SAAM: An Integrated Network Architecture for Integrated Services

Hensgen, Debra; Xie, Geoffrey G.; Kidd, Taylor; Yarger, John

SAAM: An Integrated Network Architecture for Integrated Services[*]

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

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

Abstract

The current network architecture is based predominantly on stand-alone routers. It is becoming overtaxed with the introduction of integrated services. In this paper, we propose a **S***erver and* **A***gent based* **A***ctive network* **M***anagement (SAAM) architecture that scales well with integrated services. 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. We describe a scaleable architecture for organizing the servers as well as a concrete design of the PIB. SAAM has the potential of offering a common platform where multiple network functions — such as routing, resource reservation, network management, accounting and security — can be integrated.*

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 efŢcient solution for 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. SpeciŢcally, the network often needs to reserve resources (link bandwidth, buffers, etc.) for a set of packets at particular routers in order to establish an end-to-end ţow path with a speciŢc QoS. Compared to the classical shortest path routing algorithms, QoS routing algorithms need to 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 ţexibility 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 ţow becomes unusable because of a network fault,

[*]Work supported by DARPA under the Next Generation Internet (NGI) initiative

[†]Corresonding author

[‡]Supported also by DARPA under contract number E583

a replacement path should be established within a short time frame. (In other words, the ţow should be quickly *re-routed*, preferably without involving the end user.) 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. SpeciŢcally, 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).

In this paper, we present a **S**erver and **A**gent based **A**ctive network **M**anagement (SAAM) architecture for the efŢcient support of integrated services. SpeciŢcally, 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), with which network functions such as QoS routing and re-routing of real-time ţows can be efŢciently implemented.

The use of route servers has been proposed for data networks [12, 17]. The motivation was to reduce the computational overhead for a set of closely associated routers[1]. QoS routing and re-routing were not considered. Moreover, our development of SAAM has two additional motivations. First, we envison 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.

[1]For example, a set of Internet Service Provider (ISP) routers that share a Network Access Point (NAP) [17].

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. In Section 3, we describe in detail a particular design of the PIB, and explain how SAAM can use the PIB to perform efŢcient QoS routing and re-routing of ţows 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 the 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 QoS degradation along a path, within a short time frame. Therefore, SAAM should use a proactive approach in data collection. Moreover, SAAM should aggregate the data about individual links into "ready to use" path performance information.

- **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.

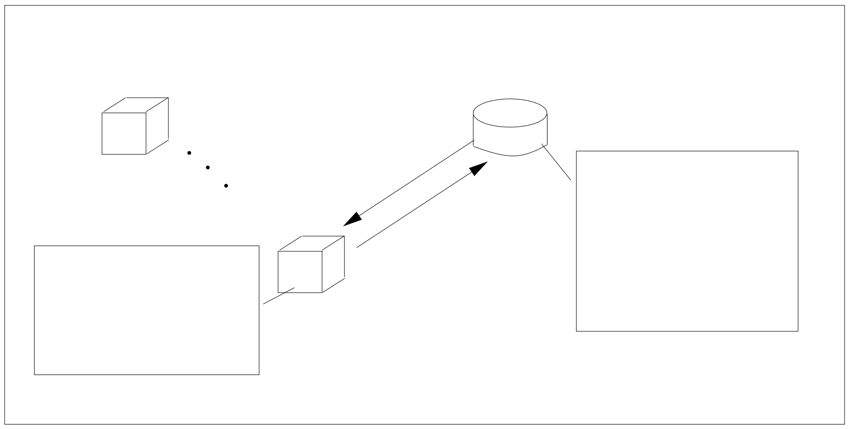

Figure 1: Logical model of SAAM

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 ţow-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 ţow-ids for data that will not be routed by SAAM.

The SAAM server builds a PIB to support QoS routing. SpeciŢcally, the server identiŢes those paths or subpaths that can potentially be used to route ţows, and maintains up-to-date performance parameters for each of them. 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]Details on how to collect such data is beyond the scope of this paper.

