

Efficient Management of Integrated Services

Using A Path Information Base*

Geoffrey G. Xie Debra Hensgen[†] Taylor Kidd[†] John Yarger

Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943-5193
{xie,hensgen,kidd,yarger}@cs.nps.navy.mil

NPS-CS-98-013 May 14, 1998

Abstract

The current network architecture is based predominantly on stand-alone routers. It is becoming overtaxed with the introduction of integrated services. This observation has led us to propose a **S**erver and **A**gent based **A**ctive network **M**anagement (SAAM) architecture that scales well with integrated services [14]. SAAM relieves individual routers from most routing and network management tasks. Instead, it employs a small number of dedicated servers to perform these tasks on behalf of the routers. In particular, these servers maintain a path information base (PIB) with which network functions, such as QoS routing and re-routing of real-time flows, can be efficiently implemented. In this paper, we focus on the design of an efficient and flexible PIB. We will not only present our design rationale and the specification of a concrete PIB building algorithm, but will also describe some important observations concerning PIBs that we have obtained by examining the statistics of sample PIBs built from three example network configurations. These observations include: (1) a single regional PIB supporting a network in the range of 20-30 routers can be maintained on a PC workstation, and (2) a PIB can provide efficient support for any QoS routing algorithm.

Keywords: Integrated services, QoS routing, network management, server-based network architecture, path information base, criticality index

*Work supported by DARPA Next Generation Internet Initiative under contract number G417.

[†]Supported also by DARPA under contract number E583.

1 Introduction

Existing data networks such as the Internet are built using sophisticated stand-alone routers. In addition to forwarding packets, each router is currently required to perform elaborate routing and management functions. As the networks grow to handle more and increasingly diverse data, more processing will be required of each router. While such a “heavy-weight router” approach scales adequately and is fault tolerant in providing best effort service, it may not be an efficient solution for implementing integrated services for the reasons that follow.

First, an integrated services network must guarantee Quality of Service (QoS) to *individual* user sessions. To meet this requirement, *QoS based routing* is required. Specifically, the network must be able to direct the sequence of packets from a given session to use a particular path of routers, where resources (link bandwidth, buffers, etc.) have been reserved for that session to achieve a particular QoS. In other words, the network should treat the sequence of packets as a single *flow*. Compared to current shortest path routing algorithms, QoS routing algorithms must deal with more constraints, and thus require much more processing on the part of each router [13, 5]. Moreover, it has been shown that it is desirable to use different QoS routing algorithms under different conditions to improve network performance [9]. Having such flexibility also requires more computation at each router. Therefore, processing overhead will become a major concern if every router is required to perform QoS routing.

Second, an integrated services network must support real-time applications that have very stringent packet delay bound requirements. Consequently, when a path for a real-time flow becomes unusable because of a network fault, a replacement path should be established within a short time frame. In other words, the flow needs to be quickly *re-routed*, preferably without involving the end user/application. Otherwise, the performance of the corresponding real-time application will suffer noticeably. Several schemes have been proposed to address this problem in the context of a network with heavy-weight routers. Specifically, they make use of dispersity routing [1] or backup channels [7]. However, these schemes also reduce network utilization and increase the processing requirements of the routers.

In summary, a heavy-weight router can easily become a performance bottleneck due to a lack of processing power. The problem is compounded by the fact that integrated services will likely require packet forwarding methods that are much more elaborate than First-In-First-Out (FIFO).

This observation has led us to propose a **S**erver and **A**gent based **A**ctive network **M**anagement (SAAM) architecture for the efficient support of integrated services [14]. Specifically, in SAAM, individual routers are relieved of most routing and network management tasks. Instead, a *small* number of dedicated servers perform these tasks on behalf of the routers. In particular, the servers maintain a *path information base* (PIB). Network functions such as QoS routing and re-routing of real-time flows, whose effectiveness are key to success of integrated services, can be efficiently implemented using the PIB.

The use of route servers has been proposed for data networks [12, 18]. The motivation was to reduce the computational overhead for a set of closely associated routers¹. QoS routing and re-routing were not considered. Moreover, our development of SAAM has two additional motivations. First, we envision SAAM to be the common platform where different network functions, such as routing, resource reservation, network management, accounting, and security, can be integrated. Second, by concentrating network management and control to a small number of servers, SAAM can potentially be used for faster deployment of new services than is currently possible.

In this paper, we focus on the design of an efficient and flexible PIB. We will not only present our design rationale and the specification of a concrete PIB building algorithm, but will also describe some important observations on PIBs that we have obtained by examining the statistics of sample PIBs built from three example network configurations. These observations include: (1) a single regional PIB supporting a network in the range of 20-30 routers can be maintained on a PC workstation, and (2) a PIB can provide efficient support for any QoS routing algorithm.

The balance of the paper is organized as follows. In Section 2, we give an overview of the SAAM architecture and discuss some important design issues related to such a server-based approach. This is done to familiarize the reader with the strategic picture of our work. In Section 3, we describe in detail a particular design of the PIB and present statistics collected from sample PIBs. Moreover, we explain how SAAM can use the PIB to perform efficient QoS routing and re-routing of flows without involving the end user.

2 Overview of SAAM Architecture

Before describing the SAAM architecture, we present a list of issues that have a direct impact on the feasibility of a server based network architecture. Many of our design choices are based on an understanding of these issues.

2.1 Design Issues

- **Responsiveness.** To support integrated services, the network must be able to detect and react to changing network conditions, especially a QoS degradation along a path, within a short time frame. Therefore, SAAM should use a pro-active approach in data collection. Moreover, SAAM should aggregate the data that it has collected about individual links into “ready to use” information on path performance.

¹For example, a set of Internet Service Provider (ISP) routers that share a Network Access Point (NAP) [18].

- **Scalability.** SAAM must be able to scale to provide a complete solution for global networks that consist of hundreds of routers. On one hand, it is desirable to have a small number of servers. On the other hand, there is an upper limit on the number of routers that a server can support. The scalability issue is also very important when determining how frequently a server should update its PIB. More frequent updates will result in more accurate information. However, they also cause more (computation and communication) overhead on the network and servers.
- **Fault-tolerance.** If not carefully designed, the failure of one SAAM server could have a devastating effect on the performance of the entire network. Therefore, servers must be deployed in such a way that the failure of one server can only affect the performance of a small set of routers for a short period of time. In addition, it should be possible to deploy redundant servers.

