same values as the link delays).

The results are shown in Figure 3.18 for the Optimum/cost algorithm; the figure indicates that when the costs are set to 1, the blocking probability is lower than when the costs are set proportionally to the link lengths. The reason is that when the costs are all equal, minimizing the cost means minimizing the amount of network resources used to route the multicast, and that should lead to a lower blocking probability. However, as indicated in Figure 3.18, this effect is relatively small - in fact, using random costs, uniformly distributed, yields basically the same results as when using unit costs.

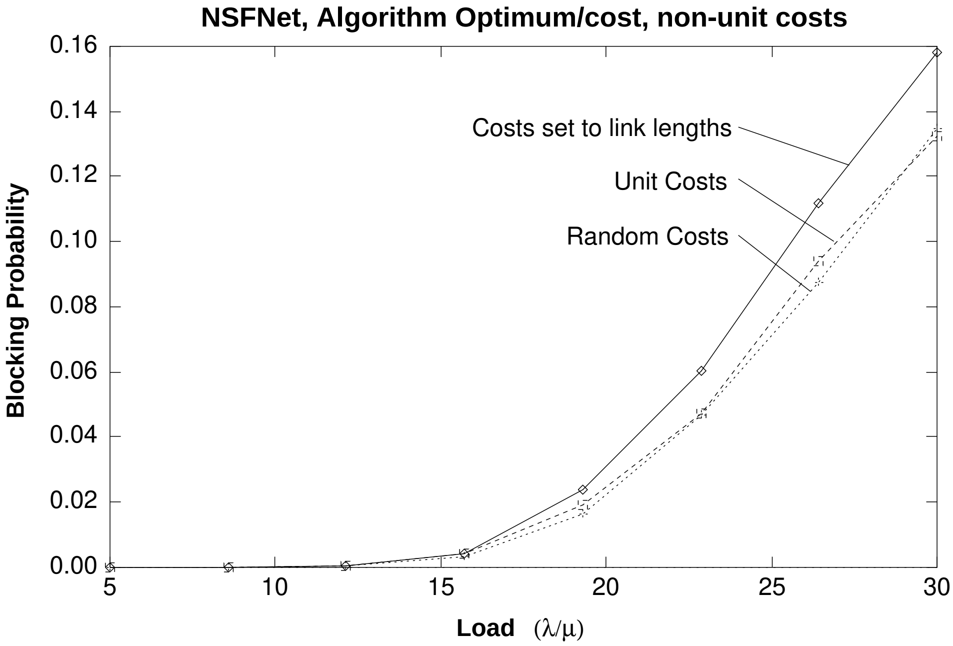


Figure 3.18: NSFNet, non-unit costs, 25,000 routes/point

Other Bandwidth Distributions

In the previous sections, we have assumed that all the streams require the same bandwidth (10% of the link capacity). In an actual network, we expect to find a mixture of bandwidths, corresponding to different qualities of video. This mixture is likely to have more streams at lower bandwidths (i.e., lower video qualities) than at higher bandwidths. Additionally, the bandwidths will belong to a discrete set of values (for example, 384 kb/s, 768 kb/s and 1.984 Mb/s for H.261; 1.5 Mb/s for MPEG I; 2 to 8 Mb/s for MPEG II).

To assess the influence (if any) of the request bandwidth distribution in the performance evaluation, we repeated the baseline case simulations, changing the stream bandwidth from its previous (deterministic) value of 10% of the link bandwidth, to a discrete random variable, assuming the values of 4.5%, 9%, 18% and 36% of the link bandwidth, with probabilities 0.3, 0.3, 0.3 and 0.1 respectively (this would correspond approximately to streams of 2 Mb/s, 4 Mb/s, 8 Mb/s and 16 Mb/s being sent over 45 Mb/s links); the average bandwidth requested is 13%. We observed the same qualitative results as when the streams request 10% of the link bandwidth. In other words, the results presented here are not sensitive to the distribution of the stream bandwidth.

3.3.8 Run Times for the Algorithms

In this section, we characterize the average run times for the algorithms as a function of the network size. The algorithms were implemented in a DEC 5000/240 workstation in C, and compiled with the highest level of optimization available. Figure 3.19 shows the average run time for each of the algorithms, for single-multicast sessions in a 6-node network, with the number of destinations chosen at random between 1 and 4. The figure indicates that the run time for the optimum algorithm is one to two orders of magnitude higher than for the heuristics; the difference increases with increasing network size.

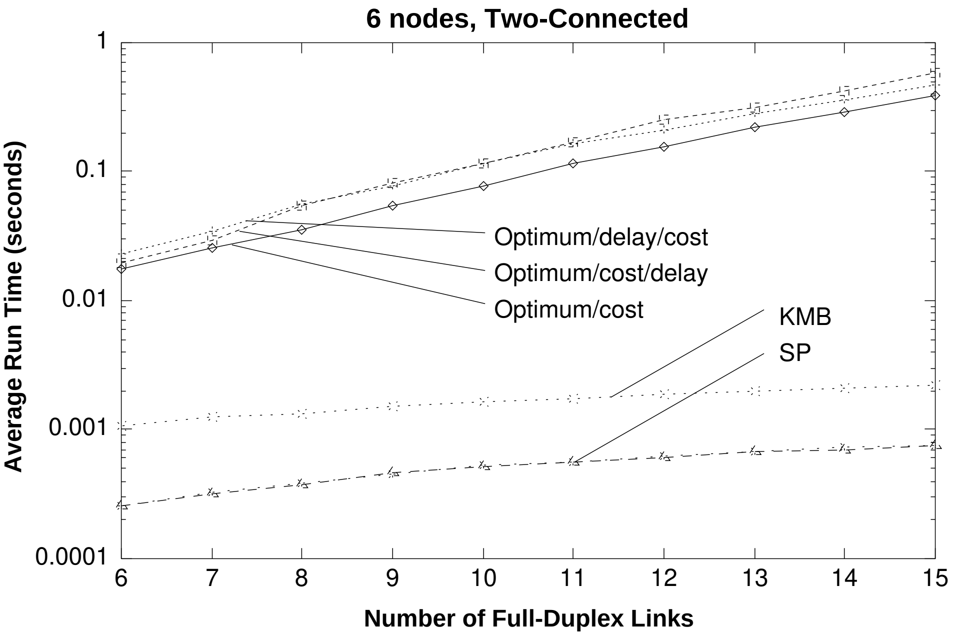


Figure 3.19: Run times for 6-node networks, 15,000 routes/point

Figure 3.20 shows the run times for the heuristic algorithms only, for single-multicast sessions in 50-node networks, where the number of destinations for each multicast is chosen at random between 1 and 10. The figure indicates that the ratio between the run times for the KMB and Shortest-Path algorithms is essentially constant; this is to be expected, because the KMB algorithm corresponds essentially to running the shortest path algorithm a number of times.

One final observation about the run times: for the optimum routing algorithm, we observed that typically the run time for successful sessions (i.e., sessions for which there is at

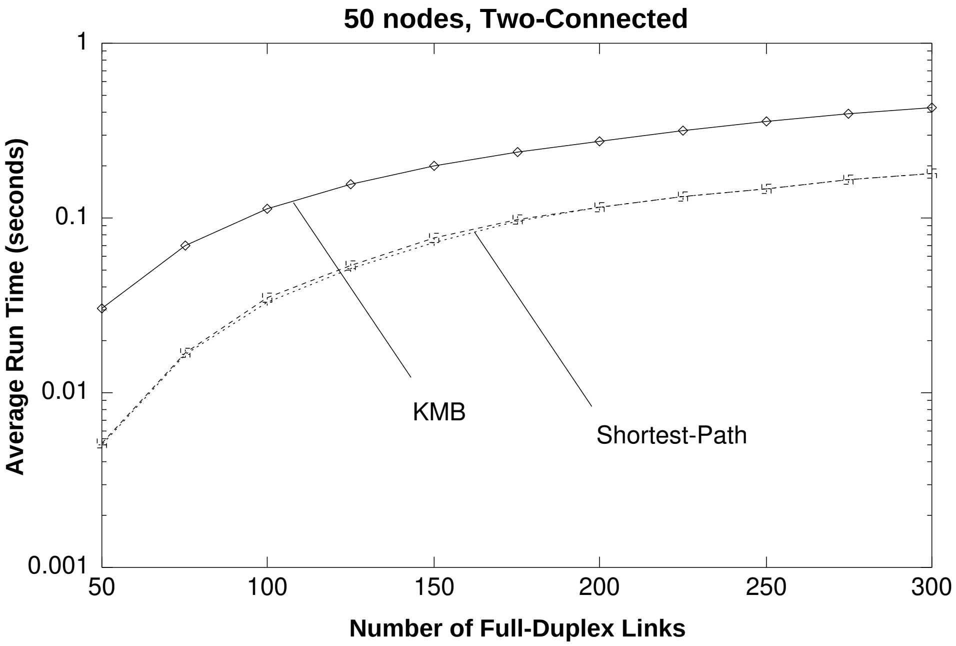


Figure 3.20: Run times for 50-node networks, 15,000 routes/point

least one solution to the routing problem, given the network usage) is much shorter than the run time when no solution exists. In other words, if there is a solution to the routing problem as formulated in Chapter 2, then the optimum routing algorithm will in most cases find it quickly; otherwise, it might take a long time to determine that no solution exists. This is not the case for the heuristics: they take approximately the same time to route a successful session and to give up and declare a session blocked; in fact, a blocked multiple-multicast session might take *less* time to process because not all routes are computed.

The difference in run times could be used to speed-up the optimum routing algorithm by imposing a time limit for finding the solution; if no feasible solution is found when the time limit is reached, the problem is declared infeasible. Such an algorithm is not "optimum" anymore, because there is always a chance that it can miss a solution, or return a sub-optimal one. We evaluated this tradeoff for our implementation of the algorithm, in the DEC 5000/240 workstation, using both the NSFNet topology and random topologies, for sessions with 4-5 multicasts, with 2-5 destinations. The results are shown in Figure 3.21, where we plot the fraction of feasible solutions that would be missed due to the imposition of a time limit, as a function of this time limit. The figure corresponds to 2,346 feasible sessions, and indicates that a reasonable limit is 500 seconds; higher limits would bring diminishing returns. With the 500-second limit, less than 0.2% of the feasible solutions are missed. We should stress that the 500-second figure applies only to our implementation in a DEC 5000/240 workstation and only to networks with 12 nodes and 15 links; for slower CPUs (or bigger networks), higher limits should be used.

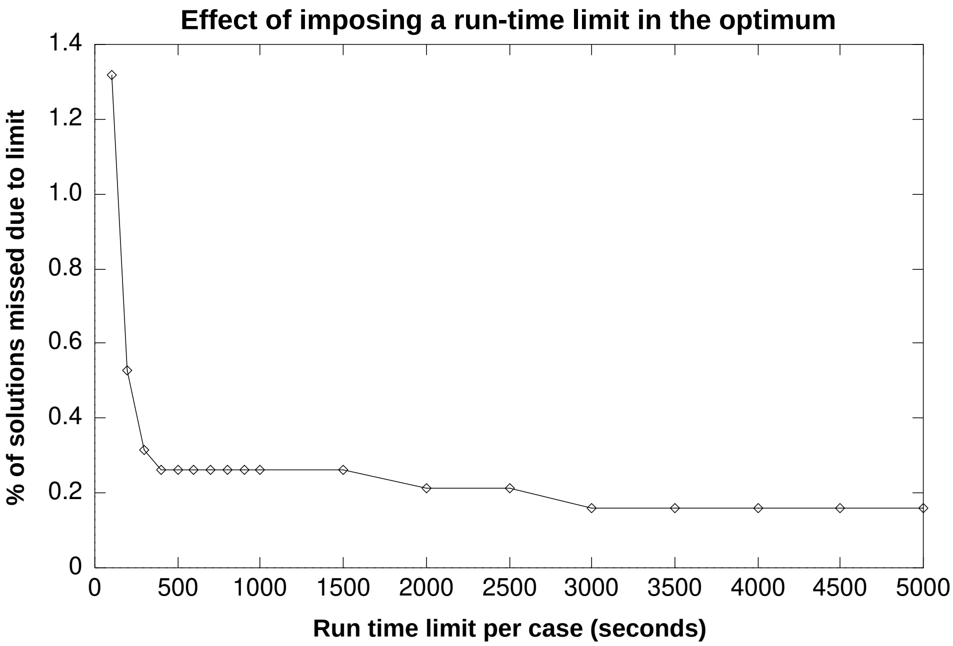


Figure 3.21: Effect of adding a run-time limit to the optimum

3.4 Conclusions

The main conclusion of this chapter is that cost-based algorithms yield, in general, lower blocking probabilities than delay-based algorithms, at the expense of higher delays; the network capacity (defined as the load λ/μ for a target blocking probability) can be 1.2 to 2.0 times higher when the former are used. However, traditional minimum-cost algorithms cannot cope with latency constraints.

We proposed an algorithm for optimum multicast routing. However, due to the large run times (one to two orders of magnitude higher than the heuristics), its main use is as a benchmark for heuristic algorithms (except possibly for small networks). We have found that, under realistic network and traffic conditions, the routes found by the heuristic algorithms are close to the optimum; the only exception is when there are active latency constraints. In this case, the best performance is achieved by the optimum.

The conclusions of this chapter can be summarized as:

- For 1-connected networks, the choice of routing algorithm makes little difference in performance; one might as well use the simplest routing algorithm (shortest path). This kind of network topology should be avoided when implementing a network, because of lower reliability and performance.

- Minimum-cost algorithms are indicated for scenarios where the lowest possible blocking probabilities are required (for a given infrastructure), and latency constraints are not a problem. One example would be a campus network, where the link delays are low, and any path will satisfy the latency constraint.

- In a scenario where the latency constraint can be a problem (e.g., in a WAN environment), the options are:

  - Use shortest path algorithms, and pay a price in higher blocking probability.

  - Use the optimum routing algorithm. This is possible only for small to moderate size networks (i.e, of the size of the NSFNet).

  An area for future work would be to devise an efficient minimum-cost routing algorithm, which is able to satisfy a delay constraint. The work by Kompella et al [37] is a step in the right direction, but their algorithm is applicable only to networks with bidirectional links, which is not the case in practice.

- The best way (from a traffic point of view) to upgrade a network under stream traffic is to add links and make it into a mesh, thus reducing the path length, instead of just increasing the bandwidth of existing links. This is true both for unicast and multicast stream traffic.

Chapter 4

Routing of Streams in WDM Reconfigurable Networks

4.1 Introduction

Due to its low attenuation (less than 0.2 dB/km) and very high bandwidth, fiber has become the medium of choice for point-to-point links at high speeds, for any distance over ≈ 100 m and for data rates of 45 Mb/s and up. Using Wavelength-Division Multiplexing (WDM), many channels can be created in the same fiber. A network node equipped with a tunable transmitter can select any of these channels for sending data, and can dynamically change this selection. The range of wavelengths addressable depends on the technology used to implement the tunable transmitter [47]. The optical signal from the transmitters in the network is combined by an optical interconnection (e.g, a WDM star coupler), and made available to a subset of the optical receivers, determined by the optical interconnection. Optical receivers can also be tunable [47]. The interconnection pattern between nodes is defined by the tuning of transmitters and receivers to specific wavelengths, and can be dynamically changed. In a traditional (fixed-topology) network, given a multimedia session, the routing algorithm is responsible for finding routes for each of its components, satisfying the session requirements. In a WDM network with tunable transmitters and receivers, the routing algorithm has an additional degree of freedom: it can choose (or modify) the topology.

In this chapter, we consider the problem of routing multimedia streams in a WDM network. In section 4.2, we discuss the characteristics of the optical components used in building a network, and describe the previous work in optical WDM networks. In section 4.3, we give the problem formulation, and show that it can be solved exactly by linear programming. Since the problem is NP-complete, the optimum algorithm has worst-case exponential runtime; additionally, its implementation is complex. Therefore, in section 4.4 we give a number of heuristic algorithms for unicast and multicast routing; these heuristic algorithms find suboptimal solutions. Evaluation of the heuristic algorithms is presented in sections 4.5, 4.6 (unicast traffic) and 4.7 (multicast traffic). For the unicast traffic case, we first derive an upper bound in the performance measure of interest, and show that the heuristic produces results that are close to the upper bound, thus obviating the need for pursuing the optimum solution. We also evaluate the performance of the WDM network in a dynamic environment, and compare it to that of a centralized switch. For the multicast traffic case, we compare the various heuristics proposed under a dynamic traffic environment. Our conclusions are presented in section 4.8.

4.2 Optical Network Components and Configurations

The optical WDM network has three basic "building blocks" [48]:

- optical interconnection;
- optical transmitters; and
- optical receivers.

In this section, we describe the characteristics of each of these components, and discuss the previous work in the area of WDM networks.

4.2.1 Optical Interconnection

The optical interconnection is responsible for mixing the light from the transmitters and splitting (dividing) it among the receivers, irrespective of the wavelength. The most common optical interconnection is the WDM star, shown in figure 4.1. The WDM star equally divides the optical power from each of the incoming ports among the output ports. The optical signal in each output port is a combination of the optical signals from each of the input ports, as shown in figure 4.1.

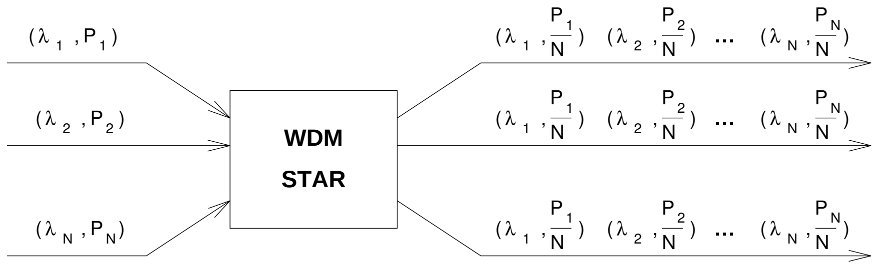


Figure 4.1: The WDM star coupler

Other optical interconnections, such as trees, multiple stars, etc., are possible; they do not necessarily divide the input power equally among all the outputs. In general, a given output will receive the signal from a subset of the inputs [49, 50].

4.2.2 Optical Transmitters

Optical transmitters are responsible for modulating the optical signal with the user data. There are two kinds of light sources for optical transmitters: LEDs and lasers. LEDs can only be amplitude-modulated because of their wide spectral widths, and cannot be modulated at very high data rates. They are unsuitable for use in a WDM network. Lasers can provide much larger output powers than LEDs, and can be modulated at much higher data rates. Moreover, due to the (relatively) narrow spectral width of the laser, it can be modulated not only in amplitude, but also in frequency or in phase.

There are several ways to build tunable lasers; the choice of methods usually represents a tradeoff between tuning speed on one side, and linewidth[1] and range of frequencies that can be addressed on the other side. Linewidth is important because, for a given modulation format, it determines the minimum channel spacing if WDM is used to combine several

[1]Due the the phase noise, the laser output spectrum is not an ideal line, but has a certain spectral width; the 3-dB spectral width is denoted by **linewidth.**

channels in the same fiber. Table 4.1 [51, 52] summarizes the characteristics of several tuning methods.

Table 4.1: Characteristics of Laser Tuning Methods

METHOD	RANGE	KIND	LINEWIDTH	SPEED
Electro-optical	7 nm	discrete	60 kHz	100 μs
Acusto-optical	70 nm	discrete	?	3 μs
2-section DFB	0.32 nm	continuous	?	$<$ 5 ns
	3.3 nm	continuous	15 MHz	?
3-section DBR	8-10 nm	quasi-cont.	20-100 MHz	10 μs
	4.4 nm	continuous	1.9 MHz	10 μs
	2 nm	continuous	2.5-6.5 GHz	15 ns

Another important transmitter parameter is the optical power. If the optical signal from a transmitter is split among multiple receivers by the optical interconnection, the optical power is one of the factors determining the number of receivers that can be reached.

4.2.3 Optical Receivers

The main component in an optical receiver is the photodiode, which converts the incoming light into an electrical signal. The photodiode responds to the optical power of the signal; it is largely independent of the wavelength. There are basically two kinds of optical receivers [48, 47]:

Direct Detection Receivers: The optical signal is applied directly to the photodiode. This kind of receiver can be used only with amplitude modulation (ASK). In a WDM system, the direct detection receiver must be preceded by an optical filter [53], which allows only a single wavelength to reach the photodiode. If the optical filter is tunable, the receiver will be tunable. Table 4.2 presents the characteristics of the various types of tunable filters [54].

Table 4.2: Tunable Filter Characteristics

TYPE	RANGE	BW (nm)	CHANNELS	LOSS	SPEED
Fabry-Perot	50 nm	< 0.01	100s	5 dB	ms
Acusto-Optics	400 nm	1	100s	5 dB	μs
Electro-Optics	10 nm	1	10	5 dB	ns
Active Semiconductors	1-4 nm	0.05	10	0 dB	ns

Coherent Receivers: Coherent receivers mix the light from a local laser with the incoming signal, prior to applying it to the photodiode. The local laser is kept synchronized to the transmit laser by means of a PLL. A coherent receiver responds only to a specific wavelength, defined by the wavelength of the local laser. The coherent receiver can be tuned by changing the wavelength of the local laser; see table 4.1 for the characteristics of tunable lasers. Coherent receivers are more complex than direct detection receivers.

An optical WDM receiver is characterized by the following parameters:

Sensitivity: Defined as the minimum power at the input of the receiver that still guarantees a bit error probability not higher than 10^{-9}. A more fundamental measure is the number of photons per bit required to achieve this error probability, but this quantity only makes sense on a limit situation where the dominant noise is the shot noise. In general, coherent receivers have higher sensitivity than direct detection receivers.

Minimum Channel Separation: Minimum separation, in wavelength, between the center frequencies of two distinct channels. Direct detection receivers use optical filters, which have large passbands, resulting on large channel separations. Coherent receivers do the filtering in the electrical domain, which allows for the use of much sharper filters; in this case, the channel separation is determined by the modulation format and the combined linewidth of the transmitter and local oscillator lasers.

Tuning Speed: Time for a receiver to switch from one wavelength to another. Coherent receivers can be tuned by adjusting their local oscillator lasers; the figures on table 4.1 are valid for this case too. However, if the local laser must be kept in phase with the transmitter laser (which is required for PSK and FSK), then the total tuning time includes also a component corresponding to the time necessary for achieving the phase lock, which will depend on the receiver structure. The tuning times for the various types of optical filters are given in table 4.2

4.2.4 Optical Network Configurations and Operation

The basic function of a network is to transport the data generated by the users and deliver it to the destination, with the appropriate quality of service. The physical implementation of the network depends on where the users are and what kind of service they expect. For example, a single shared channel is a reasonable implementation for a local area network, while in the wide-area it is more reasonable to implement a network with point-to-point links, operating in a store-and-forward fashion.

We classify the previous work into two categories: *Local Area Networks* (LANs) and *Wide Area Networks* (WANs). The work in Local Area Networks is characterized by the fact that a direct channel is established between the sender and the receiver, and communication is single-hop. The network interconnection is usually assumed to be a WDM star. The work in Wide Area Networks is characterized by the multi-hop communication aspect, and by a more general optical interconnection. The Metropolitan Area Network (MAN) is an intermediate case; some MAN schemes are multiple-hop, others are shared-channel. A comprehensive review of the work in the field of WDM networks can be found in [55] and [56]. Unless explicitly stated, all the work described in this section is theoretical. A survey of the work in experimental WDM networks can be found in [47]. Except for IBM's RAINBOW, which will be described in the next section, none of the experimental WDM networks has contributed to the field of routing: the bulk of the work done there was in the actual implementation, and since all these networks have a small number of nodes, routing is really not an issue for them.

WDM Local Area Networks/Single Hop Operation

In Local Area Networks, typically there is a direct channel between the sender and the receiver. In existing networks such as an Ethernet segment, the bandwidth of this channel is shared between all the nodes connected to it, and is an upper bound in the throughput of the network. As the user traffic increases, either the channel bandwidth must be increased, or more channels must be provided (and switching between these channels). WDM is a way of providing more channels, and the switching function can be implemented by having tunable transmitters and/or receivers. The following observations can be made:

- If the receivers are tunable, the network can provide physical multicasting, by tuning multiple receivers to the wavelength used by a given transmitter. However, there is a coordination problem, because the switching action (tuning) happens at the receiver, which must be somehow informed that the sender wishes to initiate the communication.

- If the transmitters are tunable, there is no sender-receiver coordination problem, because the switching action happens at the sender. However, multiple transmitters can potentially tune to the same wavelength; this represents a collision, and the resulting signal in general cannot be received. This problem can be dealt with by having some sort of coordination between senders (so it does not happen) or by providing some sort of multiple-access scheme, to recover from collisions.

Habbab et al [57] and later Mehravari [58] considered a WDM star network where the number of distinct wavelengths is much less than the number of stations. Each station has one tunable receiver and one tunable transmitter, and both are capable of addressing all the wavelengths in the network. Coordination between transmitters and receivers is achieved by reserving one wavelength for control; all idle nodes keep their receivers tuned to this wavelength. When a node decides to transmit, it chooses one of the data wavelengths at random and sends a packet in the control channel informing the destination of this choice. It then tunes its transmitter to the data wavelength chosen and sends the packet. Multiple-access schemes are used both in the control and in the data channels. The authors study the network throughput and delay as a function of the multiple-access schemes used in the control channel and in the data channels.

Chlamtac and Ganz [59] and later Ganz and Koren [60] considered a scenario where all the stations are synchronized, the transmitters are fixed and the receivers are tunable. All packets arrive aligned at the star coupler. Coordination between transmitters and receivers is achieved by having a common "tuning schedule", known by all nodes; the wavelengths are used in a TDM fashion. Control algorithms and approximate analysis based on Markov chains are presented.

The RAINBOW network [61] is a local/metropolitan area WDM network intended to cover a diameter of 25 km, designed and implemented by IBM. It connects up to 32 IBM PS/2's through a 32×32 passive star coupler and allows the computers to communicate circuit-switched data at a rate of 300 Mb/s/node, yielding an aggregate throughput of up to 9.6 Gb/s. The network's physical topology is the WDM star. Each computer is equipped with its own fixed frequency optical transmitter and tunable optical receiver. The optical transmitters utilize directly modulated distributed feedback (DFB) laser diodes. Wavelength selection at the receiver is accomplished with a tunable fiber Fabry-Perot filter whose cavity spacing is varied piezoelectrically. To open a circuit, a node tunes its receiver to the destination's wavelength, and starts transmitting a "request" pattern in its wavelength. Idle receivers are continuously polling the transmit wavelengths, looking for requests. Once a request is found, the node will keep its receiver tuned to the requestor's wavelength, and will acknowledge in is own wavelength. Communication now can start. The time for the receiver to identify and lock to a channel is 10 ms.