2.3 Hierarchical Organization of Servers

To address the scalability issue, SAAM organizes its servers in a hierarchy. (See Figure 2.) SpeciŢcally, at the Ţrst level, SAAM partitions the network into regions, and sets up one server[3] 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.

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. SpeciŢcally, 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.

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

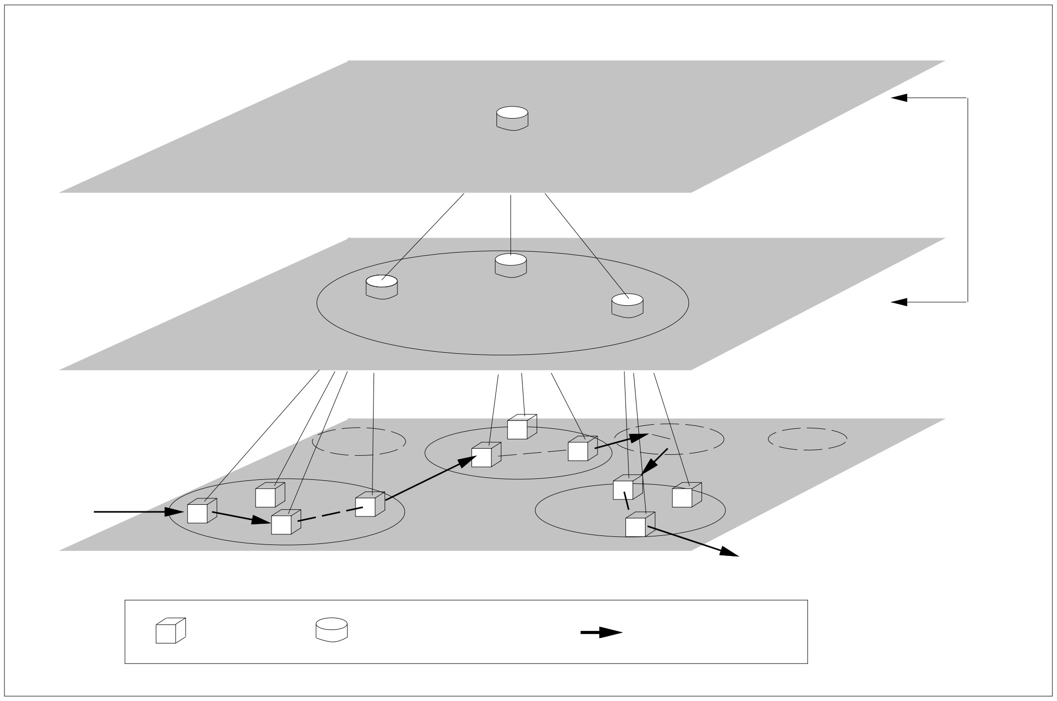

Figure 2: Hierarchical organization of SAAM servers

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 sufŢcient 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 beneŢts of the design, we also explain how SAAM can make use of the created PIB to perform efŢcient 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. SpeciŢcally, we deŢne 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.

3.1.1 Path definition

In an integrated service network, each network link is shared by a set of *logical* service pipes [4], 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) trafŢc is an example of a service pipe.

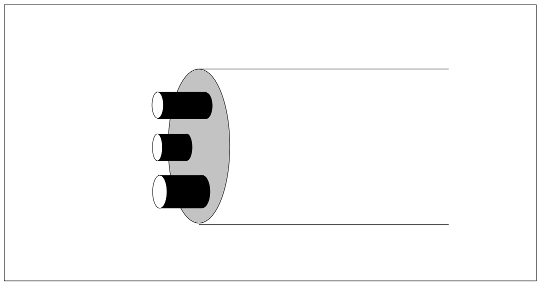

Figure 3: Link shared by service pipes

SpeciȚcally, we deȚne the following parameters for a service pipe (denoted by s):[4]

D	target upper bound on the total packet delay (in seconds); includes queueing delay, transmission delay and propagation delay; service pipe s offers only best effort service when D is unspeciȚed
E	upper bound on the percentage of packets that incur a delay greater than $s.D$; service pipe s offers only best effort service when E is unspeciȚed, and a guaranteed service when $E = 0$
B	amount of pre-allocated link bandwidth; in bits/second
R	bandwidth available for new țows; initially set to B

We deȚne 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*

$$\pi =< s_1, s_2, ..., s_K > \tag{1}$$

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

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 deȚned for a service pipe.

