The Online and Offline Properties of Routing Algorithms in MPLS

by

Serene Wing Hang Wong

Hon.B.Sc., University of Toronto, 2000

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia

July 2002

Department of Computer Science

The University of British Columbia
Vancouver, Canada

Date July 18 2002

Abstract

This thesis focuses on the route allocation problem. In particular, we consider how to optimally place a LSP on a network such that the bandwidth demand is guaranteed. The objection function is throughput. Specifically, two metrics are used: the number of rejected requests and the amount of rejected bandwidth. We evaluate the performance of five routing algorithms: the shortest path algorithm, the widest-shortest path algorithm, the shortest-widest path algorithm, the minimum interference routing algorithm, and the profile-based routing algorithm. In addition, we investigate the sensitivity of history of the two more sophisticated routing algorithms: the minimum interference routing algorithm, and the profile-based routing algorithm.

We explore the sensitivity of history through the use of the batch process scheme. The batch process scheme allocates tunnels for a batch of requests at a time, and can be viewed as a continuum with its ends being the online and offline scheme. We compare the performance of routing algorithms with varying batch sizes indicating the changes of the history information.

The routing decisions that routing algorithms make significantly influence the performance of the network. Thus, it is beneficial to evaluate the performance of routing algorithms. Furthermore, with the batch process scheme, algorithms can make use of the history information in order to make better routing decisions which would greatly enhance the utilization of the network resources.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my supervisor, Professor Alan Wagner, for his guidance throughout the thesis. I would especially like to thank him for being approachable, patient and considerate. I would also like to thank Professor William Evans, my second reader, for his valuable comments.

SERENE WING HANG WONG

The University of British Columbia
July 2002

Chapter 1

Introduction

The explosive growth in the Internet over the last few years has resulted in a massive increase in traffic as well as the development of services and applications that demand higher quality of service. A major problem is how the extra demands on the network can be accommodated to allow for this enormous growth while attaining the quality of service that applications desire. Traffic engineering is an area which manages traffic in networks to achieve better utilization of network resources, better response time and better service guarantees. With better traffic management, networks are able to route more traffic, and provide higher quality of service.

More formally, traffic engineering is an area that strives to achieve performance goals through the knowledge obtained from measuring, modeling and controlling of network traffic. We focus on the controlling of network traffic, the route allocation problem, which considers how to allocate paths for requests. Some general problems that are connected to the route allocation problem that traffic engineering seeks to solve include the constraint based routing problem, connection admission control problem, rerouting problem, and the network and capacity planning problem. The constraint based routing problem is to determine how to optimally place a

Label Switched Path (LSP) such that it satisfies a set of constraints on the network. The connection admission problem looks at whether or not to accept a request. The rerouting problem is to determine how to reroute a LSP, and the network and capacity planning problem is to determine an optimal network topology for a set of estimated requests. Our main concern in this thesis is on the constraint based routing problem with the understanding that routing allocation is a component in all of the above mentioned problems.

The routing allocation problem for an Internet Service Provider (ISP) is a big challenge due to the rapid growth of networks. An ISP has to allocate more paths for the increase in traffic and to lay LSPs that satisfy their quality of service. The current Interior Gateway Protocols (IGP), the networking protocols that are used by ISPs, have limitations. Routing Information Protocol (RIP) is one of the most commonly used IGP and was originally released in Unix BSD. Open Shortest Path First (OSPF) is another commonly used IGP and it is a more widely used IGP. Both RIP and OSPF are based on the shortest path algorithm. A major shortfall of the shortest path algorithm is congestion. The shortest path algorithm does not consider the characteristics of the traffic or the link state information when it makes its routing decision. It only considers the topology information. Therefore, current IGPs do not have sufficient capabilities for traffic engineering.

Several approaches have been developed to overcome the weaknesses of current IGPs. One approach is the development of the overlay model. This approach puts a virtual topology on top of the physical topology. The IGPs then treat the virtual topology as though it were physical. Extensions have also been made to OSPF to support traffic engineering. More recently, a new protocol, Multiprotocol Label Switching (MPLS) which has the capacity to explicitly specify a path has been developed.

MPLS is an IGP. It is situated in between layer 2 and layer 3 of the Open Systems Interconnection (OSI) model, and it uses label switching as its forwarding scheme. Label switching consists of two components, the forwarding component and the control component. The forwarding component uses the label and the forwarding table when making its forwarding decision. The label is used as an index into the forwarding table to determine the outgoing interface. The control component gives routing information to MPLS enabled routers and creates forwarding tables used by the forwarding component.

There are several advantages to MPLS for traffic engineering including, but not limited to its ability to do explicit routing, to attach attributes to traffic trunks and to make use of attributes attached to resources. Explicit routing facilitates traffic engineering control by allowing the entire path, from source to destination, to be specified. The ability to attach attributes to traffic trunks enables the specification of quality of service for each traffic trunk. The ability to make use of attributes attached to resources constrains the use of resources for LSPs, and hence allows the control of the use of resources. These capabilities make MPLS suitable for traffic engineering.

1.1 Motivation

Routing allocation problems can be online or offline. An offline routing algorithm assumes that one has complete knowledge about future requests, and an online or dynamic routing algorithm does not have any knowledge regarding the future. It routes requests one by one as they arrive. The constraint based routing problem, the connection admission control problem and the rerouting problem have both an online and offline counter part, while the network and capacity planning problem is an offline problem.

Although routing algorithms are often dynamic, in practice, the time to provision a tunnel is often not dynamic; instead, it is a batch process. A batch process refers to the handling of requests a batch at a time as opposed to one by one, as they arrive. There are two reasons for provisioning tunnels using the batch process. Firstly, there is a time lapse between a customer ordering a service from a company, and the company passing it to the network engineer to configure the network. Secondly, networks are configured periodically as network reconfiguration is error prone, and an error would cause undesirable consequences. During the time lapse, requests are collected in a batch. Thus a batch process is often used for tunnel provisioning.

The batch process is useful in many situations. It is suitable for services that are subscribed to a company on a regular basis, and for services that do not require immediate action. This batch process can be viewed as a spectrum with its ends being the online scheme and the offline scheme. The batch process has more knowledge when compared to the online scheme, but has less knowledge when compared to the offline scheme.

The batch process is advantageous in that it can gain history information from the batch of requests and it is also suitable for many services. The question that is of interest then is which algorithms are best suited for the batch process environment, or which algorithms are sensitive to history.

1.2 The Problem

In this thesis, we consider a simple constraint based routing problem. We focus on how to optimally place a LSP such that the bandwidth is guaranteed. The objective function is throughput. In particular, two metrics are used: the number of rejected requests and the amount of rejected bandwidth. We evaluate five routing algorithms:

1. Shortest path algorithm – chooses the shortest path with respect to the number of hops.

2. Widest-shortest path algorithm – chooses the shortest path. If there exists more than one shortest path, the widest path is chosen. The widest path is the one with the largest residual bandwidth in the bottleneck link.

3. Shortest-widest path algorithm – chooses the widest path. If there is more than one widest path, the shortest path is chosen.

4. Minimum interference routing algorithm – uses the maxflow computation to make its routing decision. The key idea is to minimize interference.

5. Profile-based routing algorithm – uses multi-commodity flow computation as the preprocessing phase to approximate traffic, and uses the shortest path algorithm as its online phase to route individual requests.

In addition, we focus on the sensitivity of history of the two more complex algorithms, the minimum interference routing algorithm, and the profile-based routing algorithm.

The rest of the thesis is organized as follows. Chapter two gives the background information. Chapter three discusses the shortest path algorithm, the widest-shortest path algorithm, the shortest-widest path algorithm, the minimum interference routing algorithm, and the profile-based routing algorithm in detail. In addition, Chapter three refines the research problem. Chapter four presents the design of the test cases and provides the test results as well as their explanations. Chapter five concludes our work and makes suggestions for future work.

Chapter 2

Background

The popularity of the Internet has grown tremendously over the last few years and has resulted in an enormous increase in traffic as well as the development of new applications and services. The problem is how one can accommodate this increase in traffic, while providing the quality of service that new applications and services desire. Traffic engineering is an area that manages traffic in networks with the goal to optimize the utilization of network resources and to attain higher network performance.

2.1 Traffic Engineering

Traffic engineering is an area that includes the application of technology to measure, model, characterize and control network traffic, and to achieve certain performance goals through the use of such knowledge [1].

Two essential classes of performance objectives for traffic engineering are described in [1], namely, traffic oriented objectives and resource oriented objectives. Traffic oriented objectives focus on the quality of service for particular traffic streams. Some examples in a single class best effort service model are throughput maximization, packet loss minimization and delay minimization. Resource oriented objectives focus on the utilization of network resources. In particular, it is not desirable to have one part of the network over-utilized and another part of the network under-utilized. An example of resource oriented performance objective is bandwidth maximization. There are performance objectives that belong to both the traffic and resource oriented performance objectives. An example is congestion minimization. Congestion occurs when the network does not have enough resources and when resources are not efficiently used. The aim of traffic engineering is to avoid the latter.

In this thesis, we focus on the problem of controlling traffic, in particular, the route allocation problem.

2.2 Traffic Engineering Problems

Girish, Zhou and Hu, [9], mathematically formulated four traffic engineering problems all connected to the routing problem: the constraint based routing problem, the connection admission control problem, the rerouting problem, and the network and capacity planning problem. In the following we briefly describe each of these problems.

- Constraint Based Routing Problem – In the constraint based routing problem [9], an optimal placement of a label switched path (LSP) that satisfies a set of given constraints is to be determined. The LSP determines the path that all packets assigned to that LSP must follow from ingress to egress. The constraints include the network state information, the attributes of the resources of the network, and the attributes of the LSPs [9]. If there exists several feasible LSPs that satisfy all the constraints, then the optimal LSP must further be determined. In order to choose an optimal path, there needs to be an objective function or some metrics to define what needs to be optimized. Bandwidth, delay, delay jitter and cost are some common routing metrics [21]. Their objective functions are maximizing the total bandwidth or the residual bandwidth, minimizing delay, minimizing jitter, and minimizing the cost, respectively. A routing algorithm routes paths according to these objective functions [21].

- Connection Admission Control Problem – Admission control [9] is the problem of deciding whether or not to admit a request. There are two reasons why one would reject a request. Firstly, if it is impossible to satisfy the request within the given constraints, then it must be rejected. However, even when the constraints can be satisfied, there is still the issue of whether it is best to accept this request over future requests. For example, if accepting the request results in blocking future traffic, then it might be better to reject the request in order to accept future requests.

- Rerouting Problem – Rerouting is the problem of rerouting LSPs [9]. Rerouting is to route already established tunnels to other paths. Rerouting is needed when one or more elements in the network, nodes or links, fails or recovers from earlier failure. During a failure, LSPs that use the node or the link are affected. Those affected LSPs have to re-route to other feasible paths, called backup LSPs. Routers such as Junipers allow multiple backup paths to be configured so that affected LSPs can route to the backup paths. Once the network elements have recovered, re-optimization of allocated tunnels may be desired. This is done by rerouting LSPs on more preferable links that become available once the recovery of one or more network elements occurs. Rerouting can also be used for preemptive tunnels where higher priority LSPs are able to preempt lower priority LSPs [9]. Finally, load balancing may also require rerouting [9].

- Network Design and Capacity Planning Problem – The network design and capacity planning problem is to determine an optimal network topology given an estimation of requests. This is the initial problem of constructing the network and to plan for its future growth.

The focus of this thesis is on the constraint based routing problem although the routing allocation is a component in all of the previous problems. In particular, we consider the simple problem of optimally placing an LSP such that the bandwidth demand is satisfied. The two objective functions that we use are the minimization of the number of rejected requests and the amount of rejected bandwidth.

2.3 Limitations of Current Interior Gateway Protocol

An Interior Gateway Protocol (IGP) is a networking protocol that is used within an autonomous system (AS). An AS is a network that is under a single entity of administrative control. The two most common IGPs used are RIP and OSPF. RIP is an older protocol originally released in Unix BSD. OSPF is a newer and more widely used IGP.

The Routing Information Protocol (RIP) [12] is an IGP that uses the distance vector algorithm. When making routing decisions, RIP compares alternate paths using a fixed metric, such as the number of hops and cost. There are several limitations to traffic engineering with RIP. Firstly, RIP uses a fixed metric and thus, it is not designed for making routing decisions based on metrics such as delay or load, which are real-time parameters. Secondly, RIP is based on the shortest path algorithm, and the shortcomings of which will be discussed later.

The Open Shortest Path First (OSPF) [15] is another IGP that uses the shortest path algorithm. It is a destination based algorithm and as such the routing decision is based only on the destination. Destination based algorithms impose restrictions on their routing choices because requests with the same source and destination have to route on the same path. Fortz and Thorup in [8] optimizes the weight in OSPF in order to do traffic engineering. Routing decisions are based on the shortest path which in turn is based on the assigned weights. They optimize the weights based on a given set of demands using a local search heuristic. However, since it is still a destination based algorithm, the routing decision is restricted and cannot consider the source, destination and quality of service in routing decisions.

Current IGPs are insufficient for traffic engineering because they are based on the shortest path algorithm. A major drawback of the shortest path algorithm is congestion. The shortest path algorithm makes routing decision based on topology information and does not take into consideration the availability of bandwidth or the characterization of traffic. Congestion occurs when 1) multiple requests use the same links, or 2) a request is routed on a path that does not have enough bandwidth to satisfy its demand. The shortest path algorithm is not able to use other feasible paths in the network which can accommodate the incoming requests without causing congestion. Thus, current IGPs which use the shortest path algorithm are insufficient for traffic engineering.

Several approaches have been developed to improve the deficiencies of current IGPs. An overlay model has been used. Virtual topologies are placed on top of the physical topology, and the IGP looks at virtual topologies as if they are the physical network. The use of overlay models allows for the application of various functions including constraint-based routing, admission control, and imposing traffic policies [1]. Extensions have also been made to OSPF to support traffic engineering [11]. In the extension, a traffic engineering database is formed by additional attributes that are added to OSPF advertisements. Some additional attributes include the traffic engineering metric, the maximum bandwidth which can be used and color. The traffic engineering database can be use for the monitoring of link attributes and constraint-based source routing.

Most recently, a new protocol called MPLS (Multiprotocol Label Switching) has been introduced by the Internet Engineering Task Force (IETF). MPLS makes it possible to explicitly specify the entire path. We focus on MPLS in this thesis. The next two sections give an overview of MPLS and its capabilities for traffic engineering.

2.4 MPLS

MPLS [6, 17] is a protocol that uses label switching as its forwarding scheme. It is an IGP. MPLS is in between layer 2, the link layer, and layer 3, the network layer, in the OSI (Open Systems Interconnection) model. We will briefly describe how packets are routed using label switching.

Before describing label switching, we first introduce some terminologies. A MPLS enabled router is a label-switched router (LSR). A collection of LSRs and the connections between them form a MPLS cloud within which tunnels or LSPs are created. An ingress router is the edge router that the packet enters into the MPLS network, and an egress router is the last router before the packet leaves the MPLS network.

Label switching consists of two components, the forwarding component and the control component. The forwarding component of an algorithm is to make forwarding decisions. The forwarding decision uses two kinds of information, the label and the forwarding table. A label is of a short fixed length and does not encode any network layer header information. A label is assigned to a packet as it arrives at an ingress router. As a packet arrives at each LSR, it uses the assigned label to index into the forwarding table to determine its outgoing interface. In a connectionless network layer protocol, as a packet arrives at the router, the packet's header is examined. The router consults the IP routing table to perform a longest prefix match with the packet's destination address for the next hop. Packets in MPLS do not have to consult the IP routing table every time they arrive at a router.

The control component gives routing information to LSRs and creates forwarding tables required by the forwarding component. The label switching control component includes all routing protocols that the convention routing control component uses. Since the convention routing control component does not need to create forwarding tables, the information that convention routing provides is not sufficient for label switching. A mapping is needed between the next hop and the label.

MPLS uses the notion of Forwarding Equivalence Class (FEC). All Packets in an LSR can be forwarded into some disjoint subsets. A LSR treats each subset in the same manner. These subsets are FECs. A packet is mapped into a FEC as it arrives in the ingress router. The mapping is based on the source and destination addresses as well as the type of service of the packet. FEC gives service providers a lot of flexibility. FEC allows them to route every FEC in a different manner satisfying the individual FEC's needs.

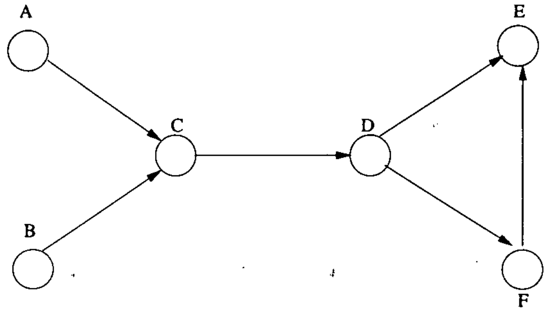


Figure 2.1: Explicit Routing

2.5 MPLS and Traffic Engineering

Let us first introduce the following terminology. A traffic trunk or a tunnel is an aggregation of flows that is placed on a LSP. The aggregation of flows belongs to the same class. In practice, traffic trunk, tunnel and LSP are synonymously used.

MPLS has several capabilities for traffic engineering. The first is its ability to do explicit routing [1]. Explicit routes can either be manually configured by a network operator or automatically established by MPLS defined control protocols. The second is its ability to attach attributes to traffic trunks [1, 20]. These attributes can characterize the QoS for traffic trunks. The third is its ability to make use of attributes attached to resources [1, 20] which can restrict the placement of LSPs on the network. Each of these mechanisms is further described.

2.5.1 Explicit Routing

Explicit routing allows the setup of LSPs to use any path from source to destination. The path can be completely specified in which all LSRs between the ingress and egress nodes are specified. The path can also be partially specified in which only a subset of LSRs on the path between the ingress and egress nodes are specified.

A LSP is setup from A to E in Figure 2.1. $A \Rightarrow C \Rightarrow D \Rightarrow F \Rightarrow E$ completely specifies the path. Each hop in the path is set to be strict or loose. If the hop is set to strict, the next hop must be the specified LSR and the specified LSR must be adjacent to it. If the hop is set to loose, other hops can be inserted before the specified LSR, and the LSR does not have to be adjacent.

Explicit Routing allows more control for traffic engineering than the control mechanisms used in current IGPs, and it can specify any path, not only the shortest path. Explicit Routing is more flexible than destination based routing because it considers both the source and the destination when making its routing decision, while destination based routing makes its decision based only on the destination. If two requests share the same source and destination, explicit routing can route them in two completely different paths but destination based routing cannot. Thus, explicit routing is more flexible and allows more control.