In summary, the work done in single-hop algorithms assumes that the tuning of transmitters and/or receivers can be very fast, and that the network either uses a multiple-access scheme (which is difficult to implement efficiently in optics) or is synchronized (which might be difficult to achieve at high speeds).

Wide Area Networks/Multi-Hop Operation

In a Wide-Area Network, due to its size and geographical distribution of nodes, it is not possible (or reasonable) to have channels shared by the nodes. For example, it is reasonable to connect all the nodes in a building in a star topology; all the fibers go to a closet where they connect to a WDM star coupler. However, it is not reasonable to connect all the major network nodes in a country to a single "central" star; the delay and loss in the fiber would be unacceptable. In this latter case, a mesh topology is more indicated; communication between neighboring nodes will happen with a minimum of delay. In a WAN, links are usually point-to-point, and communication happens in a store-and-forward manner. One of the first WDM networks proposed, the ShuffleNet [62], was a store-and-forward network. Transmitters and receivers were fixed, and the "links" were the WDM channels. It was no different than an interconnection of nodes using point-to-point links in a certain specific topology.

In [63, 64, 65, 66] it is assumed that the network reconfiguration process will be performed infrequently; during the reconfiguration, the network may even be non operational. The problem then becomes similar to a traditional topological design problem, where the traffic is Poisson and the traffic matrix is known, with additional constraints introduced by the fact that each node has a well-defined number of transmitters and receivers. They all consider that the combined optical signal from all transmitters is available to all receivers. The differences are in the following areas:

(i) The fiber plant: paper [63] also considers the design of the fiber plant (optical power budget, propagation delays). Papers [64, 65, 66] do not make any additional assumptions about the optical interconnection.

(ii) Objective function to be optimized: in [63], the objective function is to minimize the average delay; the authors assume a queue model for the nodes, which makes the delay a non-linear function of the flow in the links. They also take into account the propagation delays in the network. In [64], the network is assumed to operate under deflection routing, and the average delay is indirectly minimized by minimizing the length of the alternate paths between sources and destinations. In [65, 66], the objective function is to minimize the maximum flow over all links.

(iii) Additional constraints in the optimization: in [66] tunability restrictions are assumed, i.e., receivers can only be tuned to a subset of the available bandwidths. The other papers do not have additional constraints.

(iv) Solution method: in [63, 64] the objective function is non-linear, and the authors resort to the "simulated annealing" method to search for a sub-optimal solution. In [65, 66] the authors present an heuristic algorithm which divides the problem into two subproblems - the wavelength assignment subproblem and the routing subproblem, which are solved by linear programming.

Summary of Previous Work

In summary, the previous work in the field of WDM networks can be classified into single-hop routing (appropriate for LANs, and maybe MANs) and multi-hop routing (appropriate for MANs/WANs). For single-hop routing, one has to assume either a multiple-access scheme or synchronization between nodes; both are difficult to efficiently implement in practice. For multi-hop routing, it has been assumed that tuning of the transmitters and receivers happens over a very long time scale; the topology of the network does not change often, and when it does, it is in response to changes in the traffic matrix. The problem then becomes similar to the traditional topological design problem, with some additional constraints (i.e., the number of links leading to a node must be equal to the number of receivers in that node, and similarly for transmitters).

Restricting the WDM network to single-hop operation (tuning in a packet-by-packet basis) has practical implementation problems, and if the tuning is not fast enough, streams cannot be supported in this environment. The other extreme (reconfiguring the network only when the long-term traffic trends change) does not make use of the full switching potential of the WDM network. When dealing with streams, it is possible to reconfigure the network when requests arrive, and when streams terminate. Conceptually, this is similar to the long-term reconfiguration, but the change in traffic trends are actual stream arrivals and terminations.

4.3 Problem Formulation

In this section, we define the problem of routing streams in a WDM network. We start with the traffic model, which is the same as in Chapter 2, and then describe the assumptions we make about the optical network. We then give the problem formulation in precise mathematical terms, and show how it can be transformed into an integer linear programming problem. The approach taken is to present a sequence of linear programming formulations, starting from the simplest (unicast traffic, unit link labels, no latency constraints) and reaching the most general case.

4.3.1 The Traffic Model

For the traffic, we assume that user's requests come in *sessions*. A session is a group of streams that are logically related. We will denote by T the number of streams in the session. Stream i, $i = 1, \ldots, T$, is characterized by its source s_i, its n_i destinations $d_{i1}, d_{i2}, \ldots, d_{in_i}$, its bandwidth requirement r_i and its maximum latency constraint D_i. We assume that all the streams in the session arrive and depart simultaneously, the session arrival process is a Poisson process with rate λ, and the session duration is exponentially distributed with rate μ.

4.3.2 Network Assumptions

In this chapter, we assume that the network operates in a store-and-forward, multi-hop operation, but the reconfiguration of the network happens in a stream-by-stream basis, creating paths as the streams arrive, and removing them after they terminate. As done in the previous chapters, bandwidth in the links can be shared in a TDM fashion.

We make the following assumptions about the network, which is depicted in Figure 4.2:

- There are N nodes in the network; node i, $i = 1, \ldots, N$ is equipped with S_i optical transmitters and P_i optical receivers.
- The optical interconnection is such that all receivers have access to the light signal from all transmitters. No other assumptions are made about it. This assumption simplifies the formulation of the problem, but limits the results to the LAN/MAN environment.
- The number of distinct wavelengths, denoted by W, is larger than the number of transmitters/receivers in the network. Due to the large available in the fiber, this is a reasonable assumption.
- At any time, only one transmitter can be tuned to a given wavelength. We do not consider multiple-access operation (i.e., many transmitters tuned to the same wavelength), because this is difficult to implement efficiently in optics.

- Any given transmitter can be connected to any given receiver - there are no tunability restrictions. Current technology allows the implementation of transmitters and receivers that are tunable over wide ranges, making this a reasonable assumption.
- Usually, S_i and P_i are much less than N. Therefore, each node will have direct connectivity to a (typically small) subset of nodes.

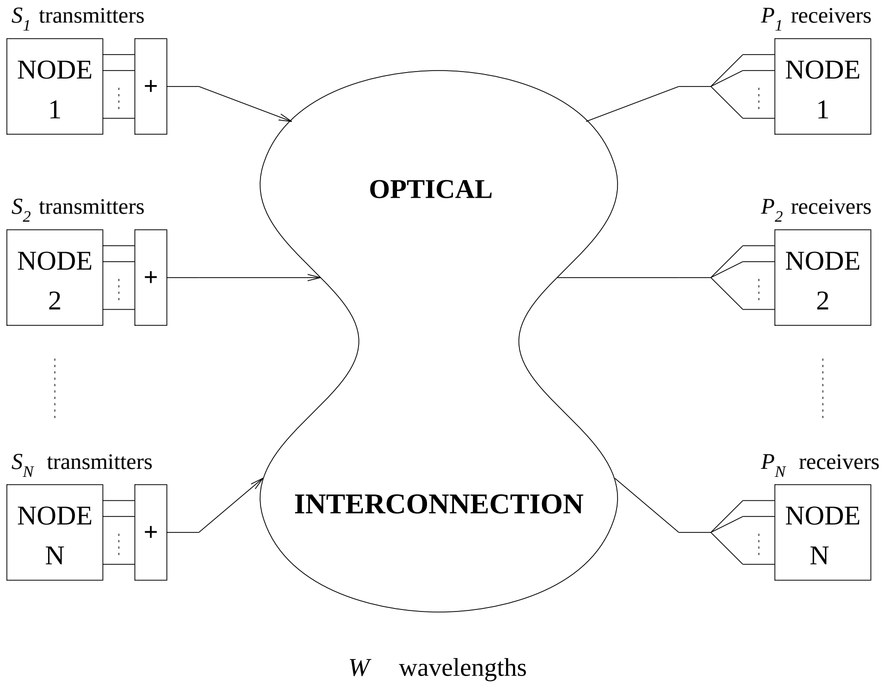


Figure 4.2: The WDM Network

4.3.3 Statement of the Problem

The problem under consideration can be stated as: "Given a session with T streams, each stream being characterized by its source, destinations, latency constraint and bandwidth requirement, find the logical network topology and routes that satisfy the stream requirements, while optimizing a given objective function."

A solution to the reconfiguration/routing process is composed of two parts: (i) the wavelength assignment, which defines which transmitters are connected (tuned) to which receivers (and thus defines the network topology), and (ii) the assignment of routes given the wavelength assignment. If multiple solutions exist for a reconfiguration/routing problem, the objective function is the criterion used to select the "best" one. As done in Chapter 2, the objective function is chosen to be a linear combination of costs and delays.

4.3.4 First Formulation: The Unicast Routing Problem

In this section, we present the mathematical formulation for the optimum reconfiguration and routing problem in its simplest case. To write this formulation, we make the following additional assumptions:

- There are no latency constraints.
- All the streams are unicast.
- The total number of transmitters in the network ($\sum_{i=1}^{N} S_i$) is equal to the total number of receivers in the network ($\sum_{i=1}^{N} P_i$) and will be denoted by K. This is a reasonable assumption under unicast traffic because all communications is one-to-one; "extra" transmitters or receivers will remain unused and thus can be ignored.
- No physical multicast is allowed, i.e., there is at most one receiver and one transmitter per wavelength. Under this assumption, it does not matter which element (i.e., the transmitter or the receiver) is tunable.
- Link costs and delays are all unity.

All these assumptions will be relaxed in latter formulations. Defining:

T	:	Number of streams in the session
$\{r_i\}$	:	Required bandwidth for stream i, $i = 1, \ldots, T$
$\{s_i\}$	:	Source node for stream i, $i = 1, \ldots, T$
$\{d_i\}$	:	Destination node for stream i, $i = 1, \ldots, T$
K	:	Total number of transmitters/receivers in the network
N	:	Number of nodes in the network
V	:	Bandwidth of each individual link
$\boldsymbol{P}$	:	Receiver distribution vector ($N \times 1$); P_i is the number of receivers in node i

$\boldsymbol{L}$: Transmitter location matrix ($N \times K$); $L_{ij} = 1$ if transmitter j is located in node i, otherwise $L_{ij} = 0$.

$\boldsymbol{R}$: Wavelength allocation matrix ($N \times K$); $R_{ij} = 1$ if transmitter j is sending in a wavelength currently being received at node i, otherwise $R_{ij} = 0$.

$\boldsymbol{B}^i$: Destination vector ($N \times 1$) for stream i, $i = 1, \ldots, T$; $B^i_{s_i} = 1$, $B^i_{d_i} = -1$, and $B^i_j = 0$ for $j = 1, \ldots, N$; $j \neq s_i$; $j \neq d_i$

$\boldsymbol{X}^i$: Routing vector ($K \times 1$) for stream i; $X^i_j = 1$ if stream i is routed through transmitter j, $i = 1, \ldots, T$, $j = 1, \ldots, K$

The problem formulation can be expressed as:

GIVEN: $K, N, V, T, \boldsymbol{L}, \boldsymbol{P}; \{s_i\}, \{d_i\}, \{r_i\}, i = 1, \ldots, T$

MINIMIZE: Average path length

$$\sum_{i=1}^{T} r_i \sum_{j=0}^{K} X^i_j \tag{4.1}$$

WITH RESPECT TO: $\boldsymbol{R}$, $\boldsymbol{X}^i, i = 1, \ldots, T$

UNDER CONSTRAINTS:

1. Communication is one-to-one, i.e., there is only one transmitter and one receiver per wavelength.

$$\sum_{i=1}^{N} R_{ij} = 1, \quad j = 1, \ldots, K \tag{4.2}$$

2. Node i has only P_i receivers.

$$\sum_{j=1}^{K} R_{ij} = P_i, \quad i = 1, \ldots, N \tag{4.3}$$

3. There should be a path from every source to every destination. This is equivalent to writing a set of flow conservation equations, for routing one unit of flow from the source to the destination of each stream in the session.

$$(\boldsymbol{L} - \boldsymbol{R})\boldsymbol{X}^i = \boldsymbol{B}^i, \quad i = 1, \ldots, T \tag{4.4}$$

4. The total bandwidth of the streams routed through a link should not exceed the link bandwidth:

$$\sum_{i=1}^{T} r_i X_j^i \leq V, \quad j = 1, \ldots, K \tag{4.5}$$

5. Integer constraints: receivers cannot be "divided".

$$\boldsymbol{R} \qquad \text{is binary} \tag{4.6}$$

No bifurcation of flow (in a packet-switched network, this condition can be relaxed, in which case the stream might be "divided" into several routes):

$$\boldsymbol{X} \qquad \text{is binary} \tag{4.7}$$

The objective function (4.1) and constraints (4.2), (4.3), (4.4) and (4.5) define a non-linear optimization problem; the objective function is linear, but the constraint set (more specifically, equations (4.4) - the flow conservation equations) is not. When constraints (4.6) and (4.7) are added, it becomes a non-linear integer optimization problem.

However, the non-linearity in the constraint set comes just from the $\boldsymbol{R}\boldsymbol{X}^i$ product in equation (4.4). By using the fact that $\boldsymbol{R}$ and $\boldsymbol{X}^i$ are binary variables, and by increasing the number of equations and free variables, we can convert the routing/reconfiguration problem into a *linear* integer programming problem. We add to the set of free variables the $N \times K$ binary matrices $\boldsymbol{Z}^i$, $i = 1, \ldots, T$, subject to the following new constraints:

$$Z_{jk}^i \leq X_k^i \tag{4.8}$$

$$Z_{jk}^i \leq R_{jk} \tag{4.9}$$

$$i = 1, \ldots, T; \quad j = 1, \ldots, N; \quad k = 1, \ldots, K$$

Equation (4.4) then becomes :

$$\boldsymbol{L}\boldsymbol{X}^i - \boldsymbol{Z}^i \mathbf{1} = \boldsymbol{B}^i \tag{4.10}$$

where $\mathbf{1}$ is a $K \times 1$ vector with 1 in all positions.

In summary, by adding $\boldsymbol{Z}^i$ to the list of free variables, replacing equation (4.4) with equation (4.10), and adding inequalities (4.8) and (4.9) to the constraint set, the reconfiguration/routing problem becomes a linear integer programming problem, which can be solved by standard techniques such as the branch-and-bound method [35]. It should be noted that, for a given fixed topology (i.e., given $\boldsymbol{R}$), this problem reduces to the well-known multicommodity flow problem.

4.3.5 Second Formulation: Routing of Multicast Streams in a WDM Network with Tunable Transmitters

We now relax the following assumptions from the previous formulation:

- Streams can be multicast.
- Streams can have maximum latency constraints, measured in hops.
- The objective function is a linear combination of costs and delays, both measured in number of hops.

We still assume that tuning is one-to-one, i.e., physical multicast is not allowed. This would be the case in a WDM network where the transmitters are tunable. Of course, if we prohibit physical multicasting, this formulation also applies to a WDM network with tunable receivers.

Defining:

K	:	Total number of transmitters/receivers in the network
N	:	Number of nodes in the network
V	:	Bandwidth of each individual link
$\boldsymbol{P}$	:	Receiver distribution vector ($N \times 1$); P_i is the number of receivers in node i
$\boldsymbol{L}$	:	Transmitter location matrix ($N \times K$); $L_{ij} = 1$ if transmitter j is located in node i, otherwise $L_{ij} = 0$.
$\boldsymbol{R}$	:	Wavelength allocation matrix ($N \times K$); $R_{ij} = 1$ if transmitter j is sending in a wavelength currently being received at node i, otherwise $R_{ij} = 0$.

T	:	Number of multicast streams.
s_i	:	Source node for multicast i
n_i	:	Number of destinations for multicast i
$\{d_{ik}\}$	:	Set of destinations for multicast i, $k = 1, \ldots, n_i$
r_i	:	Bandwidth requirement for multicast i
$\boldsymbol{X}^i$	:	$K \times n_i$ multicast routing matrix for multicast stream i. $X^i_{jk} = 1$ if transmitter j is used in the multicast path for stream i to reach destination d_{ik}, otherwise $X^i_{jk} = 0$, $k = 1, \ldots, n_i$.
Y^i	:	$K \times 1$ multicast path vector for stream i. $Y^i_j = 1$ if transmitter j is in the multicast path for stream i, otherwise $Y^i_j = 0$.
M_i	:	Delay for multicast request i, in hops.
D_i	:	Latency constraint for multicast request i, in hops.
$\boldsymbol{B}^i$	:	$N \times n_i$ source-destination matrix for multicast stream i; $B^i_{jk} = 1$ if $j = s_i$, $B^i_{jk} = -1$ if $j = d_{ik}$, and $B^i_{jk} = 0$ otherwise, $k = 1, \ldots, n_i$.
β_c	:	Weight of the cost in the optimization.
β_d	:	Weight of the delay in the optimization.

The optimum multicast routing routing problem in a WDM network can be formulated as follows:

GIVEN: $K, N, V, \boldsymbol{L}, \boldsymbol{P}, T, \beta_c, \beta_d; \{\boldsymbol{B}^i\}, \{r_i\}, \{D_i\}, i = 1, \ldots, T$

MINIMIZE:

$$\sum_{i=1}^{T} r_i \left(\beta_c \sum_{j=1}^{K} Y^i_j + \beta_d M_i \right) \tag{4.11}$$

WITH RESPECT TO: $\boldsymbol{R}; \boldsymbol{X}^i, \boldsymbol{Y}^i, M_i, \quad i = 1, \ldots, T$

UNDER CONSTRAINTS:

1. Physical communication is one-to-one, i.e., there is only one transmitter and one receiver per wavelength.

$$\sum_{i=1}^{N} R_{ij} = 1, \quad j = 1, \ldots, K \tag{4.12}$$

2. Node i has only P_i receivers.

$$\sum_{j=1}^{K} R_{ij} = P_i, \quad i = 1, \ldots, N \tag{4.13}$$

3. For every stream, there must be a path from its source to each of its destinations. This is equivalent to writing a set of flow conservation equations for routing one unit of flow from the source to each of the destinations:

$$(\boldsymbol{L} - \boldsymbol{R})\boldsymbol{X}^i = \boldsymbol{B}^i \quad i = 1, \ldots, T; \tag{4.14}$$

4. If a link is in the path from the source to any of the destinations, then it must be included in the multicast path.

$$X^i_{jk} \leq Y^i_j, \quad k = 1, \ldots, n_i, \quad j = 1, \ldots, K, \quad i = 1, \ldots, T; \tag{4.15}$$

5. The delay for a multicast is the delay to the farthest destination:

$$M_i - \sum_{j=1}^{K} X^i_{jk} \geq 0, \quad k = 1, \ldots, n_i, \quad i = 1, \ldots, T; \tag{4.16}$$

6. There is a maximum delay constraint for each of the multicast streams:

$$M_i \leq D_i, \quad i = 1, \ldots, T; \tag{4.17}$$

7. The total flow through a link cannot exceed its bandwidth:

$$\sum_{i=1}^{T} r_i \boldsymbol{Y}^i \leq V; \tag{4.18}$$

8. Integer constraints: no bifurcation of flow; a single path is taken from the source to each of the destinations.

$$\boldsymbol{X}, \boldsymbol{Y} \quad \text{are binary.} \tag{4.19}$$

Receivers cannot be "divided":

$$\boldsymbol{R} \qquad \text{is binary} \tag{4.20}$$

The objective function (4.11) and constraints (4.12) to (4.18) define a non-linear optimization problem; the objective function is linear, but the constraint set (more specifically, equations (4.14) - the flow conservation equations) is not. When constraints (4.19) and (4.20) are added, it becomes a non-linear integer optimization problem.

However, the non-linearity in the constraint set comes just from the $\boldsymbol{RX}^i$ product in equation (4.14). By using the fact that $\boldsymbol{R}$ and $\boldsymbol{X}^i$ are binary variables, and by increasing the number of equations and free variables, we can convert the routing/reconfiguration problem into a *linear* integer programming problem. We add to the set of free variables the $N \times K$ binary matrices $\boldsymbol{Z}^{ij}$, $i = 1, \ldots, T$, $j = 1, \ldots, n_i$, subject to the following new constraints:

$$Z^{ij}_{kl} \leq X^i_{jl} \tag{4.21}$$

$$Z^{ij}_{kl} \leq R_{kl} \tag{4.22}$$

$$i = 1, \ldots, T; \quad j = 1, \ldots, n_i; \quad k = 1, \ldots, N; \quad l = 1, \ldots, K$$

Equation (4.14) then becomes :

$$\boldsymbol{LX}^i_j - \boldsymbol{Z}^{ij}\mathbf{1} = \boldsymbol{B}^i_j \tag{4.23}$$

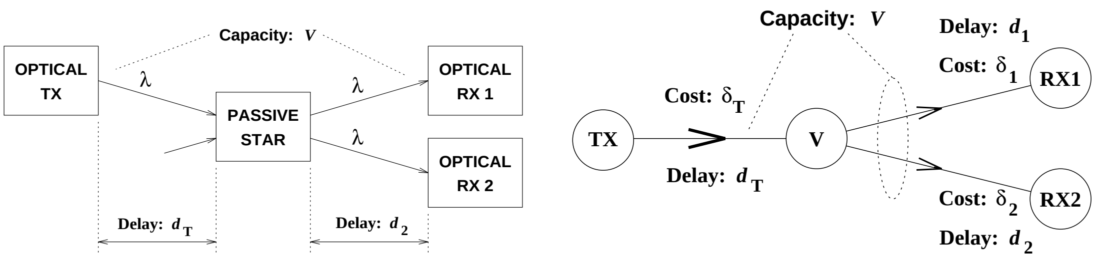
$$i = 1, \ldots, T; \quad j = 1, \ldots, n_i$$

where $\mathbf{1}$ is a $K \times 1$ vector with 1 in all positions.

In summary, by adding $\boldsymbol{Z}^{ij}$ to the list of free variables, replacing equation (4.14) with equation (4.23), and adding inequalities (4.21) and (4.22) to the constraint set, the reconfiguration/routing problem becomes a linear integer programming problem, which can be solved by standard techniques such as the branch-and-bound method [35]. It should be noted that, for a given fixed topology (i.e., given $\boldsymbol{R}$) and for unicast traffic, this problem reduces to the well-known multicommodity flow problem.

4.3.6 Third Formulation: Routing of Multicast Streams in a WDM Network with Tunable Receivers

An optical WDM network where the receivers are tunable is able to provide *physical multicasting*, by having multiple receivers tune to the same wavelength. As indicated in Chapter 2 and depicted in Figure 4.3, this physical multicasting can be modeled by creating, for each transmitter, a virtual node that is reached with delay and cost equivalent to the delay and cost from the transmitter to the "center" of the network (the WDM star). The link between the real node and the virtual node models the fact that the capacity out of the transmitter is V. The "replication" of the data happens at the virtual node.



Physical Multicasting　　　　Model for Physical Multicasting

Figure 4.3: Physical multicasting in a WDM network

The difference between this formulation in this section and the previous one is that now we allow physical multicasting. To do that, it becomes necessary to augment the network with the virtual nodes, one for each transmitter. Also, we no longer assume that the network has the same total number of transmitters and receivers.

Initially, we define:

N	:	Number of actual (not virtual) nodes in the network
P_i	:	Number of receivers in node i, $i = 1, \ldots, N$
S_i	:	Number of transmitters in node i, $i = 1, \ldots, N$
P	:	Total number of receivers in the network; $P = \sum_{i=1}^{N} P_i$
S	:	Total number of transmitters in the network; $S = \sum_{i=1}^{N} S_i$
K	:	Total number of transmitters and receivers in the network; $K = S + P$

N' : Total number of nodes (real and virtual) in the network; $N' = N + S$

To each transmitter and each receiver in the network we assign a number; transmitters are numbered from 1 to S, and receivers from $S+1$ to K. Using this numbering scheme, when we refer to "transceiver j", it can be either a transmitter or a receiver, according to the value of j. We also number the network nodes from 1 to N and the virtual nodes from $N+1$ to N'; when we refer to "node i", it can be either a real node or a virtual node, depending on the value of i. Virtual node $N+i$ corresponds to transmitter i, $i = 1, \ldots, S$. Defining:

V : Bandwidth of each individual link

T : Number of multicast streams.

s_i : Source node for multicast i

n_i : Number of destinations for multicast i

$\{d_{ik}\}$: Set of destinations for multicast i, $k = 1, \ldots, n_i$

r_i : Bandwidth requirement for multicast i

$\boldsymbol{X}^i$: $K \times n_i$ multicast routing matrix for multicast stream i. $X^i_{jk} = 1$ if transceiver j is used in the multicast path for stream i to reach destination d_{ik}, otherwise $X^i_{jk} = 0$, $k = 1, \ldots, n_i$.

Y^i : $K \times 1$ multicast path vector for stream i. $Y^i_j = 1$ if transceiver j is in the multicast path for stream i, otherwise $Y^i_j = 0$.

M_i : Delay for multicast request i, in hops.

D_i : Latency constraint for multicast request i, in hops.

$\boldsymbol{L}$: Transmitter location matrix ($N \times S$); $L_{ij} = 1$ if transmitter j is located in node i, otherwise $L_{ij} = 0$.

$\boldsymbol{R}$: Receiver location matrix ($N \times P$); $R_{ij} = 1$ if receiver j is located at node i, otherwise $R_{ij} = 0$.

$\boldsymbol{F}$: Wavelength allocation matrix ($S \times P$); $F_{ij} = 1$ if receiver $S+j$ is tuned to the wavelength of transmitter i.

$\boldsymbol{B}^i$: $N \times n_i$ source-destination matrix for multicast stream i; $B^i_{jk} = 1$ if $j = s_i$, $B^i_{jk} = -1$ if $j = d_{ik}$, and $B^i_{jk} = 0$ otherwise, $k = 1, \ldots, n_i$.

β_c : Weight of the cost in the optimization.

β_d : Weight of the delay in the optimization.