$\pi.D$ the target upper bound on the total packet delay, which is expressed by[5]

$$\pi.D = \sum_{s \in \pi} s.D. \tag{2}$$

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-deȚne $s.D$ to be a target delay upper bound based on the expected packet arrival time; hence equation 2 will continue to hold [14, 8]. Such a re-deȚnition will not complicate țow resource reservation for the following reason: For any țow f that uses path π, the delays of its packets are tightly bounded by

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

where p is any packet in the țow, $EAT(p)$ is its *expected arrival time* to the Țrst 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 trafȚc policer employed for the țow, can be determined *a priori* and subtracted from the țow's delay bound requirement at țow setup time. For example, for each packet p in a țow constrained by a leaky bucket policer with parameters of (ρ, σ), we have [14]:

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

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

$$\pi.E \approx \sum_{s \in \pi} s.E. \tag{5}$$

The derivation of the above equation is as follows. Assuming that packet losses of a țow at different

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

[5]When appropriate, we consider the path as just a set, rather than an ordered sequence, of service pipes.

service pipes are independent events[6], we have

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

Therefore

$$\begin{aligned} \pi.E &= 1 - \prod_{s \in \pi} (1 - s.E) && (7) \\ &\approx 1 - (1 - \sum_{s \in \pi} s.E) && (8) \\ &= \sum_{s \in \pi} s.E. && (9) \end{aligned}$$

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

$$\pi.B = \min_{s \in \pi} \{s.B\}. \tag{10}$$

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

$$\pi.R = \min_{s \in \pi} \{s.R\}. \tag{11}$$

$\pi.F$ the set of ţows that use π

3.1.3 Link sharing

We assume that a suitable link sharing algorithm [2, 4, 16] is implemented at every link so that a Ţrewall is established between the link's service pipes. SpeciŢcally, 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 ţows requesting a statistical service. Consequently, we assume that each link in the network is a statistical pipe, and we will represent a path by $< a_1, a_2, ..., a_K >$ where a_k is the kth router in the path. We use f to denote a statistical ţow. 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[7] π that connects the source and the destination of the ţow, and satisŢes:

$$\begin{aligned} \pi_D &\leq f.D, && (12) \\ \pi_E &\leq f.E && (13) \end{aligned}$$

[6]This is not an overly conservative assumption, considering the fact that the ţow must be sharing service with many other ţows when a packet loss occurs. We have obtained experimental results that support this claim [15].

[7]One that contains only statistical service pipes

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 sufŢces 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. SpeciŢcally, information for a long-distance path (i.e., one that crosses multiple regions) is built and managed *jointly* by three SAAM servers: a Ţrst-level server responsible for the source segment, i.e., from the source to an outgoing border gateway; another Ţrst-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. SpeciŢcally, paths that contain a loop or have a hop count[8] greater than a predetermined value H_{max} are deemed undesirable.

In our current design, a regional PIB consists of two arrays of records. The Ţrst array — called the Path Information Array (PIA) — contains current information (the values of D, E, R, the set of active ţows, etc.) of each desirable local path. The second one — called the Update Information Array (UIA) — holds, for each service pipe, a list of pointers to all PIA records that describe a path containing the service pipe. With the UIA, a SAAM server will react quickly when there is a signiŢcant change in the performance of a service pipe in its region; in particular, the server will update only the PIA records of affected paths following the pointers stored in the UIA, and re-route ţows

[8]in one region

if necessary. (See Section 3.3.) Next, we describe, in detail, the algorithm that a SAAM server will use to build its regional PIB.