2.5.2 Traffic Trunk Attributes

Traffic Trunk attributes [20, 1] are parameters that characterize the behavior of a traffic trunk. They are important because they determine the QoS of a traffic trunk. These attributes are assigned to LSPs by network operators or can be assigned in the ingress router by the underlying protocol as traffic trunks are mapped to an FEC. Several traffic trunk attributes are described below.

- Traffic Parameter Attribute [1] – The traffic parameter attribute specifies the characteristic of traffic streams. Some examples are peak rates, average rates, burst size and bandwidth. The traffic parameter attribute is important to traffic engineering because it allows the specification of resource requirement which enhances resource allocation and congestion avoidance.

- Resource Class Affinity Attribute [1, 20] – The resource class affinity attribute specifies the class of resource that the tunnel is to include or exclude. For inclusion, it restricts the tunnel to be placed on certain classes of resources, and for exclusion, it restricts the tunnel from being placed on some classes of resources. For example, in explicit inclusion, all resources that do not belong to a specified class are first pruned before computing the tunnel. This is helpful for imposing policies such as limiting the tunnels to be placed on certain specific regions in the network.

- Adaptivity Attribute [1, 20] – The adaptivity attribute is a binary value which indicates the possibility of a tunnel to be re-optimized. Re-optimization refers to the re-establishment of existing tunnels for better performance. If a tunnel is permitted to re-optimize, then it can route on a different path as resources become available. However, if a tunnel is not permitted to re-optimize, the tunnel is pinned to its original path. Tunnels can be re-optimized when the state of the network changes. The state of network changes from time to time because resources may become available. Resources that were allocated could become deallocated and resources may also recover from failure thus creating more routing options. In our thesis, tunnel re-optimization is not permitted.

- Setup Priority Attribute [20] – The priority attribute defines the order of importance of tunnels . In particular, when multiple LSPs compete for the same resources, the setup priority determines which LSP has the highest priority and should get the available resources.

- Preemption Attribute [1, 20] – The preemption attribute specifies whether a tunnel can preempt other tunnels. This permits tunnels that have higher priority to have the privilege to take on better paths.

- Resilience Attribute [1, 20] – The resilience attribute specifies whether a tunnel is to be rerouted to another path in case of failures that affect the tunnel.

- Policing Attribute [1] – The policing attribute specifies actions that are taken if a tunnel exceeds its traffic parameter values. These actions may include limiting its rate, tagging it or having no action at all. It is desirable to have policing in ingress routers to ensure that the service level agreements (SLA) are enforced. The policing attribute provides supervision in the usage in the network.

2.5.3 Resource Attribute

Resource Attributes [1] are parameters that specify the state information for topologies. They provide information that is essential for the traffic placement on resources. On the other hand, they also constrain the use of resources for LSPs which is important for traffic engineering. In this section, two resource attributes, the maximum allocation multiplier and the resource class attribute, are discussed.

- Maximum Allocation Multiplier [1] – The maximum allocation multiplier specifies the availability of resources for allocation. Examples of resources that have this attribute are bandwidth on links and LSRs' buffer resources. The availability of resources can be configured administratively enabling better control of congestion avoidance and network utilization. This attribute is often configured so that resources are under-subscribed or over-subscribed. Undersubscription of a resource implies that the aggregated demands from all LSPs for that resource is less than the capacity for that resource. Over-subscription of a resource implies that the aggregated demands from all LSPs for that resource is greater than the capacity of that resource. Under-subscription places a limit on the utilization of the resource. This may be used to reserve backup LSPs in the event of failure. Usually, over-subscription is used to better utilize resources by making use of the stochastic nature of traffic, such as when the peak rates of the two flows do not occur at the same time.

There is a trade-off between congestion avoidance and better network utilization. If one is interested in avoiding congestion by using the under-subscription approach, then at times when the actual traffic is less than the specified traffic, it will lead to network under-utilization. LSPs may be forced to take sub-optimal paths using the virtual link capacity even if there exists an optimal path using the physical link capacity. On the contrary, if one desires to have high-utilization in the network by using the over-subscription approach, then at times when the actual traffic is greater than the specified traffic, congestion will occur [20]. Thus, the maximum allocation multiplier has to be carefully configured in order to obtain a balance between congestion avoidance and better network utilization.

- Resource Class Attribute [1] – Resource class attribute is used to classify resources into classes. The resource class attribute can be viewed as assigning colors to resources, such that resources that have the same color will belong to the same class. This attribute is helpful in implementing policies such as:

    - the inclusion or exclusion policy of resource utilization for LSP placements as discussed above

    - the preference of resources of LSP placements

- the local traffic control of a specific area in a network
- the application of uniform policy of some resources that are not in the same region

2.6 Existing Routing Algorithms

This section gives an overview of some existing routing algorithms [5].

Many of the existing routing algorithms can be classified into two groups. The first group of algorithms is greedy. They optimize for a given flow and do not consider other flows. The second group of algorithms uses network flow computation, which are not greedy but are more computationally expensive.
Algorithms that belong to the first group are:

- Shortest Path Algorithm – First prune all links that do not satisfy the bandwidth demand. Then the shortest path, with respect to the number of hops, is chosen. The shortest path algorithm optimizes a given flow by routing on the shortest path without considering other flows, thus, congestion may result if multiple paths share the same links.

- Wang-Crowcroft Algorithm – This algorithm is a bandwidth-delay-constrained algorithm [19]. It first prunes all links that do not satisfy the bandwidth demand. Then Dijkstra's Algorithm is run on the residual graph. A path is feasible when its delay is less than the delay constraint. This algorithm can be reduced to the shortest path algorithm. Instead of using hop count, it uses the delay as its metric. Thus, it is greedy, as discussed in the shortest path algorithm.

- Widest-Shortest Path Algorithm – Guerin, Orda, and Williams describe this algorithm in [16]. It first selects the shortest path. If there exists more than one shortest path, the widest path is chosen. The widest path is the path that has the largest residual bandwidth in its bottleneck link. Although it chooses the widest path to avoid congestion when there exists more than one shortest path, it is still greedy when there is only one shortest path.

- Shortest-Widest path Algorithm – Wang and Crowcroft describe this algorithm in [22]. It first selects the widest path. If there exists more than one widest path, the shortest path is chosen. This algorithm is greedy in that it attempts to use the least congested path for the current request with no consideration for requests that are to come.

- Awerbuch et al. Algorithm – This algorithm is a throughput-competitive routing algorithm [2], with bandwidth being the constraint. The objective function is to maximize the amortized throughput, and the costs in links are exponential to the utilization of bandwidth. This algorithm can be reduced to the shortest path problem with a different weight function. Thus, this algorithm is also greedy.

Algorithms that belong to the second group are:

- Minimum Interference Routing Algorithm – Kar, Kodialam and Lakshman describe the minimum interference routing algorithm in [10]. Its key concept is to defer the loading of critical links of other potential flows. Critical links are computed using maxflow computations. This algorithm is not greedy because it considers other flows when optimizing a given flow.

- Profile-Based Routing Algorithm – Suri, Waldvogel and Warkhede describe the profile-based routing algorithm in [18]. A multi-commodity is first computed which outputs a reduced graph. Then, the shortest path algorithm is run on the reduced graph. The objective is to satisfy as many requests as possible. This algorithm is not greedy because the multi-commodity computation looks at all flows when making its routing decision.

In our thesis, we select algorithms from each group for our evaluations. From the first group, we choose the shortest path algorithm, the widest-shortest path algorithm, and the shortest-widest path algorithm. From the second group, we choose the minimum interference routing algorithm and the profile-based routing algorithm. Each of these five algorithms is discussed in detail in the next chapter.

Chapter 3

The Problem

There are two sections in this chapter. The first section discusses the five algorithms we are going to investigate: the shortest path algorithm, the widest-shortest path algorithm, the shortest-widest path algorithm, the minimum interference algorithm, and the profile-based algorithm. The second section is the thesis statement.

3.1 Algorithms

3.1.1 Shortest Path (SP)

The shortest path routing algorithm is a simple algorithm. Given a source, a destination, and a bandwidth request, the algorithm returns a path which starts from the source and ends at the destination, with every link in the path possessing a residual bandwidth greater than or equal to the requested bandwidth. The term, *residual bandwidth* refers to the bandwidth available for allocation. We call such paths *feasible* paths. In a network topology, several feasible paths might exist. The shortest path routing algorithm selects the shortest path among all feasible paths where the shortest path is the path that has the least number of links between the source and the destination.

The major advantage of this algorithm lies in its simplicity. The shortest path algorithm is a greedy algorithm which optimizes a given flow and does not consider other flows. This can result in poor utilization of the overall network. Since flows are treated independently, it is possible for congestion to occur when multiple flows use the same links. Shortest path is unable to avoid congestion by taking other feasible paths; as a result the congested link can become even more congested, while other parts of the network remain under-utilized.

Latency is also a concern. There are basically two kinds of delay: queuing delay and propagation delay [22]. Shortest path minimizes the propagation delay by minimizing the number of hops in a path. However, queuing delay may increase due to congestion.

3.1.2 Widest-shortest path (WSP)

In [16], Guerin, Orda, and Williams describe the widest-shortest path algorithm. Widest-shortest path differs from shortest path by choosing the widest path if there exists more than one shortest path. The widest path is the one that has the largest residual bandwidth in its bottleneck link. A bottleneck link of a path is the link with the smallest residual bandwidth. Widest-shortest path improves upon shortest path because among all equal length paths, it attempts to avoid the bottleneck link hoping to avoid congestion. The algorithm can be described as follows:

1. Prune all links that do not satisfy the required bandwidth

2. Choose the shortest path (with respect to the number of hops)

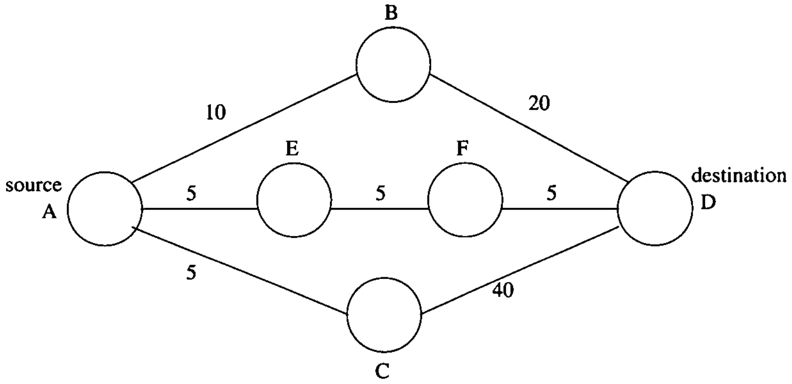


Figure 3.1: Widest-Shortest Path, residual capacities are shown on the link

3. If there exists more than one shortest path, choose the widest path (the one with the largest residual bandwidth in its bottleneck link)

Figure 3.1 shows a network topology with a request to allocate a tunnel from A to D having 3 units of demand. There are three feasible paths, $A \Rightarrow B \Rightarrow D$, $A \Rightarrow C \Rightarrow D$, and $A \Rightarrow E \Rightarrow F \Rightarrow D$. The algorithm chooses the shortest path. There are two shortest paths: $A \Rightarrow B \Rightarrow D$ and $A \Rightarrow C \Rightarrow D$. Both of these paths route through three hops. Among these two shortest paths, the algorithm chooses the widest path, $A \Rightarrow B \Rightarrow D$. This is because the bottleneck for $A \Rightarrow B \Rightarrow D$ is 10 units, and the bottleneck for $A \Rightarrow C \Rightarrow D$ is 5 units. The path with the largest residual bottleneck link capacity is chosen. Thus, this algorithm returns $A \Rightarrow B \Rightarrow D$.

Widest-shortest path improves on the shortest path algorithm when there is more than one shortest path. However, this algorithm is still based on the shortest path and the improvement only occurs when there is more than one shortest path. Thus it suffers from the same limitations as the shortest path algorithm. However, widest-shortest path may work well in networks with large numbers of equal length paths, i.e. fully meshed networks.

3.1.3 Shortest-widest path (SWP)

The shortest-widest path algorithm is described by Wang and Crowcroft in [22]. It is an improvement to shortest path since it attempts to avoid congestion by choosing the widest path. Shortest-widest path uses two metrics. The first metric used maximizes the smallest residual link capacity over all paths from source to destination. The second is to minimize the propagation delay. Recall that there are basically two kinds of delay: queuing delay and propagation delay. Queuing delay is estimated in the first metric, since if the bottleneck bandwidth in a link is small, it is likely that the queuing delay is large. These two metrics are called the width and the length of a path. The bottleneck residual bandwidth corresponds to the width of the path and the propagation delay corresponds to the length of the path. The algorithm optimizes each of these metrics in turn.

The algorithm is as follows. First prune all links that do not meet the bandwidth requirement. Then, find a path with the largest width, the largest bottleneck residual bandwidth. If several paths have the same largest bottleneck residual bandwidth, then the path with the minimum length is chosen. Note the difference between the shortest-widest path algorithm and the widest-shortest path algorithm. Widest-shortest path chooses the shortest path first whereas the shortest-widest path chooses the widest path first.

Shortest-widest path improves upon shortest path by maximizing the smallest residual link capacity in a path. However, there is a drawback to this algorithm. The algorithm first looks for a path with the largest width. The path with the largest width may be a long path from the source to destination. While on the one hand, it is avoiding congested links, on the other hand, it is using a lot of network resources by routing on longer non-shortest path links. Propagation delay also increases if the widest path is long. Shortest-widest path may be good on smaller networks. As well, one could imagine combining the width and the length by bounding the length of the widest path. There is no reason why long paths have to be accepted.

3.1.4 Minimum Interference Routing Algorithm (MIRA)

Kar, Kodialam, Lakshman describe the minimum interference routing algorithm in [10]. MIRA is an online, bandwidth guaranteed algorithm that does not split requests. MIRA maximizes the acceptance of demand by deferring the loading of critical links that might, if congested, lead to rejection of requests. Unlike the previously discussed greedy algorithms, MIRA considers other flows when making its routing decisions. Furthermore, MIRA uses information about the ingress egress pairs that previous algorithms do not use.

MIRA uses the maxflow computation. The idea of maxflow is to push as much flow as possible from a source a to a destination b [1]. The maxflow value for (a, b) decreases as a demand is routed from a to b. The maxflow value for (a, b) also decreases as a demand is routed from c to d if (c, d) routes its demand using (a, b)'s bottleneck edge. In this case, the routing of (c, d) interferes with (a, b). The key idea in MIRA is to minimize the interference. Suppose a request with ingress egress pair (a, b) arrives. MIRA chooses a path that satisfies the bandwidth requirement, and minimize the interference with the rest of the ingress egress pairs, i.e. excluding (a, b).

More formally, MIRA chooses its paths as follows. For each ingress egress

[1] Ingress egress nodes are in lowercases for general situations, and uppercases for examples.

pair excluding the ingress egress pair that belongs to the current request, a *critical set* is computed. The critical set consists of a set of critical edges. The critical set for any ingress egress pair is the minimum cut between that ingress egress pair. If there exists more than one minimum cut for that ingress egress pair, then the critical set is the union of all the minimum cuts. Intuitively, a critical set contains the bottleneck edges for that ingress egress pair. MIRA defers the loading of or "protects" critical links.

The ingress egress pairs are arranged in order of their relative importance. The importance of an ingress egress pair can be assigned by choosing functions that indicate the importance of an ingress egress pair. In this thesis, we define the importance of an ingress egress pair as the inverse of its maxflow. This implies that the smaller the maxflow is for an ingress egress pair, the more the importance of the ingress egress pair. The idea is to protect links that have small residual bandwidth.

For each ingress egress pair besides the ingress egress pair in the current request, calculate its maxflow and its critical set. For each edge in the topology, assign a weight. The weight function is:

$$w(l) = \sum_{(s,d):l \in C_{sd}} \alpha_{sd}$$

where (s,d) is an ingress egress pair, C_{sd} is the critical set for ingress egress pair, (s,d),

$$\alpha_{sd} = 1/\sigma_{sd}$$

is the importance of ingress egress pair (s,d), and σ_{sd} is the maxflow of (s,d). All the edges that do not meet the bandwidth requirement are pruned from the network and the shortest path algorithm is run on the reduced network.

Figure 3.2 shows a network topology with a request to allocate a tunnel from

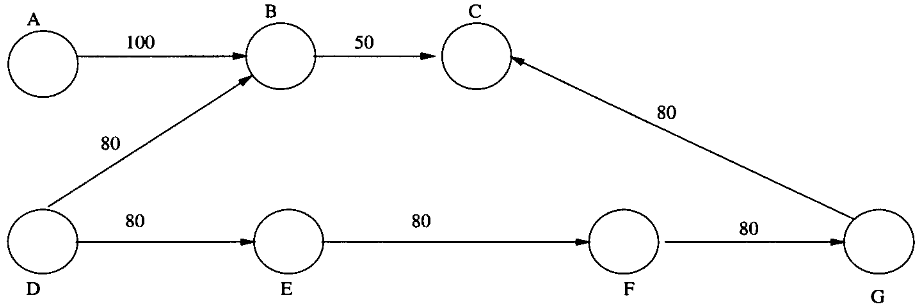


Figure 3.2: MIRA

D to C having 10 units of demand. Suppose that the only ingress egress pairs are (A, C) and (D, C). The critical set for (A, C) is $\{BC\}$ since BC is its minimum cut. The maxflow for (A, C) is 50. The importance of (A, C) is $1/50 = 0.02$. The weight for BC is 0.02 and the weights for all other links are set to a very small positive number, say 0.00001. The cost for $D \Rightarrow B \Rightarrow C$ is 0.02001, and the cost for $D \Rightarrow E \Rightarrow F \Rightarrow G \Rightarrow C$ is 0.00004. Thus MIRA chooses the path $D \Rightarrow E \Rightarrow F \Rightarrow G \Rightarrow C$ since 0.00004 is less than 0.02001.

Although MIRA's minimum interference idea is appealing, it has several limitations. Firstly, MIRA only focuses on the affect of interference for one ingress egress pair instead of a cluster of nodes. Secondly, MIRA does not use the information about demands of requests leading to situations where the protection of ingress egress pairs is unfavorable. Thirdly, MIRA does not have admission control. Fourthly, MIRA is computationally expensive since for every request, hundreds of maxflows may need to be calculated.

3.1.5 Profile-Based Routing Algorithm (PB)

Suri, Waldvogel, and Warkhede describe the profile-based routing algorithm in [18]. PB was designed to overcome the weaknesses in MIRA. PB uses a set of traffic profiles that enable PB to consider the blocking effect on a cluster of nodes instead of focusing on one ingress egress pair. Moreover, PB includes admission control. PB uses a multi-commodity flow computation that is used initially to reserve bandwidth based on estimates of the traffic demand. Once the bandwidth is reserved, shortest path is used to allocate bandwidth for individual requests.