The optimum multicast routing routing problem in a WDM network with tunable receivers can be formulated as follows:

GIVEN: $K, N, V, \boldsymbol{L}, \boldsymbol{R}, T, \beta_c, \beta_d; \{\boldsymbol{B}^i\}, \{r_i\}, \{D_i\}, i = 1, \ldots, T$

MINIMIZE:

$$\sum_{i=1}^{T} r_i \left(\beta_c \sum_{j=1}^{K} Y_j^i + \beta_d M_i \right) \tag{4.24}$$

WITH RESPECT TO: $\boldsymbol{F}; \boldsymbol{X}^i, \boldsymbol{Y}^i, M_i, \quad i = 1, \ldots, T$

UNDER CONSTRAINTS:

1. A receiver can be listening to only one wavelength:

$$\sum_{i=1}^{S} F_{ij} = 1, \quad j = 1, \ldots, P \tag{4.25}$$

2. For every stream, there must be a path from its source to each of its destinations:

$$\left[\begin{array}{c|c} \boldsymbol{L} & -\boldsymbol{R} \\ \hline -\boldsymbol{I}_S & \boldsymbol{F} \end{array} \right] \left[\begin{array}{c} \boldsymbol{Xt}^i \\ \hline \boldsymbol{Xr}^i \end{array} \right] = \left[\begin{array}{c} \boldsymbol{B}^i \\ \hline 0 \end{array} \right] \quad i = 1, \ldots, T \tag{4.26}$$

where $\boldsymbol{I}_S$ is the $S \times S$ identity matrix, $\boldsymbol{Xt}^i$ is a matrix with the first S rows of $\boldsymbol{X}^i$, and $\boldsymbol{Xr}^i$ is a matrix with the remaining P rows of $\boldsymbol{X}^i$. This matrix equation can be divided into the following two equations:

$$\boldsymbol{L}\boldsymbol{Xt}^i - \boldsymbol{R}\boldsymbol{Xr}^i = \boldsymbol{B}^i \tag{4.27}$$

$$-\boldsymbol{Xt}^i + \boldsymbol{F}\boldsymbol{Xr}^i = 0 \tag{4.28}$$

3. If a link is in the path from the source to any of the destinations, then it must be included in the multicast path.

$$X_{jk}^i \leq Y_j^i, \quad k = 1, \ldots, n_i, \quad j = 1, \ldots, K, \quad i = 1, \ldots, T; \tag{4.29}$$

4. The delay for a multicast is the delay to the farthest destination:

$$M_i - \sum_{j=1}^{K} X^i_{jk} \geq 0, \quad k = 1, \ldots, n_i, \quad i = 1, \ldots, T; \tag{4.30}$$

5. There is a maximum delay constraint for each of the multicast streams:

$$M_i \leq D_i, \quad i = 1, \ldots, T; \tag{4.31}$$

6. The total flow through a link cannot exceed its bandwidth:

$$\sum_{i=1}^{T} r_i \boldsymbol{Y}^i \leq V; \tag{4.32}$$

7. Integer constraints: no bifurcation of flow; a single path is taken from the source to each of the destinations.

$$\boldsymbol{X}, \boldsymbol{Y} \quad \text{are binary.} \tag{4.33}$$

Receivers cannot be "divided":

$$\boldsymbol{R} \qquad \text{is binary} \tag{4.34}$$

As in the previous formulation, the only non-linear equation in this optimization problem is equation (4.28), which has the $\boldsymbol{F}\boldsymbol{X}\boldsymbol{r}^i$ product. The problem is made linear by adding to the set of free variables the $S \times P$ binary matrices $\boldsymbol{Z}^{ij}$, $i = 1, \ldots, T$, $j = 1, \ldots, n_i$, subject to the following new constraints:

$$Z^{ij}_{kl} \leq Xr^i_{jl} \tag{4.35}$$

$$Z^{ij}_{kl} \leq F_{kl} \tag{4.36}$$

$$i = 1, \ldots, T; \quad j = 1, \ldots, n_i; \quad k = 1, \ldots, S; \quad l = 1, \ldots, P$$

Equation (4.28) then becomes :

$$- \boldsymbol{X} \boldsymbol{t}_j^i + \boldsymbol{Z}^{ij} \mathbf{1} = 0 \tag{4.37}$$

$$i = 1, \ldots, T; \quad j = 1, \ldots, n_i$$

where $\mathbf{1}$ is a $P \times 1$ vector with 1 in all positions.

In summary, by adding $\boldsymbol{Z}^{ij}$ to the list of free variables, replacing equation (4.28) with equation (4.37), and adding inequalities (4.35) and (4.36) to the constraint set, the reconfiguration/routing problem becomes a linear integer programming problem.

4.3.7 A General Linear Programming Formulation for the Optimum Multicast Routing Problem

In this thesis, we have presented several optimum routing formulations, based on integer linear programming, each specifically tailored to a particular scenario. In this section, we present a general formulation that encompasses all the previous ones, and can be used in any of the previous scenarios (although not very efficiently). The main shortcoming the formulation in Chapter 2 is that it cannot accommodate WDM networks. The main shortcoming of the unicast and multicast formulations presented so far for the WDM network is that it implicitly assumes unit link costs and delays. Ideally, we should be able to assign a cost and a delay to each of the transmitters and receivers in the WDM network; when there is a connection between a given transmitter and a given receiver, the delay and cost of the link created will be the sum of the transmitter and receiver costs and delays. For example, if the physical topology of the WDM network is the star, the propagation delay between two nodes corresponds to the propagation delay from the first node to the star (which is proportional to that node's distance to the star) plus the propagation delay from the star to the second node.

Another scenario not included in the previous formulations is the case of tunable transmitters and receivers, when the number of available wavelengths is *smaller* than the number of transmitters (if it is larger, then one would just tune each transmitter to a different wavelength and leave it fixed). In this case, the "transmitter virtual nodes" of the previous section represent the distinct wavelengths, and are not necessarily associated with a specific transmitter. Formally, the identity matrix $\boldsymbol{I}_S$ in equation 4.26 becomes a (non-square) transmitter assignment matrix.

Since this formulation is very similar to the previous one, we use the same symbols, with the following additions:

- $\mathcal{C}$: $K \times 1$ cost vector; $\mathcal{C}_i$ is the cost associated with transmitter i, $i = 1, \ldots, S$, and $\mathcal{C}_{S+i}$ is the cost associated with receiver i, $i = 1, \ldots, P$.
- $\mathcal{D}$: $K \times 1$ delay vector; $\mathcal{D}_i$ is the delay associated with transmitter i, $i = 1, \ldots, S$, and $\mathcal{D}_{S+i}$ is the delay associated with receiver i, $i = 1, \ldots, P$.
- W : Number of available wavelengths.
- $\boldsymbol{G}$: $W \times S$ transmitter assignment matrix; $G_{ij} = 1$ if transmitter j is sending on wavelength i.
- $\mathcal{E}$: Maximum number of receivers that can be connected to a transmitter. If transmitters are tunable, $\mathcal{E} = 1$; if receivers are tunable, $\mathcal{E} = P$. This formulation does not preclude the use of $1 \leq \mathcal{E} \leq P$.

The general formulation is:

GIVEN: $K, N, V, \boldsymbol{L}, \boldsymbol{R}, \mathcal{C}, \mathcal{D}, T, \beta_c, \beta_d, \mathcal{E}; \{\boldsymbol{B}^i\}, \{r_i\}, \{D_i\}, i = 1, \ldots, T$

MINIMIZE:

$$\sum_{i=1}^{T} r_i \left(\beta_c \mathcal{C} \boldsymbol{Y}^i + \beta_d M_i \right) \tag{4.38}$$

WITH RESPECT TO: $\boldsymbol{F}, \boldsymbol{G}; \boldsymbol{X}^i, \boldsymbol{Y}^i, \boldsymbol{Zr}^{ij}, \boldsymbol{Zt}^{ij}, M_i, \quad i = 1, \ldots, T; j = 1, \ldots, n_i$

UNDER CONSTRAINTS:

1. A receiver can be listening to only one wavelength:

$$\sum_{i=1}^{S} F_{ij} = 1, \quad j = 1, \ldots, P \tag{4.39}$$

2. The number of receivers connected to a transmitter can be at most $\mathcal{E}$:

$$\sum_{j=1}^{P} F_{ij} \leq \mathcal{E}, \quad i = 1, \ldots, S \tag{4.40}$$

3. No more than one transmitter can be sending on each wavelength:

$$\sum_{j=1}^{S} G_{ij} \leq 1, \quad i = 1, \ldots, W \tag{4.41}$$

4. A transmitter sends in only one wavelength:

$$\sum_{i=1}^{W} G_{ij} \leq 1, \quad j = 1, ldots, S \tag{4.42}$$

5. For every stream, there must be a path from its source to each of its destinations:

$$\boldsymbol{LXt}^i - \boldsymbol{RXr}^i = \boldsymbol{B}^i \tag{4.43}$$

$$-\boldsymbol{Zt}^{ij}\mathbf{1} + \boldsymbol{Zr}^{ij}\mathbf{1} = 0 \qquad i = 1, \ldots, T; \quad j = 1, \ldots, n_i \tag{4.44}$$

6. Flow to the receivers can only be sent if the link is in place:

$$Zr_{kl}^{ij} \leq Xr_{jl}^{i} \tag{4.45}$$

$$Zr_{kl}^{ij} \leq F_{kl} \tag{4.46}$$

$$i = 1, \ldots, T; \quad j = 1, \ldots, n_i; \quad k = 1, \ldots, W; \quad l = 1, \ldots, P$$

7. Flow from the transmitters can only be sent if wavelengths have been allocated:

$$Zt_{kl}^{ij} \leq Xt_{jl}^{i} \tag{4.47}$$

$$Zt_{kl}^{ij} \leq G_{kl} \tag{4.48}$$

$$i = 1, \ldots, T; \quad j = 1, \ldots, n_i; \quad k = 1, \ldots, W; \quad l = 1, \ldots, S$$

8. If a link is in the path from the source to any of the destinations, then it must be included in the multicast path.

$$X_{jk}^{i} \leq Y_{j}^{i}, \quad k = 1, \ldots, n_i, \quad j = 1, \ldots, K, \quad i = 1, \ldots, T \tag{4.49}$$

9. The delay for a multicast is the delay to the farthest destination:

$$M_i - \sum_{j=1}^{K} \mathcal{D}_j X^i_{jk} \geq 0, \quad k = 1, \ldots, n_i, \quad i = 1, \ldots, T \tag{4.50}$$

10. There is a maximum delay constraint for each of the multicast streams:

$$M_i \leq D_i, \quad i = 1, \ldots, T \tag{4.51}$$

11. The total flow through a link cannot exceed its bandwidth:

$$\sum_{i=1}^{T} r_i \boldsymbol{Y}^i \leq V \tag{4.52}$$

12. Integer constraints: no bifurcation of flow; a single path is taken from the source to each of the destinations.

$$\boldsymbol{X}, \boldsymbol{Y} \quad \text{are binary.} \tag{4.53}$$

Receivers cannot be "divided":

$$\boldsymbol{F} \qquad \text{is binary} \tag{4.54}$$

The product of the allocation matrices and the flows must be binary:

$$\boldsymbol{Zr}, \boldsymbol{Zt} \qquad \text{are binary} \tag{4.55}$$

The above formulation is completely general:

- For fixed-topology networks, one just has to fix the $\boldsymbol{F}$ matrix and set $\mathcal{E}$ to 1; the costs and delays associated with the "transmitters" are set to zero, and the actual link costs and delays are associated with the receivers. In the particular case of unicast sessions, fixed-topology networks, no latency constraints, this formulation reduces to the traditional multicommodity flow problem

- For WDM networks, one can set $\mathcal{E}$ to 1 if the transmitters are tunable, or set $\mathcal{E}$ to P if the receivers are tunable. Note that, if the receivers are tunable, even under unicast traffic it might make sense to tune two receivers to the same transmitter - two unicast streams can be sharing that transmitter's bandwidth, each addressed to a different receiver.
- If the number of wavelengths is bigger or equal to the number of transmitters, one just has to set $\boldsymbol{G}$ to $\boldsymbol{I}_S$.

4.4 Heuristic Algorithms for the Reconfiguration and Routing Problem

The reconfiguration and routing problem, as formulated in section 4.3, is NP-complete, and the exact optimum solution given there has (in the worst case) exponential run time. In this section, we present a number of simpler heuristic solutions. We start by presenting a heuristic algorithm for the unicast case, and use this heuristic algorithm to build minimum-cost and minimum-delay heuristics for the multicast case.

Given a session with one or more streams, we seek to find the logical network topology and the routes for this session. From a high level point of view, the heuristic solutions proposed here start with an arbitrary initial logical topology, and make changes to it considering the streams in the session one at a time. The changes are made using the *Shortest Path with Reconfiguration Algorithm*, a variation of Dijkstra's Shortest Path algorithm proposed by us that works in a reconfigurable network environment. In the following, we first describe the Shortest Path with Reconfiguration Algorithm, and then give the complete reconfiguration and routing heuristics.

4.4.1 The Shortest Path with Reconfiguration Algorithm

Given a source node, Dijkstra's algorithm builds a shortest path *tree* from that node. The tree starts with the source node, and at each iteration a node is added to it in such a way that the paths in the tree are the shortest from the source. When used to find the shortest path between a particular pair of nodes, the algorithm terminates when the destination node is added to the tree, at which point the remainder of the tree is discarded and only the path between the source and the destination nodes is retained.

Our objective is to compute the shortest path in a WDM network, where the topology of the network is a free variable that can also be used to minimize the path length. In the best case, we would just tune a transmitter at the source and a receiver at the destination to the same wavelength, and obtain the shortest possible path, with length equals to one hop. However, this might not always be possible, since transmitters and receivers might be already connected to other nodes. In general, we classify the transmitters and receivers in the network either as *free* or *locked*, and the shortest path algorithm can only reconfigure the free transmitters and receivers, although it might make use of the locked ones in whatever topology they happen to be. If the WDM network supports physical multicasting (i.e., if it has tunable receivers), this is taken into account in the algorithm by *considering all the transmitters as free, without regard to the other connections.* The algorithm described below does exactly this: given the WDM network in a certain logical topology, where some links are free and some are locked, and a source-destination pair, it finds the shortest path between these two nodes, reconfiguring the free links if necessary. In Appendix C, we give a formal description of the algorithm.

Step 1: Using Dijkstra's algorithm, identify: (i) the shortest path between the source and the destination, and (ii) the node closest to the source which has a free transmitter (i.e., either the source itself or the first node added to the shortest path tree that has a free transmitter); this node, if found, will be denoted by *Node A*. Note that if the network is disconnected, there might not be a path between the source and the destination.

Step 2: Using Dijkstra' algorithm in reverse from the destination to the source (i.e., building the tree in reverse), find the node closest to the destination that has a free receiver; this node, if found, will be denoted by *Node B*.

Step 3: If either node A or node B or both were not found, stop. If a path between the source and the destination was found in step 1, it is the shortest path. Otherwise, there is no path. If both node A and node B were found, proceed to step 4.

Step 4: Let L_1 denote the length of the shortest path found in step 1 (make $L_1 = \infty$ if no path was found), and L_2 denote the length of the path obtained by tuning the transmitter in node A to the receiver in node B, and using the shortest path from the source to A, the newly-created A-B link, and the path from B to the destination. If $L_1 \leq L_2$, do not reconfigure the network and use the shortest path from step 1; otherwise, tune A to B and use the path just created, as described above.

NOTE: At most one reconfiguration is needed to obtain the shortest path (and the algorithm above finds it). This can easily be shown by contradiction: assume that the shortest path between nodes S and R requires that node A be reconfigured to connect to node B, and node C be reconfigured to connect to node D. In the absence of tuning constraints, we can reconfigure node A to connect directly to node D, finding a path that is shorter, which contradicts the initial hypothesis.

4.4.2 The Reconfiguration and Routing Heuristic for Unicast Streams

Given a session, the basic idea behind the wavelength assignment heuristic is to take an arbitrary initial topology, and apply the shortest path with reconfiguration to each of the components of the session. The shortest path with reconfiguration algorithm is applied to the streams in the session in decreasing order of bandwidth. In general, the first streams to be routed will be given shorter paths, as more network resources are available. Therefore, it is better to route first the higher-bandwidth streams, to minimize their usage of network resources.

Step 1: Choose an arbitrary initial wavelength assignment. Create a vector $\boldsymbol{U}$, containing the used bandwidth on each transmitter; initially, $U_i = 0$, $i = 1, \ldots, K$. Sort the streams in the session in order of bandwidth.

Step 2: Consider the stream with the highest bandwidth requirement that was not yet processed; let us denote it by stream j. Temporarily prune from the network topology the transmitter/receiver pairs that do not have enough free bandwidth to support the stream, i.e., belonging to the set $\{i : V - U_i < r_j\}$. Mark all the transmitter/receiver pairs belonging to the set $\{i : U_i > 0\}$ as locked, and the remainder as free.

Step 3: Execute the "Shortest Path with Reconfiguration Algorithm" described above for this stream. If successful, update the the $\boldsymbol{U}$ vector as follows: $U_i \leftarrow U_i - r_j$, $i \in$ path.

Step 4: If all streams in the session have been considered, terminate; otherwise, return to step 2.

After this algorithm is run, the initial network topology is transformed into a new topology which matches the session requirements. If all the invocations of the shortest path with reconfiguration algorithm in step 3 are successful, a set of routes for the session is also available; otherwise, the heuristic fails and declares the problem infeasible.

Note that, if the logical network topology has been defined, routing a session using this topology becomes the traditional multicommodity flow problem. More specifically, the $\boldsymbol{R}$ matrix in equation (4.4) ceases to be a free variable, thus making it a linear equation; the optimization problem then becomes equations (4.1), (4.4), (4.5) and (4.7).

As done in Chapter 2 for the routing of multicast streams in fixed-topology networks, this integer linear programming problem can be solved by the traditional branch-and-bound method. Specific features of the problem can be used to prune the search space and speed-up the solution, as proposed by Crowder et al [38]. In fact, the same pruning rules presented in section 2.5.2 apply here; one just needs to remember that there is one single destination in the unicast case. The linear relaxation of the problem can also be efficiently solved by decomposition, with the difference that, in this case, there is only a single level of decomposition - a group of T unicast streams is decomposed into T unicast routing problems. The decomposition equations for this case are well-known and will not be presented here; the reader is referred to [17].

One can further optimize the routing solution by using the topology found by the heuristic, disregarding the routes found in step 3, and re-routing the session using the integer linear programming solution. In some cases, by doing this it is possible to solve a problem declared infeasible by the heuristic.

4.4.3 Using Simulated Annealing to Improve the Heuristic Solution

The *Simulated Annealing* method [67] is an optimization method designed for non-linear integer problems that are difficult to solve analytically. The method starts from a feasible solution, and perturbates this solution to see if it can be improved. Unlike traditional steepest-descent methods, simulated annealing can accept modifications to the current solution that do not improve it. This gives it the potential of moving away from a local optimum, and finding a better solution. The result given by the heuristic described in the previous section can be used as the starting point for the simulated annealing method; from that, the method can potentially identify a better solution, closer to the optimum.

The method mimics the annealing process for a metal or crystal. Initially the metal is melted, and its temperature is very high. The temperature is gradually lowered, and the metal will crystallize in a regular structure, with a minimum of energy. If the temperature is lowered too fast, the regular crystalline structure will not form. The same idea is applied to the optimization problem. The "temperature" controls the probability that a perturbation in the solution that does not improve the objective function is accepted. Initially, the "temperature" is high; it is then gradually lowered, and the solution should "coalesce" into the optimum. The algorithm works in "epochs" of constant temperature; each epoch is composed of a fixed number of perturbations in the current solution. A perturbation, in the case of the WDM network, corresponds to exchanging the connections of two transmitter-receiver pairs, as depicted in Figure 4.4 [68]. After the network topology is changed, the routes can be re-optimized using integer programming, as described in the previous section.

The algorithm keeps track of the "best" solution it has seen; let us denote the objective function of this solution as L^*. If a perturbation improves the objective function, it is

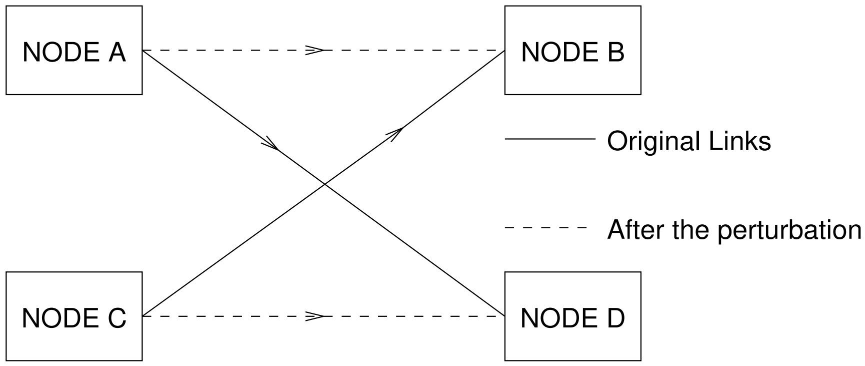


Figure 4.4: The perturbation

accepted. Otherwise, it may be accepted with probability $\exp(-\frac{L-L^*}{T_e})$, where L is the new value of the objective function after the perturbation, and T_e is the current temperature. For routing in the WDM network, the "temperature" has no physical meaning; it is just a number selected by trial and error, based on the order of magnitude of the values of the objective function. To implement a simulated annealing optimization, one must find (experimentally, by trial and error) the appropriate values for the following parameters:

- Size (i.e., number of perturbations) of the annealing epoch.
- Temperatures:
  - Initial and final temperatures.
  - Rate of temperature decrease between annealing epochs.

4.4.4 Heuristic Algorithms for Multicast Routing In WDM Networks

In this section, we present minimum-cost and minimum-delay heuristic algorithms for WDM networks, based on the Shortest Path with Reconfiguration algorithm. The algorithms presented here are for a single multicast stream; bandwidth constraints are taken care of by pruning from the network those links with insufficient free capacity, and multiple streams are handled sequentially, as done for the unicast case.

4.4.4.1 Minimum Delay Heuristic Algorithm

On a fixed-topology network, minimum-delay multicast routing can be achieved by finding the shortest path from the source to each of the destinations and then merging these paths. For a single multicast, this is an exact algorithm, i.e., the routes found are optimal. The same idea can be applied to the WDM network: one would use the shortest path with reconfiguration algorithm between the source and each of the destinations, and merge the paths. However, unlike its fixed-topology counterpart, this algorithm might not be optimal even for a single stream, depending on whether or not physical multicasting is available.

In the fixed-topology network, using shortest path (with link delays as link labels) leads to minimum-delay paths. If this process is repeated from a source node to all its destinations, the delay to each destination will be minimum, and thus the delay of the multicast path (defined as the delay to the farthest destination) will be minimum. The paths can be computed independently, and then merged. In a WDM network, the Shortest Path with Reconfiguration algorithm is also optimum for a single destination. However, if physical multicasting is not allowed (or possible), as routes are computed, links get locked and the topology changes; the tree found is dependent on the order in which the routes to the individual destinations are computed. In this case, the global optimum cannot be decomposed into a number of individual subproblems. If physical multicasting is allowed (i.e., the receivers are tunable), then this algorithm is optimum, because: (i) all transmitters are always free, regardless of the tuning of the receivers, and (ii) if the path to two different destinations require the tuning of the same receiver, they will require the *same* tuning, and thus will not interfere with each other.

4.4.4.2 Minimum-Cost Routing Heuristic Algorithm

The minimum-cost multicast routing heuristic algorithm presented here is based on the Takahashi-Matsuyama (TM) minimum Steiner tree heuristic. The basic idea in the original TM algorithm is to start building a tree with the source node, and at each iteration add to the tree the destination closest to it. The same idea can be used in the WDM network, but here we use the shortest path with reconfiguration algorithm.

Formally, the minimum-cost heuristic is:

INPUTS: A WDM network, where some transmitters and receivers are locked into a certain topology, and some are free; and a multicast to be routed, characterized by its source node s, and its n destinations $\{d_1, d_2, \ldots, d_n\}$.

OUTPUTS: The updated topology for the WDM network, and the multicast path from the source to the destinations.

ALGORITHM:

Step 1: Add the source node s to the multicast path. Create a virtual node V where the path is "collapsed", i.e., all the nodes in the multicast path are removed from the network and node V "inherits" their transmitters and receivers.

Step 2: Find the shortest path with reconfiguration from V to all the multicast destinations not yet in V. Each path is computed independently from the others, i.e., without taking into account the changes in topology required by the other paths.

Step 3: From all the paths found in step 2, choose the shortest and discard the others. "Collapse" all the nodes in the path into V.

Step 4: If all the destinations have been added to V, stop. The desired multicast path is the result of merging all the paths in V. If there are still destinations not in V, return to step 2.

4.5 Performance Evaluation of The Reconfiguration and Routing Heuristic for Unicast Traffic

In this section, we present an evaluation of the reconfiguration and routing heuristic described in the previous section, considering a single session in an empty network. Ideally, one would compare the results of the heuristic with the exact (optimum) solution; however, we derive an upper bound in performance, which is much simpler to compute than the optimum, and use it in the evaluation. We also compare the performance of the WDM reconfigurable network with a fixed-topology network with the same number of nodes and links; for the evaluation, we chose the ShuffleNet [62]. The routes in the ShuffleNet were computed using integer programming [17, 35]. In summary, the main objectives of this section are to compare the performance of the heuristic proposed in the previous section for unicast traffic with the upper bound, and with the performance of a fixed-topology network. We also seek to evaluate the improvement in the heuristic brought upon by the simulated annealing method.

4.5.1 Evaluation Scenarios and Performance Measures

The first step in the evaluation is defining the evaluation scenarios and performance measures under which the algorithms are to be compared:

Evaluation Scenarios

For the evaluation, we consider networks with $N = 8$ nodes; each node has 2 optical transmitters and 2 optical receivers ($K = 16$). We consider the routing of a single session over an idle network (this is equivalent to making the session arrival rate, λ, much lower than the average session duration, $1/\mu$). The session is composed of T streams, $10 \leq T \leq 20$, and the sources and destinations of the streams are uniformly distributed over the network. The bandwidth requirement for each stream is chosen at random between 0 and 100% of the link bandwidth, using the following bimodal distribution (m is the average bandwidth requirement, expressed as a fraction of the link bandwidth V):

$$p_R(r) = \begin{cases} \frac{1-m}{m} & \text{if } r < m \\ \frac{m}{1-m} & \text{if } r \geq m \end{cases} \tag{4.56}$$

The average bandwidth required by the session, as a fraction of the total bandwidth in the network, is given by mT/K; we denote this quantity as the *Offered Load* to the network. For the evaluation, we vary the offered load between 0 and 0.9.