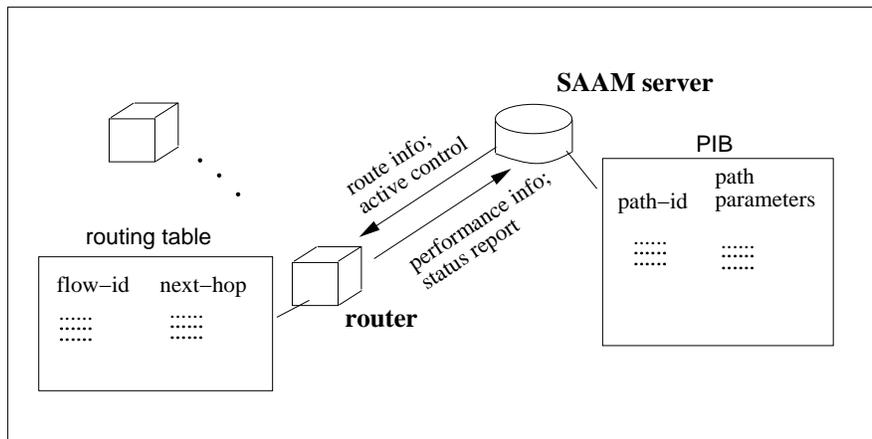


Figure 1: Logical model of SAAM

2.2 Logical Model of SAAM

SAAM consists of light-weight routers and a small set of heavy-weight servers. Logically, each router is a client of a single SAAM server process. (See Figure 1.) Next, we describe how a particular router and the SAAM server process interact in this model. For brevity, we will focus on those aspects related to QoS routing.

SAAM requires (preferably dedicated and real-time) duplex communication channels between each router and its server. We assume that these channels are established when the router joins the network. The router does not participate in QoS routing; it updates its flow-based routing table with route data passed down from the server. Note that the router can still participate in conventional routing if backward

compatibility is required. In such a case, the router must pre-allocate a set of flow-ids for data that will not be routed by SAAM.

The SAAM server builds a PIB to support QoS routing. Specifically, the server identifies those paths or subpaths that can potentially be used to route flows, and maintains up-to-date performance parameters for each of path/subpath. The server computes path performance parameters by aggregating link level performance data passed up from each router.²

We will present more details on how to build the PIB in Section 3.

2.3 Hierarchical Organization of Servers

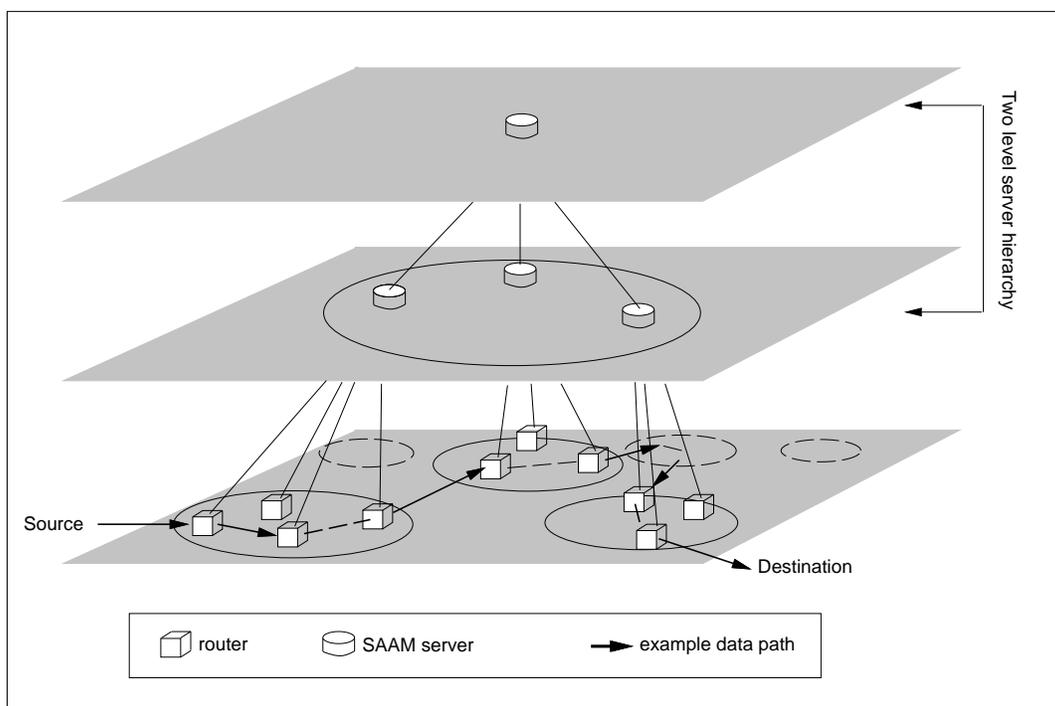


Figure 2: Hierarchical organization of SAAM servers

To address the scalability issue, SAAM organizes its servers in a hierarchy. (See Figure 2.) Specifically, at the first level SAAM partitions the network into regions, and sets up one server³ for each region. (A region is represented by a circle in Figure 2.) The current approach to network partitioning using Autonomous Systems [10] can easily be extended to perform this task. Once established, the SAAM server will perform network functions on behalf of the routers in its region.

²Details on how to collect such data is beyond the scope of this paper.

³SAAM also sets up one or more backup servers if high fault tolerance is required.

Similar to today's architecture, each SAAM region has a subset of routers, called border gateways, through which data can come in and go out of a region. SAAM uses a parent server at the top level to perform the network functions that enable communication between these routers.

The main advantage of the above architecture is that it allows SAAM to build a scalable PIB. The details are described in Section 3.2. The hierarchical architecture also permits SAAM to be gradually deployed into today's networks. Specifically, SAAM can be implemented initially in one part of a network. The top-level SAAM server will function as a speaker for all routers in the SAAM part of the network, i.e., it will become the sole participant in the information exchange with routers in the other (non-SAAM) part of the network.

While we examined the simplest two-level server hierarchy, it should be noted that this architecture can support a greater number of levels, as the situation demands.

3 Design of Path Information Base

As discussed earlier, an essential component of our SAAM architecture is the PIB. When designing a PIB, one must consider the following two issues:

1. *Performance.* The PIB will be used by a wide range of network functions that include routing, resource reservation and network management. To ensure good performance of these functions, the PIB should (i) maintain sufficient information, and (ii) supply that information in a timely manner.
2. *Cost.* The overhead of building and maintaining the PIB should be carefully analyzed and controlled. In particular, the PIB must scale well as the network size grows.

In this section, we describe a PIB design that takes advantage of the SAAM architecture to achieve high performance and control cost. To illustrate the benefits of the design, we also explain how SAAM can make use of the created PIB to perform efficient QoS routing and re-routing. For ease of discussion and without loss of generality, we assume a two-level SAAM server hierarchy like the one shown in Figure 2.