PB, like MIRA, is an online algorithm. However, PB has two phases: an offline phase and an online phase. The offline phase is a multi-commodity flow computation, and the online phase is the shortest path algorithm. The basic idea of the multi-commodity flow algorithm is to push as much flows from sources to destinations as possible while minimizing a cost function.

PB uses the output of the multi-commodity flow computation to enforce admission control as well as to guide the shortest path algorithm. The output of the multi-commodity flow pre-allocates bandwidth for flows and the shortest path algorithm is executed on the pre-allocated bandwidth. When the pre-allocated bandwidth is not able to satisfy the demand, the request is rejected even though there may be sufficient bandwidth to satisfy the request. The offline and online phases are discussed in detail.

Offline, Preprocessing Phase

The first phase is the multi-commodity flow preprocessing phase. PB uses a set of traffic profiles. The information in the traffic profile can be obtained by monitoring the network, deduced from historical trend data or can be based on service level agreements (SLAs). SLAs are service specifications that both customers and providers have to adhere to. Each traffic profile is a tuple, (class ID, ingress node, egress node, requested bandwidth). The class ID specifies the class that the ingress egress pair belongs to. Some examples of classes include having all requests with the same ingress egress pair in one class, having all requests with the same egress node in one class and having all requests with the same bandwidth in one class. The requested bandwidth is an aggregated bandwidth between the ingress egress pair for that class. Since each traffic profile contains aggregated information, i.e. each traffic profile does not correspond to a particular flow, the aggregated flows in the traffic profile can be split. The splitting does not imply that each flow is split, since the information is aggregated, it simply implies that several flows are routed on other paths.

It might not be possible to route all requests in the set of traffic profiles, thus additional edges are added to the graph. For each ingress egress pair, an edge from the ingress to the egress node is added. This edge is called the excess edge where the capacity and the cost for it are both infinity. The cost for all non-excess edges is one. The objective of this phase is to push as much flow in the set of traffic profiles as possible, and also to minimize the cost. Each traffic class is one commodity. Assume that there are k commodities. The cost function is as follows:

$$minimize \sum (cost(e) \sum_{i=1}^{k} x_i(e))$$

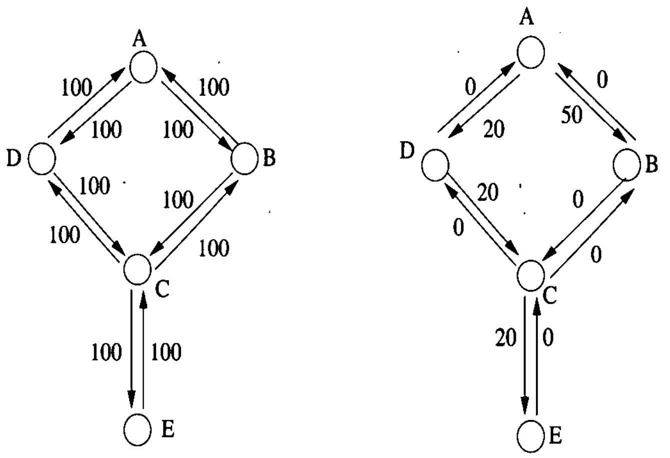
where

$$cost(e) = \{ {}^{1\ if\ e\ is\ not\ an\ excess\ edge}_{\infty\ if\ e\ is\ an\ excess\ edge}$$

is the cost of edge e, and $x_i(e)$ is the amount of flow in edge e for commodity i.

Online Phase

The online phase uses $x_i(e)$ from the offline phase. Initially, the capacity for each edge is set to $x_i(e)$ for each class resulting in a reduced graph for each class. Each



(a) Initial Capacities

(b) Reduced graph from output of the multi–commodity flow algorithm

Ingress Node	Egress Node	Required bandwidth
A	B	30
A	E	20
A	B	40

(c) Requests

Ingress Node	Egress Node	Aggregated bandwidth
A	B	50
A	E	20

(d) Traffic Profile

Figure 3.3: An Example for PB

request is assumed to map to a unique class and is routed on the pre-allocated bandwidth specified in the reduced graph using the shortest path algorithm.

If the demand for class i on edge e is greater than $x_i(e)$, then even though the real capacity is greater or equal to the demand, PB would not allow the demand to route on edge e. The values generated from the offline phase are used as thresholds for the online phase. Furthermore, a request that belongs to class i cannot use bandwidth that is reserved for class j.

Example

For the sake of simplicity, consider the following example using one class. Figure 3.3 (a) shows a network topology with its initial capacities. Figure 3.3 (b) shows the pre-allocated bandwidth output from the multi-commodity flow algorithm. Figure 3.3 (c) shows the requests, and Figure 3.3 (d) shows the set of traffic profiles. During the online phase, a tunnel $A \Rightarrow B$ is placed for the first request, a tunnel $A \Rightarrow D \Rightarrow C \Rightarrow E$ is placed for the second request, and the third is rejected since there is not enough pre-allocated bandwidth that can satisfy it although there is actually sufficient bandwidth to satisfy it.

PB uses class information. The mapping function that maps LSPs to classes is an important factor for the success of PB. This is because the multi-commodity flow pre-allocates bandwidth for each class, and a request that belongs to class A can only use bandwidth that is reserved for class A. Properties of the definition of classes that are of interest include the number of classes that LSPs are mapped into and which LSPs should be mapped into the same class.

We have mentioned the improvements of PB over MIRA. Let's look at some of its disadvantages. Firstly, PB assumes that the splitting of a commodity implies the splitting of a group of flows, not an individual flow. This is because PB uses an aggregation of bandwidth in the traffic profile. This causes a problem when an individual flow has a large demand. Secondly, PB's performance depends on the accuracy of the information provided in the set of traffic profiles.

3.2 Refinement of the Problem

We have seen four traffic allocation problems in the previous chapter, the constraint based routing problem, connection admission control problem, rerouting problem, and the network and capacity planning problem. The constraint based routing problem, the connection admission control problem, and the rerouting problem have both an online and offline counter part. The network and capacity planning problem are static problems since they have an estimation of requests. An offline scheme assumes that one has complete knowledge of the requests. An online scheme does not assume any prior knowledge regarding the future, specifically, the future sequence of requests. Each request allocates a tunnel as it arrives.

Although routing algorithms often consider tunnel provisioning to be dynamic, however, in practice, static provisioning is commonly used to provision tunnels corresponding to a service. The time for tunnel provisioning is often not dynamic, but rather, it is a batch process. A batch process allocates tunnels for a batch of requests instead of allocating a single tunnel for a single request as it arrives. This batch process can be viewed as a continuum with its ends being the offline and online scheme. It has more knowledge than the online scheme but less knowledge than the offline scheme. The knowledge that the batch process has is the history information.

There are two reasons for provisioning tunnels using a batch process and not dynamically. First, there is a time lapse from the moment when a company takes in service requests, to when the requests are passed on to the network engineer, to when the network engineer configures the network. Secondly, networks are usually only reconfigured at certain times, and usually do not reconfigure previous services as reconfiguration is prone to error and could lead to undesirable consequences. We do not consider the reconfiguration of existing tunnels, but it could, however, use the same algorithms as presented in the thesis.

Batch processing of tunnel provisioning is useful for many services. It is useful for services that are subscribed by customers on a regular basis and it is useful for services that do not require immediate action. For example, if a company receives ten services that require actions for tomorrow, then the company can process a batch of ten requests.

Our main concern is which of these algorithms are best suited for this type of environment. The batch process makes use of two kinds of history information. The first is the different sizes of batches. The second is the information that a specific algorithm gives, for example, the ingress egress pair, communication matrix, bandwidth, and the class that a LSP belongs to. The question thus can be rephrased to ask how sensitive are the algorithms to history information.

In this thesis, we address the route allocation problem. In particular, we focus on how a LSP can be optimally placed with respect to minimizing the number of rejected requests and rejected bandwidth, while satisfying the demand. We evaluate five algorithms, SP, WSP, SWP, MIRA and PB. In addition, we study two of the more sophisticated algorithms, MIRA and PB, with respect to their sensitivity to history information.

Chapter 4

Experimental Design

4.1 Introduction

In this section we describe the experimental set-up. All the algorithms were written in Java and we made use of BRITE for topology generation [13]. We used the Graph Theory Package compiled by Brian W. Bush to implement these algorithms [3], and used PPRN for the multi-commodity computation applied in the profile-based algorithm [4]. Experiments were conducted to compare the different algorithms, and only a representative subset of the total number of experiments run is presented. Topologies are shown in Appendix A.

4.2 Methodology

The experimental methodology used to investigate the algorithms is specified in this section. There are a number of parameters to be considered. These include the following:

1. Algorithms – The sensitivity of history of the online, batch process, and offline schemes are compared in these algorithms:

   - SP – the online, batch process and offline scheme for SP
   - WSP – the online, batch process and offline scheme for WSP
   - SWP – the online, batch process and offline scheme for SWP
   - MIRA online – the online scheme for MIRA
   - MIRA batch – the batch process scheme and the offline scheme for MIRA
   - PB online – the online scheme for PB
   - PB batch – the batch process scheme and the offline scheme for PB

2. Topology – In [14], Medina, Matta and Byers show that BRITE (Boston University Representative Internet Topology gEnerator) [13] generates topologies that resemble the Internet more closely than two very commonly used tools, namely, Waxman and Transit-Stub, as well as the grid. Their results are based on how closely topologies generated from those tools match the four power-law relationships of the Internet topology described by Faloutsos et al. in [7]. Brite is available at *http://www.cs.bu.edu/brite/download.html.*

   We used 15 topologies. Each topology consists of 30 nodes. Brite's bandwidth assignment is continuous. We want a discrete bandwidth assignment since in the real world, bandwidth assignment is not continuous. The three discrete bandwidths that are used are 10 Gb, 1 Gb and 100 Mb. We select the heavy tail distribution for bandwidth assignment in Brite and convert the output from Brite into the three discrete bandwidths. We choose the heavy tail distribution for bandwidth assignment because there are more edges with small bandwidth than large bandwidth in the networks.

The Objective function was throughput. In particular, the two metrics that measure throughput are the number of rejected requests and the amount of rejected bandwidth.

In order to compare the different algorithms, we fixed a number of the parameters and then varied the algorithms and topologies in order to compare the results.

The following parameters were fixed.

1. Distribution of Requests

    - Number of requests: 1000
    - Sequence of randomly generated requests: fixed
    - Size of requests: 1Mb, 5Mb, 10Mb
    - Assignment of request size: given by Zipf's law. Zipf's law states that $P_n \sim 1/n^a$ where P_n is the frequency in which n occurs, n is the item than has the n^{th} rank, and a is close to 1. Intuitively, Zipf's law states that only a few items occur very frequently. Most items seldom occur. Thus, by using Zipf's law, we are saying that small request size occur very frequently while large request size occur rarely. In 1000 requests, there are:

        - 546 1Mb requests
        - 273 5Mb requests
        - 181 10Mb requests

    Thus, the total bandwidth requested for these 1000 requests is 3721 Mb.

2. History Information – Batch Sizes

- Batch Sizes:
  - 1, 20, 50, 100, 500, 1000
  - For a batch size of 1, the algorithm corresponds to the online scheme since the algorithm only knows about the current request and has no knowledge about future requests. For a batch size of 1000, the algorithm corresponds to the offline scheme because the algorithm has complete knowledge of the future, and knows exactly which requests are coming. In between the online and offline scheme, we can consider processing requests in different sized batches. For each batch, we can use the information about the requests in the batch. The larger the batch size, the more history information is available to the algorithm.

4.3 Comparison Among all Algorithms

In this section, we compare the performance among SP, WSP, SWP, MIRA with batch size 100 and PB with batch size 100 with its class definition being the ingress egress pair. We make four observations.

1. WSP performs the best out of SP, WSP and SWP, as shown in Table 4.1. For the number of rejections, it performs better than SP and SWP in 12 out of 15 topologies. For the amount of bandwidth rejected, it performs better than SP and SWP in 13 out of 15 topologies. This is expected as WSP is an improvement to SP because it avoids congestion by taking the widest path, the path that has the largest bandwidth in its bottleneck link, when there exists more than one shortest path. It performs better than SWP although

	Topology	SP	WSP	SWP
No. of Rejections	1	116	92	215
Rejected Bandwidth(Mb)	1	152	124	407
No. of Rejections	6	151	122	238
Rejected Bandwidth(Mb)	6	235	166	510
No. of Rejections	7	80	44	162
Rejected Bandwidth(Mb)	7	80	44	260
No. of Rejections	10	17	12	144
Rejected Bandwidth(Mb)	10	17	12	256
No. of Rejections	14	47	33	178
Rejected Bandwidth(Mb)	14	71	37	338

Table 4.1: Performance for SP, WSP, SWP

SWP also avoids congestion because SWP, by avoiding congestion, may end up using long paths resulting in the use of more network resources.

2. SWP performs the worst out of SP, WSP and SWP, Table 4.1. For the number of rejections, it performs the worst in 13 out of 15 topologies, and 14 out of 15 topologies for the amount of bandwidth rejected. SWP takes into consideration the congestion, however, SWP may end up using a longer path than the shortest path by avoiding congested areas leading to the increase use in network resources.

	Topology	MIRA	PB
No. of Rejections	1	131	60
No. of Rejections	3	152	67
No. of Rejections	6	131	73
No. of Rejections	7	69	32
No. of Rejections	14	33	30

Table 4.2: Performance of MIRA and PB, No. of Rejections

3. PB performs better with the number of rejections as its metric. For the number of rejections, PB performs better than MIRA in 6 out of 10 topologies,

	Topology	MIRA	PB
Rejected Bandwidth(Mb)	1	187	223
Rejected Bandwidth(Mb)	3	164	167
Rejected Bandwidth(Mb)	6	187	255
Rejected Bandwidth(Mb)	7	77	210
Rejected Bandwidth(Mb)	14	37	150

Table 4.3: Performance of MIRA and PB, Rejected Bandwidth

	Topology	WSP	PB
No. of Rejections	1	92	60
No. of Rejections	3	69	67
No. of Rejections	6	122	73
No. of Rejections	7	44	32
No. of Rejections	14	33	30

Table 4.4: Performance of WSP and PB, No. of Rejections

Table 4.2, and performs better than WSP in 7 out of 10 topologies, Table 4.4. The 10 topologies are the topologies that do not have "stop" as the result. The "stop" represents a bug in PPRN package that we use for the multi-commodity flow algorithm. We have contacted the author of the package regarding this bug. However he was unable to fix it at the time the experiments were run. This package is available at *http://www-eio.upc.es/~jcastro/pprn.html.* However, for the amount of rejected bandwidth, both MIRA and WSP perform better in all 10 topologies, Table 4.3, Table 4.5. This is because PB assumes that information is aggregated and the splitting of commodity is actually the splitting of groups of flows instead of individual flow. Thus, PB may favor requests that have smaller bandwidth demand which enables it to accept more requests.

	Topology	WSP	PB
Rejected Bandwidth(Mb)	1	124	223
Rejected Bandwidth(Mb)	3	77	167
Rejected Bandwidth(Mb)	6	166	255
Rejected Bandwidth(Mb)	7	44	210
Rejected Bandwidth(Mb)	14	37	150

Table 4.5: Performance of WSP and PB, Rejected Bandwidth

	Topology	MIRA	WSP
No. of Rejections	1	131	92
Rejected Bandwidth(Mb)	1	187	124
No. of Rejections	3	152	69
Rejected Bandwidth(Mb)	3	164	77
No. of Rejections	6	131	122
Rejected Bandwidth(Mb)	6	187	166
No. of Rejections	7	69	44
Rejected Bandwidth(Mb)	7	77	44
No. of Rejections	14	33	33
Rejected Bandwidth(Mb)	14	37	37

Table 4.6: Performance of MIRA and WSP

4. Greedy algorithms do not always perform worse than algorithms that are based on network flow computations, Table 4.6. For the number of rejections, WSP performs better than MIRA in 11 out of 15 topologies, and they have the same performance in topology 14. For the amount of bandwidth rejected, WSP performs better than MIRA in 9 out of 15 topologies and performs the same in 2 topologies. This is because MIRA needs the ingress egress pair to perform maxflow computations, however, as we will see later, history information does not always enhance performance.

4.4 History

This section discusses the role of history and how it appears in algorithms SP, WSP, SWP, MIRA and PB. We also evaluate the effect of history in these five algorithms.

4.4.1 SP, WSP, SWP

SP, WSP and SWP only use the network topology state information. They do not use any algorithm specific history information such as ingress egress pairs or bandwidth demands. SP, WSP, and SWP are not sensitive to history because they do not use any other information except the topology state information. Thus, the batch process of tunnel provisioning does not help in achieving a better utilization of the network.

		No. of Requests per Batch					
	Algorithm	20	50	100	200	500	1000
No. of Rejections	SP	46	46	46	46	46	46
Rejected Bandwidth (Mb)	SP	100	100	100	100	100	100
No. of Rejections	WSP	25	25	25	25	25	25
Rejected Bandwidth (Mb)	WSP	70	70	70	70	70	70
No. of Rejections	SWP	44	44	44	44	44	44
Rejected Bandwidth (Mb)	SWP	128	128	128	128	128	128

Table 4.7: Topology 15 for SP, WSP and SWP

Table 4.7 shows that as the amount of history increases, both the number of rejected requests and the amount of rejected bandwidth remain constant for SP, WSP and SWP. Table 4.7 is the result for topology 15. Of course, this behavior occurs in 15 out of 15 topologies for SP, WSP and SWP.

4.4.2 MIRA

Like SP, WSP and SWP, MIRA uses the network topology state information. An additional knowledge MIRA uses is the ingress and egress nodes. The ingress and egress nodes are used as the specific history information for MIRA. We separate MIRA into two cases, MIRA online and MIRA batch. MIRA online corresponds to the online scheme, and MIRA batch corresponds to the batch process and the offline scheme.

4.4.3 MIRA Online

MIRA predetermines a set of ingress egress nodes before it considers any request. In MIRA online, the predetermined set of ingress egress nodes is all possible pairs of ingress egress nodes in the topology. This corresponds to knowing nothing about the history because all possible pairs of ingress egress nodes are used. MIRA has to protect all these ingress egress pairs even though there might be no requests from some of the ingress egress pairs. Thus, this is the online scheme in that it does not know anything about the history.

MIRA online adds all ingress egress pairs, and this is used in the experiments as a model for MIRA online where it has no history information, i.e. batch size of 1. Note that batch size of 1 is not reasonable since no interference is possible. As a result, there is a discontinuity from MIRA online for no history to batch size of 20.