Consider a particular SAAM region and its SAAM server. Assume that there are M routers in the region. At boot-up time, the server assigns a unique index $i \in \{1, 2, \ldots, M\}$ to each router; and for each router i, it computes and stores the following set[9]:

$$\begin{aligned} &Parents(i) \\ &\quad = \{j \mid \text{there is a service pipe from router } j \text{ to router } i\} \end{aligned} \tag{14}$$

Afterwards, the server uses the following recursive search algorithm to build its regional PIB. Recall that the objective is to build the PIA and UIA, which we now formally deﬁne as follows:

$$\begin{aligned} &PIA(i, j, h) \\ &\quad = \{\pi \mid \pi \text{ goes from } i \text{ to } j \text{ in } h \text{ hops}\}, \qquad (15) \\ &\qquad 1 \le i, j \le M \text{ and } 1 \le h \le H_{max}. \\ &UIA(k, l) \\ &\quad = \{(i, j, h, n) \mid \text{service pipe} < k, l > \text{is part of the } n\text{th} \\ &\qquad \text{path of } PIA(i, j, h)\}, \qquad (16) \\ &\qquad 1 \le k, l \le M. \end{aligned}$$

The details of the algorithm are speciﬁed in Figure 4. 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 l does not exceed 3, H_{max} is between 6 and 8, and M should be less than 50 for any given SAAM region. Therefore, the algorithm is not much of a burden for the SAAM server. Furthermore, the algorithm needs to be run only when there is a topological update, e.g., a service pipe being permanently added or removed, and other such infrequent events. A service failure will be considered a short-lived condition and will require only modiﬁcation to the parameter values of paths that contain it.

[9]For efﬁciency, we implement a set as an array.

3.3 Routing and Re-Routing of Flows

The PIB created by $Build\_SAAM\_PIB()$ is much more comprehensive than those built using shortest path algorithms.[10] 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 a different scheme at 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 SAAM can use the PIB to perform (i) fast routing of a long distance ﬂow, and (ii) ﬂow re-routing in the event of a link failure or service malfunction.

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 source region SAAM server forwards the request to the parent server, selects[11] from its PIB a path (denoted by π^{src}) that has the smallest D among those that go from the source to an outgoing border gateway, and then sends the information of 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 responses 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}) from an incoming border gateway to the destination, and then sends the information of π^{des} to the parent server. Finally, after receiving the information of 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 $(\pi^{src}.E + \pi^{des}.E)$, and then uses an appropriate QoS

[10]It would be too costly for every router to maintain such a PIB.

[11]The server can pre-select such paths if faster response is desired.

Algorithm specification

```
    *Build_SAAM_PIB ()*
1   $V \leftarrow \{1, 2, ..., M\}$ ;
2   **for** ( each router $i \in V$ ) **do**
3       $a[0] \leftarrow i$ ;
4       $h \leftarrow 1$ ;
5       $Add\_Path(a, h)$ ;

    *Add_Path (a, h)*                // add all paths that go to $a[0]$ in $h$ hops
1   $W \leftarrow Parents(a[h-1])$;
2   **for** ( each router $j \in W$ ) **do**
3       **if** ( $Cause\_No\_Loop(a, h, j)$ )
4           **then**  $a[h] \leftarrow j$ ;
5                 $PIA(a[h], a[0], h) \leftarrow PIA(a[h], a[0], h) \bigcup \{< a[h], a[h-1], ..., a[0] >\}$ ;
6                 $UIA(a[h], a[h-1]) \leftarrow UIA(a[h], a[h-1]) \bigcup \{(a[h], a[0], h, \|PIA(a[h], a[0], h)\|)\}$ ;
7                 **if** ( $h < H_{max}$ )
8                     **then**  $h \leftarrow h + 1$ ;
9                           $Add\_Path(a, h)$ ;
```

where

$$Cause\_No\_Loop(a, h, j) = \begin{cases} \textbf{False} & \text{if } \exists q \text{ such that } 1 \leq q \leq h-2 \text{ and } a[q] = j \\ \textbf{True} & \text{otherwise} \end{cases}$$

Figure 4: Algorithm for building PIA and UIA

routing scheme to search for a suitable path between the gateways.