3.1 Preliminary

First, we describe the system model for our PIB design. Specifically, we define a path in the context of an integrated services network, and identify a set of important path parameters that will be managed by the PIB.

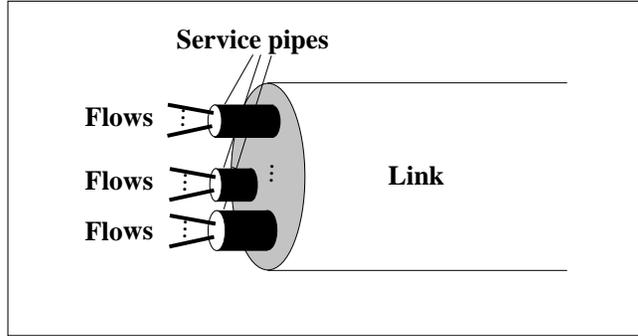


Figure 3: Link shared by service pipes

3.1.1 Path definition

In an integrated service network, each network link is shared by a set of *logical* service pipes [4, 2, 17], each of which provides a particular level of network performance measured by packet delay and packet loss rate. (See Figure 3.) An ATM virtual path that is dedicated to Constant Bit Rate (CBR) traffic is an example of a service pipe.

Specifically, we define the following parameters for a service pipe (denoted by s):⁴

- D target upper bound on the total packet delay (in seconds); includes queuing delay, transmission delay and propagation delay; s offers only best effort service when D is unspecified, i.e., have an invalid value such as -1
- E upper bound on the percentage of packets that incur a delay greater than $s.D$; s offers only best effort service when E is unspecified, and a guaranteed service when $E = 0$
- B amount of pre-allocated link bandwidth; in bits/second
- R bandwidth available for new flows; initially set to B

We define a path in an integrated services network as follows.

Definition 1 A path is an ordered sequence of service pipes. Specifically, an arbitrary path (denoted by π) is represented by

$$\pi = \langle s_1, s_2, \dots, s_K \rangle \quad (1)$$

where s_k is the k th service pipe in the path, $k = 1, 2, \dots, K$.

When appropriate, we consider the path as just a set, rather than an ordered sequence, of service pipes. In such a case, $\pi = \{s_1, s_2, \dots, s_K\}$.

⁴In this paper, we follow the convention of using the “.” operator to associate a parameter with an object.

3.1.2 Path parameters

Next, we list the set of path parameters that will be maintained in the PIB. Most of these parameters are generalizations of what have been defined for a service pipe.

$\pi.D$ the target upper bound on the total packet delay, which is expressed by

$$\pi.D = \sum_{s \in \pi} s.D. \quad (2)$$

Note that when a rate-based packet service discipline (e.g., Weighted Fair Queueing) is used at each service pipe, the target end-to-end delay upper bound of a path can be much smaller than the sum of the target per-hop delay bounds. In such a case, we re-define $s.D$ to be a target delay upper bound based on the expected packet arrival time; hence equation 2 will continue to hold [15, 8]. Such a re-definition will not complicate flow resource reservation for the following reason: For any flow f that uses path π , the delays of its packets are tightly bounded by

$$\sum_{s \in \pi} s.D + \max(0, \max_{p \in f} (EAT(p) - A(p))). \quad (3)$$

where p is any packet in the flow, $EAT(p)$ is its *expected arrival time* to the first router of π , and $A(p)$ is the *actual arrival time*. The upper bound on $(EAT(p) - A(p))$, which depends on the type of traffic policer employed for the flow, can be determined *a priori* and subtracted from the flow's delay bound requirement at flow setup time. For example, for each packet p in a flow constrained by a leaky bucket policer with parameters of (ρ, σ) , we have [15]:

$$EAT(p) - A(p) \leq \frac{\sigma}{\rho}. \quad (4)$$

$\pi.E$ the upper bound on the percentage of packets that incur a delay greater than $\pi.D$, which is expressed by

$$\pi.E \approx \sum_{s \in \pi} s.E. \quad (5)$$

The derivation of the above equation is as follows. Assuming that packet losses of a flow at different service pipes are independent events⁵, we have

$$(1 - \pi.E) = \prod_{s \in \pi} (1 - s.E). \quad (6)$$

⁵This is not an overly conservative assumption, considering the fact that the flow must be sharing service with many other flows when a packet loss occurs. We have obtained experimental results that support this claim [16].

Therefore

$$\pi.E = 1 - \prod_{s \in \pi} (1 - s.E) \quad (7)$$

$$\approx 1 - (1 - \sum_{s \in \pi} s.E) \quad (8)$$

$$= \sum_{s \in \pi} s.E. \quad (9)$$

$\pi.B$ the total effective bandwidth, which is defined by

$$\pi.B = \min_{s \in \pi} \{s.B\}. \quad (10)$$

$\pi.R$ the currently available effective bandwidth, which is defined by

$$\pi.R = \min_{s \in \pi} \{s.R\}. \quad (11)$$

$\pi.F$ the set of flows that are currently using π .

3.1.3 Link sharing

We assume that a suitable link sharing algorithm [2, 4, 17] is implemented at every link so that a firewall is established between the link's service pipes. Specifically, the performance guarantees of one service are independent of those of other services. For brevity and without loss of generality, we will focus exclusively on how to build a PIB for flows requesting a statistical service. Consequently, we assume that each link in the network is a statistical pipe, and we will represent a path by $\langle a_1, a_2, \dots, a_K \rangle$ where a_k is the k th router in the path. We use f to denote a statistical flow. There are two QoS parameters associated with f : the delay bound requirement of $f.D$ and the loss bound requirement of $f.E$. The objective of QoS routing and re-routing is to allocate, and re-allocate if necessary, a statistical path π — π containing exclusively statistical service pipes — to connect the source and the destination of the flow while satisfying:

$$\pi_D \leq f.D, \quad (12)$$

$$\pi_E \leq f.E \quad (13)$$

3.2 Building Path Information Base

We follow a divide-and-conquer strategy to control the cost of building and managing the PIB. The strategy is based on the following observation: With the hierarchical architecture of SAAM, it suffices for the SAAM server of each region to build and manage a relatively small regional PIB that contains information

for only local paths in the region. Specifically, information for a long-distance path (i.e., one that crosses multiple regions) is built and managed *jointly* by three SAAM servers: a first-level server responsible for the source segment, i.e., from the source to an outgoing border gateway; another first-level server for the destination segment, i.e., from an incoming border gateway to the destination; and the parent server for the middle segment between the border gateways. In the remainder of this section, we will focus on how to build a regional PIB.