4.4.4 MIRA Batch

The batch process makes use of the history information that is in the current batch when making its routing decision. In order to use the batch process, some modifications are made to MIRA. The modification is called MIRA batch. Instead of predetermining a set of ingress egress nodes to be used, a dynamically chosen set of ingress egress nodes is used. The set of ingress egress nodes for each batch is tailored for that particular batch. For each batch, the set of ingress egress pairs used includes those ingress egress pairs that appear at least once in the current batch of requests. This implies that all ingress egress nodes that are used will correspond to some requests in the future, and protecting them is beneficial for satisfying those later requests. If the batch size is 1000, then it corresponds to the offline scheme because it knows the entire history as there are a total of 1000 requests.

The larger the size of the batch implies that one would have more history information. In our modification, only ingress egress nodes that belongs to the current batch of requests is protected by MIRA when handling the requests in the current batch. This means that if there is some ingress egress nodes that are not in the current batch, but are in the next batch, then they will not be protected in the current batch. Thus, it is expected that the larger the size of the batch, the better MIRA performs.

Evaluation of History –MIRA Online

Topology	No. of Rejections	Rejected Bandwidth
3	141	153
5	51	67
6	115	171
9	0	0

Table 4.8: MIRA Online

One would speculate that MIRA is sensitive to history because it uses the ingress egress nodes. Furthermore, MIRA batch should perform better than MIRA online since MIRA batch corresponds to the batch process and offline scheme. However, this is not the case.

Table 4.8 shows the number of rejections out of 1000 requests and the rejected bandwidth out of 3721Mb for MIRA online for topology 3, 5, 6 and 9. Compare these numbers with Table 4.9. MIRA batch does not always perform better than MIRA online. In topology 3, MIRA batch does not perform better than MIRA online for each batch size 50, 100 and 200. In topology 5, for the number of rejections, MIRA online is better than MIRA batch for batch sizes 50 and 100. For bandwidth rejection, MIRA batch with batch sizes 20, 50 and 100 does not perform better than MIRA online. In topology 6, for the number of rejections, MIRA online is better than MIRA batch for all batch sizes. Finally, in topology 9, MIRA online is not better or worse than MIRA batch because an excellent performance is achieved for both MIRA online and MIRA batch as all entries are 0.

Topology 3, 5 and 6 are not the only topologies in which MIRA batch does not perform better than MIRA online. This is true for 13 out of 14 topologies. Topology 9 is not included since the number of rejections and the amount of rejected bandwidth are 0 for both MIRA online and MIRA batch.

This set of experiments shows that MIRA is not sensitive to history. This can also be seen in the next set of experiments. The reason for MIRA not being sensitive to history is presented after the presentation of the results for MIRA batch.

Evaluation of History – MIRA Batch

MIRA batch corresponds to the batch process scheme and the offline scheme. Batch sizes 20, 50, 100, 200, 500 and 1000 correspond to varying the amount of history information. One would speculate that the larger the size of the batch, the more knowledge regarding the future, and hence, the better the performance.

		No. of Requests per Batch					
	Topology	20	50	100	200	500	1000
No. of Rejections	3	111	177	152	157	121	122
Rejected Bandwidth(Mb)	3	127	205	164	161	141	142
No. of Rejections	5	44	62	61	35	42	52
Rejected Bandwidth(Mb)	5	91	78	77	55	58	68
No. of Rejections	6	142	158	131	135	127	124
Rejected Bandwidth(Mb)	6	214	254	187	195	183	176

Table 4.9: MIRA Batch

However, this is not the case. Table 4.9 shows that for topology 3, 5 and 6, there is no trend that shows the larger the size of the batch, the better the performance for both the number of rejected requests and the amount of rejected bandwidth. In fact, in 11 out of 14 topologies for the rejected bandwidth and 10 out of 14 topologies for the number of rejections, it is observed that the larger the size of the batch does not imply better performance. Topology 9 is again not included since MIRA batch performs so well that all entries are 0.

MIRA's Sensitivity to History

It is clear from the experimental results that history does not play an important role in MIRA. Particularly, as the amount of history information increases, the performance does not increase. There is no trend to show that performance for both the number of rejected requests and the amount of rejected bandwidth improves with increasing batch size or amount of history information. This implies that in some situations, MIRA performs better with more history information, while in other situations, MIRA performs better with less history information. Table 4.9 shows that for topology 3, the performance for batch size 100 is worse than the performance for batch size 20. In this case, MIRA performs better when it knows less. On the

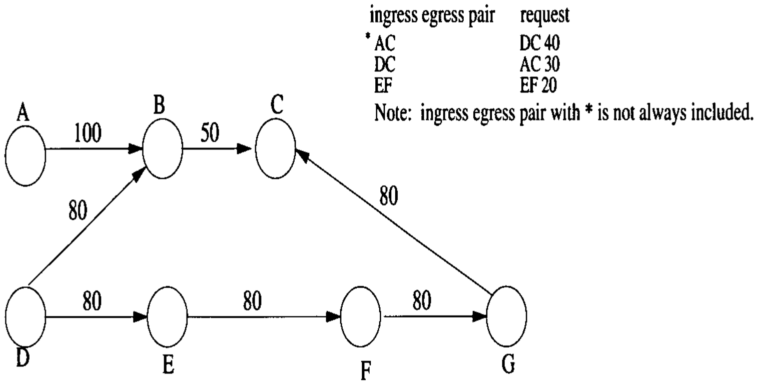

ingress egress pair request
*AC DC 40
DC AC 30
EF EF 20
Note: ingress egress pair with * is not always included.
A
B
C
100
50
80
80
80
80
80
D
E
F
G

Figure 4.1: An Example for MIRA to Perform Better with More History

other hand, the performance for batch size 500 is better than the performance for batch size 100. In this case, MIRA performs better when it knows more. Table 4.9 gives more examples of this finding.

First, let us first consider MIRA's enhanced performance with more history. Recall that the history information used in MIRA is the ingress egress pair, and one of the key ideas of MIRA is to minimize interference. Assume that MIRA knows about the ingress egress pair (a, b), and MIRA is currently routing the request (e, f). When MIRA routes (e, f), it tries to protect all ingress egress pairs that it is aware of, including (a, b), by minimizing the interference on their critical edges. If MIRA is not aware of (a, b), that is, if MIRA has less history information, then MIRA will not protect (a, b). Then, other requests may use up the critical edges of (a, b) causing (a, b) to be rejected. In situations like this, MIRA performs better with more history information.

Figure 4.1 shows a network, the ingress egress pairs and the requests. The case in which MIRA knows about the ingress egress pair (A, C) will be discussed first. The current request is (D, C). When (D, C) chooses its path, it considers ingress egress pairs (A, C) and (E, F). The critical set of (A, C) is $\{BC\}$, with its maxflow value being 50. The critical set of (E, F) is $\{EF\}$, with its maxflow value being 80. Therefore, the weight for BC is 0.02, the weight for EF is 0.0125, and the weight for all other links are set to a small positive value, say 0.00001. (D, C) chooses $D \Rightarrow E \Rightarrow F \Rightarrow G \Rightarrow C$ as opposed to $D \Rightarrow B \Rightarrow C$ since the weight for $D \Rightarrow E \Rightarrow F \Rightarrow G \Rightarrow C$ is 0.01253 and the weight for $D \Rightarrow B \Rightarrow C$ is 0.02001. The request (A, C) is accepted since BC has enough residual bandwidth for it. Thus, all requests are satisfied. However, if MIRA does not know about (A, C), then when (D, C) makes its routing decision, it does not consider (A, C). Thus, the weight for all links are 0.00001 except for EF whose weight is 0.0125. Thus, (D, C) routes on $D \Rightarrow B \Rightarrow C$ instead of $D \Rightarrow E \Rightarrow F \Rightarrow G \Rightarrow C$. In this case, (A, C) is rejected. Thus, in this example, MIRA performs better with more history information.

This can be seen in Table 4.9. In topology 3, the number of rejections and the amount of rejected bandwidth decreases as the batch goes from 50 to 100, 200 to 500 and 100 to 500. This is also shown in topology 5 as the batch size goes from 50 to 200 and 50 to 500, in topology 6 as the batch size goes from 50 to 500, 20 to 1000 and 50 to 100.

Now, let us consider situations in which MIRA performs better with less history. Three of these situations are discussed.

Assume MIRA acknowledges the ingress egress pair (a, b), and assume that the maxflow of (a, b) is small. Since the maxflow of (a, b) is small, the weight of the critical edges for (a, b) is high. Thus, other ingress egress pairs route around these critical edges and choose other longer paths. The three situations are as follows:

1. The demand of (a, b) is much smaller than its maxflow.

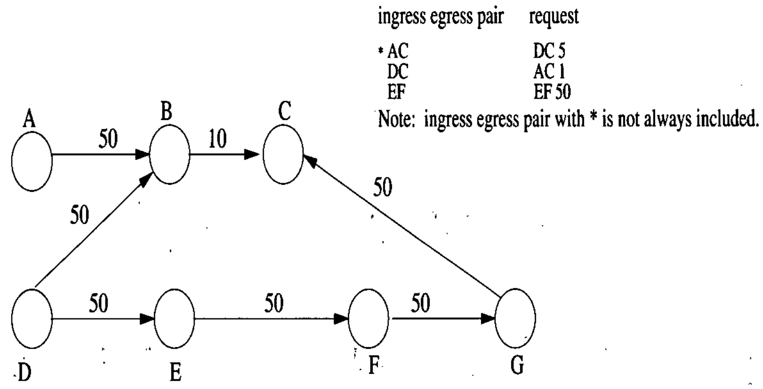

ingress egress pair request
* AC DC 5
DC AC 1
EF EF 50
Note: ingress egress pair with * is not always included.
A
B
C
D
E
F
G
50
10
50
50
50
50
50

Figure 4.2: MIRA Performs Better With Less History, Example 1

2. Rejecting (a, b) is better than protecting it.

3. The demand of (a, b) is larger than its maxflow.

Situation 1: The demand of (a, b) is much smaller than its maxflow

If MIRA does not know about (a, b), (a, b) will still be accepted despite the inability of MIRA to protect its critical edges. (a, b) will be accepted because its demand is much smaller than its maxflow. Furthermore, other ingress egress nodes do not have to take longer paths which would use up network resources, and in turn reject other requests. Thus, in this situation, MIRA performs better with less history.

Figure 4.2 shows a network topology, the set of ingress egress pairs, and the set of requests. Consider the case if MIRA knows about (A, C). When (D, C) makes its routing decision, it considers the ingress egress pairs (A, C) and (E, F). The critical set for (A, C) is $\{BC\}$, and the critical set for (E, F) is $\{EF\}$. The maxflow value for (A, C) is 10 and the maxflow value for (E, F) is 50. Thus, the weight for BC is 0.1 and the weight for EF is 0.02. All other links are assigned a weight of a small positive number, 0.00001. There are two feasible paths for (D, C),

$D \Rightarrow B \Rightarrow C$ whose weight is 0.10001 and $D \Rightarrow E \Rightarrow F \Rightarrow G \Rightarrow C$ whose weight is 0.02003. Thus, (D, C) chooses $D \Rightarrow E \Rightarrow F \Rightarrow G \Rightarrow C$. EF now has a residual bandwidth of 45, thus (E, F) is blocked because it demands 50 units of flow. In this case, both (D, C) and (A, C) are accepted but (E, F) is blocked.

Now consider the case in which MIRA does not acknowledge (A, C). When (D, C) makes its routing decision, it considers the ingress egress pair (E, F). The critical set for (E, F) is $\{EF\}$, and its maxflow value is 50. Thus, the weight for EF is 0.02 while all the other weights are 0.00001. Again, there are two feasible paths for (D, C), $D \Rightarrow B \Rightarrow C$ whose weight is 0.00002 and $D \Rightarrow E \Rightarrow F \Rightarrow G \Rightarrow C$ whose weight is 0.02003. Thus, (D, C) chooses $D \Rightarrow B \Rightarrow C$. In this case, (E, F) is not blocked since the residual capacity for EF is 50, and (A, C) is not blocked since the residual capacity for BC is 5 and (A, C)'s demand is only 1. Thus, all three requests are accepted. This example illustrates the first case for MIRA to perform better with less history.

Situation 2: Rejecting (a, b) is better than protecting it

If MIRA does not know about (a, b), (a, b) will be rejected since MIRA will not be able to protect (a, b). However, by rejecting (a, b), other ingress egress pairs do not have to take longer paths which use up network resources which in turns reject other requests. MIRA may perform better by rejecting (a, b) rather than protecting it. Thus, if MIRA does not know about (a, b), MIRA may perform better.

Figure 4.3 shows a network topology, a set of ingress egress pairs and a set of requests. Consider the case which MIRA knows about the ingress egress pair (A, C). When (D, C) makes its routing decision, it considers the ingress egress pairs (A, C), (E, F) and (F, G). The critical set for (A, C) is $\{BC\}$ and its maxflow

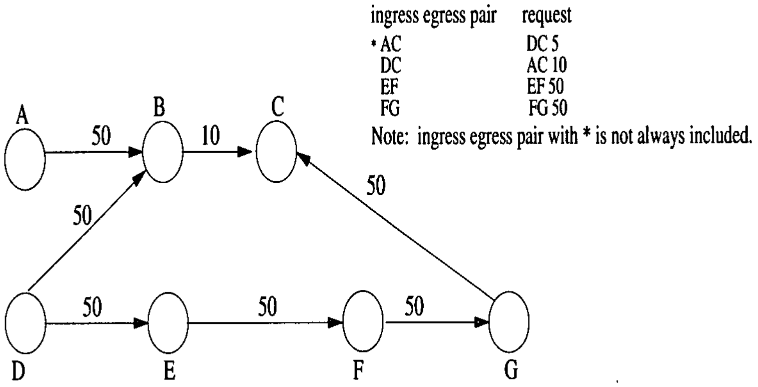


Figure 4.3: MIRA Performs Better With Less History, Example 2

value is 10. The critical set for (E, F) is $\{EF\}$ and its maxflow value is 50. The critical set for (F, G) is $\{FG\}$ and its maxflow value is 50. Thus, the weights for the links are as follows: BC has a weight of 0.1, EF has a weight of 0.02, FG has a weight of 0.02, and the rest of the links have a weight of 0.00001. (D, C) routes on $D \Rightarrow E \Rightarrow F \Rightarrow G \Rightarrow C$ as supposed to $D \Rightarrow B \Rightarrow C$ since the weight for $D \Rightarrow E \Rightarrow F \Rightarrow G \Rightarrow C$ is 0.04002 and the weight for $D \Rightarrow B \Rightarrow C$ is 0.10001. Thus, (A, C) is accepted and routes on $A \Rightarrow B \Rightarrow C$, (E, F) and (F, G) are rejected since the residual bandwidths for EF and FG are both 45 and their demands are 50. Thus, (A, C) is accepted but both (E, F) and (F, G) are rejected.

Consider the case in which MIRA is not aware of (A, C). When (D, C) makes its routing decision, it considers (E, F) and (F, G). The weights for EF and FG are both 0.02. The weights for the rest of the links are 0.00001. Thus, (D, C) chooses $D \Rightarrow B \Rightarrow C$ to route on since its weight is 0.00002 and the weight for $D \Rightarrow E \Rightarrow F \Rightarrow G \Rightarrow C$ is 0.04002. This results in the rejection of (A, C) since the residual bandwidth on BC is 5 and (A, C)'s bandwidth demand is 10. However, both (E, F) and (F, G) are accepted. Thus, in this case, MIRA accepts more requests

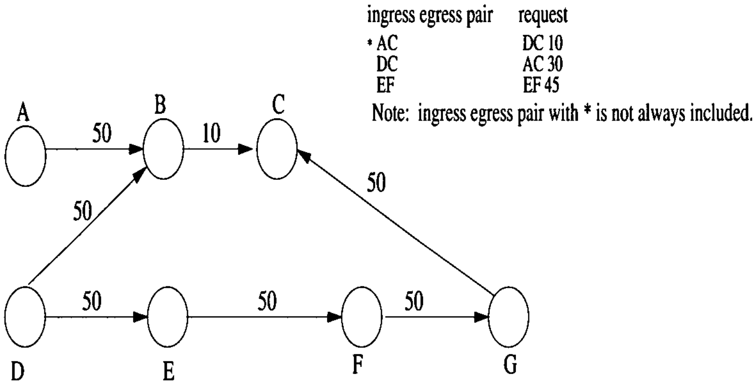

ingress egress pair request
* AC DC 10
DC AC 30
EF EF 45
Note: ingress egress pair with * is not always included.
A
B
C
D
E
F
G
50
10
50
50
50
50
50

Figure 4.4: MIRA Performs Better With Less History, Example 3

when it knows less.

Situation 3: The demand of (a, b) is larger than its maxflow

No matter how MIRA protects the critical edges of (a, b), (a, b) will not be accepted as the demand of (a, b) is larger than its maxflow. Since MIRA knows about (a, b), MIRA will protect it causing other requests to take longer paths. Longer paths imply using up network resources which in turn may reject other requests. Thus, if MIRA does not know about (a, b), it will perform better.

Figure 4.4 shows a network topology, ingress egress pairs and requests. Assume MIRA knows about (A, C). When (D, C) makes its routing decision, it considers (A, C) and (E, F). Thus, the weight for BC is 0.1 and the weight for EF is 0.02. All other links have weight 0.00001. (D, C) chooses $D \Rightarrow E \Rightarrow F \Rightarrow G \Rightarrow C$ since its weight is 0.02003 which is less than 0.10001, the weight for $D \Rightarrow B \Rightarrow C$. Thus, both (A, C) and (E, F) are rejected. If MIRA does not know about (A, C), (D, C) routes on $D \Rightarrow B \Rightarrow C$. In this case, only (A, C) is blocked. Thus, in this case, MIRA performs better when it knows less.

Therefore, in all three situations, one can see that MIRA does not always perform better when it has more history. This can be seen in our experiments, Table 4.9. In topology 3, the number of rejected requests and the amount of rejected bandwidth increase as the batch size increases from 20 to 50 and from 20 to 100. This can also be seen in topology 5 where the batch size increases from 200 to 1000 and from 500 to 1000, and in topology 6, where batch size increases from 100 to 200 and from 20 to 50.

4.4.5 PB

Like all previously discussed algorithms, PB uses the network topology state information. Additionally, PB requires a set of traffic profiles. There are basically three kinds of knowledge in a traffic profile. They are:

1. class information
2. ingress egress pairs
3. the aggregated bandwidth requirement for each ingress egress pair in that class

All three kinds of knowledge are used as the specific history information for PB. We separate PB into two cases, PB online and PB batch. PB online corresponds to the online scheme, and PB batch corresponds to the batch process and offline scheme.