Performance Measures

The most basic performance measure is the *Session Acceptance Probability.* Given a large sample space of sessions, the session acceptance probability is the fraction of this sample space that can be routed in the network (i.e., the feasible region of the optimization problem described in 4.3 is not empty). Since we seek to minimize the *Average Path Length* (in hops), this is another useful performance measure. Note that these two performance measures are related: for a given algorithm, the average path length indicates the usage of network resources when routing a session. If this value is high, it is likely that the blocking probability will also be high.

4.5.2 An Upper Bound on the Session Blocking Probability

Given a session composed of T streams as described above, it might be impossible to route this session, regardless of the network reconfiguration algorithm. In this section, we establish a necessary condition for a session to be accepted.

If a session is to be accepted and routed, for each node in the network there should be at least one way of distributing the streams that originate from it (terminate at it) among its transmitters (receivers). For example, it is not possible to have three streams requesting 60% of a link's bandwidth originating at a node that has only two transmitters, although the three streams combined request less bandwidth than the total available. Formally, a **necessary** condition for the existence of a solution for the routing/reconfiguration problem in section 4.3, is that, for every node k, $k = 1, \ldots, N$, at least one feasible solution is found for each of the problems below:

Problem 1: define $\mathcal{A}_k$ to be the set of streams that originate at node k; find a set α_{ij}, $i \in \mathcal{A}_k$, $j = 1 \ldots, S_k$ such that:

$$\sum_{i \in \mathcal{A}_k} \alpha_{ij} r_i \leq V, \quad j = 1, \ldots, S_k \tag{4.57}$$

$$\sum_{j=1}^{S_k} \alpha_{ij} = 1, \quad \alpha_{ij} \text{ is binary}, \quad \forall i \in \mathcal{A}_k$$

Problem 2: define $\mathcal{B}_k$ to be the set of streams that terminate at node k; find a set β_{ij}, $i \in \mathcal{B}_k$, $j = 1 \ldots, P_k$ such that:

$$\sum_{i \in \mathcal{B}_k} \beta_{ij} r_i \leq V, \quad j = 1, \ldots, P_k \tag{4.58}$$

$$\sum_{j=1}^{P_k} \beta_{ij} = 1, \quad \beta_{ij} \text{ is binary}, \quad \forall i \in \mathcal{B}_k$$

Although in general these problems could be solved by linear integer programming, for the purposes of this chapter we just implemented an exhaustive search, due to the relatively small number of streams per session in the cases evaluated.

4.5.3 Numerical Results

In this section, we present numerical acceptance probability and average path length results for the scenarios described above. In all cases, we are routing a single session on an empty network.

As indicated in section 4.4.3, before the simulated annealing method can be employed, we need to determine reasonable values for its parameters, based on test runs. These parameters are the number of perturbations in the annealing epoch, and the temperatures. Moreover, we need to estimate the size of the session sample space to estimate the acceptance probability. Therefore, we performed three sample runs, varying the number of perturbations in the annealing epoch, and the number of sessions in the sample space. We restricted ourselves to a single annealing epoch, with the temperature fixed at 1. Each session had $T = 12$ requests, and the average bandwidth per stream was set to $m = 0.35$. The average path length and acceptance probability are given in table 4.3, for the ShuffleNet, the reconfiguration and routing heuristic, and the simulated annealing (which uses the heuristic as a starting point). Based on the results from table 4.3, we decided to fix the number of perturbations at 100 and the number of sessions tried for each load at 150. We chose to keep the simulated annealing solution at one single epoch of temperature 1 until we could compare the results obtained with the upper bound derived in section 4.5.2.

We have simulated the scenarios described in section 4.5.1, and obtained both the session acceptance probability and the average number of hops, as a function of the offered load, for sessions composed of 10, 15 and 20 streams. We also obtained the same performance measures for an 8-node ShuffleNet, with 2 transmitters and 2 receivers per node (i.e., the same size as the WDM network under evaluation), using exactly the same sessions. The

Table 4.3: Number of Sessions and Annealing Epoch Size

			ShuffleNet		WDM Network			
					Heuristic		Sim. Anneal.	
Run	Pert.	Sessions	Len.	Prob.	Len.	Prob.	Len.	Prob.
1	100	100	2.0209	34%	1.11343	87%	1.10671	89%
2	100	500	2.03492	33.4%	1.10925	89.4%	1.10812	91.4%
3	1000	100	1.99385	36%	1.0983	86%	1.09366	90%

Len.: Average path length.

Prob.: Acceptance probability.

simulation results are given in Figures 4.5, 4.6 and 4.7, where we plot the session acceptance probability as a function of the offered load for the ShuffleNet and for the WDM reconfigurable network, for 10, 15 and 20 streams per session respectively, as well as the upper bound from section 4.5.2. For the WDM network, the plots show the session acceptance probability for the heuristic and for the simulated annealing. Since the heuristic solution was the starting point for the simulated annealing, its results have to be better or equal than those of the heuristic. The main conclusions from these plots are:

- The fact that the session acceptance probability for the proposed heuristic is close to the upper bound indicates that there is no need to pursue the optimum solution, since the room for improvement is *at most* the difference between the two solutions.

- The simulated annealing, in general, improved very little over the heuristic solution. Since there is little room for improvement anyway, we decided not to pursue annealing solutions with multiple epochs.

- As expected, the reconfigurable network significantly outperforms the fixed-topology network (the ShuffleNet), even under uniformly-addressed traffic. For example, for a 90% session acceptance probability, the reconfigurable network can carry twice the load of the ShuffleNet, for 15-stream sessions.

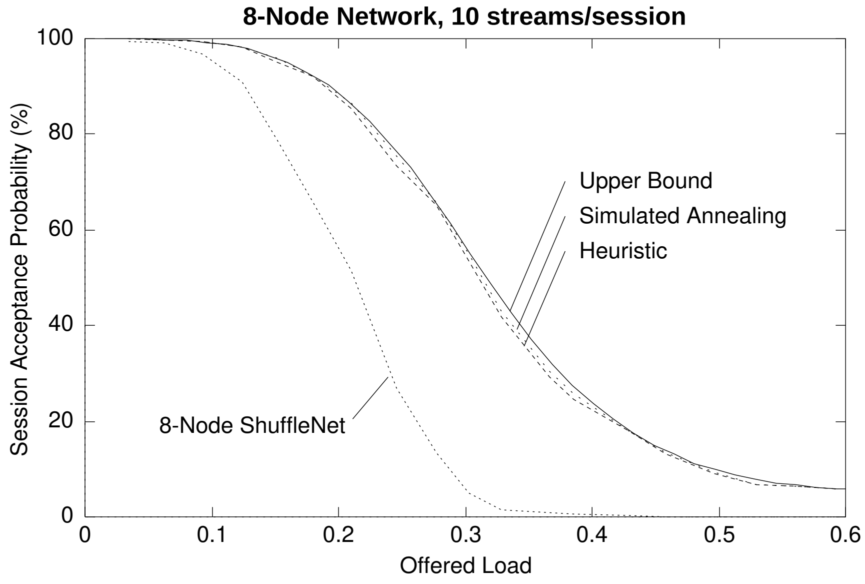


Figure 4.5: Session Acceptance Probability for 10 streams per session (150 sessions per point)

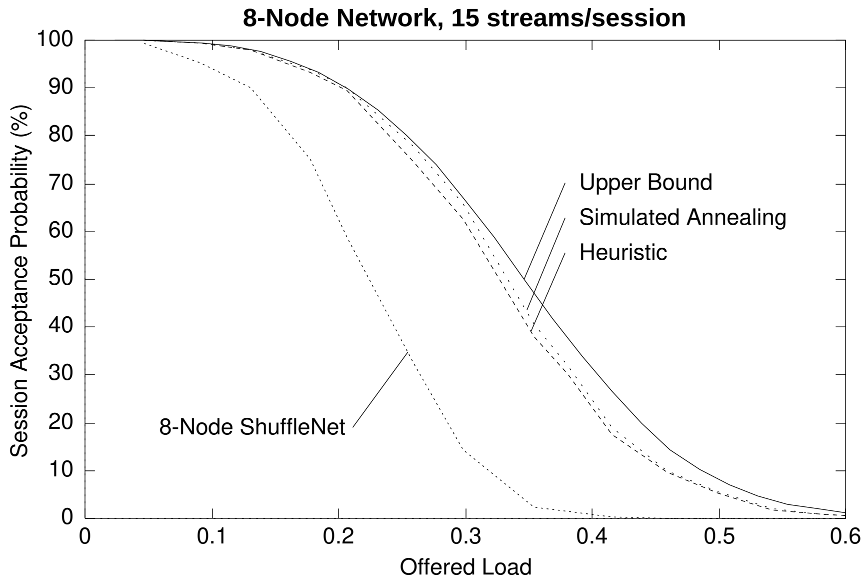


Figure 4.6: Session Acceptance Probability for 15 streams per session (150 sessions per point)

Figure 4.8 shows a comparison of the session acceptance probability for 10, 15 and 20 streams/session, both for the WDM network (using the routing and reconfiguration heuristic) and the ShuffleNet. As shown in Figure 4.8, for the same load, a session with a higher number of streams will have a higher probability of being accepted, because the bandwidth of the individual streams will be lower, thus allowing more freedom in arranging them. This is

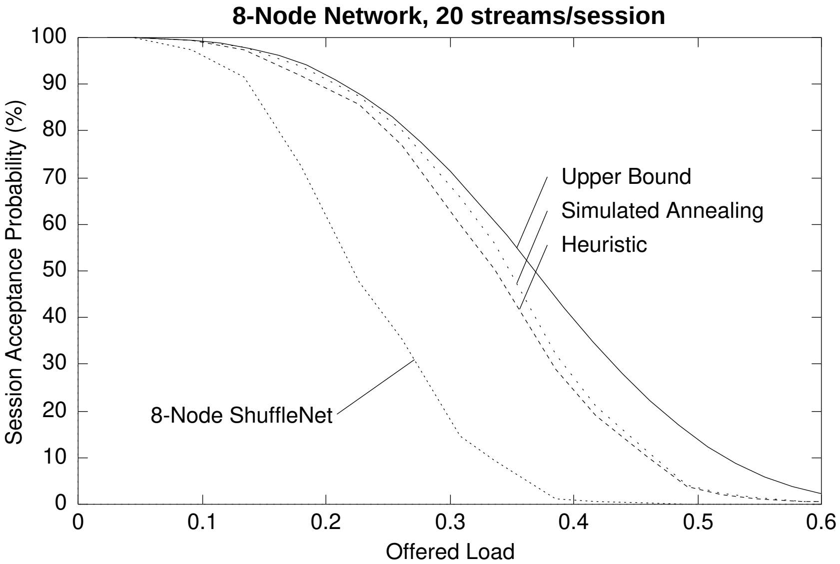


Figure 4.7: Session Acceptance Probability for 20 streams per session (150 sessions per point)

always true for the ShuffleNet. For the reconfigurable network, however, at very high loads, this trend is reversed - the performance for sessions with smaller number of streams is better because (for the sessions accepted) it is possible to dedicate a link to each stream.

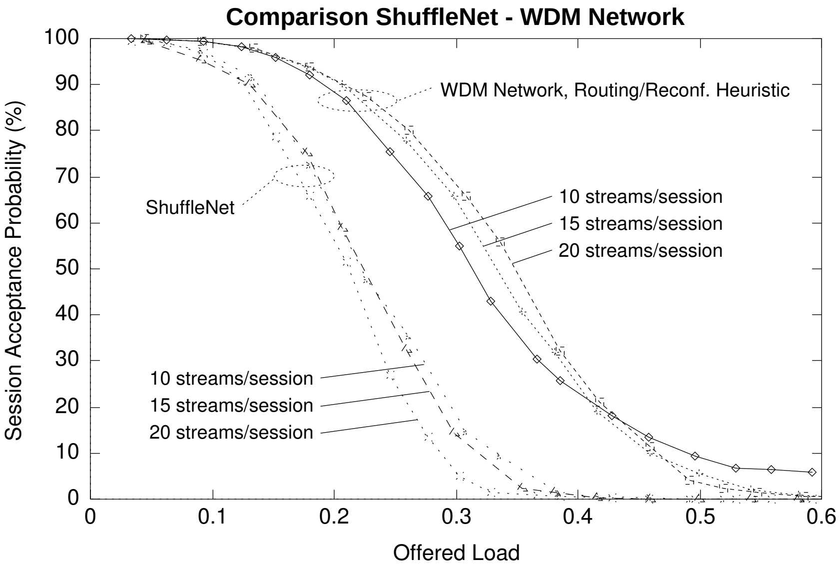


Figure 4.8: Comparison of the Session Acceptance Probability for the ShuffleNet and the WDM Network, 150 sessions/point

Figure 4.9 shows the average path length as a function of the offered load, for 20 streams per session (the plots for 10 and 15 streams/session are similar), and it further confirms our observation that the performance of the reconfigurable network is better than the fixed-topology one; while the average path length for the ShuffleNet is around 2 hops, the path length for the reconfigurable network, even at high loads, is close to 1 hop, which is, of course, the minimum possible. Again, there is very little improvement by using the simulated annealing method.

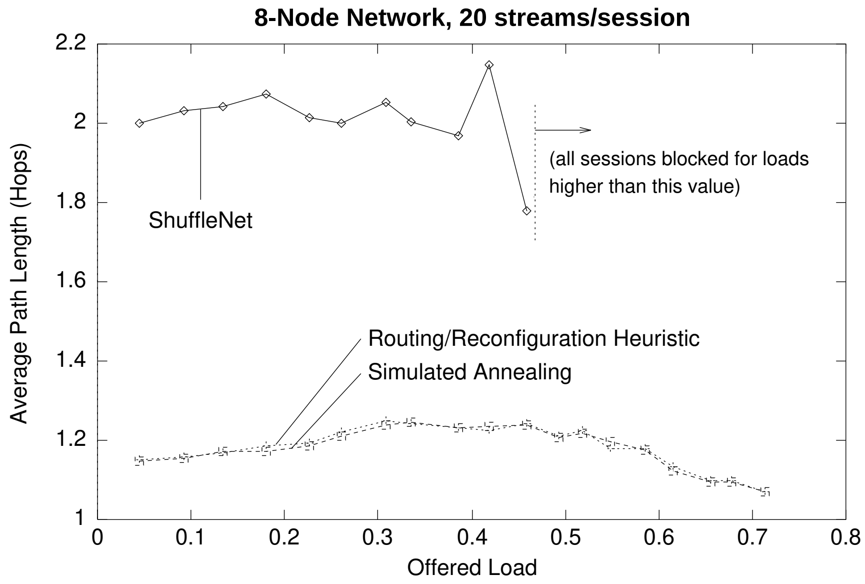


Figure 4.9: Average Path Length for 20 streams/session, 150 sessions/point

4.6 Evaluation of the WDM Network for a Dynamic Traffic Model Under Unicast Traffic

In the previous section, we evaluated the reconfiguration/routing algorithm in a static environment, i.e., the routing of a single session with a fixed number of streams on an idle network. Although this evaluation was useful to show that there is no motivation to pursue the optimum, it is not an indication of the actual performance of the algorithm in a realistic environment. In this section, we evaluate the WDM network and the routing/reconfiguration in a realistic environment, where sessions arrive, are routed (or dropped), and if accepted stay in the network for a certain period of time. We first describe the operation of the network in such a scenario, keeping in mind the traffic requirements, and then evaluate its performance, which we compare to that of a centralized switch.

4.6.1 Operation of the WDM Network in a Dynamic Environment

The main difference between a WDM network and a traditional mesh network is that the former is able to dynamically change its topology; with current optical components, this can be accomplished in sub-millisecond time. The ability to change the network topology at will during operation, in a de-centralized fashion, gives rise to the following issues:

Control of the network: Control of the network is distributed; therefore, messages about the network topology have to be exchanged by the nodes. There are two ways to do that: (i) have a separate control network, whose topology is fixed (e.g., a ring), and send the control messages over this network; and (ii) keep the optical network strongly-connected, and reserve a certain amount of bandwidth in each link for management purposes. In this latter case, reconfigurations that partition the network should not be allowed. We chose to keep the network topology strongly-connected at all times; therefore, in Appendix C we describe a simple modification of the Shortest Path with Reconfiguration algorithm to keep the network connected by performing a secondary reconfiguration, if necessary. Therefore, all transmitters and receivers in the optical network are kept connected (tuned) at all times, even if they are not being used for data traffic.

Re-routing established sessions in use: For stream traffic, if a link in use is reconfigured, the stream has to be interrupted for re-routing. Video/audio streams can tolerate interruptions and delay variations as long as the receiving end pre-buffers a certain amount of data, to keep playing during interruptions. For interactive traffic, however, the amount of pre-buffering cannot be very large as it adds latency to the communication; it is generally recognized that the latency for interactive communications should be less than 200 ms. The decision of *when* to reconfigure the network represents a tradeoff between performance (in terms of session blocking probability) and the number of times an established stream is re-routed during its lifetime. The extremes for this tradeoff are:

- Reconfigure only the idle transmitters and receivers : this has the advantage that existing connections are never disturbed, but at high loads, it is unlikely that a link is completely idle; the network will find itself "locked" in a random configuration that is far from optimal, and will generally perform worse than a regular fixed-topology network under the same traffic conditions.
- Reconfigure the network at each arriving session: one could consider the arriving session and the sessions already established in the network as a new "larger" session to be routed on an empty network. This has the advantage that the network is always optimal, but a given stream might be re-routed an excessive number of times.

Considering the issues listed above, we propose the following model of operation for the reconfigurable network, which is illustrated in Figure 4.10:

- The network starts with some arbitrary strongly-connected topology.
- When a session arrives, it is either routed or blocked; blocked sessions are cleared.
- The reconfiguration/routing algorithm used to accept a session and route it is:

  Step 1: Try to route the session on the current network topology, using the shortest path with reconfiguration algorithm; unused links can be reconfigured. Prior to routing each stream in the session, temporarily prune from the network topology those links that do not have enough free bandwidth to support it. A session can be accepted only if the routing of all its component streams is successful. If the session was successfully routed, accept this route. The streams already established in the network will not be disturbed. Otherwise, proceed to step 2.

Step 2: Since no path was found in step 1, the only alternative to accept this session would be to re-arrange existing connections. We use the unicast heuristic presented in section 4.4 to compute the new topology and routes, considering as our "session" the existing streams and the new session being added.

Step 3: If the reconfiguration/routing in step 2 was successful, we accept the new session and implement the reconfiguration. Otherwise, we block the incoming session and the network topology remains unchanged.

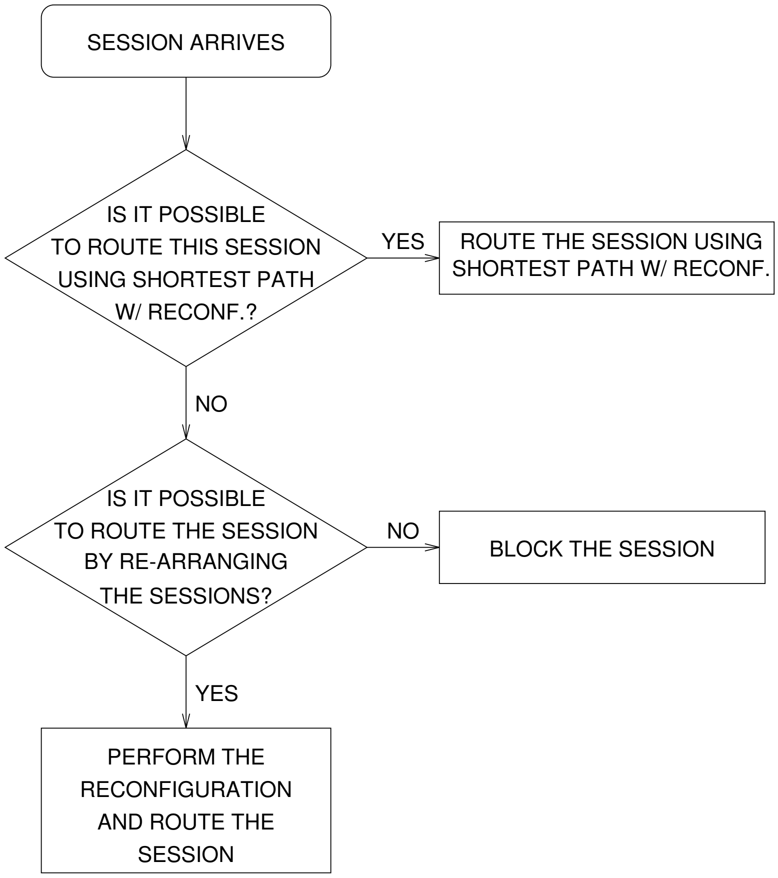


Figure 4.10: Dynamic operation of the WDM Reconfigurable Network

In summary, we reconfigure the network only when the cost of not doing so is to block a session; this way, we achieve the same blocking probability as if we were to reconfigure at every arriving session, while minimizing the number of times a given stream is re-routed.

4.6.2 Numerical Results

In this section, we present simulation results for the WDM network in a dynamic environment, where sessions arrive according to a Poisson process, and if not blocked, stay for an exponentially-distributed amount of time. The performance measures of interest are:

- Session blocking probability: probability that an arriving session cannot be routed and is blocked.
- Average time between successive reconfigurations.
- Average path change for re-routed streams; the path change is defined as the difference between the longest and the shortest paths experienced by the stream during its lifetime. This is a measure of the delay jitter introduced by the reconfiguration. This jitter has to be taken into account when defining the receive buffer sizes for video and audio streams.

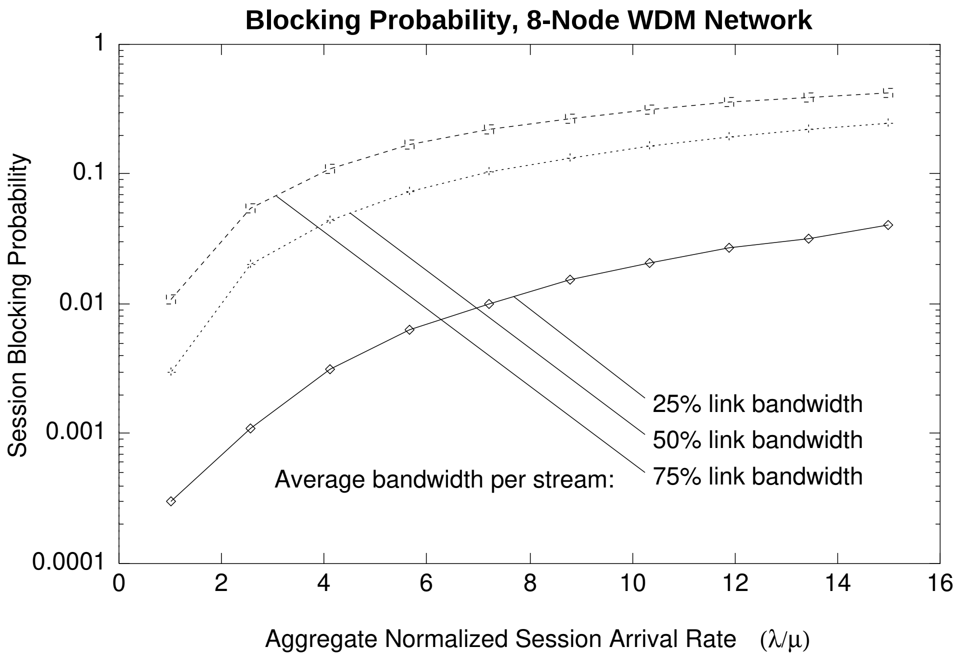


Figure 4.11: Session Blocking Probability (50,000 routes per point)

Figure 4.11 shows the session blocking probability in the WDM network as a function of the session arrival rate, and Figure 4.12 gives the breakdown of the blocking probability as a function of the stream bandwidth, for case where the average bandwidth requested is 50%. As Figure 4.12 indicates, streams requesting higher bandwidths are more likely to be blocked.

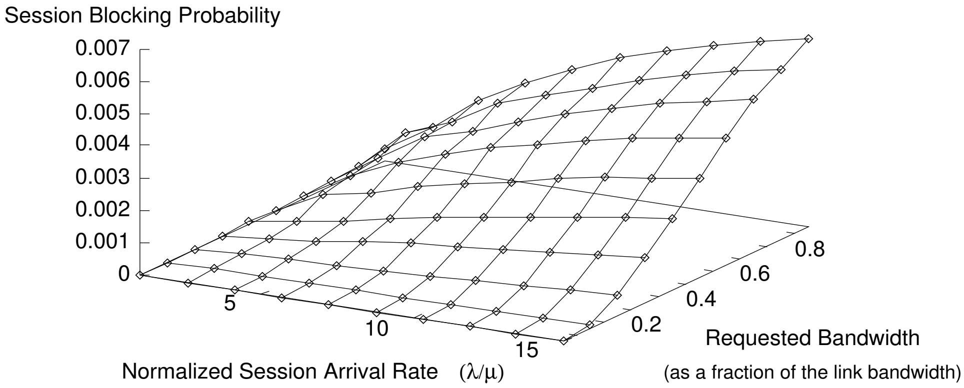


Figure 4.12: Session blocking probability, as a function of the stream bandwidth and the session arrival rate; average session bandwidth of 50%

Figure 4.13 shows the average path length as a function of the session arrival rate, for various values of the average stream bandwidth. At low arrival rates, the average path length is close to 1, since there is a high likelihood that the transceivers will be free. As the traffic load increases, the path length increases, as transceivers become locked and the streams are forced to take longer paths. Figure 4.13 also shows that the higher the average stream bandwidth, the lower the path length; the reason is that, since high-bandwidth streams use more resources, the network "fills up" faster, and only the sessions that lead to shorter path lengths can be accepted.

Figure 4.14 shows the average time between re-routes as a function of the session arrival rate. Note that the time is given as a multiple of the session duration, i.e., a value of 10 means that the average time between reconfigurations is 10 times the lifetime of a session; in the average, approximately only one out of 10 sessions will be re-routed during its lifetime. The figure shows that, at low loads, re-routing is seldom employed, becoming more frequent only at very high traffic loads. An interesting effect shown in Figure 4.14 is that, for a given session arrival rate, the average time between re-routes exhibits a minimum in relation to the average stream bandwidth. This happens because, when the average bandwidth is lower, the network blocks less frequently; therefore, there are longer intervals between re-routing.