We also identify and exclude undesirable paths from each regional PIB to reduce the size of the PIB. Specifically, those paths that contain a loop or have a regional hop count greater than a predetermined value H_{max} are deemed *invalid*.

Definition 2 *A path is valid if and only if: (1) it has a hop count less than or equal to H_{max} , and (2) all the routers in the path are distinct.*

Next, we describe, in detail, the content of a PIB and the steps that a SAAM server takes to build its regional PIB. Consider a particular SAAM region and its SAAM server. Assume that there are M routers in the region.

The regional PIB contains two types of path information arrays. The first type of arrays — called the Path Information Arrays (PIAs) — are defined as follows:

$$PIA(i, j) = \{\text{reference to } \pi \mid \pi \text{ is a valid path and it goes from } i \text{ to } j\}, \quad (14)$$

$$1 \leq i, j \leq M.$$

In other words, the set of all valid paths is partitioned into subsets, each of which is referenced by a single PIA. With PIAs, the SAAM server can quickly find a suitable path when given the source and destination router identifications. The meaning of “suitable” is dependent on the particular routing algorithm used by the server. There will be more discussion on this point in Section 3.3.

The other type of information arrays — called the Update Information Arrays (UIAs) — also contain references to valid paths. There is one such array for each service pipe in the region. UIAs are defined as follows:

$$UIA(k, l) = \{\text{reference to } \pi \mid \pi \text{ is a valid path and it contains service pipe } \langle k, l \rangle\}, \quad (15)$$

$$1 \leq k, l \leq M.$$

In other words, each UIA contains references to all valid paths that use a particular service pipe. Our motivation for having UIAs is to reduce the processing overhead of the server in identifying paths that are affected, and thus require information update, when there is a significant change in the performance of a service pipe.

It should be noted that while there could be many references to a path, there is only one physical information record associated with it. The record holds current information (the values of D , E , R , the set of active flows, etc.) about the path.

The server takes two major steps to build the PIB. First at network boot-up time, the server assigns a unique index $i \in \{1, 2, \dots, M\}$ to each router; and for each router i , it computes and stores the following set:

$$Parents(i) = \{j \mid \text{there is a service pipe from router } j \text{ to router } i\} \quad (16)$$

```

Build_SAAM_PIB()
1   $V \leftarrow \{1, 2, \dots, M\}$  ;
2  for ( each router  $i \in V$  ) do
3       $a[0] \leftarrow i$  ;
4      Process_Path(1) ;

Process_Path(h)           // process all valid paths that go to  $a[0]$  in  $h$  to  $H_{max}$  hops
1   $W \leftarrow Parents(a[h - 1])$ ;
2  for ( each router  $j \in W$  ) do
3      if ( Cause_No_Loop( $h, j$ ) )
4          then  $a[h] \leftarrow j$  ;
5              Insert_Element(<  $a[h], a[h - 1], \dots, a[0]$  >, PIA( $a[h], a[0]$ ));
6              for (  $q = h; q > 0; q = q - 1$  ) do
7                  Insert_Element(<  $a[h], a[h - 1], \dots, a[0]$  >, UIA( $a[q], a[q - 1]$ ));
8              if (  $h < H_{max}$  )
9                  Process_Path( $h + 1$ ) ;

 $a[]$  is a global utility array,
Insert_Element( $x, y$ ) is a function that inserts an element  $x$  into the set  $y$ , and
Cause_No_Loop( $h, j$ ) =  $\begin{cases} \text{false} & \text{if } \exists q \text{ such that } 1 \leq q \leq h - 2 \text{ and } a[q] = j \\ \text{true} & \text{otherwise} \end{cases}$ 

```

Figure 4: Algorithm for building PIA and UIA

Afterwards, the server uses the algorithm specified in Figure 4 to build its regional PIB (i.e., PIAs and UIAs). It should be noted that the creation of a physical information record for each path is omitted from

the algorithm specification. The creation should happen right before step (5) of *Process_Path(h)* when a path is processed for the first time. It involves computations as described in Section 3.1.2. We assume that the parameters for each service pipe are available when the algorithm is run. (They are either collected or sent from the associated router at network bootstrap time.)

The algorithm has an average complexity of $O(M \cdot g^{H_{max}})$, where g is the average size of the *Parents* set for a router. Typically, g does not exceed 3.