4.4.6 PB Online

In PB online, the predetermined set of traffic profiles uses all possible ingress egress nodes in the topology and uses a constant as the aggregated bandwidth. This corresponds to the online case because PB online knows nothing about the history.

Topology	Demanded Bandwidth	No. of Rejections	Rejected Bandwidth
1	1	838	3294
1	5	568	1986
2	1	812	3342
2	5	573	2029
3	1	813	3337
3	5	585	2047
4	1	832	3375
4	5	582	2069
5	1	825	3397
5	5	576	2153

Table 4.10: PB Online, Topology 1, for having 1 class

Table 4.10 shows the result for using 1 and 5 as the aggregated bandwidth. PB online using one class performs poorly for all 15 topologies. The result is not surprising since one would expect PB online to perform poorly when there is no history information. The performance of PB relies on the amount of information the set of traffic profiles provides.

4.4.7 PB Batch

History information obtained from the current batch is used for the batch process scheme. In order to use the batch process scheme, some modifications to PB are made. The modified version of PB is known as PB batch. Instead of predetermining a set of traffic profiles, PB batch dynamically builds the set of traffic profiles.

For each batch, the set of traffic profiles only includes those ingress egress pairs that appear at least once in the current batch of requests. The required bandwidth for each ingress egress pair in a class is no longer an estimate but it is the aggregated bandwidth from the current batch of requests. All ingress egress pairs that are in the set of traffic profiles will be used by some requests in the future.

When the batch size is 1000, this corresponds to the offline scheme.

PB requires two phases: the preprocessing phase and the online phase. For 1000 requests, the preprocessing phase is executed once, then each 1000 requests are routed one at a time in the online phase. PB batch requires more than two phases. In PB batch, for each batch of requests, these two phases are needed. For example, if the size of the batch is 500, then for each 500 requests, a preprocessing phase and an online phase are needed.

PB Classes

The definition of classes is not well defined in [18] and we attempt to experimentally determine how it affects the quality of the solution. In particular, we investigate how the definition of classes or the mapping function for mapping requests to classes affect PB's sensitivity to history. There are two issues regarding classes of PB that are to be tested, the number of classes which requests are mapped to, and the mapping function for defining which requests should belong to the same class.

PB batch is expanded to test the properties of classes in PB. It is expanded such that requests are mapped into classes using different mapping functions. Six mapping functions are used:

1. Map all requests into one class.

2. Map all requests that have the same bandwidth requirement into one class. There are at most three classes according to this mapping function since bandwidth requirement can only be 1Mb, 5Mb or 10Mb.

3. Map all requests that have the same egress node into one class. There are at most thirty classes since the topologies contain 30 nodes.

4. Map all requests that have the same ingress and egress nodes into one class. There are at most eight hundred and seventy classes according to this mapping function since the topologies contain 30 nodes. Thus, there are 30 * 29 = 870 different ingress egress pairs.

5. Randomly assign requests to three different classes.

6. Randomly assign requests to five hundred different classes.

By using different mapping functions to map requests to different number of classes, one class, three classes, thirty classes, five hundred classes and eight hundred and seventy classes, one can see how the number of classes affects PB's sensitivity to history. Furthermore, by varying the mapping function for defining which requests should belong to the same class, one can see how the definition of classes affects PB's sensitivity to history.

Evaluation of History - PB Batch with Topologically and Randomly Assigned Classes

This set of experiments investigates whether PB's sensitivity of history is affected by the way requests are being classified. Two groups of mapping functions are used in assigning classes, one topologically and the other randomly.

1. Topologically Assigned Classes – Mapping functions that topologically assigned classes ensure requests that are in the same class share some topological properties. Two such mapping functions are studied here.

   The first mapping function is to map all requests that have the same egress node into one class. The second mapping function is to map all requests that have the same ingress and egress nodes into one class. One would expect that the more history PB has, the better PB performs.

Topology		No. of Requests per Batch					
		20	50	100	200	500	1000
4	No. of Rejections	39	34	26	22	stop	stop
4	Rejected Bandwidth(Mb)	255	210	160	135	stop	stop
8	No. of Rejections	47	32	33	33	19	13
8	Rejected Bandwidth(Mb)	282	200	150	138	155	100
9	No. of Rejections	39	38	22	stop	stop	stop
9	Rejected Bandwidth(Mb)	249	217	112	stop	stop	stop
15	No. of Rejections	41	30	stop	stop	37	26
15	Rejected Bandwidth(Mb)	201	160	stop	stop	159	143

Table 4.11: PB Batch, Egress Node As Mapping Function

Topology		No. of Requests per Batch					
		20	50	100	200	500	1000
4	No. of Rejections	37	31	28	24	stop	stop
4	Rejected Bandwidth(Mb)	240	195	170	145	stop	stop
8	No. of Rejections	48	31	34	33	24	9
8	Rejected Bandwidth(Mb)	278	195	165	133	170	80
14	No. of Rejections	58	34	30	stop	stop	stop
14	Rejected Bandwidth(Mb)	249	214	150	stop	stop	stop
15	No. of Rejections	41	30	stop	stop	32	25
15	Rejected Bandwidth(Mb)	201	160	stop	stop	138	138

Table 4.12: PB Batch, Ingress Egress Pair As Mapping Function

Both Table 4.11 and Table 4.12 demonstrates that the more history PB has, the better it performs as speculated. As the size of the batch increases, indicating more history information is available, the better the performance of PB. For the egress node as the mapping function, this is shown in 11 out of 15 topologies for the amount of rejected bandwidth. For the ingress egress pair as the mapping function, this is shown in 12 out of 15 topologies for the amount of rejected bandwidth. Thus, PB is sensitive to history when these two mapping functions of are used.

2. PB Batch with Randomly Assigned Classes – Mapping functions that randomly assign classes do not characterize any property of requests that are in the same class. A mapping function that randomly maps requests into five hundred classes is used. Experiments for this mapping function are run 5 times, and the result presented here is the average of the 5 experiments, with the outliers removed.

		No. of Requests per Batch					
	Topology	20	50	100	200	500	1000
No. of Rejections	2	86.2	81.4	61.8	56.3	stop	stop
No. of Rejections	4	39.3	33.8	33	30.6	stop	stop
No. of Rejections	8	72.2	44	42.4	28.4	26.8	20.2
No. of Rejections	15	47.8	43.2	stop	33.3	38.8	24.2

Table 4.13: No. of Rejections for PB Batch with 500 Randomly Assigned Classes

		No. of Requests per Batch					
	Topology	20	50	100	200	500	1000
Rejected Bandwidth(Mb)	1	497.7	324.8	415.6	315.8	stop	273.8
Rejected Bandwidth(Mb)	2	355.6	335.8	244.2	239.3	stop	stop
Rejected Bandwidth(Mb)	4	245	209	176.8	156.2	stop	stop
Rejected Bandwidth(Mb)	14	236.4	220	206.2	171.2	stop	stop

Table 4.14: Rejected Bandwidth for PB Batch with 500 Randomly Assigned Classes

Again, one would expect that the larger the size of the batch, the better the performance of PB batch. Results are shown in Table 4.13 and Table 4.14. This trend is shown in 11 out of 15 topologies for the amount of bandwidth rejected. Thus PB is also sensitive to history when requests are randomly assigned to five hundred classes.

For both groups of mapping functions, the more the history, the better PB performs is shown. PB is sensitive to history when requests that are in the same class share some topological properties, and when requests that are in the same class may not have any property in common. Thus PB's sensitivity to history does not depend on the definition of which requests should belong to the same class.

Evaluation of History - PB Batch with a Few Number of Classes

In this section, we investigate PB's sensitivity to history and the number of classes PB uses. We see in the previous section that PB is sensitive to history when a considerable number of classes are used:

- The mapping function that maps all requests having the same egress node into one class has at most thirty classes
- The mapping function that maps all requests having the same ingress egress pair into one class has at most eight hundred and seventy classes
- The mapping function that randomly assign classes has five hundred classes

We are to investigate PB's sensitivity to history when only a few number of classes are used. Let's start with PB batch using one class.

		No. of Requests per Batch					
	Topology	20	50	100	200	500	1000
No. of Rejections	1	69	99	126	129	177	290
No. of Rejections	6	68	108	134	174	stop	stop
No. of Rejections	14	56	83	127	138	stop	stop
No. of Rejections	15	61	84	stop	stop	158	287

Table 4.15: No. of Rejections for PB Batch with 1 Class

	No. of Requests per Batch					
	20	50	100	200	500	1000
Topology 1	393	554	732	619	523	498
Topology 2	389	536	730	568	stop	stop
Topology 3	260	440	613	stop	stop	stop
Topology 4	301	394	493	436	stop	stop
Topology 5	264	372	stop	stop	stop	stop
Topology 6	306	532	609	583	stop	stop
Topology 7	249	504	680	stop	stop	stop
Topology 8	408	558	643	623	530	438
Topology 9	420	540	786	stop	stop	stop
Topology 10	299	415	663	stop	stop	stop
Topology 11	259	497	690	stop	stop	stop
Topology 12	300	562	stop	stop	stop	stop
Topology 13	453	534	stop	stop	stop	stop
Topology 14	320	476	654	559	stop	stop
Topology 15	263	349	stop	stop	545	469

Table 4.16: Amount of Bandwidth Rejected for PB Batch with 1 Class

One would expect that the larger the size of the batch, the better the performance of PB batch. However, such a trend is not seen from the above results when PB batch uses only one class, Table 4.15 and Table 4.16. Moreover, the trend for the number of rejection is just the opposite, the more history, the worse the performance. In fact, this situation is seen in 13 out of 15 topologies for the number of rejections.

Consider what happens when PB batch uses three classes. There are two mapping functions that map requests to at most three classes. The first mapping function is to map all requests that have the same bandwidth requirement into one class. Recall that the bandwidth requirement can only be 1Mb, 5Mb or 10Mb. Thus, this mapping function maps requests to at most three classes. The second mapping function randomly assigns requests to three classes. For this mapping function, we ran the experiment five times, and the results presented here is the average of the five experiments.

		No. of Requests per Batch					
	Topology	20	50	100	200	500	1000
No. of Rejections	1	59	94	118	159	stop	161
Rejected Bandwidth(Mb)	1	296	354	448	686	stop	706
No. of Rejections	2	84	107	128	140	stop	stop
Rejected Bandwidth(Mb)	2	354	380	475	482	stop	stop
No. of Rejections	4	49	91	96	117	stop	stop
Rejected Bandwidth(Mb)	4	263	305	384	482	stop	stop
No. of Rejections	6	62	116	132	175	stop	stop
Rejected Bandwidth(Mb)	6	289	361	409	523	stop	stop

Table 4.17: PB Batch with Bandwidth as the Mapping Function

		No. of Requests per Batch					
	Topology	20	50	100	200	500	1000
No. of Rejections	1	56.6	60.8	85.4	105.6	147	228.8
No. of Rejections	4	41	49.4	54.2	81	stop	stop
No. of Rejections	6	61.8	78.2	90.2	114	stop	stop
No. of Rejections	8	52	59.6	69	109.8	141.2	171.6

Table 4.18: No. of Rejections for PB Batch with 3 Randomly Assigned Classes

Again, one would expect that the larger the size of the batch, the better the performance of PB batch. However, such a trend is not seen from the above results when PB batch uses only three classes, Table 4.17, Table 4.18 and Table 4.19. Moreover, the trend for both mapping functions is the opposite; the more history PB has, the worse it performs. In fact, for mapping all requests that have the same bandwidth requirement into one class, the trend, the more history, the worse it performs, is seen in 15 out of 15 topologies for the number of rejections and 12 out of 15 topologies for the amount of rejected bandwidth. For randomly mapping requests into three classes, this trend is seen in 15 out of 15 topologies for the number of rejections. An explanation is given next.

		No. of Requests per Batch					
	Topology	20	50	100	200	500	1000
Rejected Bandwidth(Mb)	3	227.8	270.4	319.2	stop	stop	stop
Rejected Bandwidth(Mb)	6	263.6	299.4	392.6	475.8	stop	stop
Rejected Bandwidth(Mb)	11	247.4	269.8	335.2	stop	stop	stop
Rejected Bandwidth(Mb)	14	245.6	250.6	298	510	stop	stop

Table 4.19: Rejected Bandwidth for PB Batch with 3 Randomly Assigned Classes

Recall that PB requires two phases: the preprocessing phase and the online phase. The preprocessing phase is a multi-commodity flow computation on the set of traffic profiles. The online phase is based on the output from the preprocessing phase. The output of the multi-commodity flow computation provides the bandwidth information for each edge for each class. Denote $x_i(e)$ to be the amount of bandwidth on edge e for class i output by the multi-commodity flow algorithm. The online phase is the shortest path algorithm running on the reduced graph. Requests that belong to class i cannot use bandwidth that is reserved for class j.

When only a few classes are used, a large amount of bandwidth reservation is made from the multi-commodity flow computation for each class. This is because there are a large number of requests in a class since there are only a few number of classes. This is what happened in our experiments since we used a small number of classes, one and three, and we have 1000 requests. Thus, for the case of using one class, 1000 requests are in that class, and for the case of three classes, on average, there are 333 requests per class. The multi-commodity flow computation knows which bandwidth is reserved for which commodity within a class. However, the shortest path algorithm does not know. It only considers the shortest path that is available on the reduced graph. For a request that belongs to class i, the shortest path algorithm finds the shortest path on the reduced graph with bandwidth that is reserved for class i. The shortest path algorithm is not able to tell which bandwidth is reserved for which request. It only knows which bandwidth is reserved for which class. Furthermore, due to the large amount of bandwidth that is reserved for class i, the shortest path may choose to route on bandwidth that is reserved by the multi-commodity flow computation for other requests within the same class. The online phase has somehow overridden the intelligence from the preprocessing phase. Therefore, when a few number of classes are used, it is not the case that the more history information, the better PB performs. This is because the more history information implies more requests per batch which implies more bandwidth per class.

Figure 4.5 is an example to illustrate this situation. For the sake of simplicity, an example in which PB batch that uses one class is presented. Capacities that are not shown in Figure 4.5 are of no significance for our purposes. There are two sets of requests, A and B. A contains requests that are in two batches of sizes two and three. B contains requests that are in one batch of size five.

Let's look at Requests A first. For the first batch, the multi-commodity flow algorithm reserves 20 units of bandwidth in the path $I \Rightarrow B \Rightarrow C \Rightarrow D$ for ID, and reserves 10 units of bandwidth in the path $E \Rightarrow B \Rightarrow C \Rightarrow D$. Thus, the online phase will route ID on the path $I \Rightarrow B \Rightarrow C \Rightarrow D$ for ID and route ED on the path $E \Rightarrow B \Rightarrow C \Rightarrow D$. When the second batch is handled, AD is rejected since the only path from A to D is $A \Rightarrow B \Rightarrow C \Rightarrow D$ and the capacity for BC is 30. When handling the first batch, PB did not know about AD since it is in the second batch. Thus, Requests A rejects 1 request, AD 30.

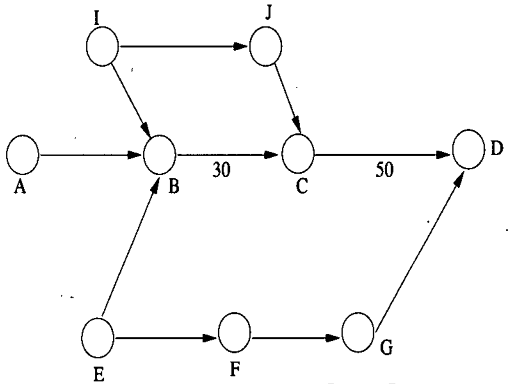
Let's now look at Requests B. PB is able to see all 5 requests since they are all in one batch. One would expect Requests B to perform better than Requests

A since it knows more. The multi-commodity flow algorithm reserves 20 units of bandwidth in the path $I \Rightarrow J \Rightarrow C \Rightarrow D$ for ID, reserves 10 units of bandwidth in the path $E \Rightarrow F \Rightarrow G \Rightarrow D$ for ED, reserves 30 units of bandwidth in the path $A \Rightarrow B \Rightarrow C \Rightarrow D$ for AD, reserves 10 units of bandwidth in EB and reserves 20 units of bandwidth in IB. After finishing the preprocessing phase, the online phase is performed. The shortest path route ID using the path $I \Rightarrow B \Rightarrow C \Rightarrow D$ since this path uses 4 hops and the path $I \Rightarrow J \Rightarrow C \Rightarrow D$ also uses 4 hops. The shortest path algorithm uses bandwidth that is not reserved for ID, but it is intended for AD and IB. The shortest path route ED using the path $E \Rightarrow B \Rightarrow C \Rightarrow D$ for ED since this path uses 4 hops and the path $E \Rightarrow F \Rightarrow G \Rightarrow D$ uses 4 hops as well. The shortest path algorithm uses bandwidth that is not reserved for ED but it is intended for AD and EB. Thus, Requests B rejects 3 requests, AD 30, EB 10 and IB 20.

Thus, from this example, we can see that when PB uses a few number of classes, it is not true that the larger the size of the batch, the better the performance of PB . This also explains the trend the more history, the worse PB performs. The experiments show that PB is not sensitive to history when only a few number of classes are used

4.4.8 History

After looking at the five algorithms and their sensitivity to history individually, we are going to compare their sensitivity to history in this section. From the above discussions as well as from Figure 4.6 and Figure 4.7, one can see that SP, WSP and SWP are not sensitive to history at all. The number of rejected requests and the amount of rejected bandwidth do not fluctuate as the amount of history information



Requests A

Ingress Egress Pair	Demanded Bandwidth
ID	20
ED	10
xxx	xxx
AD	30
EB	10
IB	20

Requests B

Ingress Egress Pair	Demanded Bandwidth
ID	20
ED	10
AD	30
EB	10
IB	20

Figure 4.5: Example Illustrating the More Information PB has, the Worse it Performs When Using a Few Number of Classes

changes. On the other hand, we have seen that MIRA is also insensitive to history. However, MIRA's number of rejected requests and the amount of rejected bandwidth fluctuate as the amount of history information changes. Out of the five algorithms, PB is the algorithm that is most sensitive to history. It is expected since PB contains more history information than the other algorithms. PB is able to show the trend, the more history, the better it performs under appropriate class definitions.