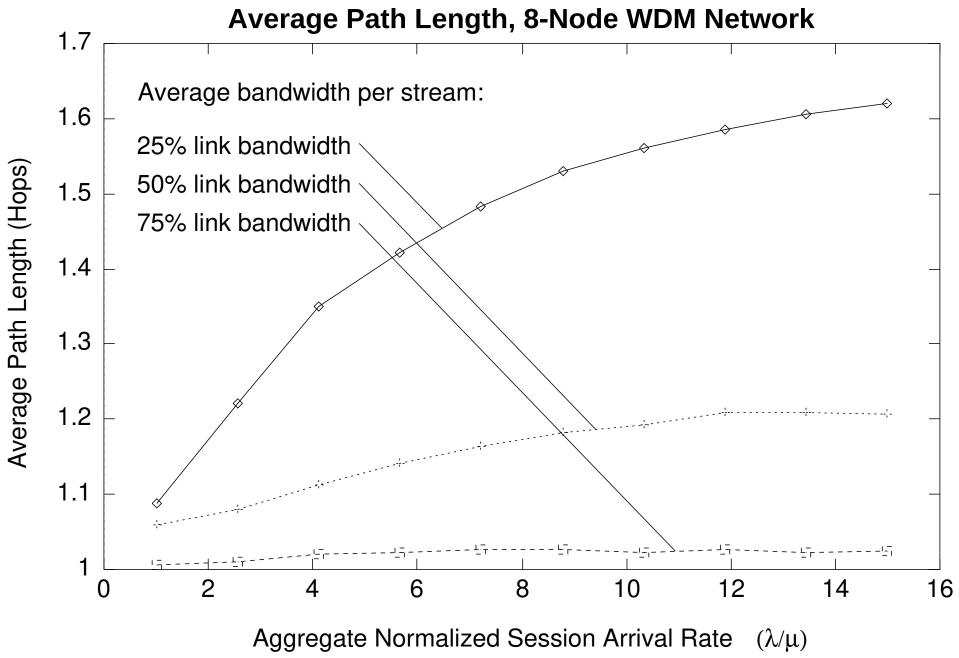


Figure 4.13: Average Path Length in the WDM Network

As the bandwidth increases, blocking increases, but the new sessions can still be accepted after re-routing. However, at high bandwidths, it is less likely that the arriving session can be accepted even with re-routing; re-routing is again a less frequent event.

We also computed the average change in the path (difference between the maximum and the minimum paths during the lifetime of the stream), and found it to be under 0.6 hops in all cases, as shown in Figure 4.15, where we plot the average path change as a function of the session arrival rate. The figure indicates that, at low loads, most streams are not re-routed; as the load increases, the average path change increases. Lower bandwidth streams will suffer higher path changes than streams requesting higher bandwidths.

We note that the WDM network can be thought of as a "distributed switch", where the switching function is performed at the nodes and the "center" of the network is completely passive. However, the same function could be performed by a centralized switch (such as an ATM switch); routing is trivial (all paths are of the form source → switch → destination), and, if the switch is non-blocking, there is no contention inside the network: as long as there is available bandwidth in the link from the source to the switch and in the link from the switch to the destination, the stream can be accepted. So, it is important to determine if,

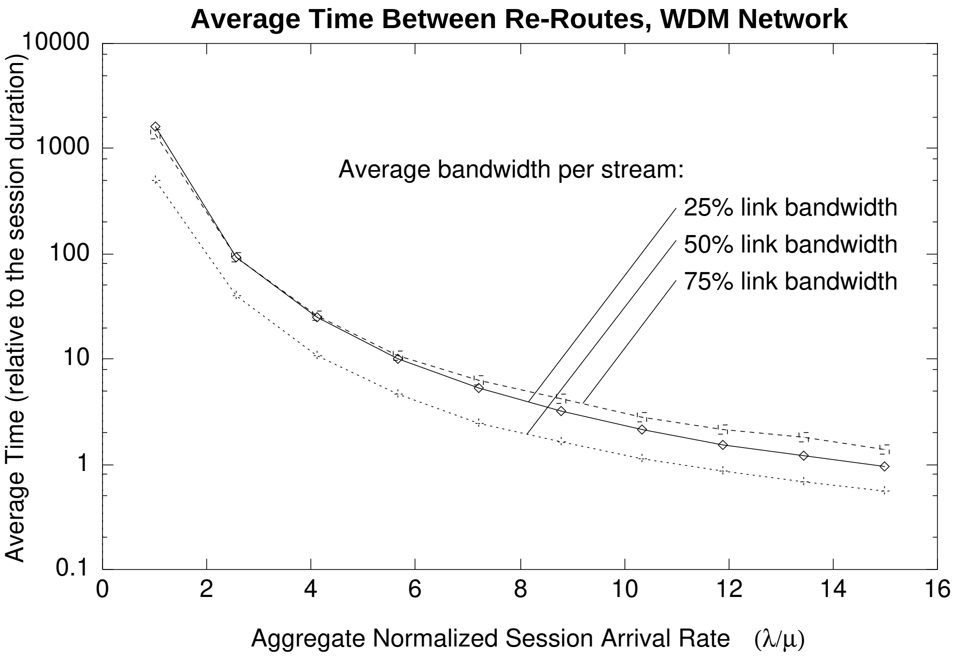


Figure 4.14: Average Time Between Re-Routes in the WDM Network

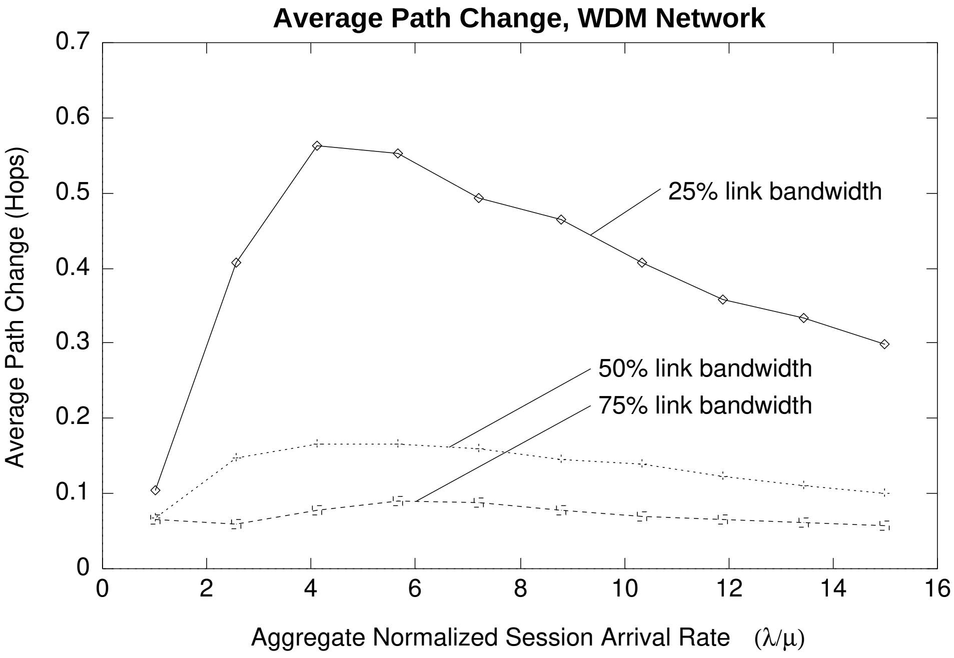


Figure 4.15: Average Path Change in the WDM Network

from a performance point of view, there is any advantage in using a WDM network.

The two network configurations, using a "distributed switch" (WDM) and a centralized switch, are shown in Figure 4.16. For the centralized switch, there are two pairs of optical transmitters/receivers, one to carry the traffic from the node to the switch, and the other to carry the traffic from the switch to the node. A WDM network with the same number of optical transmitters and receivers would have two transmitters and receivers per node. We claim that the complexity of the two networks depicted in Figure 4.16 is approximately the same. The optical transceivers are more complex in the WDM case, but the "center" of the network is passive. On the other hand, in the centralized switch scenario, all the complexity is moved to the center of the network, and the optical transceivers are simpler. To complete the evaluation scenario, we still have to choose the transmitter/receiver data rates in the WDM case (V_{WDM}) and in the centralized switch case (V_{SW}). Two cases are possible:

- Same data rate for all transmitters and receivers in both situations, $V_{SW} = V_{WDM}$. This scenario corresponds to using the "same" transmitters and receivers both for the switch and for the WDM network. The switch has the advantage of no internal blocking, and a shorter path length; the WDM network has the advantage of having twice the output bandwidth per node, but some of this capacity has to be used to forward traffic from other nodes. We will denote this switch as "switch 1".

- Same output bandwidth for each node in both situations, $V_{SW} = 2V_{WDM}$ ($S_i = P_i = 2$), i.e., although the same number of transmitters and receivers is used in both networks, the ones connected to/from the switch have twice the data rate as the ones in the WDM network. The performance of this switch is the same as the upper bound on the performance of the WDM network previously discussed. We will call this "switch 2".

We have simulated the WDM network described above, as well as switch 1 and switch 2. In all cases, the same requests are offered to the three networks. For the all the evaluations in this section, sessions are composed of a single stream.

Figure 4.17 shows the session blocking probability for the three networks as a function of the session arrival rate. The plot shows that, as expected, the performance of the WDM network is lower than that of switch 2, but not significantly. The performance of switch 1, however, is much lower than that of the WDM network. In Figure 4.17, the average bandwidth requested per stream was set to 25% of the link bandwidth; we repeated the evaluation for other values of the stream bandwidth and found similar results.

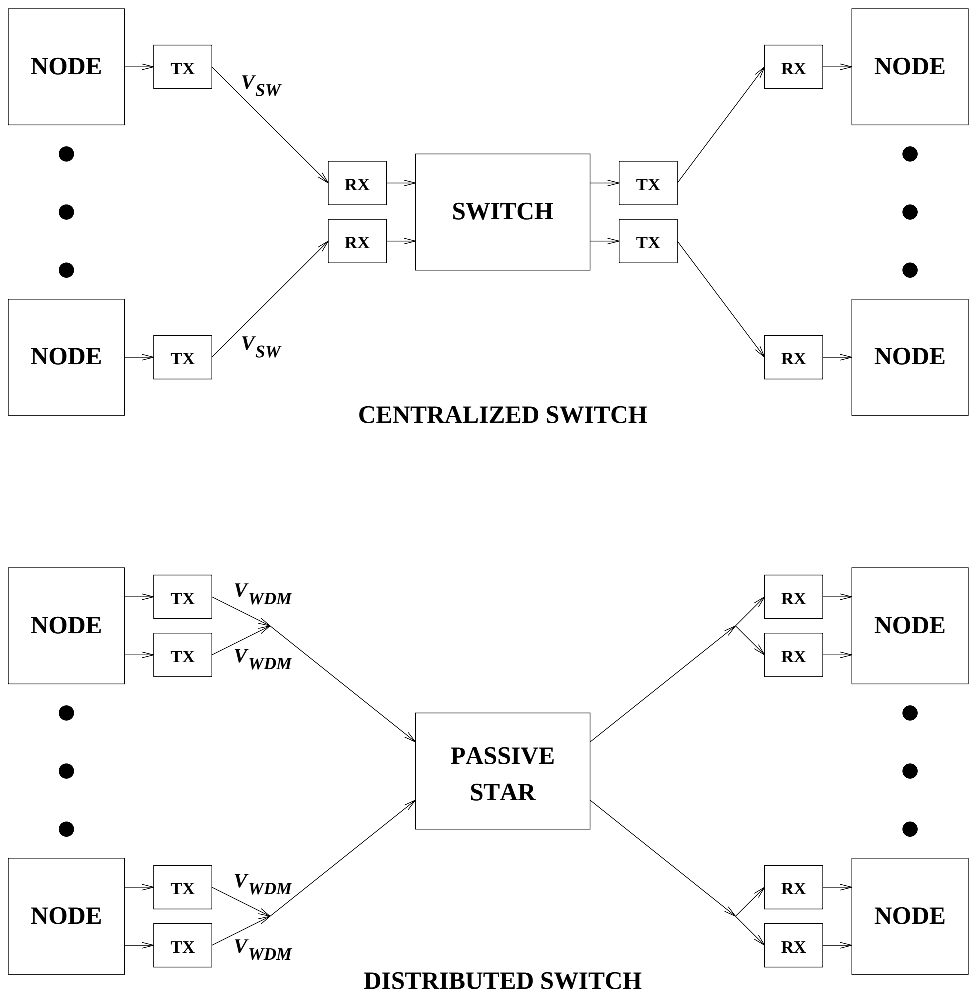


Figure 4.16: Distributed versus Centralized Switching

In summary, we have shown that the performance of a WDM reconfigurable network is much superior to that of a centralized switch with roughly the same amount of optical hardware (switch 1 in the discussion), and is comparable to the performance of a switch using optical transmitters and receivers at twice the rate of the WDM counterparts (switch 2 in the discussion). The WDM network will introduce an additional delay jitter due to stream re-routing. We have shown that, under reasonable traffic conditions, this re-routing represents a small effect, and will affect more the low-bandwidth streams than the higher-bandwidth ones. Moreover, if a specific stream cannot be re-routed for some reason[2], one just needs to

[2]The amount of buffering in the end stations limits the number of times a stream can be re-routed. If a stream is destined to a station with very small buffers, it might not have enough "jitter budget" to support re-routing.

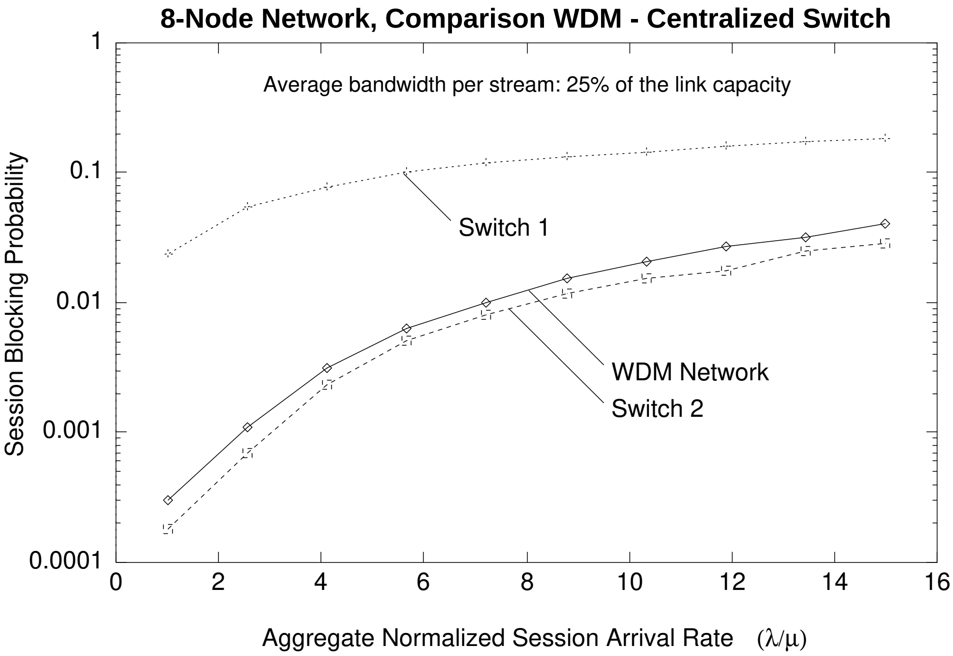


Figure 4.17: Blocking Probability results (single stream per session)

mark the transmitters/receivers it is using as locked, and that stream will not be disturbed by network reconfiguration.

4.7 Evaluation of the Multicast Routing Heuristic Algorithms for WDM Networks

In this section, we present an evaluation of the heuristic algorithms proposed in section 4.4 for multicast routing in a WDM network. As done in Chapter 3, we first evaluate the performance in a baseline case, and then determine how variations in the baseline case change the performance measures.

4.7.1 The Baseline Case

The baseline case used here is similar to the one used in the evaluation in Chapter 3. The following elements characterize the baseline case:

Traffic Model: We use the same traffic model as in the baseline case for Chapter 3: single-stream sessions, sources and destinations uniformly distributed in the network, stream bandwidths set to 10% of the link bandwidth, number of destinations uniformly distributed between 1 and 10. Sessions arrive according to a Poisson process, and if accepted remain in the network for an exponential period of time. No latency constraints were assumed.

Network Scenario: 12-node networks, each node having 2 transmitters and 2 receivers. The network starts with a strongly-connected topology generated at random. We consider the following three kinds of networks:

- networks with tunable transmitters (no physical multicast);
- networks with tunable receivers (physical multicast allowed); and
- fixed-topology networks (corresponding to the initial topology of the WDM network, which is a strongly-connected topology generated at random).

Transmitter/receiver costs (vector $\mathcal{C}$ in section 4.3.7 were set to 0.5, making the cost of a path equal to the number of hops in the path. The distances between the nodes and the WDM star coupler were generated at random, uniformly between 0 and 15; for each node, delays for its transmitters and receivers (vector $\mathcal{D}$ in section 4.3.7) were set to the distance to the star coupler.

Algorithms Evaluated: For each network scenario, we evaluated the minimum-cost and the minimum-delay heuristic. If the network under consideration allowed physical multicast, it was employed. For the fixed-topology case, the minimum-delay heuristic reduces to the traditional shortest path, and the minimum-cost heuristic to the Takahashi-Matsuyama algorithm. As done in the WDM unicast case, we allowed re-routing of existing streams only when doing so would allow the network to accept a session that it would otherwise block. For the baseline case, we do not impose the constraint that the network must remain strongly connected.

Performance Measures: The main performance measure is the blocking probability. Other performance measures, such as average time between reconfigurations, algorithm run time and average cost/delay of the established routes were also obtained.

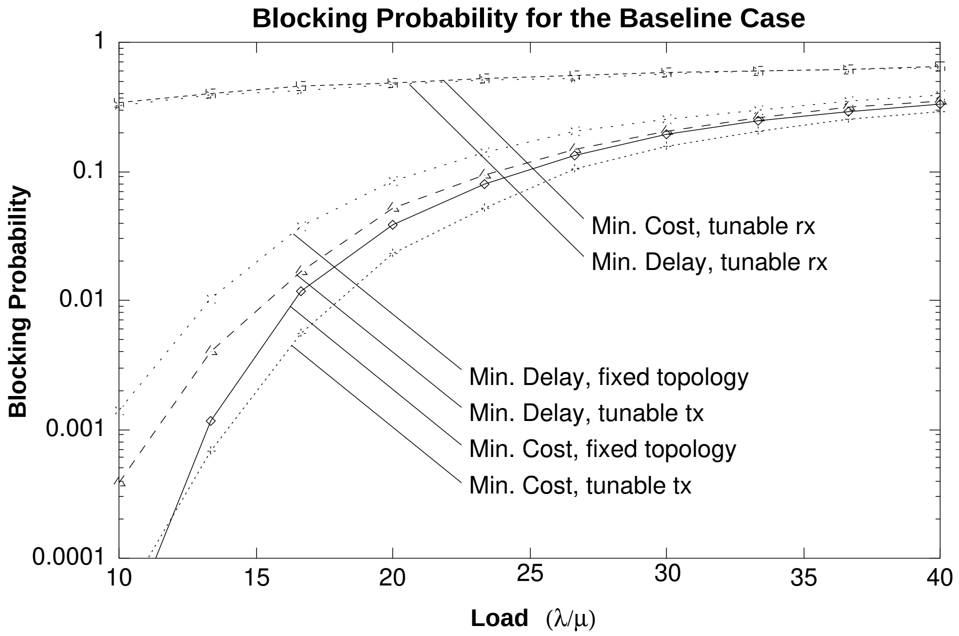


Figure 4.18: Blocking probability for the baseline case: single-multicast sessions, stream bandwidth 10% of the link capacity, 50,000 routes/point

The blocking probability as a function of the session arrival rate in the baseline case is shown in figure 4.18, for the various algorithms and network scenarios. The main conclusions from that figure are:

- Networks where physical multicast is allowed present much higher blocking than networks where it is not allowed. For example, when $\lambda/\mu = 10$, the blocking probability is under 0.2% for the fixed-topology networks and for the networks with no physical multicast, while it is about 30% for networks with physical multicast. For such networks, there is basically no difference between minimum-cost and minimum-delay algorithms. The main reason for this result is that in the baseline case, the number of transmitters and receivers in the network is the same; therefore, if two receivers are tuned to the same transmitter, there is a transmitter in the network that it not connected and cannot be used. The gain in flexibility does not make up for the lost capacity.

- Minimum-cost algorithms perform better than minimum-delay algorithms, as expected. Also, for the same kind of algorithm (i.e., minimum cost or minimum delay), the WDM networks perform better than the fixed-topology networks. However, it is interesting to note that the blocking probability for the fixed-topology under minimum-cost routing is *lower* than the blocking probability of the WDM network under minimum-delay routing.

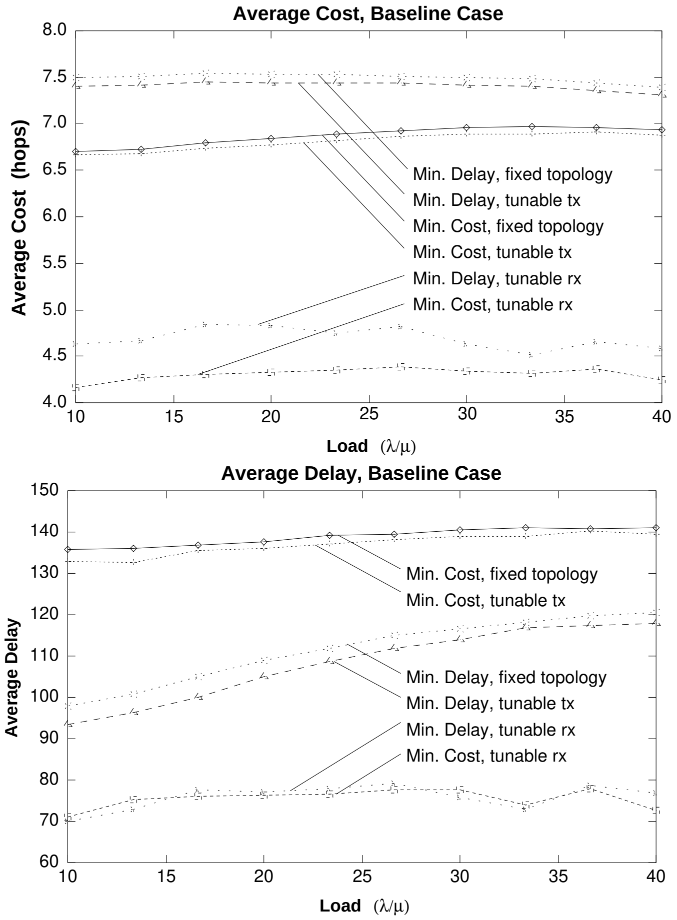


Figure 4.19: Comparison of costs and delays for established sessions in the baseline case

The plots in Figure 4.18 have shown that the blocking probability for the networks where physical multicast is allowed is much higher than when it is not. This seems counter-intuitive: the networks with physical multicast have an *additional* degree of freedom, therefore they should have "better" performance. Figure 4.19 shows where this better performance is seen: in the cost and delay of accepted sessions. Minimizing the blocking probability is not the objective of minimum cost/delay algorithms. The two plots in Figure 4.19 show clearly that the lowest costs and delays are achieved in the networks where physical multicast is allowed. However, this does not mean at all that blocking probability will be low! We should point out that, while in a traditional (fixed-topology) network the cost measure has a well defined meaning (usage of network resources), in the optical network this meaning is lost, because it does not capture the fact that the tunability of transmitters and/or receivers is a resource that can be used (and "spent"). The delay measure, however, has still the same meaning.

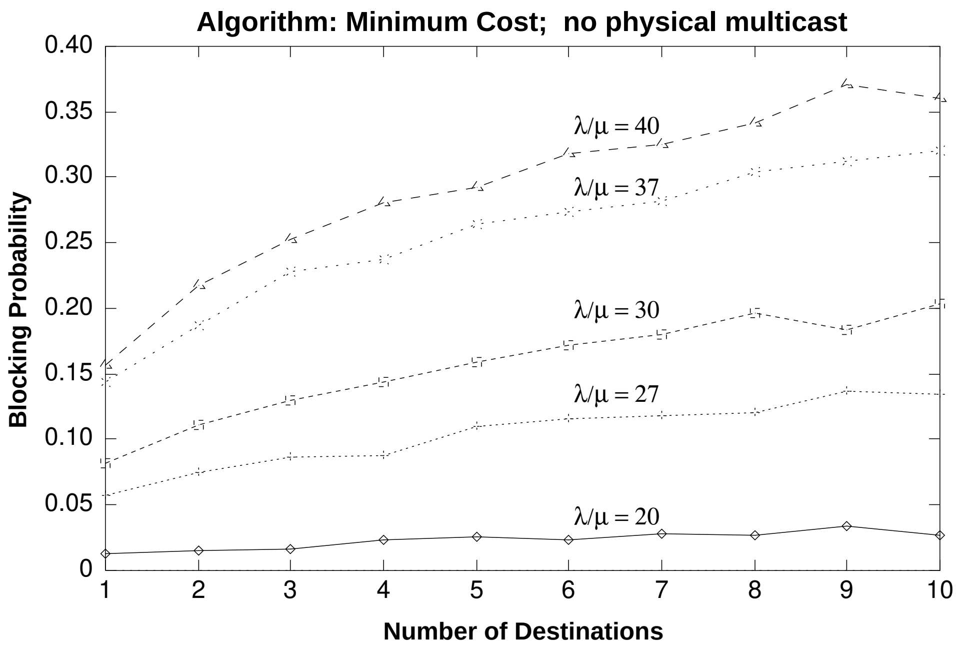


Figure 4.20: Blocking probability as a function of the number of destinations; baseline case, no physical multicast, minimum cost

Figure 4.20 shows the blocking probability as a function of the number of destinations for various load values. The routing algorithm is the minimum cost heuristic, with tunable transmitters (no physical multicast). The figure indicates that the blocking probability is a weak function of the number of destinations[3]. The same result holds for the other algorithms and network scenarios in the baseline case.

Figure 4.21 shows the average time between stream re-routes as a function of the session arrival rate in the baseline case. The time is normalized to the session duration, i.e., an

[3] In Chapter 3 we have shown that this also happens with fixed-topology networks.