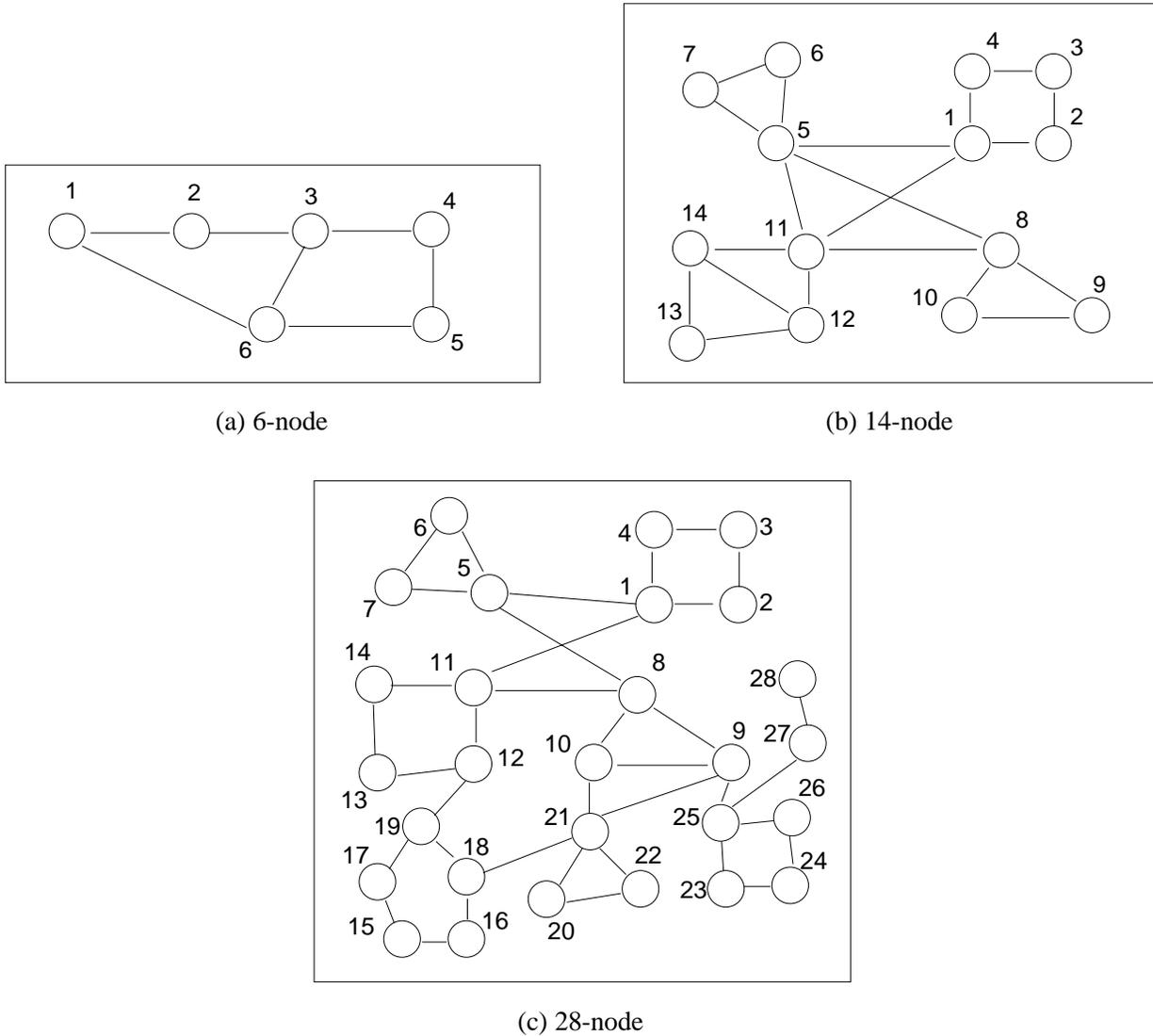


Figure 5: Network configurations used in experiments

3.2.1 Experimental results

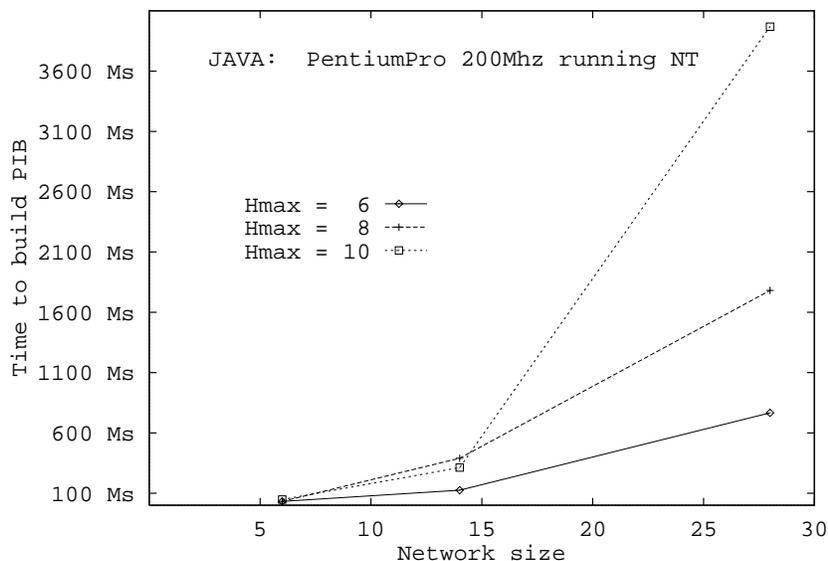


Figure 6: Algorithm performance

We have performed a set of simulation experiments to evaluate the performance of the PIB creation algorithm. The results are reported in this section.

We used three different network configurations in the experiments. They are depicted in Figure 5, representing respectively a small (6-node), a medium (14-node) and a large (28-node) region. Each link in the networks is considered to have two pipes, one for each direction. We varied the value of H_{max} between 6, 8, and 10 for each configuration.

The PIB creation algorithm is coded in Java and was run on a Dell PentiumPro 200 system using the Java Virtual Machine (version 1.1.6). The execution times of the algorithm for different network sizes are plotted in Figure 6. The times are moderate until H_{max} gets large and the network size goes beyond 30 nodes. It should be noted that a full PIB creation or re-creation needs to be performed only if there is a permanent network topology change or a network reboot. Therefore, the speed requirement is not stringent.

We have also examined the size statistics of the created sample PIBs. The statistics for the PIA sizes are plotted in Figure 7. From the graph, we observe that the number of valid paths between a pair of routers grows moderately as either network size or H_{max} increase. In other words, because it significantly reduces the search space, the PIB facilitates the rapid execution of QoS routing algorithms on the SAAM server.