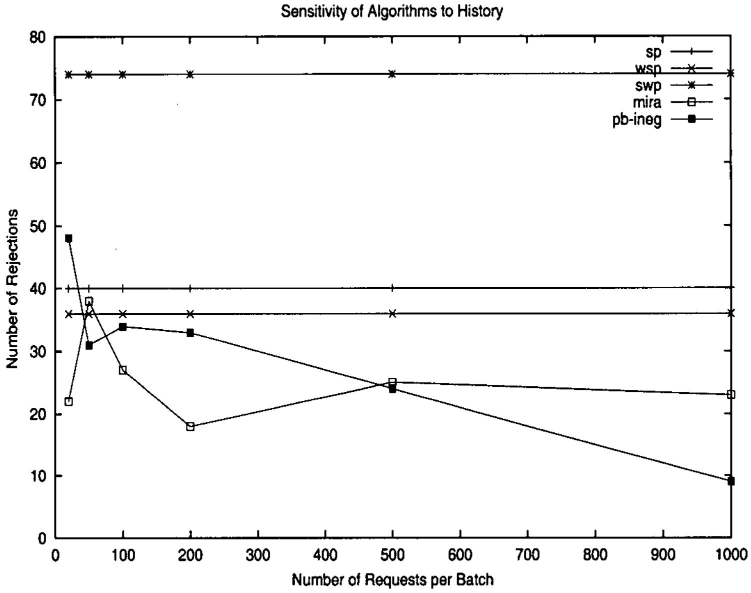


Figure 4.6: Algorithms and their Sensitivity in terms of the Number of Rejections, Topology 8

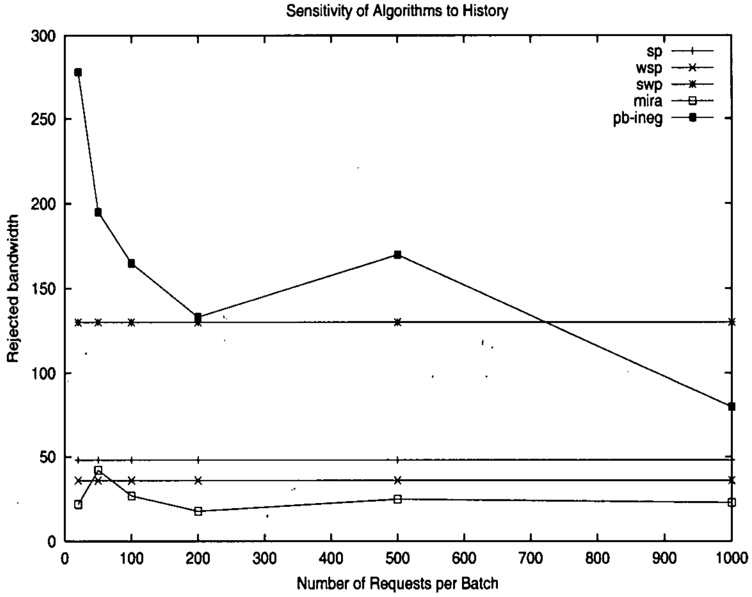


Figure 4.7: Algorithms and their Sensitivity in terms of the Amount of Bandwidth Rejected, Topology 8

Chapter 5

Conclusion

In this thesis, we considered the route allocation problem of optimal placing of a LSP with guaranteed bandwidth. The metrics that are used are the number of rejected requests and the amount of rejected bandwidth. We evaluate five routing algorithms: the shortest path algorithm, the widest-shortest path algorithm, the shortest-widest path algorithm, the minimum interference routing algorithm and the profile-based routing algorithm. Furthermore, we look at the sensitivity of history in routing algorithms. In particular, we investigate which algorithms are well-suited for the batch process scheme.

For the evaluation of SP, WSP, SWP, MIRA and PB, we see that WSP performs the best out of SP, WSP and SWP. We also see that SWP performs the worst out of SP, WSP and SWP although both WSP and SWP attempt to avoid congestion. In avoiding congestion, SWP may ultimately take longer paths which in turn would use up network resources and this may hinder its performance. Moreover, we see that PB performs better with the number of rejected requests as its metric. This is because the information in the set of traffic profiles is aggregated and PB assumes the splitting of an aggregation of flow instead of an individual flow can be split. Furthermore, we see that an algorithm which uses the network flow computation does not necessarily perform better than a greedy algorithm.

As for the sensitivity of history in routing algorithms, one can see that SP, WSP and SWP are not sensitive to history. The number of rejected requests and the amount of rejected bandwidth remain constant as the amount of history information changes. Since they are not sensitive to history, the batch process scheme does not help in attaining better network utilization. This result is as speculated since no history information is incorporated into the algorithms.

On the other hand, we have seen that MIRA is also insensitive to history although it uses history information when it routes requests. As the size of the batch increases, indicating the amount of history information increases, the performance does not increase conjunctively. Thus, we conclude that history does not play an important role in MIRA. In some situations, MIRA performs better with more history information, and in other situations, MIRA performs better with less history information.

MIRA could be improved in order to be more sensitive to history. The improvement requires MIRA to use the demand of the ingress egress pair as the history information in addition to the ingress egress pair itself. Recall there are three situations in which MIRA performs better with less history information.

1. The demand size for (a, b) is smaller than its maxflow, and MIRA does not need to protect (a, b).

2. Rejecting (a, b) is better than protecting it.

3. The demand size for (a, b) is larger than its maxflow, and MIRA does not need to protect (a, b).

Consider the first and the last situations. If MIRA uses the demand of the ingress egress nodes as its history information, then it can acknowledge that the demand for (a, b) is smaller or larger than its maxflow, and that protecting (a, b) may not be advantageous. For the second situation, MIRA needs some admission control mechanism to decide that rejecting (a, b) is better than protecting it. The knowledge of demand may also be helpful in this aspect.

Out of the five algorithms, PB is most sensitive to history. PB is able to demonstrate the trend that the more history PB has, the better it performs under appropriate class definitions. We have explored how the definition of classes affect PB's sensitivity to history, and find that the number of classes in which requests are mapped to affects PB's sensitivity to history. When a few classes are used, the performance of PB is worse as it receives more history information. This is true when requests are mapped into one class, or when requests with the same bandwidth requirement are mapped into one class or requests are randomly mapped into three classes. When the ingress egress pair or the egress node is used as the mapping function, more classes are used. When more classes are used, the trend, the more history, the better the performance can be shown. Furthermore, when requests are randomly mapped into five hundred classes, the trend can also be seen. Therefore, we conclude that the number of classes used affects PB's sensitivity to history. However, the ways requests are classified does not affect PB's sensitivity to history. PB shows the trend when more classes are used, regardless of whether requests that are in a class share some topological information or whether requests are randomly assigned into classes.

In conclusion, we see that algorithms that do not use history in their algorithms are not sensitive to history and that algorithms that do use history in their algorithms are not implicitly sensitive to history, with MIRA being an example of such a condition. Furthermore, algorithms that show the trend, the more history, the better their performance, may possess other factors that affect their sensitivity to history, with PB being a prime example. Thus, the intuitive idea which suggests that algorithms with more future knowledge will perform better is not necessarily true. Performance depends on how well the history information is used.

The routing decisions made by routing algorithms are crucial to the performance of the network. Thus it is important for routing algorithms to make wise decisions. Therefore, we have evaluated the performance of five routing algorithms. Furthermore, with the batch process scheme, routing algorithms are able to make use of more history information when making its routing decision. Routing algorithms that are able to use history information well greatly enhances the utilization of the network resources and the performance of the network. With better performance of the network, the network is able to accommodate more traffic and attain higher quality of service.

5.1 Future Work

From our studies, we see the potential advantage of improving the performance of algorithms through the availability of more history information by using the batch process scheme. We have achieved some preliminary understanding of how the use of history affects the performance of routing algorithms. We have also seen that whether history information can facilitate the performance of an algorithm depends on how well the history information is used. Two questions regarding the improvement of performance through the use of history are raised.

1. MIRA uses the ingress egress pair as its history information while PB uses the class information, the ingress egress pair as well as the aggregated bandwidth as its history information. What kind of history information should be incorporated into routing algorithms?

2. How can algorithms use history better?

Improvements can be made to PB so that PB can use history better. It would be advantageous if PB can be sensitive to history under all circumstances. However, we have seen that PB is sensitive to history only under appropriate class definitions. PB is not sensitive to history when a few classes are used. When only a few classes are used, there are a large number of requests per class. Thus, the multi-commodity flow computation reserves a large amount of bandwidth per class. The shortest path algorithm is not able to distinguish which bandwidth is reserved for which request within a class. Therefore, the intelligence of the multi-commodity flow computation is somehow overridden by the shortest path algorithm. Consequently, one might be inclined to eliminate the shortest path algorithm. However, the shortest path algorithm serves two purposes. Firstly, the multi-commodity flow algorithm uses a set of traffic profiles that is an estimation of the traffic as opposed to the actual traffic. Thus, the shortest path algorithm is needed to handle the actual traffic. Secondly, the multi-commodity flow algorithm allows the splitting of a commodity. The splitting of a commodity can become problematic when large requests are split because individual requests are not allowed to be split. Thus, the shortest path algorithm is needed to ensure that an individual request can route on a single path. Therefore, the shortest path algorithm is indispensable despite its negative effects on the multi-commodity flow algorithm.

We propose several modifications to PB so that the shortest path algorithm is not necessary, and in turn, PB can be sensitive to history under all situations. The modifications are as follows:

1. Instead of using the multi-commodity flow algorithm, use an unsplittable multi-commodity flow algorithm. With this modification, the shortest path algorithm is no longer needed to ensure that an individual request routes on a single path since the unsplittable multi-commodity flow algorithm does not split commodities.

2. Increase the output of the unsplittable multi-commodity flow algorithm by a certain percentage. With this change, the shortest path algorithm is no longer needed to handle the actual traffic. The set of traffic profiles that is used by the unsplittable multi-commodity flow algorithm is an estimation of the traffic. The actual traffic may be more than the estimated traffic. Thus, the output of the unsplittable multi-commodity flow algorithm is increased by a certain percentage in order to accommodate the traffic that is not being estimated.

3. Route requests according to the increased output from the unsplittable multi-commodity flow algorithm.

With these adjustments, PB does not need the shortest path algorithm, and thus, PB is able to use all intelligence from the unsplittable multi-commodity flow algorithm.

Implementation and testing of these modifications are required in order to determine whether PB can be sensitive to history under all circumstances with these modifications.

Another area that requires further study is the development of algorithms for the batch process scheme. Routing algorithms for this scheme are needed because both the online and offline routing schemes are not suitable. History information is not used by dynamic routing but is used by the batch process scheme. Offline routing algorithms use history information and time complexity is not an issue. However, time is an issue for the batch process scheme. Consequently, neither the online nor the offline routing algorithms are ideal for this scheme, and therefore algorithms specialized for the batch process scheme need to be developed.

More convincing routing algorithms are also needed. There is a vast amount of literature on routing algorithms, but the current deployed routing algorithms are still based on the shortest path algorithm.

Bibliography

[1] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J.McManus. Requirements for traffic engineering over MPLS. *RFC 2702*, September 1999.

[2] B. Awerbuch, Y.Azar, S. Plotkin, and O. Waarts. Throughput-competitive on-line routing. *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California*, November 1993.

[3] Brian W. Bush. Graph theory package.

[4] Jordi Castro and Narcis Nabona. PPRN. *Statistics and Operations Research Department, Universitat Politecnica de Catalunya, Barcelona (Spain)*, September 1994.

[5] Shigang Chen and Klara Nahrstedt. An overview of quality of service routing for next-generation high-speed networks: Problems and solutions. *IEEE Networks, Special Issue on Transmission and Distribution of Digital Video*, Nov/Dec 1998.

[6] Bruce Davie and Yakov Rekhter. *MPLS Technology and Applications.* Morgan Kaufmann Publishers, 2000.

[7] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. *AGM SIGCOMM, Cambridge, MA*, September 1999.

[8] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. *IEEE INFOCOM*, March 2000.

[9] Muckai K Girish, Bei Zhou, and Jian-Qiang Hu. Formulation of the traffic engineering problems in MPLS based ip networks. *Proceedings of the Fifth IEEE Symposium on Computers and Communications*, 2000.

[10] Koushik Kar, Murali Kodialam, and T.V. Lakshman. Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications. *IEEE Journal on Selected Areas in Communications*, 18(12), Dec 2000.

[11] Dave Katz, Derek Yeung, and Kireeti Kompella. Traffic engineering extensions to OSPF. *Internet Draft*, October 2001.

[12] G. Malkin. Rip version 2. *Internet Standards Track Protocol*, November 1998.

[13] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. Brite (boston university representative internet topology generator). *Boston University, http://www.cs.bu.edu/brite/download.html.*

[14] Alberto Medina, Ibrahim Matta, and John Byers. On the origin of power laws in internet topologies. *ACM Computer Communication Review*, 30(2), April 2000.

[15] J. Moy. OSPF version 2. *Internet Standards Track Protocol*, April 1998.

[16] R.Guerin, Ariel Orda, and D. Williams. QoS routing mechanisms and OSPF extensions. *Proceedings of 2nd Global Internet Miniconference(joint with Globecom'97)*, November 1997.

[17] Eric C. Rosen, Arun Viswanathan, and Ross Callon. Multiprotocol label switching architecture. *Internet Draft*, July 2000.

[18] Subhash Suri, Marcel Waldvogel, and Priyank Ramesh Warkhede. Profile-based routing: A new framework for MPLS traffic engineering. *Lecture Notes in Computer Science*, 2156, 2001. Quality of Future Internet Services.

[19] Z. Wang and J. Crowcroft. QoS routing for supporting resource reservation. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, September 1996.

[20] Xipeng Xiao, Alan Hannan, Brook Bailey, and Lionel M. Ni. Traffic engineering with MPLS in the internet. *IEEE Network Magazine*, pages 28–33, March 2000.

[21] Xipeng Xiao and Lionel M. Ni. Internet QoS: A big picture. *IEEE Network*, 13(2):8–18, March 1999.

[22] Z.Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE JSAC*, 14(7):1288–1234, September 1996.

Appendix A

Topologies

These are the topologies that are used in this thesis.