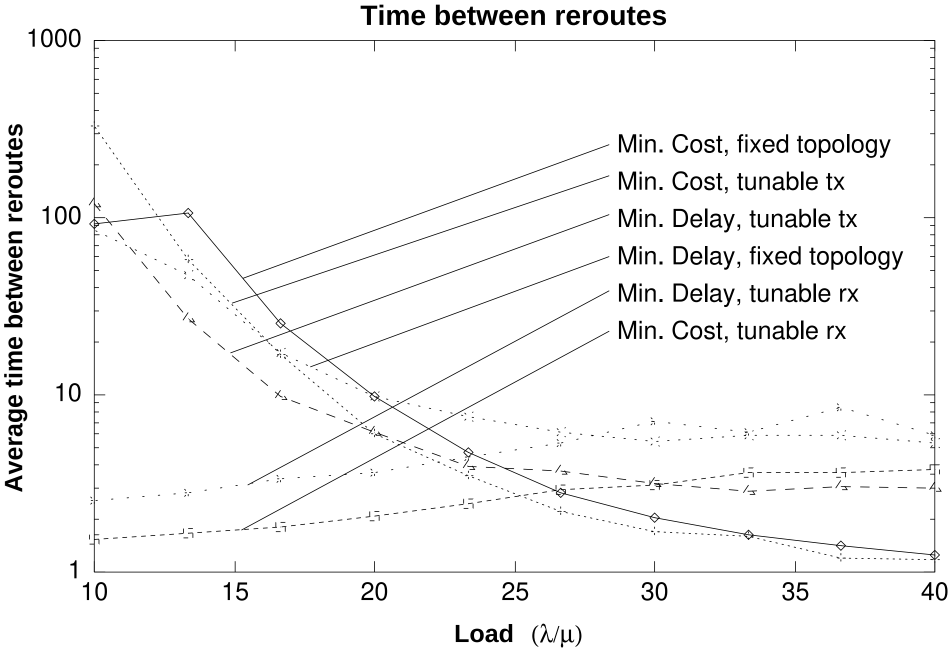


Figure 4.21: Average time between stream re-routes, baseline case

average re-route time of 10 indicates that the average time between re-routing of established streams is 10 times the average session duration (or, on the average, only one session in 10 will suffer rerouting in its lifetime). The figure indicates that only a small fraction of the streams will be re-routed during their lifetimes, for reasonable traffic loads. Moreover, it is very unlikely that a stream will will be rerouted more than once. Re-routing is more severe in networks where physical multicast is allowed, due to the high blocking. We also determined that the average path change when a stream is re-routed is very small; the average change in cost is under 0.2 hops for the topologies with no physical multicast, and under 0.6 hops for the topologies with physical multicast.

4.7.2 Changing the Network Size

In this section, we investigate the effects of increasing the network size. The network size can be increased by: (i) increasing the number of nodes, and/or (ii) increasing the number of transmitters and receivers per node. We consider the following two variations to the baseline case:

- Increase the number of transmitters and receivers per node to 3, keeping the number of nodes fixed at 12.
- Increase the number of nodes from 12 to 20, keeping the number of transmitters and receivers per node fixed at 2.

We keep the traffic model the same as in the baseline case.

Increasing the Number of Transmitters/Receivers

Figure 4.22 shows the blocking probability as a function of the session arrival rate for a 12-node network, with 3 transmitters and 3 receivers per node. We note that the difference in performance between the fixed-topology network and the WDM network has decreased considerably. This is explained by the fact that the increased node degree leads to a richer topology, with shorter paths; the additional degree of freedom of rearranging links in the WDM network is less important in this scenario. Other observations:

- Re-routing happens less frequently than in the baseline case.
- Costs (in hops) are approximately the same as the baseline case.
- For higher loads, the blocking probability is more sensitive to the number of destinations.

Increasing the Number of Nodes

Figure 4.23 shows the blocking probability as a function of the session arrival rate for a 20-node network, with 2 transmitters and receivers per node, under the same traffic as the baseline case. The same qualitative comments about the algorithms made in the baseline case can be repeated here. We note that the advantage of the WDM network over the fixed-topology network has increased. Other observations:

- Since the paths get "longer", the average change in cost can be as high as 1 hop for the networks where physical multicast is allowed, and 0.6 hops where it is not.

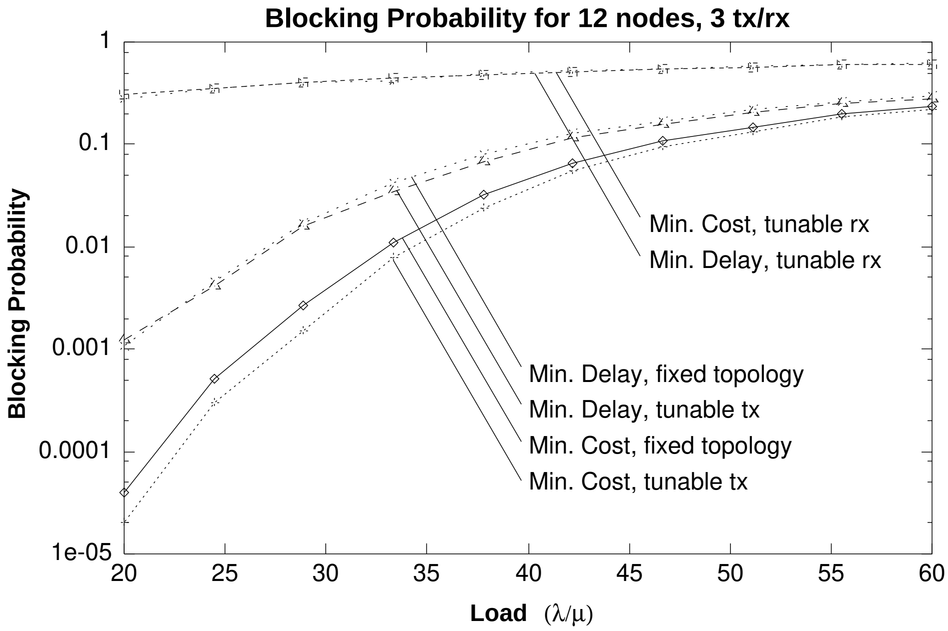


Figure 4.22: Blocking probability for 12-node networks, 3 transmitters/receivers per node, 50,000 routes/point

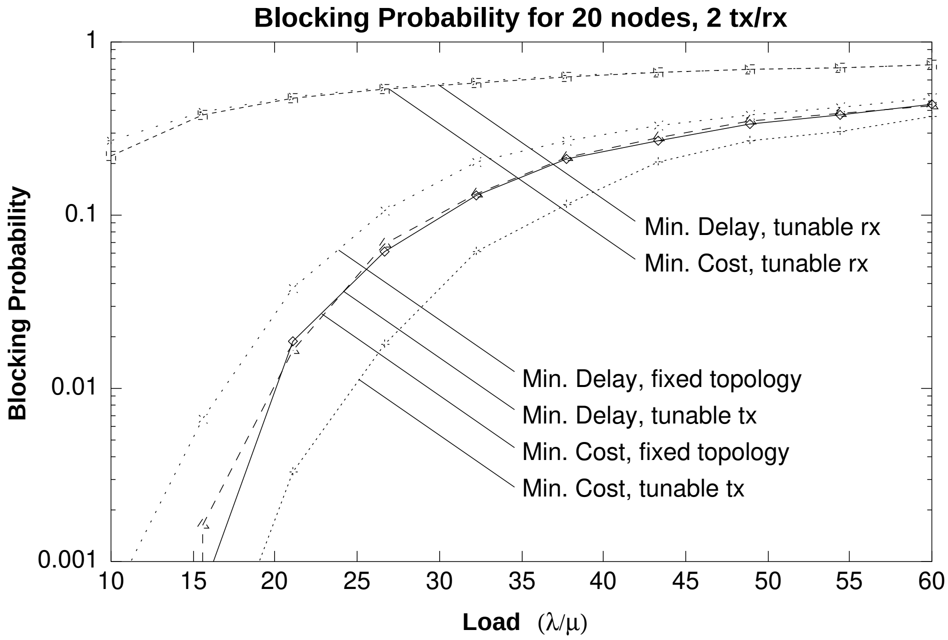


Figure 4.23: Blocking probability for 20-node networks, 2 transmitters/receivers per node, 15,000 routes/point

- For higher loads, the blocking probability is more sensitive to the number of destinations. For example, at $\lambda/\mu = 60$, using the minimum cost algorithm with no physical multicast (tunable transmitters), the blocking probability is about 20% for unicasts, and increases to about 45% for 10-destination multicasts.

4.7.3 Keeping the Network Connected

In the runs for the baseline case, we did not require the network to stay strongly connected as its topology changes. In this section, we impose the additional constraint that at every invocation of the Shortest Path with Reconfiguration Algorithm, the resulting network is tested and, if necessary, a secondary reconfiguration is implemented to keep connectivity, as described in Appendix C. The blocking probability results are shown in Figure 4.24. The results for fixed-topology networks are the same as in the baseline case and are repeated here for the sake of comparison.

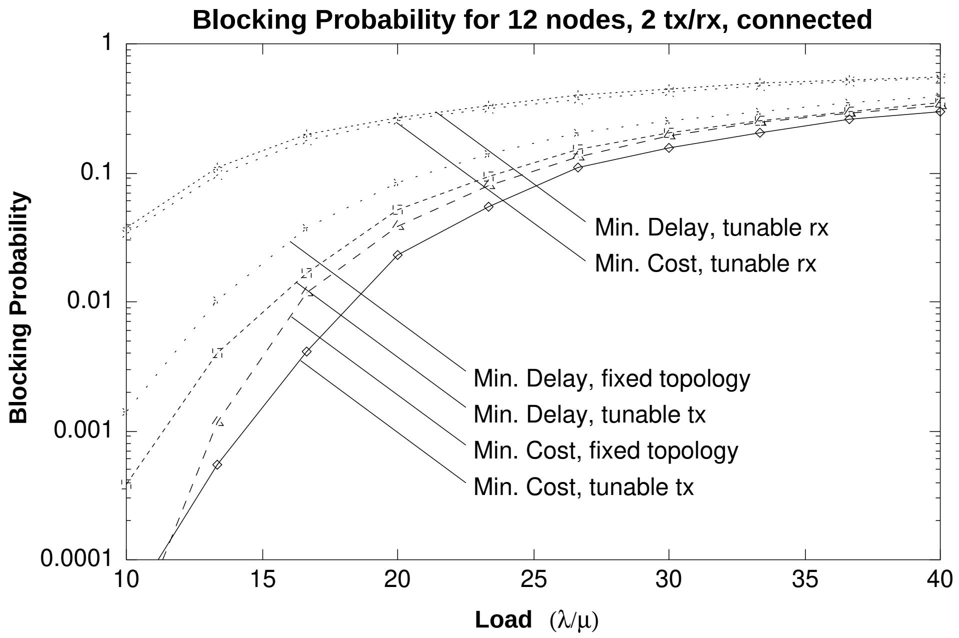


Figure 4.24: Blocking probability for 12-node networks, 2 transmitters/receivers per node; strong connectivity maintained, 50,000 routes/point

The main observation from Figure 4.24 is that while there is basically no change in the results for networks where physical multicast is not possible, there is a significant improvement for networks where it is used. This is made clearer in figure 4.25, where we compare the blocking probability for the minimum-cost heuristics in the connected and not connected cases.

Other observations include:

- For networks allowing physical multicast, keeping the topology connected increases the average time between re-routes and decreases the change in cost, especially for minimum-delay algorithms.

- Keeping the network connected increases the cost/delay obtained by the minimum cost/delay heuristics.

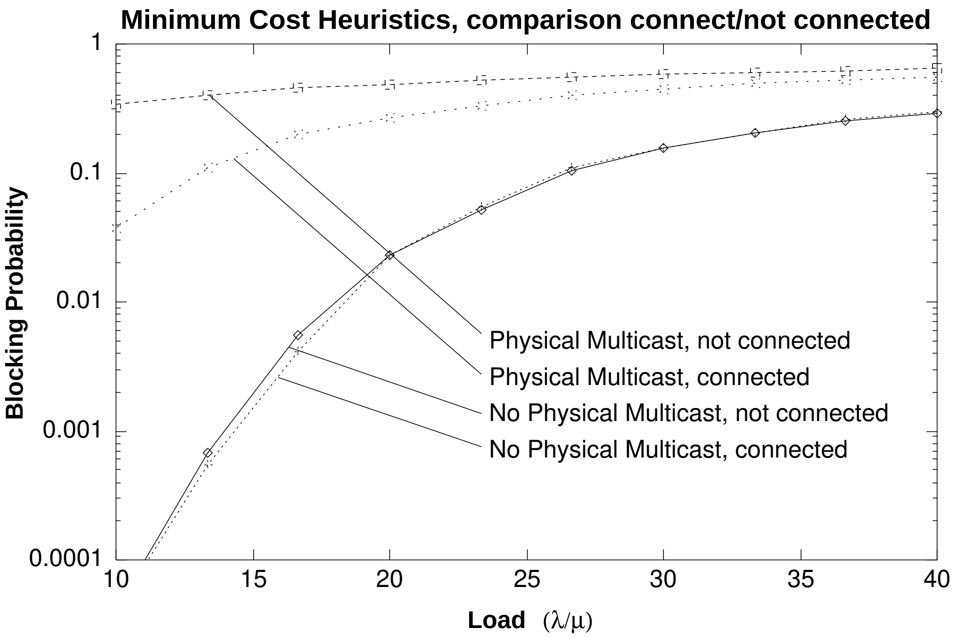


Figure 4.25: Comparison of blocking probability for minimum cost heuristics, when the network is kept connected versus when it is not

4.7.4 Changing the Number of Receivers

In the previous sections, we have shown that allowing physical multicast in the WDM network when the number of transmitters and receivers is equal greatly degrades the performance. Keeping the network connected offsets some of this degradation, but it is still severe. However, we expect that in a network where receivers are abundant (i.e., more receivers than transmitters), allowing physical multicast should improve the performance.

To investigate this fact, we computed the blocking probability for a number of 12-node networks, keeping the number of transmitters per node fixed at 2, and varying the number of receivers per node from 2 to 8. The session arrival rate was kept fixed, and no connectivity constraints were imposed in the network. The blocking probability results (as a function of the number of receivers) are shown in figure 4.26. The baseline case corresponds to 2 receivers per node.

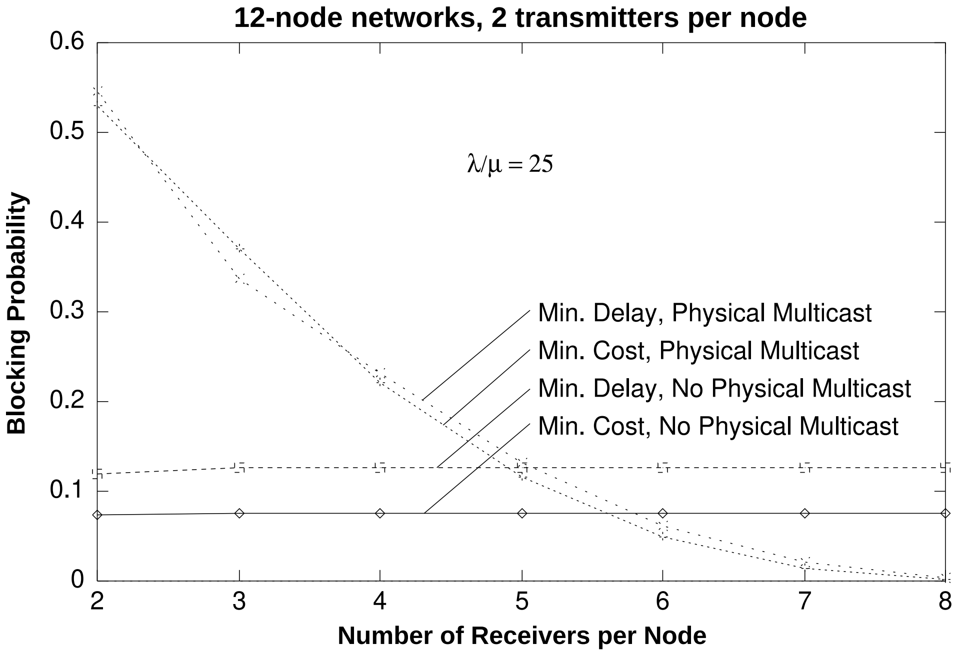


Figure 4.26: Increasing the number of receivers per node, keeping the number of transmitters fixed, 20,000 routes/point

Figure 4.26 shows that, as expected, networks where physical multicast is not possible cannot make effective use of extra receivers. When physical multicast is allowed, adding receivers will improve the performance. For this particular case, the point where the performance of the two networks becomes the same is between 5 and 6 receivers. For higher loads, the point of equal performance shifts to a lower number (for example, for $\lambda/\mu = 40$, the crossover point is between 4 and 5 receivers).

The results shown in figure 4.26 were obtained considering: (i) the network topology was not constrained to be connected at all times, and (ii) receiver/transmitter costs were assigned as in the baseline case (i.e., all costs set to 0.5). We know that keeping the network topology connected improves the performance when physical multicast is allowed, and we would like to investigate the role played by the link costs in the performance.

Figure 4.27 contains two plots, one for connected topologies, and the other for topologies that are not required to stay connected. Both plots contain the blocking probability as a function of the number of receivers in the network, for the minimum-cost heuristic on a network allowing physical multicast; the number of transmitters was fixed at 2. Each plot contains three curves, corresponding to the following transmitter/receiver cost assignments:

- Receiver cost 0.5, transmitter cost 0.5 (this is the assignment used in the evaluation so far).
- Receiver cost 1.0, transmitter cost 0.0.
- Receiver cost 0.0, transmitter cost 1.0.

The results from figure 4.27 indicate that when the network is kept connected, there is little sensitivity to the way the link cost is split between the transmitters and receivers. However, when the network is not restricted to connected topologies, the best assignment is to assign all the cost to the receivers (making it "expensive" to employ the physical multicasting).

If physical multicast is not allowed, the minimum-cost algorithm is not sensitive to the assignment of costs.

4.8 Conclusions

In this chapter, we considered the problem of routing multimedia streams in a WDM network, taking into account their requirements of bandwidth, multipoint communications, and latency, and making use of the additional degree of freedom (the topology) of the network. We presented an exact solution for the optimum routing and reconfiguration problem, based on linear integer programming. Since this solution is complex (and has exponential worst-case run time because the problem is NP-complete), we also proposed heuristic algorithms to find sub-optimal solutions, and derived an upper bound in performance. For unicast traffic, we showed that the heuristic solution is close to the upper bound, which obviates the need

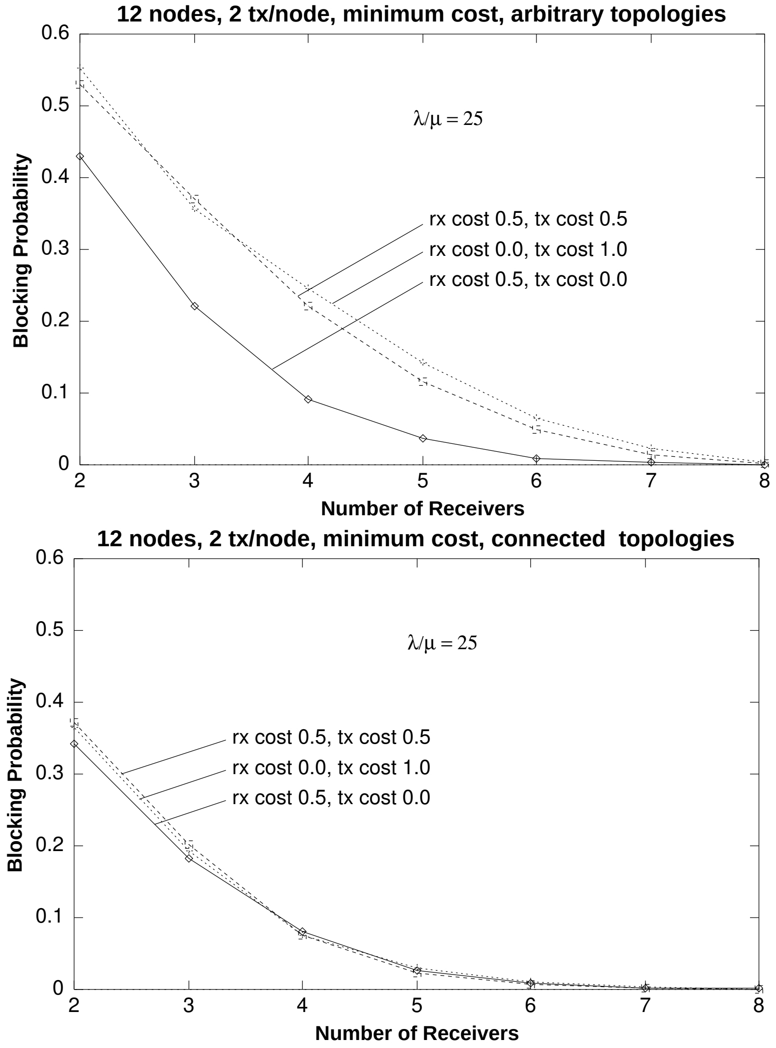


Figure 4.27: Effect of changing the transmitter/receiver costs, minimum cost algorithm, 20,000 routes/point

to pursue the more complex exact solution. Finally, we evaluated the performance of the WDM network in a realistic environment, and showed that it compares favorably with a centralized electronic switch of equivalent complexity.

The main conclusion from the evaluation of multicast routing algorithms in the WDM network concerns the use of physical multicasting in a network where receivers are tunable. The unrestricted use of physical multicasting can decrease the cost and delay of accepted sessions, but at the price of a much higher blocking probability. The previous sections have shown that, by keeping the network connected and/or by properly assigning link costs, the penalty paid in performance can be reduced. However, the blocking probability is still much higher than in the case where no physical multicasting is allowed, except when the number of receivers is large. The fundamental reason for this behavior is simply that, by allowing physical multicasting, the network can be re-arranged in a topology where all receivers are in use, but some transmitters are not connected to any receiver. This represents a net *decrease* in network capacity, and leads to increased blocking probability. Keeping the network strongly connected and/or shifting the cost to the receivers improve the performance because less transmitters are allowed to remain disconnected. By the same token, when there are plenty of receivers, it is unlikely that transmitters will be disconnected. In summary:

- Networks where physical multicasting is allowed are able to make use of extra receivers as they are added; however, performance with a small number of receivers per node is very low; techniques such as keeping the network strongly connected and/or properly assigning transmitter/receiver costs can help, but not much.

- Networks where physical multicasting is not allowed have much better performance when the number of receivers is small, but cannot make use of additional receivers.

These observations indicate that the best way to use physical multicasting is:

1. The network should be kept strongly connected at all times.

2. At all times, there should be *at least one* receiver connected to each transmitter.

3. Physical multicasting can be used as long as it does not violate rules 1 and 2 above. Note that this is true regardless of the type of traffic (unicast or multicast) in the network. Even unicast traffic can benefit from the extra flexibility of allowing physical multicasting, as long as rules 1 and 2 are satisfied.

By using the above rules, we can guarantee good performance both when the number of receivers is small and when it is large. Note that, if the number of receivers is equal to the number of transmitters, no physical multicasting will be allowed.

Based on the results of this chapter, we derive the following set of guidelines for designers/implementors of WDM networks:

- If the network has tunable transmitters and fixed receivers, the total number of transmitters and receivers in the network should be the same. To upgrade a node's capacity, one has to add transmitters and receivers to it in pairs.

- It is recommended that the network be built with tunable receivers and fixed transmitters. In this case, a node can be upgraded just by adding receivers. When operating the network, physical multicasting should be allowed only if there are more receivers than transmitters, as described above.

- Under multicast traffic, the minimum cost heuristics achieve lower blocking. It remains to be seen if delay is a problem.

Chapter 5

Conclusions and Future Work

5.1 Introduction

Supporting multimedia streams in data communications networks has implications in all the layers of the OSI model. This thesis has focused in the aspect of routing, which resides at the network layer.

The routing function is composed of three elements: (i) the routing algorithm, responsible for finding the routes given the stream requirements; (ii) the resource reservation protocol, responsible for reserving the resources required to support the stream and for maintaining state information about the reserved resources, and (iii) the routing protocol, responsible for transporting routing information between the network nodes.

The vast majority of today's data networks operate in datagram mode: packets are delivered in a "best-effort" manner. Since data is highly bursty, it is generally not possible to reserve resources. Therefore, there is no need for a resource reservation protocol, and the routing algorithm only has to find a path from the source to the destination, without regard for traffic, bandwidth, latency, etc. The routing protocol keeps the nodes informed of the network topology, so that they can recompute routes should it change (e.g., in case of link failure).

Support for multimedia streams at the network layer requires new routing algorithms, which can take into account the stream requirements as well as the current network usage when finding the routes; resource reservation protocols; and new routing protocols, to support the routing algorithms and resource reservation protocols. In this thesis, we focused on the routing algorithms, assuming that appropriate resource reservation and routing protocols are in place.

In a traditional network, the only degree of freedom exercised by the routing algorithm is selection of routes. An optical network using Wavelength-Division Multiplexing (WDM) has an additional degree of freedom: by properly tuning transmitters and/or receivers, it is possible to create and remove links between nodes; the topology itself becomes a degree of freedom for the routing algorithm. We have also studied the problem of routing audio/video streams in a reconfigurable WDM network.

In the following two sections, we summarize the conclusions pertaining the routing of streams in fixed-topology networks and WDM networks. We also touch upon directions for future research.

5.2 Routing of Multicast Streams in Fixed-Topology Networks - Summary of Conclusions

A routing algorithm for multimedia streams must satisfy the requirements of multipoint communications, bandwidth and latency. It also must be efficient, due to the relatively high bandwidths involved in video transport. No existing algorithm can meet all these requirements simultaneously. As a matter of fact, the optimum multicast routing problem is, in general, NP-complete (a simpler case, minimum-cost routing of a single multicast stream, corresponds to the well-known Steiner tree problem in graphs, which is NP-complete).

We have shown that the optimum multicast stream routing problem can be formulated as a linear programming problem. If the flow between each source-destination pair is restricted to a single route, the problem becomes an integer programming problem. This integer programming problem can be solved using traditional methods: simplex for the linear relaxation (when the integer constraints are disregarded), and branch-and-bound for the integer problem. We have proposed an efficient solution technique, with two parts:

- A two-step decomposition, to speed up the linear relaxation. The problem of routing a multiple-multicast session is first decomposed into single-multicast problems. Each of these single-multicast routing problems is further divided into unicast routing problems, from the source to each of the destinations. If the flow of data is allowed to use multiple routes, the linear relaxation is all that is needed for routing.