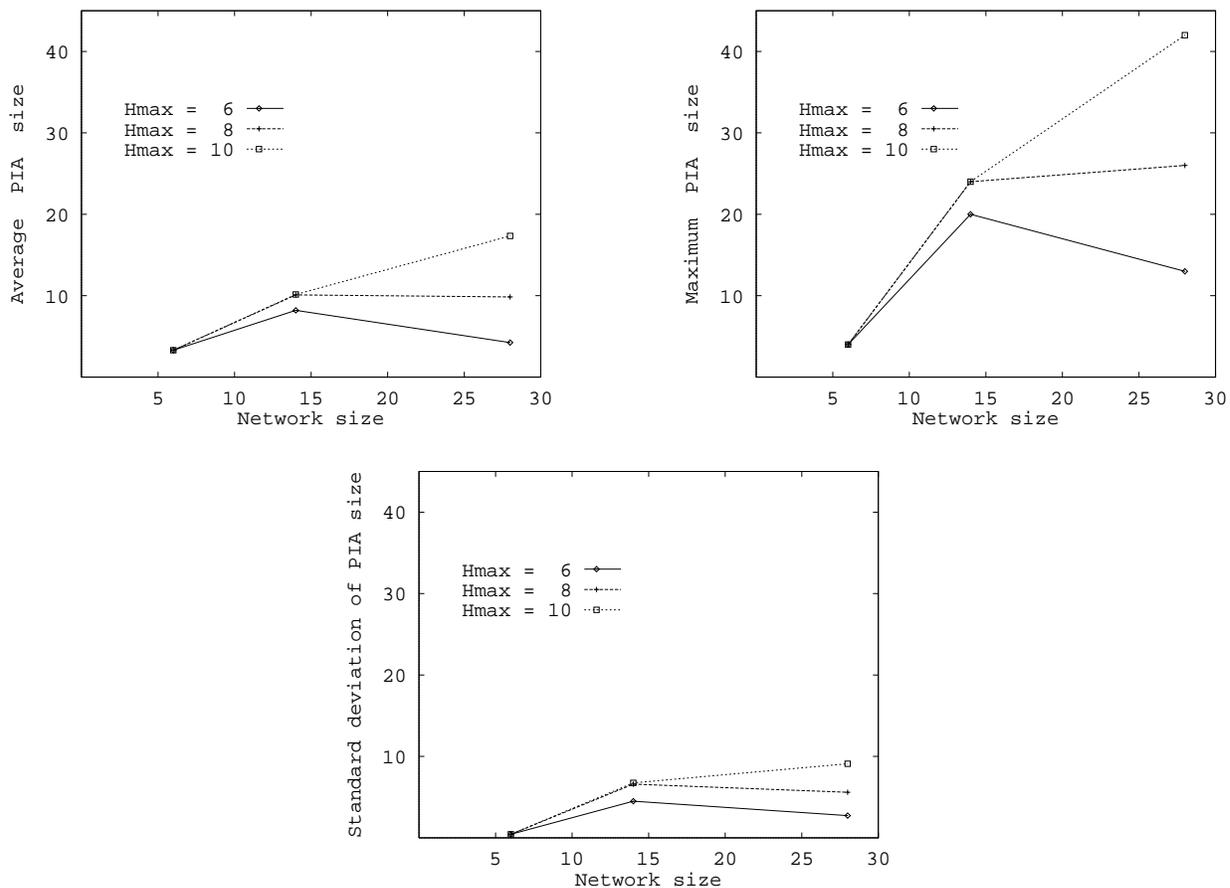


Figure 7: PIA statistics based on sample networks

The statistics for the UIA sizes are plotted in Figure 8. From the graph, we observe that a change in the performance of a service pipe affects the performance of a large number of valid paths even for a moderate network size and a small H_{max} value. Therefore, the UIA cannot be used for the online re-routing of flows in the event of a service pipe malfunction. Instead, the UIA should be used to do background (off-line) parameter updates for the paths that would be affected. Fortunately, any online re-routing would be routing protocol dependent, and as such, the number of *active paths*⁶ actually affected is significantly lower than indicated by the UIA. This is because between any pair of routers, say i and j , the paths that are used by a routing protocol usually constitute a very small subset of the $PIA(i, j)$. We will discuss online re-routing further in Section 3.3.2.

⁶A path is active if currently carries at least one flow.

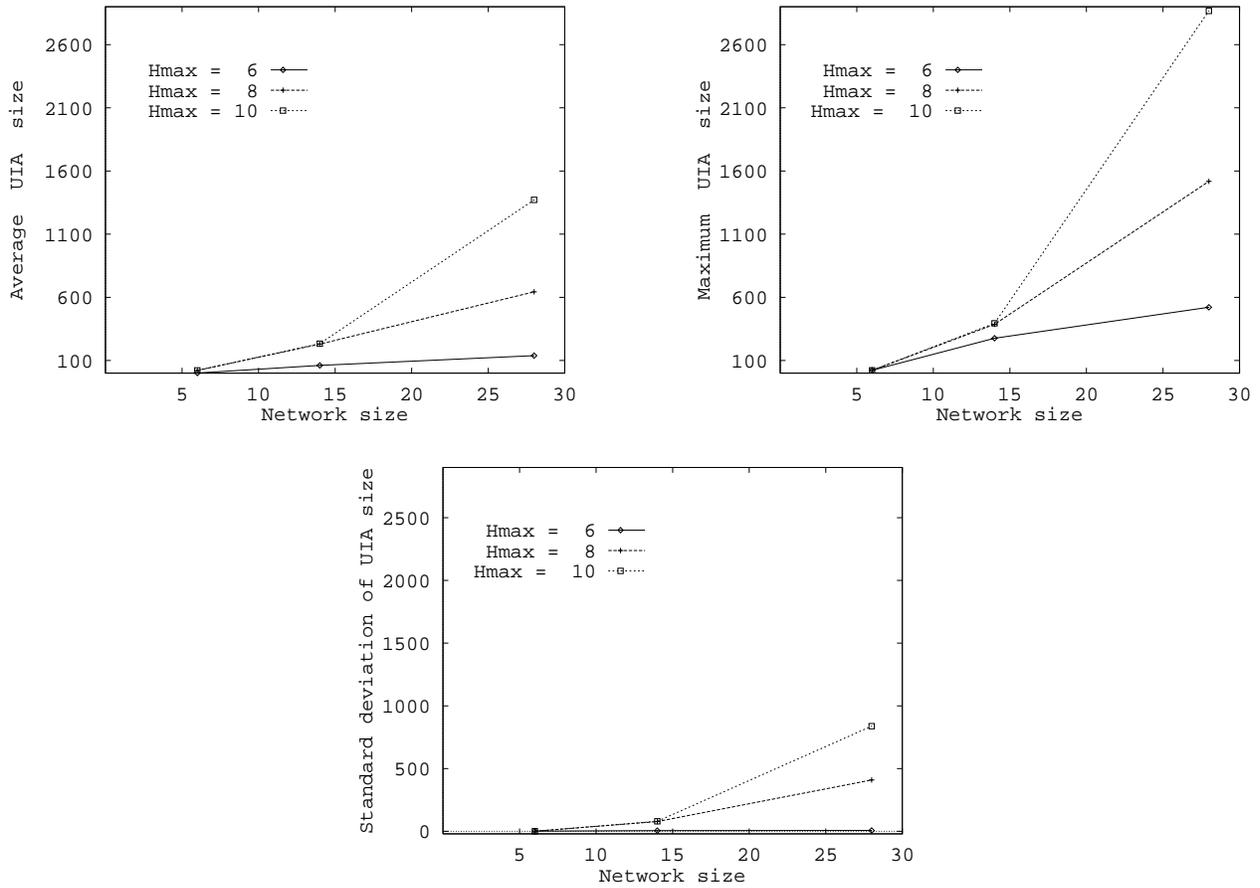


Figure 8: UIA statistics based on sample networks

3.3 Routing and Re-Routing of Flows

The PIB created by *Build_SAAM_PIB()* is much more comprehensive than those that can be built using shortest path algorithms. (It would be too costly for every router to maintain such a PIB.) Consequently, SAAM is able to use a more flexible QoS routing strategy. Specifically, SAAM supports the integration of multiple QoS routing schemes, each of which has its own very efficient PIB built on top of the SAAM PIB. A SAAM server will choose different schemes for different times depending on the current state of its region. Such flexibility is quite desirable, as observed in [9]. The details of the integration are dependent on the specifics of QoS routing schemes, and are beyond the scope of this paper. Next, we will explain how the PIB enables SAAM to perform (1) fast routing of a long distance flow, and (2) flow re-routing in the event of a link failure or service malfunction.