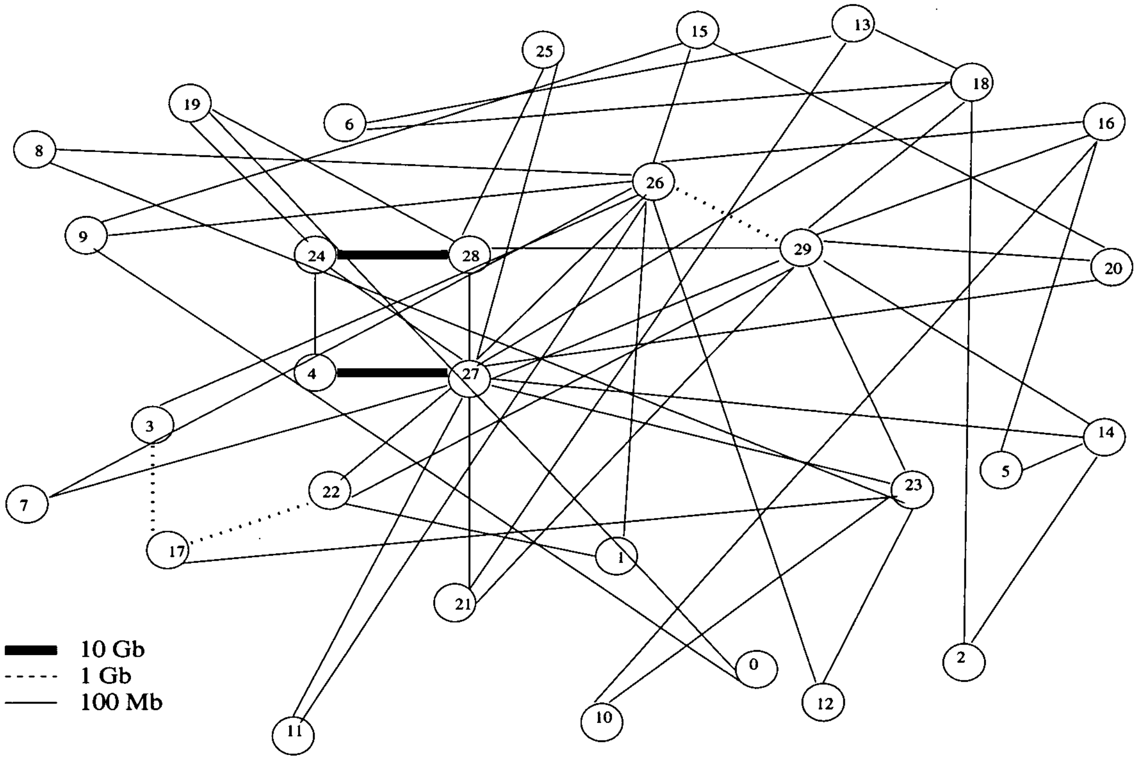


Figure A.1: Topology 1

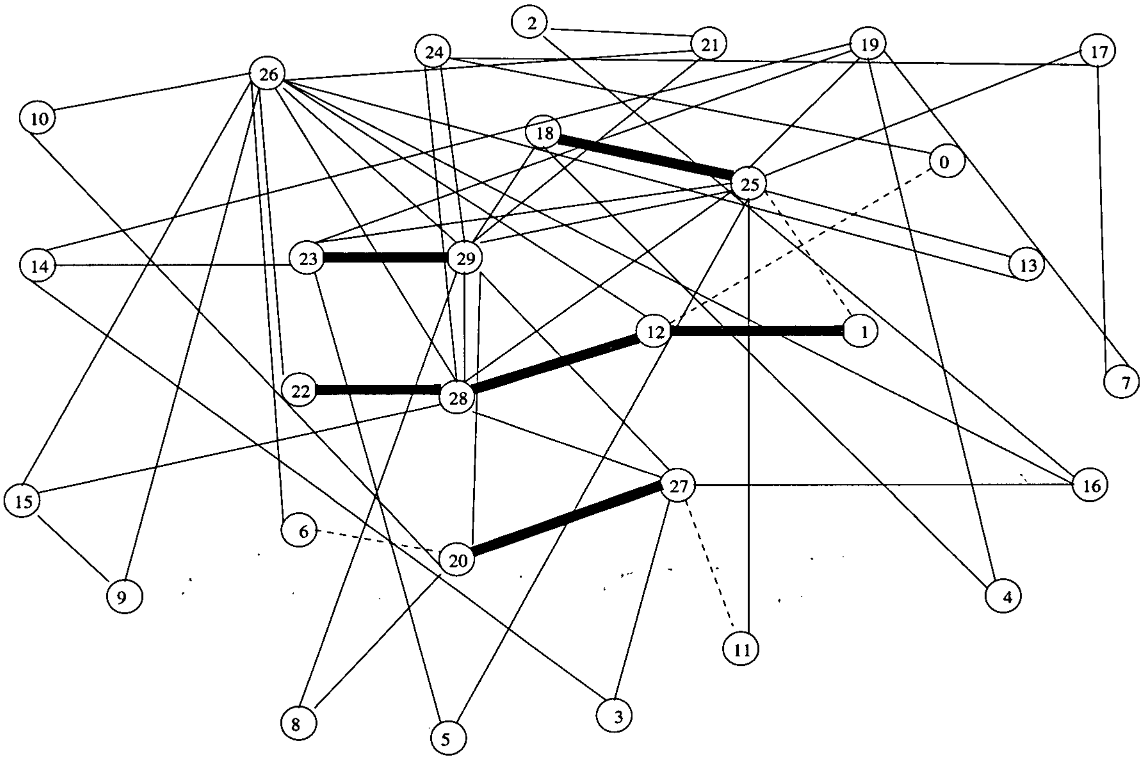


Figure A.2: Topology 2

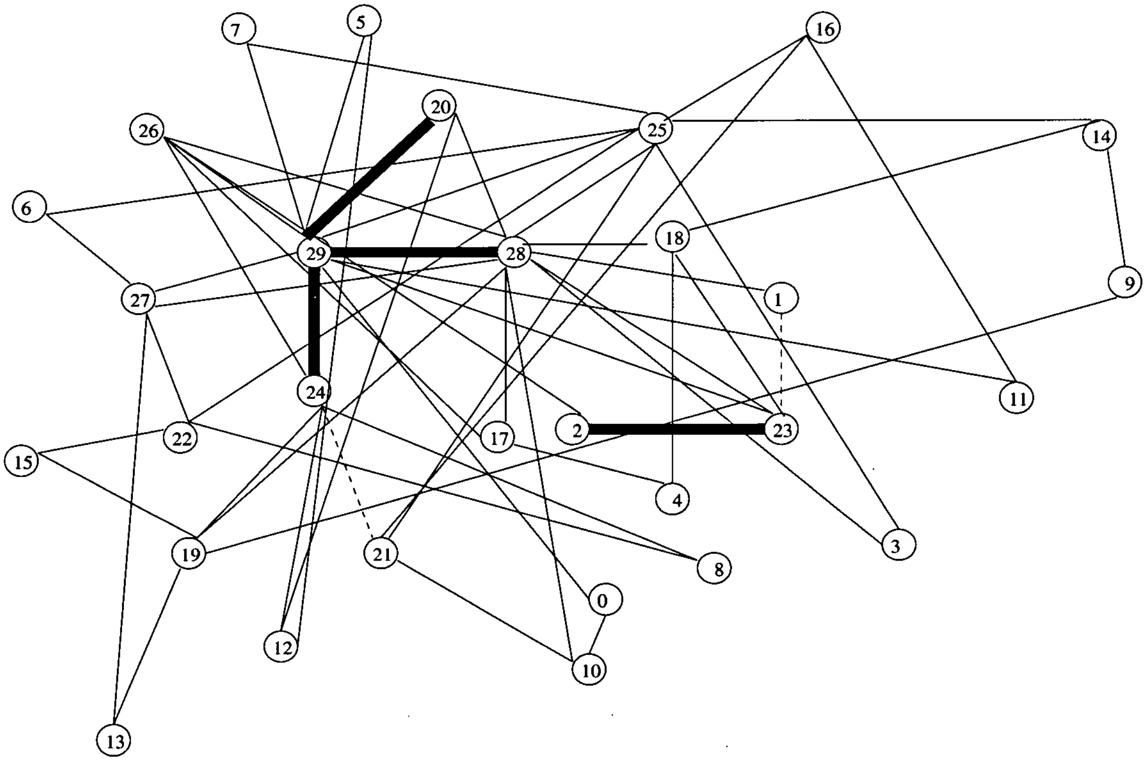


Figure A.3: Topology 3

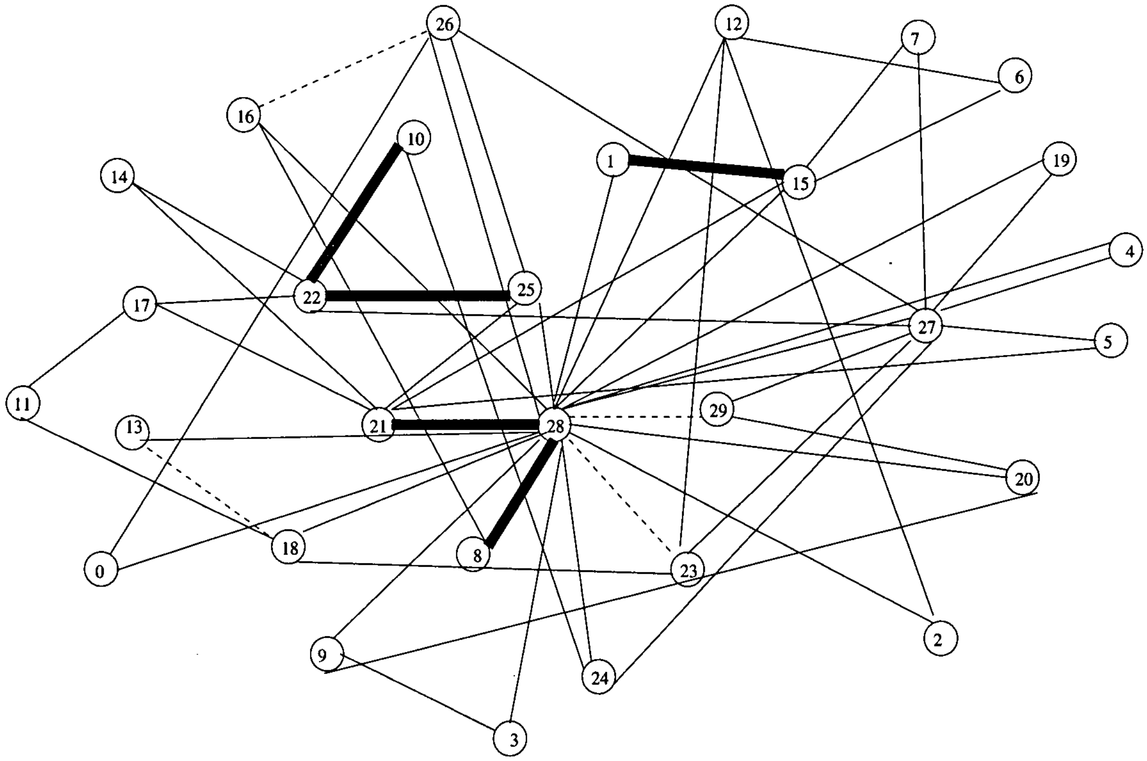


Figure A.4: Topology 4

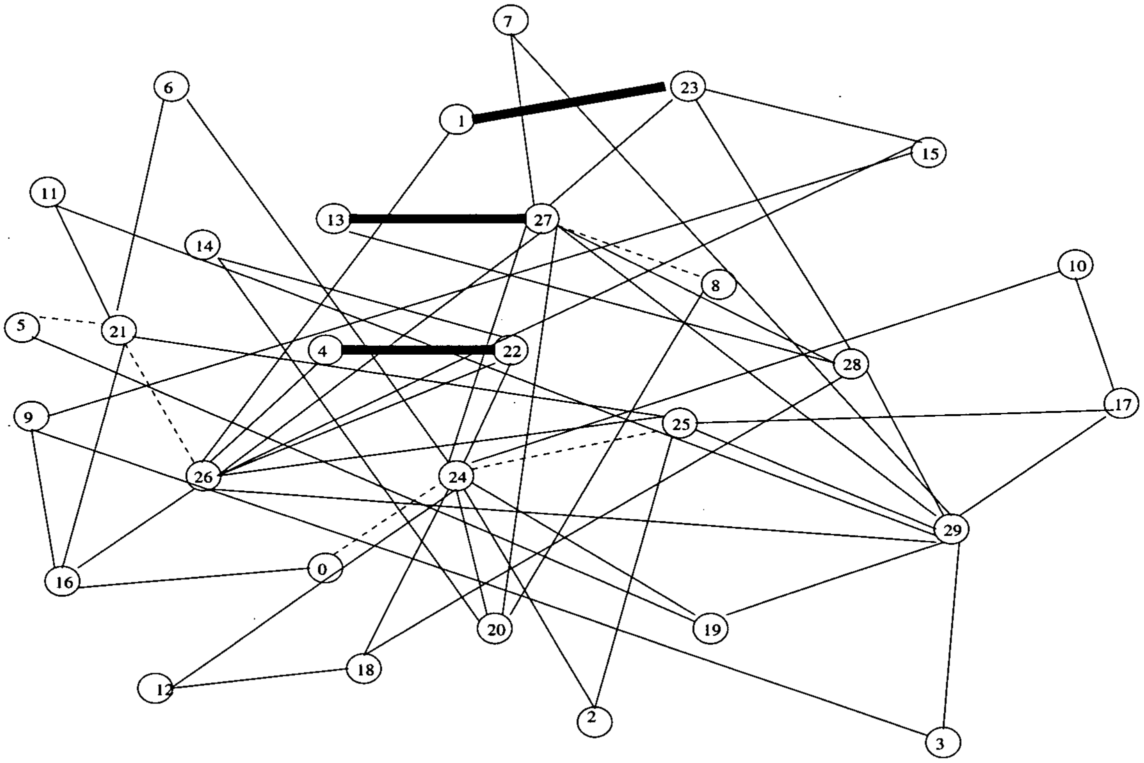


Figure A.5: Topology 5

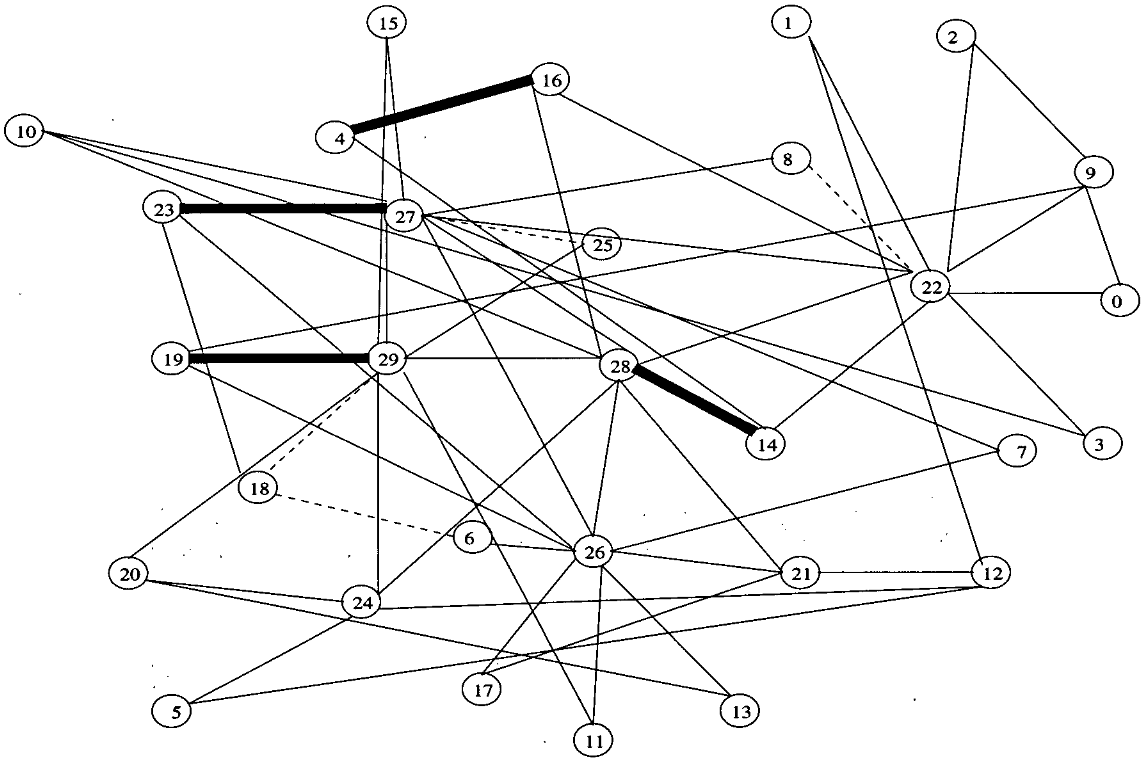


Figure A.6: Topology 6

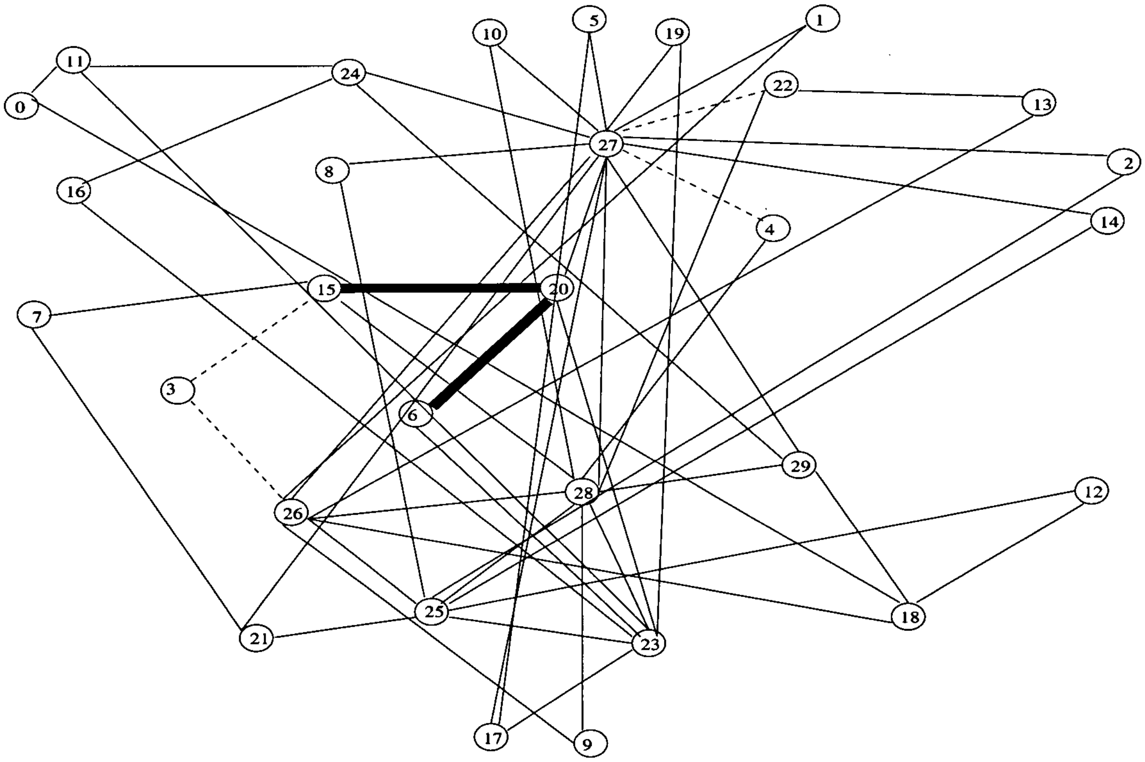


Figure A.7: Topology 7

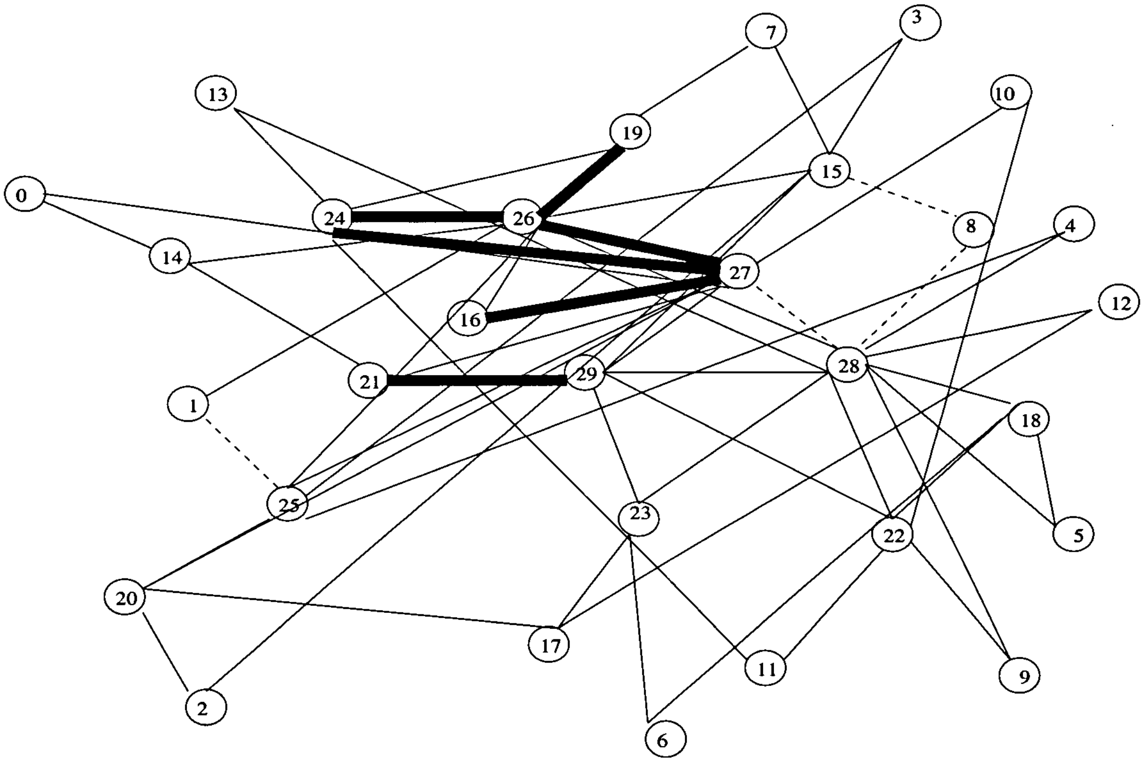