- Improved value-fixing rules, to speed up the search for the integer solution.

We have evaluated the speed-up gained by these techniques, and shown that, in most cases, they significantly reduce the time to find the optimum solution. However, since the problem is NP-complete, the worst-case run time is still exponential with the size of the problem; the optimum solution derived here can be used as a benchmark for other algorithms. The run times for heuristic multicast stream routing algorithms are usually much lower than those of the optimum, but the results are in general inferior, especially for multiple-multicast sessions.

We have used the optimum routing algorithm to evaluate the existing heuristics in realistic network and traffic scenarios. Previous work in the area focused in evaluating the cost and delay of a single multicast in an empty network. The first step was to define "realistic" network scenarios. Obviously, a network topology derived from an existing network is realistic; however, the set of existing networks is rather limited. To properly evaluate the routing algorithms, it is necessary to apply them to a wide range of topologies; therefore, methods for generating random topologies "resembling" existing networks are needed. Waxman [39] proposed an algorithm to generate "realistic" random network topologies, based on the observation that in existing networks, a node is more likely to be connected to other nodes that are "closer" than nodes that are "farther away". However, our evaluation showed that the more important property of a real network is that it is two-connected (for reliability purposes), rather than being made of "shorter" links. For the traffic, we considered a dynamic scenario, where sessions arrive, are routed if enough resources exist, and terminate. In this scenario, the session blocking probability is the main performance value.

The evaluation of multicast routing algorithms showed that, as expected, the blocking probability is lower for cost-based algorithms, since they minimize the usage of network resources. On the other hand, they might lead to excessive delay; routing algorithms that minimize the usage of network resources while meeting latency constraints are needed. Based on the work presented on this thesis, the recommendations for selecting a routing algorithm are:

- For 1-connected networks, the choice of routing algorithm makes little difference in performance; one might as well use the simplest routing algorithm (shortest path). This kind of network topology should be avoided when implementing a network, because of lower reliability and performance.
- Minimum-cost algorithms are indicated for scenarios where the lowest possible blocking probabilities are required (for a given infrastructure), and latency constraints are not a problem. One example would be a campus network, where the link delays are low, and any path will satisfy the latency constraint.
- In a scenario where the latency constraint can be a problem (e.g., in a WAN environment), one can either use the optimum proposed in this thesis (if the network is small) or use shortest-path algorithms.

We also found that, from a traffic point of view, the best way to upgrade a network which is carrying stream traffic is to add links, instead of increasing the bandwidth of the existing links. This happens because adding new links will not only increase the network capacity, but also decrease the average path length (thus decreasing the load). Of course, the implementation cost of adding new links can be very different (i.e., much higher) than increasing the bandwidth of the existing links.

5.3 Routing of Multimedia Streams in WDM Networks - Summary of Conclusions

We considered the problem of routing unicast streams in a WDM reconfigurable network. The WDM network has an additional degree of freedom over conventional networks: its topology can be dynamically changed by the routing algorithm to create the routes needed. Previous work in the area has focused either in very fast reconfiguration, with single-hop routing, or very slow reconfiguration, using multi-hop routing. The first scheme suffers from practical implementation problems; the second does not use the full capacity of the WDM network. We considered reconfiguration in the session time scale: the network is reconfigured as sessions arrive and depart, to create the appropriate routes.

We have shown that the optimum routing and reconfiguration problem can be written as a linear integer programming problem. We have also proposed an heuristic solution to the problem, based on the Shortest Path with Reconfiguration Algorithm, which is an extension of Dijkstra's shortest path for reconfigurable networks. For unicast traffic, we have shown that the proposed heuristic produces results that are close to the optimum, by using an analytical upper bound for the performance measures of interest. This obviates the need to pursue the more complicated optimum solution. We have also shown that the reconfigurable network outperforms a fixed-topology network of the same size.

We have also evaluated the performance of the WDM network in a realistic environment, where sessions arrive, are routed, and terminate. We have shown that, if re-routing is allowed, a session will be re-routed at most once in its lifetime, for a wide range of traffic values. We also compared the performance of the WDM network with that of a centralized electronic switch of equivalent complexity, and found that the WDM network outperforms the switch.

We have proposed a number of algorithms for routing of multicast streams in the WDM network. A WDM network with fixed transmitters and tunable receivers is able to provide physical multicast, by having multiple receivers tune to the same transmitter. We identify two cases: networks with tunable receivers, and networks with fixed receivers. For each of the cases, we present a linear integer programming formulation for the optimum multicast routing problem, and a minimum-cost and a minimum-delay heuristic. The minimum-cost heuristics are extensions of the Takahashi-Matsuyama minimum-cost heuristic, using the Shortest Path with Reconfiguration algorithm.

Finally, we have evaluated the heuristic algorithms proposed for multicast routing in WDM networks, and found that indiscriminate use of physical multicasting greatly degrades the performance. The key point is that no transmitters should be left disconnected; therefore, physical multicasting should be used only in networks (with tunable receivers) where the number of receivers exceed the number of transmitters. We also found that, as the node degree of the network is increased, the gain realized by changing the network topology decreases. Our recommendations for the WDM network can be summarized as:

- WDM networks with tunable receivers and fixed transmitters are potentially better than networks where the transmitters are tunable and the receivers are fixed, due to the possibility of physical multicasting. When operating the network, the best performance is obtained by having at least one receiver tuned to each transmitter.

- WDM networks with tunable receivers can be upgraded by adding additional receivers; on the other hand, if the transmitters are tunable, the network can only be upgraded by adding transmitter/receiver pairs.

- Under multicast traffic, the best performance in terms of blocking probability is achieved by the minimum-cost algorithms, and by keeping the network fully connected at all times.

5.4 Future Work

The work done in routing of streams, both on fixed-topology and on WDM networks, can be extended in the following directions:

- Study the scenario where the delays are a function of the flows in the link.

- More work should be done in proposing minimum-cost heuristics with a latency constraint. The heuristic by Kompella et al [37] is a step in the right direction for fixed-topology networks, but it only applies to topologies with undirected links.

- The work done in this thesis pertains exclusively to routing, and the optimum is defined in relation to the current session. Another optimization axis is time: if the statistics of the session arrival process are known, one could use a tool like dynamic programming to accept or reject incoming sessions. In other words, to minimize the overall blocking probability it might be useful to block a session even though there are enough free resources in the network to accept it.

- The optimum multicast routing algorithm proposed in this thesis can be applied to multi-stage ATM switches under multicast traffic, to provide routing between switching modules. Since ATM switches have regular structures, it might be possible to propose efficient multicast routing heuristics based on the structure of the switch, and use the optimum proposed here to evaluate them.

- The routing algorithms discussed in this thesis are all centralized, i.e., they must have global knowledge not only of the network topology but also of the usage of each link. In practice, sessions arrive at different locations in the network, and knowledge of this fact takes some time to propagate. Thus, at a given point in time, there might be conflicting requests propagating through the network, because sessions arrived simultaneously at different locations. Under this light, the results presented in this thesis can be seen as an upper bound on the actual performance. Alternatives to tackle this problem include:

  - Upon arrival of the session, the node computes the routes as described here, and sends out the reservation. Reservations are broadcast over the whole network. Conflicting reservations arriving to a node are resolved in a deterministic fashion. Eventually, all the nodes get the reservations, and learn of the conflict. Since they all run the same algorithm, they will know which of the conflicting reservations "wins".
  - Instead of broadcasting every session arrival and departure, the nodes may elect to broadcast only the state of the links, and only when the available bandwidth becomes lower than some threshold.

- For fixed-topology networks, performance may be improved if the network is allowed to re-route streams already in place. How to provide re-routing, and quantifying its benefits, is for further study.

Additionally, the work in routing of streams in WDM networks can be extended in the following areas:

- The evaluation presented in Chapter 4 for unicast traffic should be extended to networks of other sizes, not only 8 nodes.
- A better minimum-delay multicast routing heuristic should be proposed for the tunable transmitter case.

Appendix A

The Decomposition Procedure

In this appendix we show an extension of the decomposition problem shown in [17], used in the solution of the linear relaxation of the optimum multicast routing problem.

A.1 Formulation

Consider the following optimization problem (bold letters represent matrices or vectors):

GIVEN: $\boldsymbol{A}_i, \boldsymbol{b}_i, \boldsymbol{C}_i, \boldsymbol{D}_i, \boldsymbol{E}, \boldsymbol{F}, \boldsymbol{V}, \boldsymbol{U}_i$

MINIMIZE:

$$\sum_{i=1}^{T} \boldsymbol{C}_i \boldsymbol{x}_i + \boldsymbol{E}\boldsymbol{y} \tag{A.1}$$

WITH RESPECT TO: $\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{s}$

SUBJECT TO:

$$\boldsymbol{A}_i \boldsymbol{x}_i = \boldsymbol{b}_i, \quad i = 1, \ldots, T \tag{A.2}$$

$$\sum_{i=1}^{T} \boldsymbol{D}_i \boldsymbol{x}_i + \boldsymbol{F}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{V} \tag{A.3}$$

$$0 \leq \boldsymbol{x}_i \leq \boldsymbol{U}_i$$

$$\boldsymbol{y} \geq 0$$

$$\boldsymbol{s} \geq 0$$

Where $\boldsymbol{A}_i$ is $N \times K$, $\boldsymbol{b}_i$ is $N \times 1$, $\boldsymbol{C}_i$ is $1 \times K$, $\boldsymbol{D}_i$ is $M \times K$, $\boldsymbol{E}$ is $1 \times L$, $\boldsymbol{F}$ is $N \times L$, $\boldsymbol{y}$ is $L \times 1$, $\boldsymbol{x}_i$ is $K \times 1$, and $\boldsymbol{s}, \boldsymbol{V}$ are $M \times 1$. Additionally, we will assume that all elements in vector $\boldsymbol{V}$ are non-negative.

A.2 The Decomposition Procedure

Define $\boldsymbol{X}_i = \{\boldsymbol{x}_i : \boldsymbol{A}_i\boldsymbol{x}_i = \boldsymbol{b}_i, \quad 0 \le \boldsymbol{x}_i \le \boldsymbol{U}_i\}$. Since $\boldsymbol{x}_i$ is limited, the set $\boldsymbol{X}_i$ will have a finite number k_i of extreme points. Therefore, any $\boldsymbol{x}_i \in \boldsymbol{X}_i$ can be written as a convex combination of those extreme points, as follows:

$$\begin{aligned} \boldsymbol{x}_i &= \sum_{j=1}^{k_i} \lambda_{ij}\boldsymbol{x}_{ij} && \text{(A.4)} \\ \lambda_{ij} &\ge 0 && \text{(A.5)} \\ \sum_{j=1}^{k_i} \lambda_{ij} &= 1 && \text{(A.6)} \end{aligned}$$

The $\boldsymbol{x}_{ij}$ in equation (A.4) are the extreme points of $\boldsymbol{X}_i$.

Introducing equations (A.4)-(A.6) into equations (A.1) and (A.3)[1], the optimization problem becomes:

GIVEN: $\boldsymbol{x}_{ij}, \boldsymbol{C}_i, \boldsymbol{D}_i, \boldsymbol{E}, \boldsymbol{F}, \boldsymbol{V}$

MINIMIZE:

$$\sum_{i=1}^{T}\sum_{j=1}^{k_i} \boldsymbol{C}_i(\lambda_{ij}\boldsymbol{x}_{ij}) + \boldsymbol{E}\boldsymbol{y} \qquad \text{(A.7)}$$

WITH RESPECT TO: λ_{ij}, $\boldsymbol{y}, \boldsymbol{s}$

SUBJECT TO:

$$\begin{aligned} \sum_{i=1}^{T}\sum_{j=1}^{k_i} \boldsymbol{D}_i(\lambda_{ij}\boldsymbol{x}_{ij}) + \boldsymbol{F}\boldsymbol{y} + \boldsymbol{s} &= \boldsymbol{V} && \text{(A.8)} \\ \sum_{j=1}^{k_i} \lambda_{ij} &= 1, \quad i = 1, \ldots, T && \text{(A.9)} \\ \lambda_{ij} &\ge 0 \\ \boldsymbol{y} &\ge 0 \\ \boldsymbol{s} &\ge 0 \end{aligned}$$

Equations (A.7), (A.8) and (A.9) can be written in matrix format as:

MINIMIZE:

[1] Equation (A.2) is automatically satisfied by the $\boldsymbol{x}_{ij}$.

$$\begin{bmatrix} \boldsymbol{C}_i\boldsymbol{x}_{ij} & \boldsymbol{E} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{y} \\ \boldsymbol{s} \end{bmatrix} \tag{A.10}$$

SUBJECT TO:

$$\begin{bmatrix} \overline{\boldsymbol{A}} & \boldsymbol{F} & \boldsymbol{I}_M \\ \overline{\boldsymbol{D}} & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{y} \\ \boldsymbol{s} \end{bmatrix} = \begin{bmatrix} \boldsymbol{V} \\ \boldsymbol{1} \end{bmatrix} \tag{A.11}$$

where:

$\overline{\boldsymbol{A}} = [\boldsymbol{D}_i\boldsymbol{x}_{ij}]$

$\boldsymbol{I}_M$ is the $M \times M$ identity matrix

$$\overline{\boldsymbol{D}} = \begin{bmatrix} 1\cdots1 & 0 & \cdots & 0 \\ 0 & 1\cdots1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \underbrace{0}_{k_1} & \underbrace{0}_{k_2} & \cdots & \underbrace{1\cdots1}_{k_T} \end{bmatrix} = \begin{bmatrix} \underbrace{\boldsymbol{e}_1\cdots\boldsymbol{e}_1}_{k_1} & \cdots & \underbrace{\boldsymbol{e}_T\cdots\boldsymbol{e}_T}_{k_T} \end{bmatrix}$$

$\boldsymbol{e}_i$ is the i^{th} unit vector (i.e., a vector that has a "1" in row i and "0" elsewhere)

Let us assume that a basic feasible solution in terms of the λ_{ij}'s and $\boldsymbol{y}, \boldsymbol{s}$ is available[2], and let $[\ \boldsymbol{\omega} \quad \boldsymbol{\alpha}\ ]$ be the vector of dual variables corresponding to this basic solution ($\boldsymbol{\omega}$ is $1 \times M$ and $\boldsymbol{\alpha}$ is $1 \times T$). To determine the entering variable, one has to compute the following vector:

$$\begin{bmatrix} \omega & \alpha \end{bmatrix} \begin{bmatrix} \overline{\boldsymbol{A}} & \boldsymbol{F} & \boldsymbol{I}_M \\ \overline{\boldsymbol{D}} & 0 & 0 \end{bmatrix} - \begin{bmatrix} \boldsymbol{C}_i\boldsymbol{x}_{ij} & \boldsymbol{E} & 0 \end{bmatrix} \tag{A.12}$$

The positive elements in the vector defined by equation (A.12) correspond to variables that can enter the basis and improve the objective function. Substituting the values for $\overline{\boldsymbol{A}}$ and $\overline{\boldsymbol{D}}$ in equation (A.12), we find:

$$\begin{bmatrix} (\omega\boldsymbol{D}_i - \boldsymbol{C}_i)\boldsymbol{x}_{ij} + \alpha_i & \omega\boldsymbol{F} - \boldsymbol{E} & \omega \end{bmatrix} \tag{A.13}$$

[2]This solution can be obtained using the well-know two-phase method, for example

From (A.13) we learn:

1. If $(\boldsymbol{\omega D}_i - \boldsymbol{C}_i)\boldsymbol{x}_{ij} + \alpha_i > 0$, then λ_{ij} can enter the basis;
2. If $(\boldsymbol{\omega F} - \boldsymbol{E})_k > 0$, then y_k can enter the basis;
3. If $\omega_l > 0$, then s_l can enter the basis.

Given $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$, conditions 2 and 3 above are immediate to compute. Since $\boldsymbol{x}_{ij}$ is an extreme point of $\boldsymbol{X}_i$, condition 1 above can be rewritten as:

a. Solve the following problem:
Maximize $(\boldsymbol{\omega D}_i - \boldsymbol{C}_i)\boldsymbol{x}_i + \alpha_i$
Subject to $\boldsymbol{A}_i\boldsymbol{x}_i = \boldsymbol{b}_i, \quad 0 \leq \boldsymbol{x}_i \leq \boldsymbol{U}_i$

b. If the objective value of the problem solved in (a.) is positive, then the λ_{ij} corresponding to the optimum $\boldsymbol{x}_i$ in that problem can enter the basis. Otherwise, no λ_{ij} from subproblem i can enter the basis.

A.3 Solution Algorithm

In the following description, we will denote by *Master Problem* the problem described by equations (A.10) and (A.11).

INITIALIZATION STEP:

Begin with a basic feasible solution to the master problem. Store the basis inverse $\boldsymbol{B}^{-1}$, $\overline{\boldsymbol{b}} = \boldsymbol{B}^{-1}\begin{bmatrix} \boldsymbol{V} \\ \boldsymbol{1} \end{bmatrix}$ and $[\ \boldsymbol{\omega} \quad \boldsymbol{\alpha}\ ] = \hat{\boldsymbol{c}}_B\boldsymbol{B}^{-1}$, where $\hat{c}_{ij} = \boldsymbol{C}_i\boldsymbol{x}_{ij}$ for the basic λ_{ij} variables, and $\hat{c}_k = E_k$ for the basic y_k variables, in the revised simplex array shown below:

$$
\begin{array}{|c|c|}
\hline
\begin{bmatrix} \omega & \alpha \end{bmatrix} & \hat{c_B} \boldsymbol{B}^{-1} \overline{\boldsymbol{b}} \\
\hline
\boldsymbol{B}^{-1} & \boldsymbol{B}^{-1} \overline{\boldsymbol{b}} \\
\hline
\end{array}
\tag{A.14}
$$

MAIN STEP:

- Select the the entering variable according to the rules given in section A.2. If there is no candidate to enter the basis, stop – the optimum solution has been reached.
- If a variable has been selected to enter the basis, compute its column, adjoin it to the revised tableau, and pivot. This will update all the variables in the tableau. The columns are computed as follows:
  - If λ_{ij} will enter the basis, the column is $\boldsymbol{B}^{-1} \begin{bmatrix} \boldsymbol{D}_i \boldsymbol{x}_{ij} \\ \boldsymbol{e}_i \end{bmatrix}$
  - If y_k will enter the basis, the column is $\boldsymbol{B}^{-1} \begin{bmatrix} \boldsymbol{F}_k \\ 0 \end{bmatrix}$
  - If s_l will enter the basis, the column is $\boldsymbol{B}^{-1} \begin{bmatrix} \boldsymbol{e}_l \\ 0 \end{bmatrix}$
- Repeat the previous steps until the optimum has been reached.

A.4 Finding an Initial Solution

The procedure described in the previous section assumes that an initial basic feasible solution is available. In this section, we make use of the two-phase method to identify this initial solution. Let us consider the problem in the format described by equation (A.11). We add a number of artificial variables that enable us to immediately identify an initial feasible basis

for this extended problem. We then optimize to drive the artificial variables out of the basis. We add T artificial variables to the problem, denoted by the $T \times 1$ vector $\boldsymbol{a}$, as follows:
MINIMIZE:

$$\begin{bmatrix} 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \begin{bmatrix} \lambda_{ij} \\ \boldsymbol{y} \\ \boldsymbol{s} \\ \boldsymbol{a} \end{bmatrix} \tag{A.15}$$

SUBJECT TO:

$$\begin{bmatrix} \overline{\boldsymbol{A}} & \boldsymbol{F} & \boldsymbol{I}_M & 0 \\ \overline{\boldsymbol{D}} & 0 & 0 & \boldsymbol{I}_T \end{bmatrix} \begin{bmatrix} \lambda_{ij} \\ \boldsymbol{y} \\ \boldsymbol{s} \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{V} \\ \mathbf{1} \end{bmatrix} \tag{A.16}$$

Since all elements of $\boldsymbol{V}$ are non-negative by hypothesis, one can immediately identify a basis in equation (A.16): it will be composed by the $\boldsymbol{s}$ variables (with $s_i = V_i$) and by the artificial variables $\boldsymbol{a}$ (with $a_i = 1$). Therefore,

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{I}_M & 0 \\ 0 & \boldsymbol{I}_T \end{bmatrix} = \boldsymbol{I}_{M+T} = \boldsymbol{B}^{-1} \tag{A.17}$$

$$\hat{\boldsymbol{c}_B} = \begin{bmatrix} \underbrace{0 \cdots 0}_{M} & \underbrace{1 \cdots 1}_{T} \end{bmatrix} \tag{A.18}$$

$$\begin{bmatrix} \boldsymbol{\omega} & \boldsymbol{\alpha} \end{bmatrix} = \hat{\boldsymbol{c}_B} \boldsymbol{B}^{-1} = \hat{\boldsymbol{c}_B} \tag{A.19}$$

$$\overline{\boldsymbol{b}} = \boldsymbol{B}^{-1} \begin{bmatrix} \boldsymbol{V} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{V} \\ \mathbf{1} \end{bmatrix} \tag{A.20}$$

$$\hat{\boldsymbol{c}_B} \boldsymbol{B}^{-1} \overline{\boldsymbol{b}} = \begin{bmatrix} \underbrace{0 \cdots 0}_{M} & \underbrace{1 \cdots 1}_{T} \end{bmatrix} \begin{bmatrix} \boldsymbol{V} \\ \mathbf{1} \end{bmatrix} = T \tag{A.21}$$

Equations (A.17) to (A.21) can be represented by the following revised simplex array:

$$
\begin{array}{|c|c|}
\hline
0\cdots 0 \quad 1\cdots 1 & T \\
\hline
\boldsymbol{I}_{M+T} & \begin{array}{c} \boldsymbol{V} \\ \mathbf{1} \end{array} \\
\hline
\end{array}
$$

To determine the entering variable, we must compute:

$$
\begin{bmatrix} \boldsymbol{\omega} & \boldsymbol{\alpha} \end{bmatrix} \begin{bmatrix} \overline{\boldsymbol{A}} & \boldsymbol{F} & \boldsymbol{I}_M & 0 \\ \overline{\boldsymbol{D}} & 0 & 0 & \boldsymbol{I}_T \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 1\cdots 1 \end{bmatrix} \tag{A.22}
$$

which resolves into:

$$
\begin{bmatrix} \boldsymbol{\omega} \boldsymbol{D}_i \boldsymbol{x}_{ij} + \alpha_i & \boldsymbol{\omega} \boldsymbol{F} & \boldsymbol{\omega} & \boldsymbol{\alpha} - \mathbf{1} \end{bmatrix} \tag{A.23}
$$

The positive entries in the vector on (A.23) correspond to variables that can enter the basis. Therefore,

- If $\boldsymbol{\omega} \boldsymbol{D}_i \boldsymbol{x}_{ij} + \alpha_i > 0$, λ_{ij} can enter the basis, and the column to pivot is $\boldsymbol{B}^{-1} \begin{bmatrix} \boldsymbol{D}_i \boldsymbol{x}_{ij} \\ \boldsymbol{e}_i \end{bmatrix}$.
- If $(\boldsymbol{\omega} \boldsymbol{F})_k > 0$, y_k can enter the basis, and the column to pivot is $\boldsymbol{B}^{-1} \begin{bmatrix} \boldsymbol{F}_k \\ 0 \end{bmatrix}$.
- If $\omega_l > 0$, s_l can enter the basis, and the column to pivot is $\boldsymbol{B}^{-1} \begin{bmatrix} \boldsymbol{e}_l \\ 0 \end{bmatrix}$.
- If $\alpha_m - 1 > 0$, the artificial a_m can enter the basis, and the column to pivot is $\boldsymbol{B}^{-1} \begin{bmatrix} 0 \\ \boldsymbol{e}_m \end{bmatrix}$.

The solution procedure indicated earlier in this appendix used here. At optimality, one of the following three situations will happen:

1. All the artificials are out of the basis. An initial feasible solution has been found. One just has to compute $[\ \boldsymbol{\omega} \quad \boldsymbol{\alpha}\ ] = \hat{\boldsymbol{c}_B} \boldsymbol{B}^{-1}$ and $\hat{\boldsymbol{c}_B}(\boldsymbol{B}^{-1}\overline{\boldsymbol{b}})$, and proceed according to what was indicated in section A.3.

2. There is at least one artificial in the basis at non-zero level. This means that the original problem is infeasible, i.e., has no solution.

3. There is at least one artificial in the basis at zero level. We can proceed to the main optimization as described in item 1, but during the pivoting process, we always select one of the remaining artificials as the leaving variable, if possible.

Appendix B

Algorithms for Generation of Random Topologies

In this Appendix, we present the algorithms used to generate the random topologies for the evaluation in chapter 3.

B.1 Generating a Completely Random Topology - Full Duplex Links

INPUTS: Number of nodes, N, number of full-duplex links, K, and dimensions of the rectangle (S_x, S_y) in which to place the nodes. It is assumed that $K \geq N - 1$.

OUTPUT: The network topology.

Step 1: Generate N node locations at random (horizontal coordinate is generated uniformly between 0 and S_x; vertical coordinate is generated uniformly between 0 and S_y).

Step 2: Create a set of nodes denoted by T, initially empty. Choose one of the nodes at random and add this node to T.

Step 3: Choose at random a node in T and a node not in T, and place a full-duplex link between them in the network topology. Add the node that was not in T to T. Repeat this step until all N nodes are in T.

Step 4: In step 3, $N-1$ links were placed. Place the remaining $K-N+1$ full-duplex links in the topology by choosing their endpoints at random, but allowing only one link between any pair of nodes.

B.2 Generating a Random Topology, Short Links

INPUTS: Number of nodes, N, number of links, K, and dimensions of the rectangle (S_x, S_y) in which to place the nodes. It is assumed that $K \geq N-1$, and the links are full-duplex.

OUTPUT: The network topology.

Step 1: Generate N node locations at random (horizontal coordinate is generated uniformly between 0 and S_x; vertical coordinate is generated uniformly between 0 and S_y).

Step 2: Using Prim's algorithm [9], generate a minimum-distance spanning tree and add it to the topology. This is computed by considering the distance graph (i.e., an auxiliary graph where there is a link between every pair of nodes whose cost is the distance between the nodes) and finding its minimum-cost spanning tree. In this step, $N-1$ links are placed.