3.3.1 Fast routing of long distance flows

Denote f to be a long distance statistical flow. SAAM uses the following steps to find a path for the flow. First, upon receiving the request to set up f , the SAAM server for the source region forwards the request to the parent server; selects⁷ from its PIB a path (denoted by π^{src}) that has the smallest D among those going from the source to an outgoing border gateway; and then sends the information about π^{src} to the parent server. The parent server, after receiving the forwarded request, determines in which region the destination resides, forwards the request to the server of that region, and then waits for a response from the source and destination. The destination region SAAM server, upon receiving the forwarded request, selects from its PIB a minimum-D path (denoted by π^{des}) extending from an incoming border gateway to the destination, and then sends the information about π^{des} to the parent server. Finally, after receiving the information about π^{src} and π^{des} , the parent server updates $f.D$ and $f.E$ by subtracting from them, respectively, $(\pi^{src}.D + \pi^{des}.D)$ and $(\pi^{src}.E + \pi^{des}.E)$. It then uses an appropriate QoS routing scheme to search for a suitable path between the gateways.

3.3.2 Re-routing of flows

The network needs to re-route flows when a link fails or a service pipe malfunctions.⁸ Re-routing on a flow by flow basis would be inefficient and not suitable for real-time traffic because the number of flows that require re-routing can be quite large. With UIAs, a SAAM server can re-route on a path by path basis. Specifically, suppose $\langle k, l \rangle$ is a service pipe that fails. Upon detecting the failure, the server will select from the PIAs a replacement path with the minimum D for each path contained in $UIA(k, l)$. However, as we have demonstrated, the size of the UIA could be very large. Therefore, this approach based on UIAs may not be the best for large networks.

In reality, as we discussed in Section 3.2.1, the number of paths that require re-routing may be much smaller than indicated by $UIA(k, l)$. Recall that the actual number of active paths is dependent on the routing protocol used. We will refer to the number of active paths affected as the *Criticality Index* (CI) of service pipe $\langle k, l \rangle$. To examine the usefulness of the CI, we computed the CI values of the three sample networks using a shortest path routing algorithm. The statistics are plotted in Figure 9. The graph confirms our intuition that the CI value of a service pipe should be much smaller than its UIA size, implying that online re-routing on a path by path basis is feasible for a network of a moderate size.

A SAAM server can also adopt a backtracking scheme when $\langle k, l \rangle$ has a large CI. The scheme works as follows. The SAAM server first tries to find a replacement path from k to l with a minimum D

⁷The server can pre-select such paths if faster responses are desired.

⁸A service pipe malfunction is usually caused by a software problem such as a bug in the implementation of a packet scheduling algorithm.

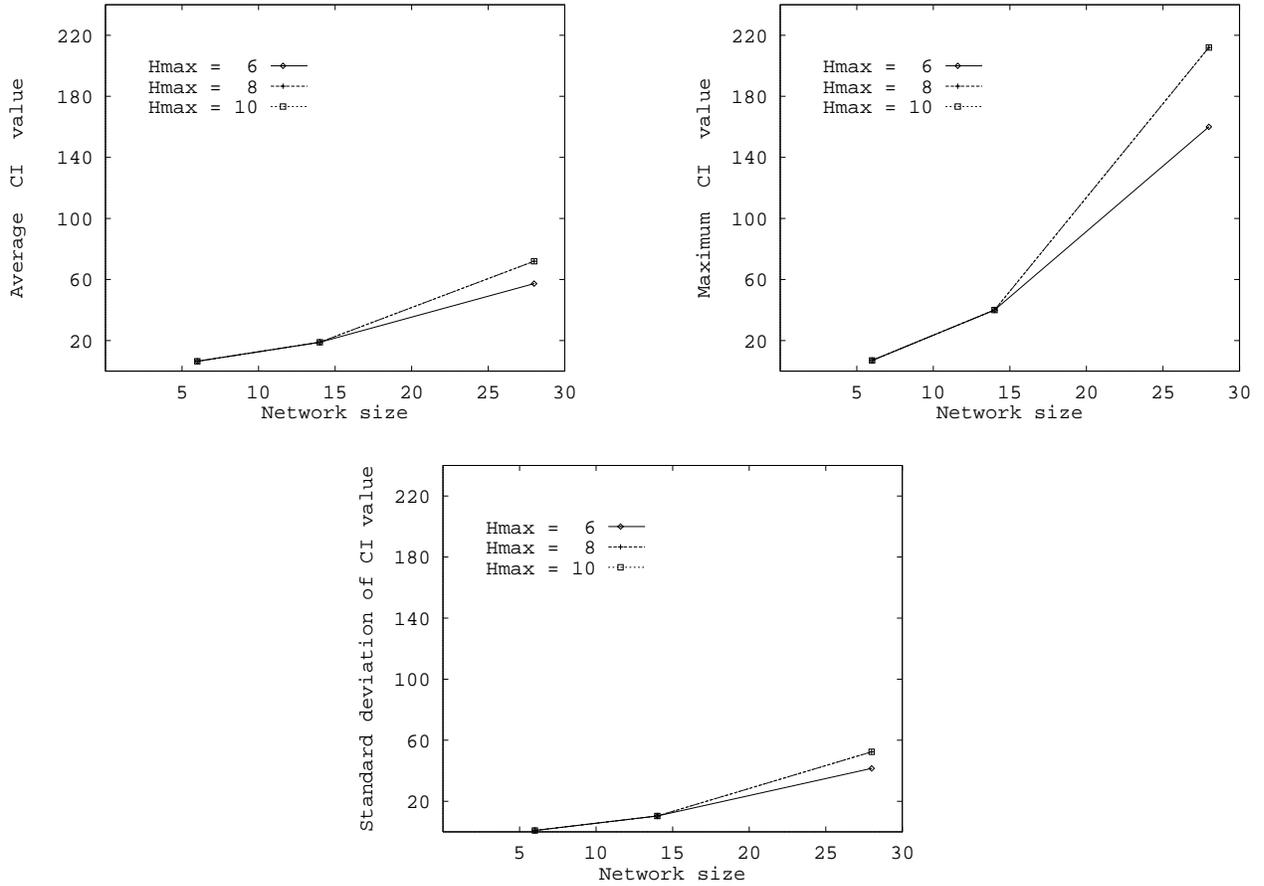


Figure 9: CI statistics for simple shortest path routing

and a hop count no greater than the design parameter⁹ H_{bak} . If unsuccessful, the server then tries to find a replacement path from each parent of k to l . The server continues the same process until a replacement path is found in all subcases or the number of backtracking steps has exceeded the value of the design parameter B_{max} . In the latter case, the SAAM server will then have to perform re-routing on a path by path basis.