Figure A.8: Topology 8

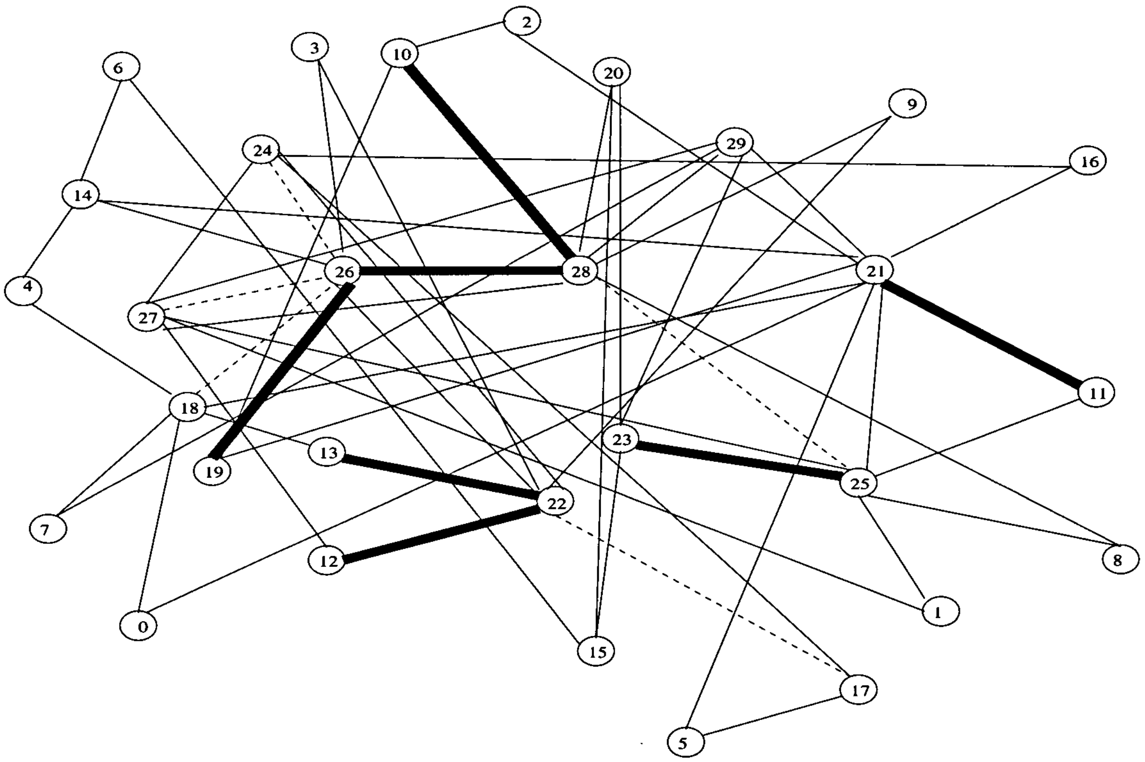


Figure A.9: Topology 9

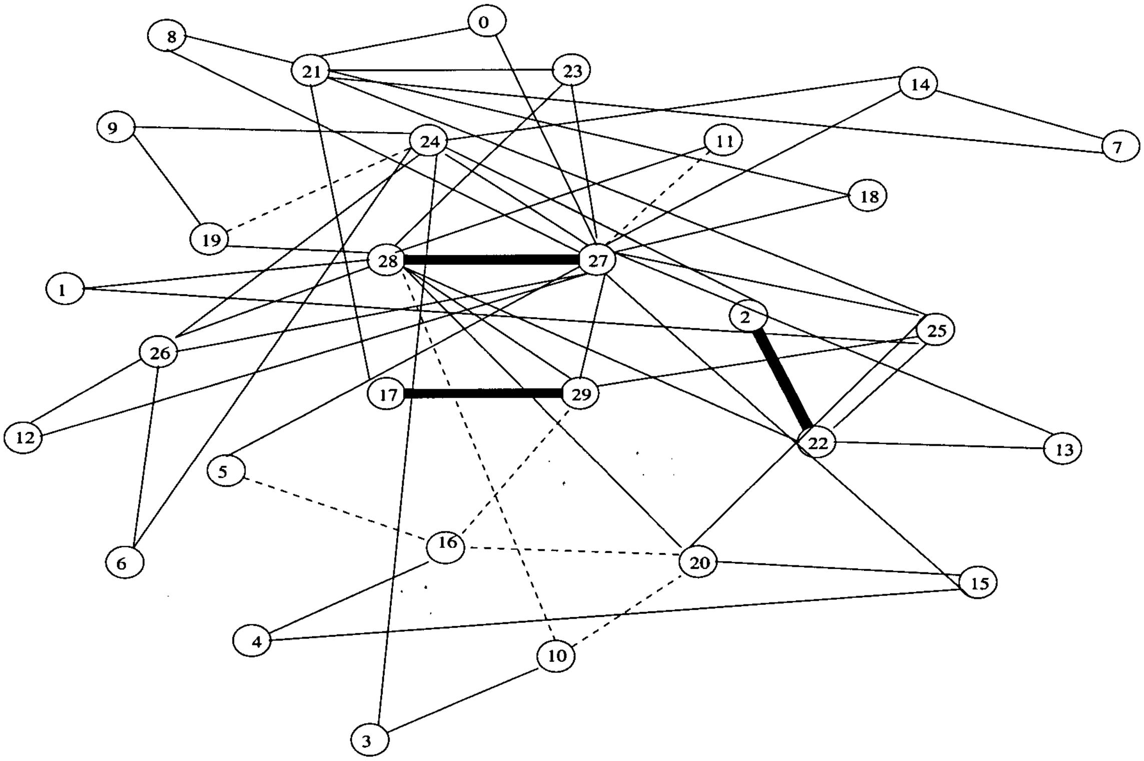


Figure A.10: Topology 10

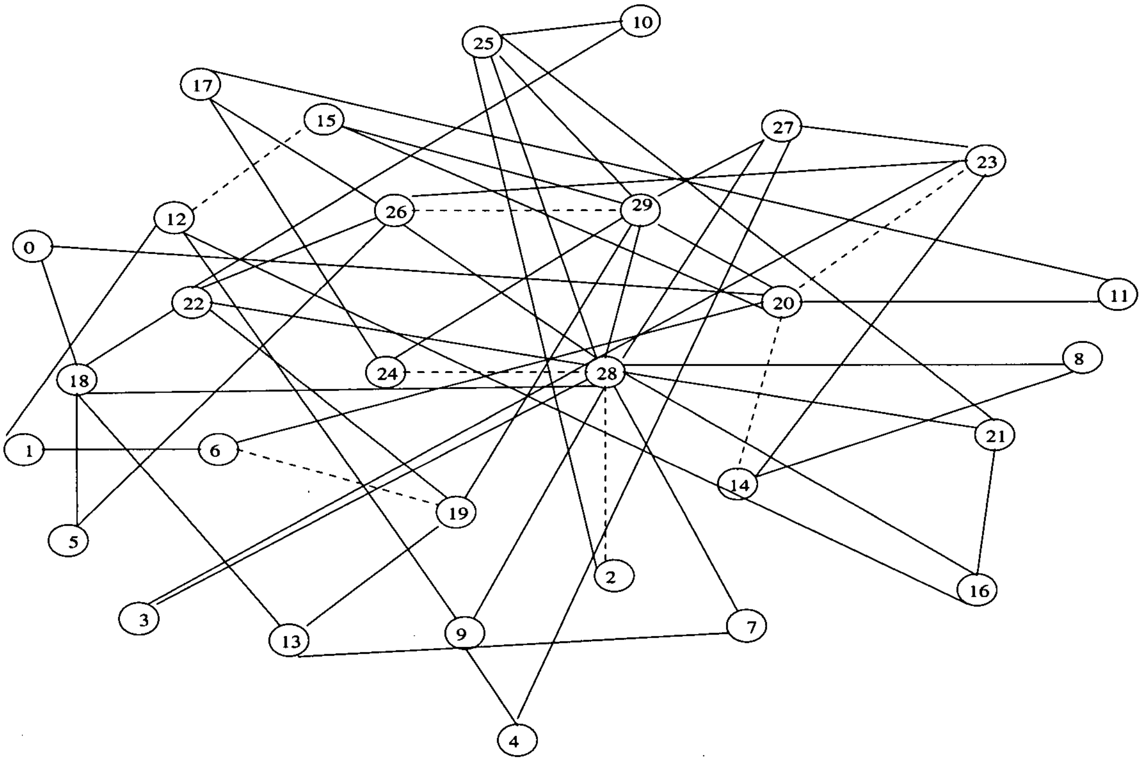


Figure A.11: Topology 11

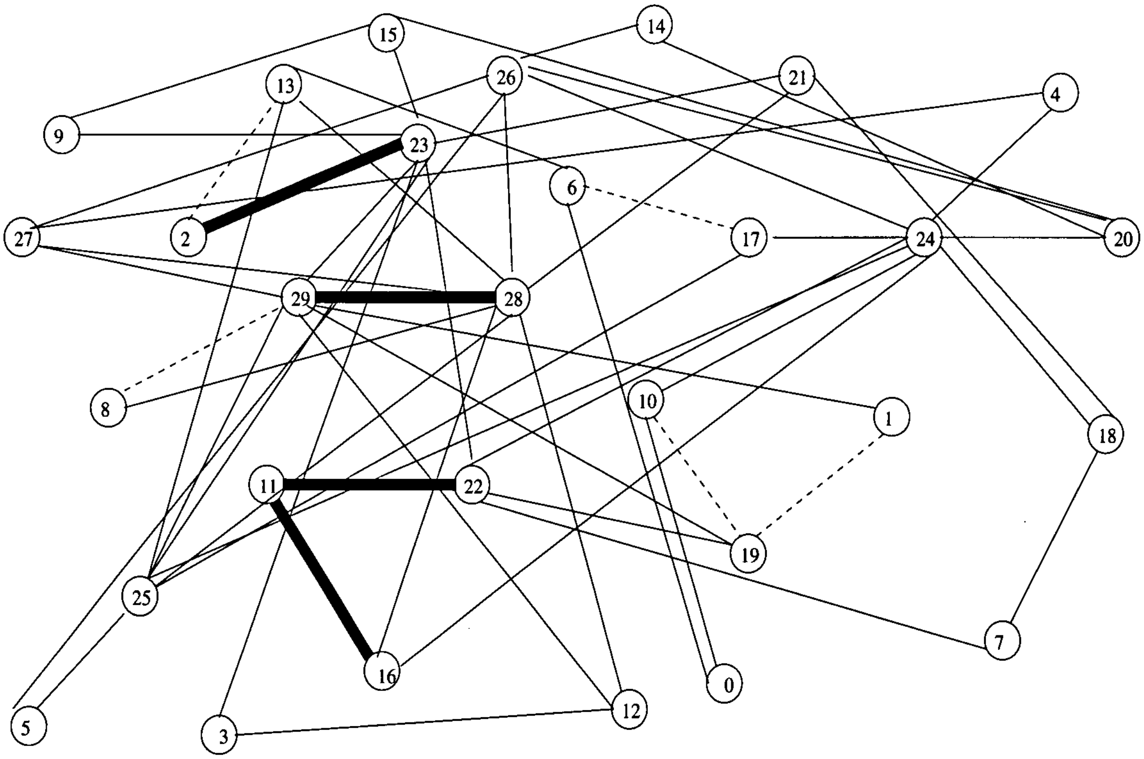


Figure A.12: Topology 12

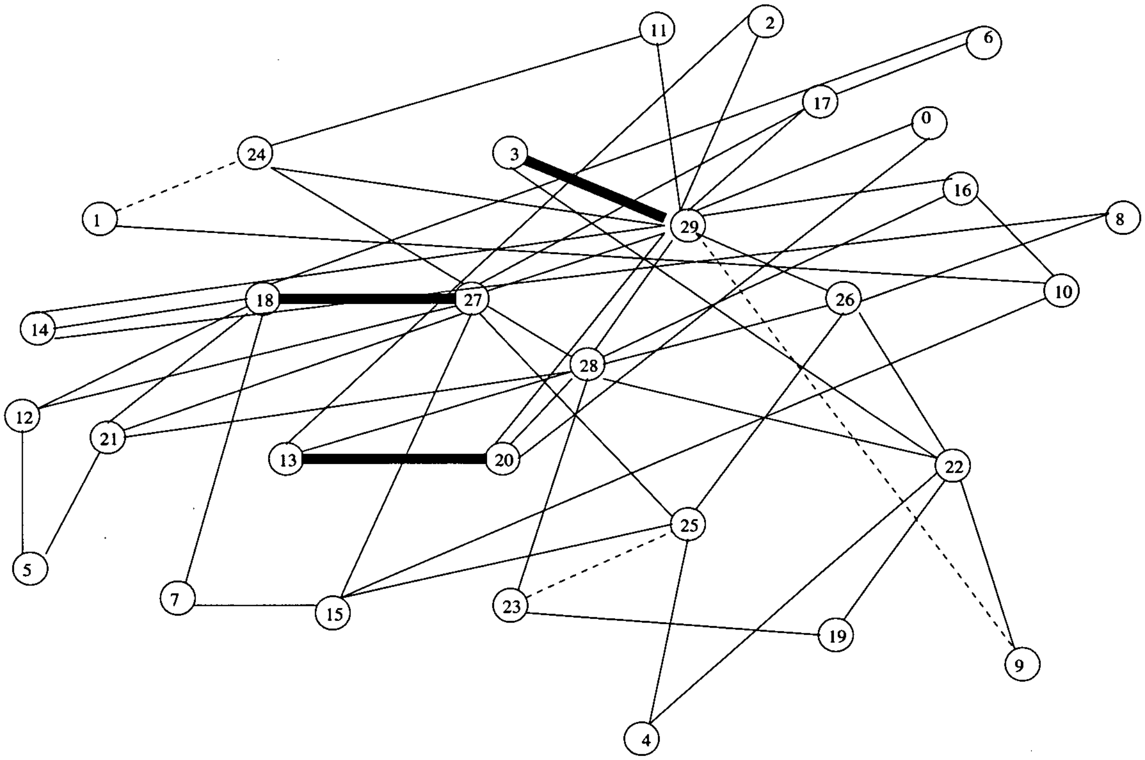


Figure A.13: Topology 13

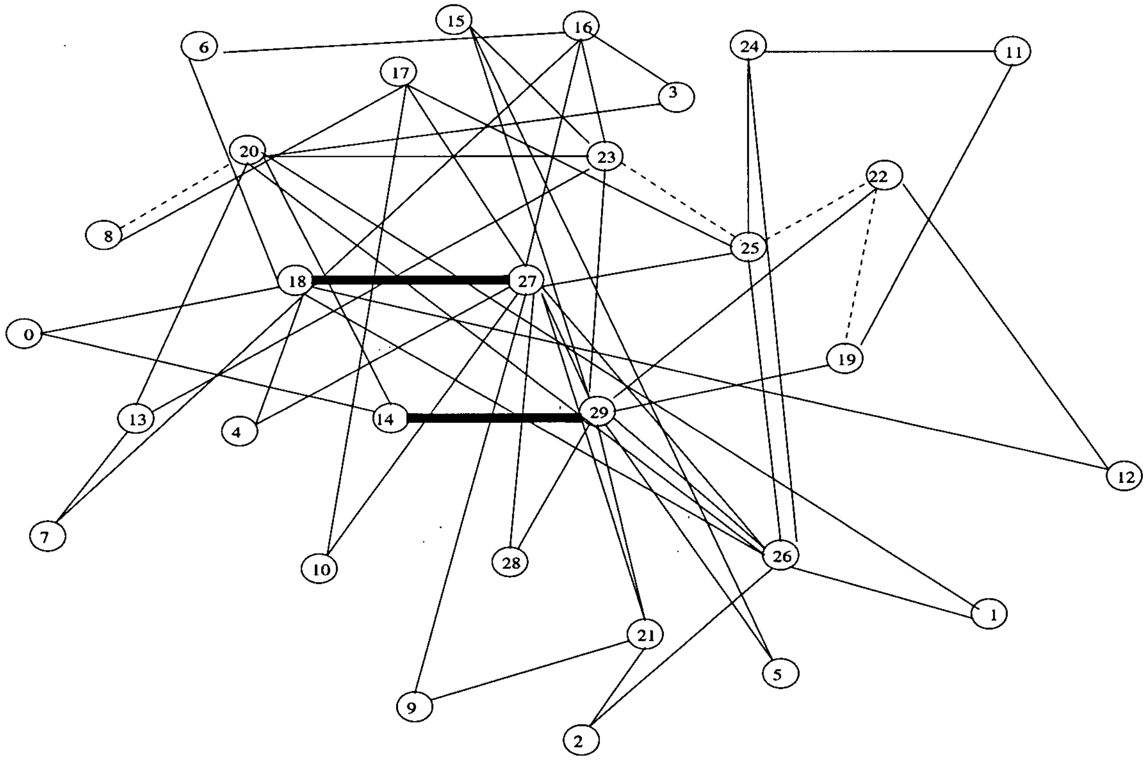


Figure A.14: Topology 14

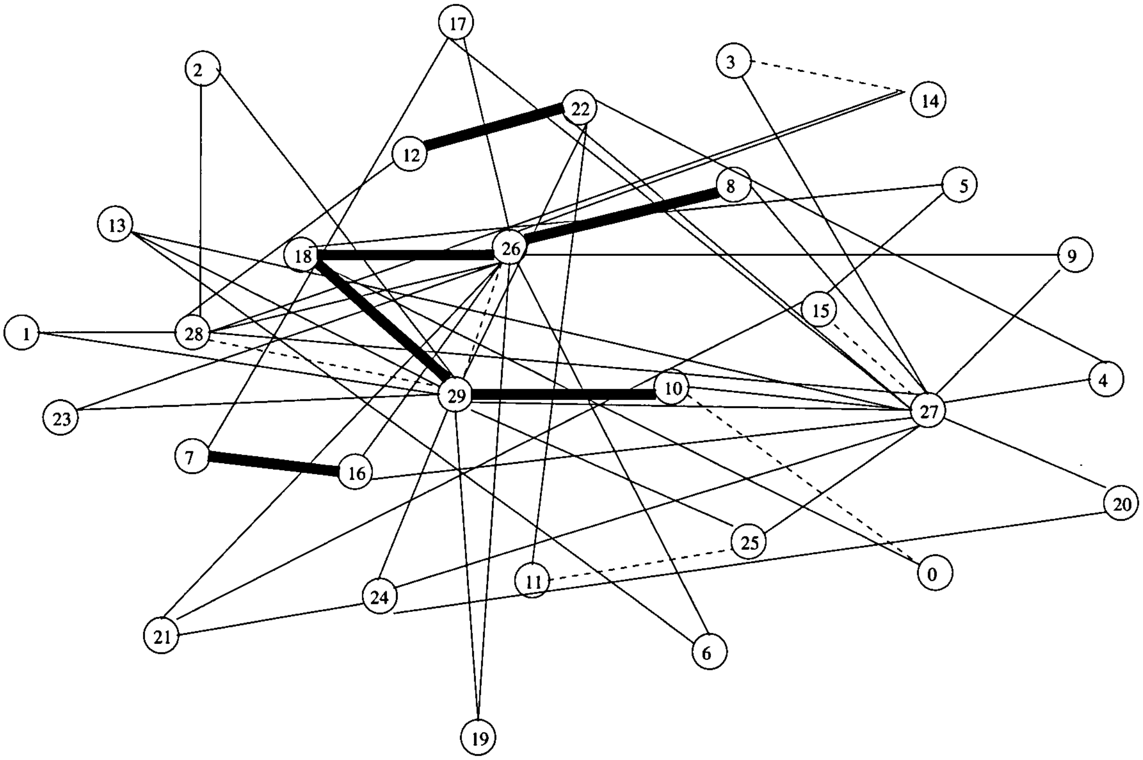


Figure A.15: Topology 15