Step 3: Choose a node at random, say, node n. Let S_n denote the set of nodes to which node n is not connected, say, $\{s_1, s_2, \ldots, s_k\}$. If $d_1, d_2, \ldots, d_n$ are the distances from node n to nodes $s_1, s_2, \ldots, s_k$, choose a node from the set S_n at random, with probabilities proportional to $1/d_1, 1/d_2, \ldots, 1/d_k$, say, s_i, and add a full-duplex link in the topology between n and s_1. Repeat this step until the desired K links are placed, but allowing only one link between any pair of nodes.

B.3 Generating a Two-Connected Topology

INPUTS: Number of nodes, N, number of links, K, and dimensions of the rectangle (S_x, S_y) in which to place the nodes. It is assumed that $K \geq N - 1$, and the links are full-duplex.

OUTPUT: The network topology.

Step 1: Generate N node locations at random (horizontal coordinate is generated uniformly between 0 and S_x; vertical coordinate is generated uniformly between 0 and S_y).

Step 2: Generate a ring as follows: first, find the two nodes closest to each other and add a full-duplex link between them in the topology. This forms a full-duplex path. Then proceed to form a ring, adding the nodes closest to each extremity of the path, until the ring closes. In this step, N links are placed.

Step 3: Choose a node at random, say, node n. Let S_n denote the set of nodes to which node n is not connected, say, $\{s_1, s_2, \ldots, s_k\}$. If $d_1, d_2, \ldots, d_n$ are the distances from node n to nodes $s_1, s_2, \ldots, s_k$, choose a node from the set S_n at random, with probabilities proportional to $1/d_1, 1/d_2, \ldots, 1/d_k$, say, s_i, and add a full-duplex link in the topology between n and s_1. Repeat this step until the desired K links are placed, but allowing only one link between any pair of nodes.

Appendix C

Formal Description of The Shortest Path with Reconfiguration Algorithms

In this Appendix, we describe the basic Shortest Path with Reconfiguration algorithm, an extension to Dijkstra's shortest path algorithm for reconfigurable networks, and the secondary reconfiguration algorithm, which is used to keep the network strongly-connected.

C.1 The Basic Shortest Path with Reconfiguration Algorithm

The algorithm described in this section finds the shortest path between two nodes in a reconfigurable optical network. Some transmitters and receivers in the network might already be tuned due to other communications in progress; these cannot be reconfigured, but they might be used in whatever topology they happen to be. The algorithm has two degrees of freedom: the path, and the topology, potentially reconfiguring the transmitters and receivers that are idle. A numeric *label* is assigned to each transmitter and receiver in the network. If a transmitter with label l^t is tuned to a receiver with label l^r, an unidirectional link with label $l^t + l^r$ is created. The shortest path minimizes the sum of the labels in the path. The label can represent, for example, the propagation delay; in this case, if the network topology is a star, the label associated with the transmitter is the propagation delay from the node to the star, and the label associated with the receiver is the propagation delay from the star to the node.

INPUTS:

- A WDM network with N nodes; node i has S_i optical transmitters and P_i optical receivers, $i = 1, \ldots, N$. The optical transmitters and receivers are either *locked* or *free*; the locked transmitters and receivers are tuned, forming a given topology, and the free ones are not connected at all and can be reconfigured. If the network supports physical multicasting, *all* transmitters are to be considered free.
- A set of nonnegative transmitter labels $\{l^t_{ij}\}$, $i = 1, \ldots, N$, $j = 1, \ldots, S_i$.
- A set of nonnegative receiver labels $\{l^r_{ij}\}$, $i = 1, \ldots, N$, $j = 1, \ldots, P_i$.
- A source node s and a destination node d.

OUTPUT: The path from s to d with the minimum length (sum of the transmitter and receiver labels in the path), using, if necessary, free transmitters and receivers.

ALGORITHM:

Step 1: For all transmitters in the network, do the following: if transmitter j in node i is free, add a *transmitter virtual node* V^t_{ij} to the network topology, and a link (using transmitter j) from node i to node V^t_{ij}, with label l^t_{ij}.

Step 2: For all receivers in the network, do the following: if receiver j in node i is free, add a *receiver virtual node* V^r_{ij} to the network topology, and a link (using receiver j) from node V^r_{ij} to node i, with label l^r_{ij}.

Step 3: Create a set of nodes P, initially empty, and a set of nodes T, initially containing all the nodes in the network. For each node in the network, associate a label l; initially, assign $l(s) = 0$, $l(i) = \infty$ for $i \neq s$. Create also two sets of nodes A and B, initially empty.

Step 4: Let k be the node with the smallest $l(k)$ (if there are many, choose one at random). If $l(k) = \infty$, there is no path between s and d using the locked receivers and transmitters; go to step 7. Otherwise, move k from the set T to the set P. If node k is a virtual node and the set A is empty, also put node k in A.

Step 5: If k is not a virtual node, update the node labels as follows: $\forall j \in P$, if there is a link from node k to node j, make $l(j) \leftarrow \min\{l(j), l(i) + l_{kj}\}$, where l_{kj} is the label of the link between k and j. If k is a virtual node, there is no need to update the node labels.

Step 6: If $k \neq d$, return to step 4; otherwise, the shortest path using the locked transmitters and receivers has been found; proceed to step 7.

Step 7: Create a set of nodes P', initially empty, and a set of nodes T', initially containing all the nodes in the network. For each node in the network, associate a label l'; initially, assign $l'(d) = 0$, $l'(i) = \infty$ for $i \neq d$.

Step 8: Let k be the node with the smallest $l'(k)$ (if there are many, choose one at random). If $l'(k) = \infty$, go to step 10. Otherwise, move k from the set T' to the set P'. If $k = s$, go to step 10. If node k is a virtual node and the set B is empty, put node k in B and go to step 10.

Step 9: If k is not a virtual node, update the node labels as follows: $\forall j \in P'$, if there is a link from node j to node k, make $l'(j) \leftarrow \min\{l'(j), l'(i) + l_{jk}\}$. Go to step 8.

Step 10: If the set A or the set B or both are empty, terminate. The shortest path is the one found in step 6, if any. If both A and B are non-empty, let us denote by a the node in A and by b the node in B. Let $L_1 = l(d)$, $L_a = l(a)$ and $L_b = l'(b)$. If $L_1 \leq L_a + L_b$, the path found in step 6 is the shortest and there is no need to reconfigure the network. Otherwise, a shorter path can be created by connecting the virtual nodes a and b. If a corresponds to the virtual node V^t_{ij} and b corresponds to the virtual node V^r_{kl}, then transmitter j in node i must be tuned to receiver l in node k, and the shortest path will be the path from s to i, the newly-created i-k link, and the path from k to d.

C.2 The Secondary Reconfiguration Heuristic

For control purposes, it is necessary to keep the WDM network strongly connected at all times, so that all the nodes can exchange control messages. The transmission of control messages can be done by flooding, as in the OSPF routing protocol [12].

In the basic shortest path with reconfiguration algorithm presented in section C.1, the free transmitters and receivers are considered to be idle. In general, however, they will be connected in a certain topology, to keep the network strongly connected. When the shortest path algorithm described in section C.1 is applied, it might change the network topology, and the resulting topology can potentially be disconnected. We denote the reconfiguration from the shortest path algorithm as the *primary reconfiguration*. The heuristic in this section will try to identify a *secondary reconfiguration* to keep the network strongly connected.

As indicated above, the network will have as many free transmitters and receivers connected as possible (but available for reconfiguration). If the total number of transmitters in the network is equal to the total number of receivers, then all free transmitters and receivers can be used. Reconfiguring the network so that a transmitter and a receiver can be connected might also include reconfiguring the previous connections held by them, as illustrated in figure C.1.

If the primary reconfiguration causes the network to become disconnected, then this new topology has at most two disconnected subsets; a way to implement the secondary reconfiguration would be to identify two good candidate nodes to "bridge" the two subsets. Therefore, we need an algorithm that can:

1. Test a topology for connectivity.

2. If the topology is disconnected, locate the appropriate nodes where the "bridging" between the subsets can be performed.

Both of these functions can be efficiently performed by a simplified version of the Floyd-Warshall shortest path algorithm [9]. In the the next two sections, we describe the simplified Floyd-Warshall algorithm and give the secondary reconfiguration heuristic.

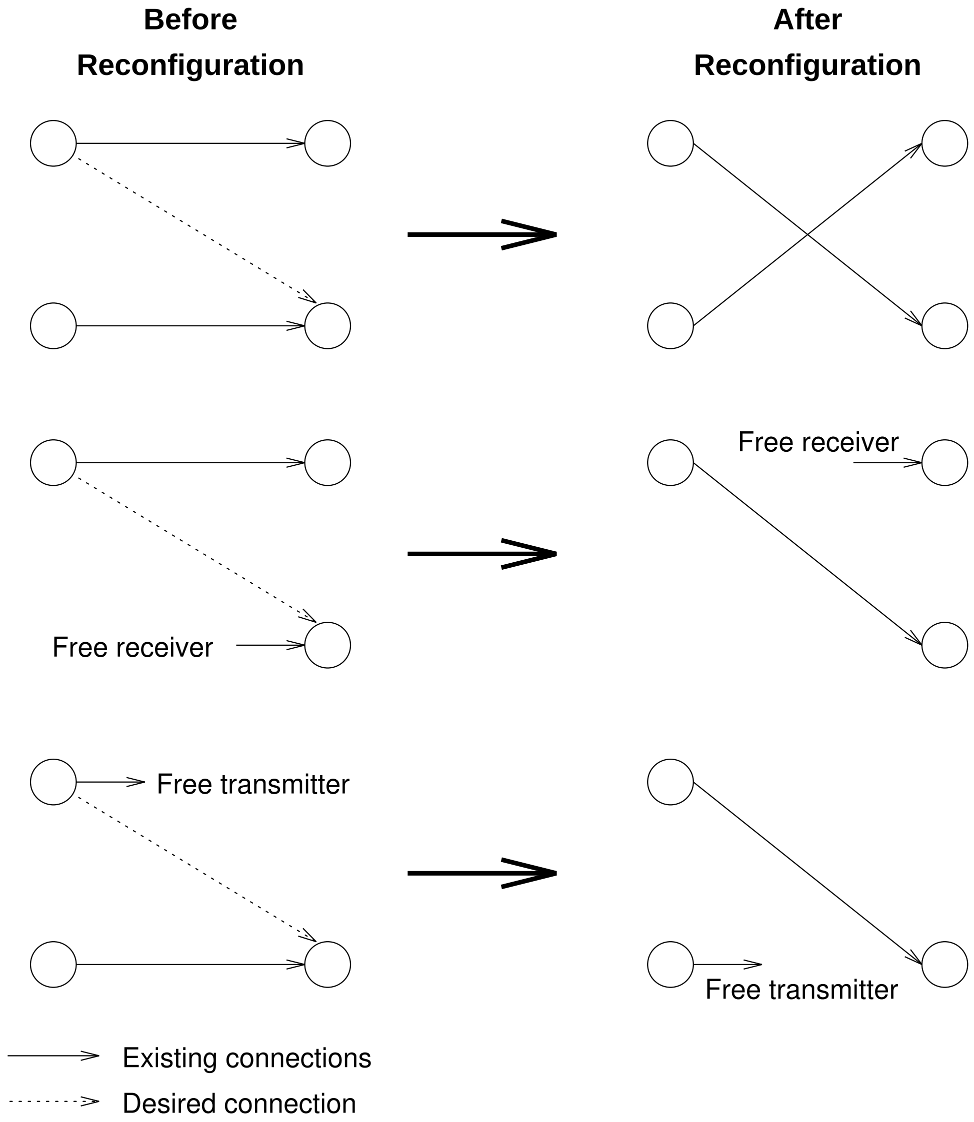


Figure C.1: Reconfiguring the network

C.2.1 The Simplified Floyd-Warshall Algorithm

INPUT: The network topology (N is the number of nodes).

OUTPUT: The connectivity matrix $\boldsymbol{D}$, where $D_{ij} = 1$ if there is at least one path from node i to node j, and $D_{ij} = 0$ otherwise.

ALGORITHM:

Step 1: Define the matrix $\boldsymbol{D}(0)$ as follows:

$$D_{ij}(0) = \begin{cases} 1 & \text{if node } i \text{ is connected to node } j\text{, or } i = j \\ 0 & \text{otherwise} \end{cases}$$

Step 2: Calculate the matrix $\boldsymbol{D}(k)$ from the matrix $\boldsymbol{D}(k-1)$ as follows:

$$D_{ij}(k) = \begin{cases} 1 & \text{if } D_{ik}(k-1) \textbf{ and } D_{kj}(k-1) \text{ are both } 1 \\ D_{ij}(k-1) & \text{otherwise} \end{cases}$$

Step 3: Repeat step 2 for $k = 1, \ldots, N$; matrix $\boldsymbol{D}(N)$ is the desired connectivity matrix $\boldsymbol{D}$.

C.2.2 Description of the Secondary Reconfiguration Heuristic

The secondary reconfiguration heuristic is executed as a part of the shortest path with reconfiguration algorithm in section C.1. If that algorithm decides on a primary reconfiguration, the secondary reconfiguration algorithm is invoked; if it fails, the primary reconfiguration is not permitted. More specifically, in step 10, if $L_1 > L_a + L_b$, the secondary reconfiguration algorithm is executed. If it is successful, then the shortest path algorithm proceeds as originally described, possibly with a secondary reconfiguration in place. If it fails, then the primary reconfiguration is dropped; the final path between the source and destination will be the path found using the current topology (or no path at all). Note that this is not optimal in any sense; one could elect to search for other reconfigurations besides the primary that would lead to longer paths but would not partition the network.

The secondary reconfiguration algorithm is:

INPUTS: The network topology, the list of locked and free transmitters and receivers, and the primary reconfiguration.

OUTPUTS: A flag, indicating failure or success; in case of success, the algorithm may recommend a secondary reconfiguration.

ALGORITHM:

Step 1: Execute the primary reconfiguration in the network topology given and use the simplified Floyd-Warshall algorithm to compute the connectivity matrix $\boldsymbol{D}$. If all the entries in $\boldsymbol{D}$ are 1, return and indicate success; the new topology is connected and there is no need for a secondary reconfiguration. Otherwise, mark the transmitter and receiver used in the primary reconfiguration as locked and proceed to step 2.

Step 2: Search the $\boldsymbol{D}$ matrix for an entry d_{ij} such that $d_{ij} = 0$, node i has a free transmitter and node j has a free receiver. If the WDM network supports physical multicasting, consider all transmitters as free.

Step 3: Reconfigure the network so that a link between nodes i and j is created (exchanging any other connection as depicted in figure C.1). Compute the new connectivity matrix $\boldsymbol{D}'$.

Step 4: If all the entries in $\boldsymbol{D}'$ are 1, stop. The secondary reconfiguration has been found. Return the secondary reconfiguration and indicate success. Otherwise, return to step 2 to search for another entry. If all zero entries in $\boldsymbol{D}$ have been considered already, terminate and indicate failure.

Bibliography

[1] ISO, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications," *ISO/IEC 8802-3*, 4th ed., 1993-07-08.

[2] "Token Ring Access Method," IEEE 802.5, 1985.

[3] "Fiber Distributed Data Interface (FDDI)," ANSI X3 T9.5, 1988.

[4] İ. Dalgıç, W. Chien and F. Tobagi, "Evaluation of 10Base-T and 100Base-T Ethernets Carrying Video, Audio and Data Traffic," *IEEE INFOCOM 94*, Toronto, Canada, June 1994, pp 1094-1102.

[5] C. Noronha and F. Tobagi, "The Evolution of Campus Networks Towards Multimedia," *Proceedings of COMPCON Spring'93*, San Francisco, CA, February 1993.

[6] "100Base-X Physical Layer Specification for Fast Ethernet," sponsored by the Fast Ethernet Alliance (version 1.0, October 15, 1993).

[7] "4T+ Physical Layer Specification for UTP Category 3/4/5 Wiring," sponsored by the Fast Ethernet Alliance (December, 1993).

[8] F. Backes, "Transparent Bridges for Interconnection of IEEE 802 LANs," *IEEE Network*, vol. 2, no. 1, Jan. 1988, pp 5-9.

[9] Bertsekas, D., and Gallager, R., *Data Networks*, Prentice-Hall, New Jersey, 1987.

[10] C. Hedrick, "Routing Information Protocol," RFC 1058, Rutgers University, June 1988.

[11] G. Malkin, "RIP Version 2 - Carrying Additional Information," RFC 1388, Xylogics, Inc., January 1993.

[12] J. Moy, "OSPF Version 2," RFC 1583, Proteon, Inc., March 1994.

[13] Almquist, P., "Type of Service in the Internet Protocol Suite," *RFC 1349*, July 1992.

[14] Deering, S.E., and Cheriton, D.R., "Multicast Routing in datagram internetworks and extended LANs," *ACM Trans. on Computer Systems*, May 1990, vol.8, no.2, p. 85-110.

[15] S. Deering, "Host Extensions for IP Multicasting," RFC 1112, Stanford University, August 1989.

[16] J. Moy, "Multicast Extensions to OSPF," *RFC 1584*, Proteon, Inc., March 1994.

[17] M. S. Bazaraa, J. J. Jarvis and H. D. Sherali, *Linear Programming and Network Flows, 2nd ed.*, John Wiley & Sons, New York, 1990.

[18] C. Topolcic, "Experimental Internet Stream Protocol, Version 2 (ST-II)," RFC 1190, CIP Working Group, October 1990.

[19] L. Zhang et al, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification," Internet Draft, July 1994.

[20] L. Zhang et al, "RSVP: A New Resource ReSerVation Protocol," *IEEE Network*, vol. 7, no. 5, Sept. 1993, pp 8-18.

[21] J. Forgie, "ST - A Proposed Internet Stream Protocol", IEN 119, M. I. T. Lincoln Laboratory, September 1979.

[22] L. Delgrossi et al, "An Implementation of ST-II for the Heidelberg Transport System," *Technical Report 43.9303*, IBM European Networking Center, Heidelberg, 1993.

[23] L. Delgrossi et al, "Reservation Protocols for Internetworks: A Comparison of ST-II and RSVP," *Fourth Int. Workshop on Network and Operating System Support for Digital Audio and Video*, Lancaster, UK, November 1993.

[24] D. Mitzel et al, "An Architectural Comparison of ST-II and RSVP," *IEEE INFOCOM 94*, Toronto, Canada, June 1994.

[25] Hwang, F.K., and Richards, D.S., "Steiner Tree Problems," *Networks*, Jan. 1992, vol.22, no.1, p. 55-89.

[26] Karp, R.M., "Reducibility among combinatorial problems", in *Complexity of Computer Communications*, R.E. Miller and J.W. Thatcher (editors), pp.85-103, Plenum Press, New York 1972.

[27] S.E. Dreyfus and R.A. Wagner, "The Steiner Problem in Graphs," *Networks*, vol. 1, pp 195-207, 1972.

[28] M.L. Shore et al, "An Algorithm for the Steiner Problem in Graphs," *Networks*, vol.12, pp 323-33, 1982.

[29] J.E. Beasley, "An Algorithm for the Steiner Problem in Graphs," *Networks*, vol.14, pp 147-59. 1984.

[30] L. Kou, G. Markowsky and L. Berman, "A Fast Algorithm for Steiner Trees," *Acta Informatica* 15, pp 141-5, 1981.

[31] Rayward-Smith, V.J., and Clare, A., "On Finding Steiner Vertices," *Networks*, Fall 1986, vol. 16, no.3, pp. 283-94.

[32] H. Takahashi and A. Matsuyama, "An Approximate Solution for the Steiner Problem in Graphs," *Math. Japonica*6, 1980, pp. 573-7.

[33] M. Dror, B. Gavish and J. Choquette, "Directed Steiner Tree Problem on a Graph: Models, Relaxations and Algorithms," *INFOR*, vol.28, no. 3, Aug. 1990, pp. 266-81.

[34] N. Maculan and P. Souza, "An Approach for the Steiner Problem in Directed Graphs," *Annals of Operations Research*, vol. 33, 1991, pp. 471-80.

[35] F.S. Hillier and G.J. Lieberman, *Introduction to Operations Research*, 5th ed., New York, McGraw-Hill, 1990.

[36] K. Bharath-Kumar and J.M. Jaffe, "Routing to Multiple Destinations in Computer Networks," *IEEE Trans. on Comm.*, vol. COM-31, no. 3, March 1983, pp 343-51.

[37] V.P. Kompella, J.C. Pasquale and G.C. Polyzos, "Multicast Routing for Multimedia Communication," *IEEE/ACM Trans. on Networking*, vol. 1, no. 3, June 1993, pp 286-92.

[38] H. Crowder, E. L. Johnson, M. Padberg, "Solving Large-Scale Zero-One Linear Programming Problems," *Operations Research*, Vol. 31, No. 5, Sept-Oct 1983, pp 803-34.

[39] Waxman, B.M. "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, (Dec. 1988) vol.6, no.9, p. 1617-22.

[40] M. Doar and I. Leslie, "How Bad is Naïve Multicast Routing?" *IEEE INFOCOM '93*, San Francisco, CA, USA, pp 82-9.

[41] Yiu-Wing Leung, et al. "Efficient algorithms for multiple destinations routing," *ICC 91*, Denver, CO, USA, 23-26 June 1991, p. 1311-17 vol.3.

[42] Chow, C.-H., "On multicast path finding algorithms," *IEEE INFOCOM '91*, Bal Harbour, FL, USA, 7-11 April 1991, pp. 1274-83.

[43] M.H. Ammar et al, "Routing Multipoint Connections Using Virtual Paths in an ATM Network," *IEEE INFOCOM '93*, San Francisco, CA, USA, pp 98-105.

[44] K. Bharath-Kumar and J.M. Jaffe, "Routing to Multiple Destinations in Computer Networks," *IBM Research Report RC9098*, Oct. 1981.

[45] McKinley, P.K., and Liu, J.W.S., "Multicast tree construction in bus-based networks," *Communications of the ACM*, (Jan. 1990) vol.33, no.1, p. 29-42.

[46] Xiaofeng Jiang, "Routing broadband multicast streams," *Computer Communications* (Jan.-Feb. 1992) vol.15, no.1, p. 45-51.

[47] L. Kazovsky et al, "WDM Local Area Networks," *IEEE LTS*, vol. 3, no. 2, pp 8-15, May 1992.

[48] W. B. Jones, *Introduction to Optical Fiber Communication Systems*, Holt, Rinehart and Winston, New York, 1988.

[49] Y. Birk et al, "Bus-Oriented Interconnection Topologies for Single-Hop Communication among Multi-Transceiver Stations," *IEEE INFOCOM '88*, New Orleans, LA, March 1988, pp 558-67.

[50] M. Mehdi Nassehi et al, "Fiber Optic Configurations for Local Area Networks," *IEEE Journal on Selected Areas in Communications*, Nov. 1985, vol. 3, no. 6, pp 941-9.

[51] K. Kobayashi and I. Mito, "Single Frequency and Tunable Laser Diodes," *Journal of Lightwave Tech.*, vol. LT-6, no. 11, Nov. 1988, pp 1623-33.

[52] T. P. Lee and C. E. Zah, "Wavelength-Tunable and Single-Frequency Semiconductor Lasers for Photonic Communications Networks," *IEEE Communications Mag.*, Oct. 1989, pp 42-51.

[53] D.A. Smith et al, "Integrated-Optic Acoustically-Tunable Filters for WDM Networks," *IEEE JSAC*, vol. 8, no. 6, Aug. 1990, pp 1151-1159.

[54] H. Kobrinski and K.-W. Cheung, "Wavelength-Tunable Optical Filters: Applications and Technologies," *IEEE Communications Mag.*, Oct. 1989, pp 53-63.

[55] B. Mukherjee, "WDM-based local lightwave networks. I. Single-hop systems," *IEEE Network*, vol.6, no.3, May 1992, pp 12-27.

[56] B. Mukherjee, "WDM-based local lightwave networks. II. Multihop systems," *IEEE Network*, vol.6, no.4, July 1992, pp 20-32.

[57] I. Habbab, M. Kavehrad, C. Sundberg, "Protocols for Very High-Speed Optical Fiber Local Area Networks Using a Passive Star Topology," *IEEE Journal on Lightwave Tech.*, vol. LT-5, no. 12, Dec. 1987, pp 1782-1793.

[58] N, Mehravari, "Improved Multiple Access Schemes for Very High-Speed Optical Fiber Local Area Networks Using a Passive Star Topology," *IEEE GLOBECOM 1989*, Dallas, Nov. 27-30, 1989.

[59] I. Chlamtac and A. Ganz, "Design Alternatives of Asynchronous WDM Star Networks," *IEEE ICC 1989*, Boston, June 11-14, 1989.

[60] A. Ganz and Z. Koren, "WDM Passive Star - Protocols and Performance Analysis," *IEEE INFOCOM 1991*, Bal Harbour, Florida, April 9-11, 1991.

[61] N. R. Dono et al., "A wavelength division multiple access network for computer communication," *IEEE JSAC*, vol. 8, no. 6, pp. 983-994, Aug. 1990.

[62] A.S. Acampora, M.J. Karol, M.G. Hluchyj, "Terabit Lightwave Networks: The Multihop Approach," *AT&T Technical Journal*, vol. 66, issue 6, Nov/Dec 1987, pp 21-34.

[63] J. Bannister, L. Fratta, M. Gerla, "Topological Design of the Wavelength-Division Optical Network," *IEEE INFOCOM 1990*, San Francisco, June 3-7, 1990.

[64] J. Bannister, L. Fratta, M. Gerla, "Routing in Large Metropolitan Area Networks Based on Wavelength-Division Multiplexing Technology," in *High-Capacity Local and Metropolitan Area Networks - Architecture and Performance Issues*, edited by G. Pujolle, Springer-Verlag, 1990.

[65] J.-F. Labourdette and A. Acampora, "Logically Rearrangeable Multihop Lightwave Networks," *IEEE Trans. on Communications*, vol.39, no.8, pp 1223-30, Aug. 1991.

[66] J.-F. Labourdette and A. Acampora, "Partially Reconfigurable Multihop Lightwave Networks," *IEEE GLOBECOM 1990*, San Diego, Dec. 2-5, 1990.

[67] S. Kirkpatrick, C. Gelatt, M. Vecchi, "Optimization by simulated annealing," *Science*, 220(4598):671-680, May 1983.

[68] J. Bannister and M. Gerla, "Design of the Wavelength-Division Optical Network," *IEEE ICC'90*, Atlanta, April 16-19, 1990.