3.4 Discussion

Our approach, especially in routing long distance flows, gives a higher priority to meeting the delay requirement of a flow than to meeting other (e.g., loss) requirements. Next, we give a mathematical justification for this choice. Let π^* be the best path found by SAAM for a flow f . Consider the following two cases when f uses π^* :

⁹We are conducting experiments to evaluate the impact of H_{bak} on the performance.

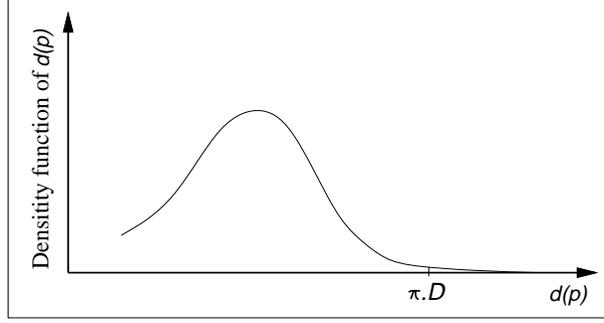


Figure 10: Typical delay distribution of a path

1. $\pi^*.D \leq f.D$ and $\pi^*.E > f.E$. We have for any packet p in the flow,

$$\Pr(d(p) > f.D) = \Pr(d(p) > \pi^*.D) - \Pr(\pi^*.D \geq d(p) \geq f.D) \quad (17)$$

$$\leq \pi^*.E - \Pr(\pi^*.D \geq d(p) > f.D) \quad (18)$$

where $d(p)$ is defined to be the end-to-end delay of p . The typical distribution curve of packet delays for π has a long but *uniformly decreasing* tail near $\pi.D$ along the delay axis. (See Figure 10.) Therefore, the value of $\Pr(\pi^*.D \geq d(p) > f.D)$ could be significant compared to $\pi^*.E$ even if $\pi^*.D$ were a little smaller than $f.D$. In other words, there should be a very high likelihood that the loss requirement of f will be satisfied by π^* if $\pi^*.D$ is much smaller, say 20% smaller, than $f.D$.

2. $\pi^*.D > f.D$ and $\pi^*.E \leq f.E$. We have for any packet p in the flow,

$$\Pr(d(p) > f.D) = \Pr(\pi^*.D \geq d(p) > f.D) + \Pr(d(p) > \pi^*.D) \quad (19)$$

$$\leq \Pr(\pi^*.D \geq d(p) > f.D) + \pi^*.E. \quad (20)$$

From an observation similar to that was described in the previous case, the value of $\Pr(\pi^*.D \geq d(p) > f.D)$ can be much larger than $\pi^*.E$, especially when $\pi^*.D$ is significantly larger than $f.D$. In such a case, the actual packet loss rate of f will likely exceed $f.E$.

4 Conclusions

We have presented the design of an efficient and flexible PIB. Such a PIB is a critical component of our server and active agent based network management architecture which we believe provides a good solution to meet the stringent requirements for managing an integrated services network.

References

- [1] Anindo Banerjea. Simulation study of the capacity effects of dispersity routing for fault tolerant realtime channels. In *Proceedings ACM SIGCOMM '96*, pages 194–205, Stanford, CA, August 1996.
- [2] Jon C.R. Bennett and Hui Zhang. Hierarchical packet fair queueing algorithms. *IEEE/ACM Transactions on Networking*, 5(5):675–689, October 1997.
- [3] J. Case *et al.* The simple network management protocol. Technical Report RFC 1157, Internet Draft, May 1990.
- [4] Sally Floyd and Van Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, August 1995.
- [5] R. Guerin, S. Kamat, A. Orda, T. Przygienda, and D. Williams. QoS routing mechanisms and OSPF extensions. Technical report, March 1997. Internet Draft *draft-guerin-qos-routing-ospf-01.txt*.
- [6] R. Guerin and A. Orda. QoS-based routing in networks with inaccurate information. In *Proceedings of IEEE INFOCOM '97*, Kobe, Japan, April 1997.
- [7] Seungjae Han and Kang G. Shin. Fast restoration of real-time communication service from component failures in multi-hop networks. In *Proceedings ACM SIGCOMM '97*, pages 77–88, Cannes, France, September 1997.
- [8] Simon S. Lam and Geoffrey G. Xie. Group priority scheduling. *IEEE/ACM Trans. on Networking*, 5(2):205–218, April 1997.
- [9] Qingming Ma and Peter Steenkiste. On path selection for traffic with bandwidth guarantees. In *Proceedings of 5th IEEE International Conference on Network Protocols*, pages 191–202, Atlanta, GA, October 1997.
- [10] Larry L. Peterson and Bruce S. Davie. *Computer Networks, A Systems Approach*. Morgan Kaufmann, 1997.
- [11] Chotipat Pornavalai, Goutam Chakraborty, and Norio Shiratori. QoS based routing algorithm in integrated services packet networks. In *Proceedings of 5th IEEE International Conference on Network Protocols*, pages 167–174, Atlanta, GA, October 1997.
- [12] Hughes Network Systems. Distributed routers, centralized control. Technical report, January 1996. HTML document: http://www.data.com/Hot_Products/Routers/Distributed_Routers.html.

- [13] Zheng Wang and Jon Crowcroft. Quality of service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, (7):1228–1234, September 1996.
- [14] Geoffrey G. Xie, Debra Hensgen, Taylor Kidd and John Yarger. SAAM: An Integrated Network Architecture for Integrated Services. To be presented at *6th IEEE/IFIP International Workshop on Quality of Service*, Napa, CA, May 1998. Available from <http://www.cs.nps.navy.mil/people/faculty/xie/pub>.
- [15] Geoffrey G. Xie and Simon S. Lam. Delay guarantee of Virtual Clock server. *IEEE/ACM Trans. on Networking*, 3(6):683–689, December 1995.
- [16] Geoffrey G. Xie and Simon S. Lam. Admission control and loss management for an application-level statistical service. In *Proceedings of 5th IEEE International Conference on Network Protocols (ICNP '97)*, pages 142–151, Atlanta, GA, October 1997.
- [17] Geoffrey G. Xie and Simon S. Lam. Real-time block transfer under a link sharing hierarchy. *IEEE/ACM Trans. on Networking*, 6(1):30–41, February 1998.
- [18] J. Yu, B. Manning, and Y. Rekhter. Router server technical overview. Technical report, January 1998. HTML document: <http://www.rsng.net/overview.html>.
- [19] Wei Zhao and Satish K. Tripathi. Routing guaranteed quality of service connections in integrated services packet networks. In *Proceedings of 5th IEEE International Conference on Network Protocols*, pages 175–182, Atlanta, GA, October 1997.