Re-routing of flows

The network needs to re-route ﬂows when a link fails or a service pipe malfunctions.[12] 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 could be quite large. With the UIA, a SAAM server can re-route on a path by path basis. Speciﬁcally, suppose $< i, j >$ is the service pipe that fails. Upon detecting the failure, the server will select from PIA a replacement path with the minimum D for each path contained in $UIA(i, j)$.

A SAAM server can also adopt a backtracking scheme for ﬂow re-routing. The scheme works as follows. The SAAM server ﬁrst tries to ﬁnd a replacement path from i to j with the minimum D and a hop count no greater than the design parameter[13] H_{bak}. If unsuccessful, the server would then try to ﬁnd a replacement path from each parent of i to j; and continues the same process until either replacement paths are found in all subcases or the number of backtracking steps has exceeded the value of the design parameter

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

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

B_{max}. In the latter case, the SAAM server would then try re-routing on a path by path basis using $UIA(i, j)$.

Discussions

Between meeting the delay and loss requirements of a ţow, our approach gives a higher priority to the former. Next, we give a mathematical justiŢcation for such consideration. Let π^* be the best path found by SAAM for a ţow f. Consider the following two cases when f uses π^*:

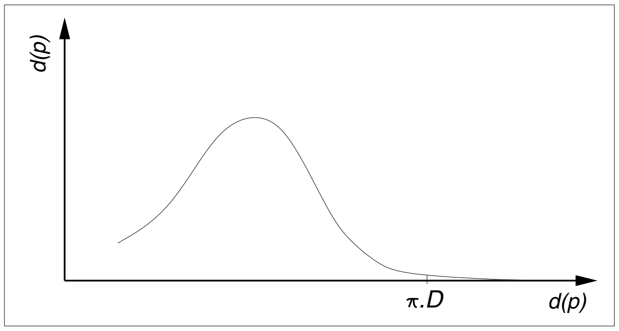

Figure 5: Typical delay distribution of a path

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

$$\begin{aligned} &\text{Pr}(d(p) > f.D) \\ &\quad = \text{Pr}(d(p) > \pi^*.D) - \text{Pr}(\pi^*.D \ge d(p) \ge f.D) \quad (17) \\ &\quad \le \pi^*.E - \text{Pr}(\pi^*.D \ge d(p) > f.D) \quad (18) \end{aligned}$$

where $d(p)$ is deŢned 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 5.) Therefore, the value of $\text{Pr}(\pi^*.D \ge d(p) > f.D)$ could be signiŢcant 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 satisŢed by π^* if $\pi^*.D$ is much smaller, say 20% smaller, than $f.D$.

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

$$\begin{aligned} &\text{Pr}(d(p) > f.D) \\ &\quad = \text{Pr}(\pi^*.D \ge d(p) > f.D) + \text{Pr}(d(p) > \pi^*.D) \quad (19) \\ &\quad \le \text{Pr}(\pi^*.D \ge d(p) > f.D) + \pi^*.E. \quad (20) \end{aligned}$$

From a similar observation described in the previous case, the value of $\text{Pr}(\pi^*.D \ge d(p) > f.D)$ can be much larger than $\pi^*.E$ especially when $\pi^*.D$ is signiŢcantly 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 SAAM, a server based network architecture for integrated services. Unlike other current approaches, 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. We envison SAAM to be the common platform where different network functions such as routing, resource reservation, network management, and security can be integrated.

Acknowledgment

The authors would like to thank David St. John and the anonymous reviewers for their constructive comments.

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 trafŢc 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 and Simon S. Lam. Delay guarantee of Virtual Clock server. *IEEE/ACM Trans. on Networking*, 3(6):683–689, December 1995.

[15] 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. Also available from *http://www.cs.nps.navy.mil/people/faculty/xie/pub*.

[16] Geoffrey G. Xie and Simon S. Lam. Real-time block transfer under a link sharing hierarchy. *IEEE/ACM Trans. on Networking*, 6(1), February 1998. An earlier version in *Proceedings of IEEE INFOCOM '97*, April 1997.

[17] J. Yu, B. Manning, and Y. Rekhter. Router server technical overview. Technical report, January 1998. HTML document: *http://www.rsng.net/overview.html*.

[18] 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.