An Efficient Bandwidth Management Scheme for Real-Time Internet Applications

Fugui Wang
Computer Science and Engineering Department
Michigan State University
East Lansing, MI 48824
wangfugu@cse.msu.edu

Prasant Mohapatra
Department of Computer Science
University of California
Davis, CA 95616
prasant@cs.ucdavis.edu

Sarit Mukherjee and Dennis Bushmitch
Panasonic Information and Networking Technologies Lab
Two Research Way
Princeton, NJ 08540
{sarit, db}@research.panasonic.com

Abstract

Differentiated services (DiffServ) was recently proposed by the IETF as a scalable solution for the Internet QoS. Within the DiffServ architecture, premium services is a service class which is proposed for interactive real-time applications like Internet telephony or video conference. In order to ensure the service quality of premium services, each DiffServ domain need to appropriately negotiate a service level agreement(SLA) with its customers and neighboring domains. Because the resources for premium service is usually a small part of the total network bandwidth, dynamic SLA negotiation is preferred to maximize the resource utilization. However, a completely dynamic SLA negotiation scheme introduces scalability problem for the bandwidth broker(BB). In this paper, we introduce the concept of "pipe" as a viable solution for this scalability problem. We propose a simple threshold-based updating scheme for the pipe which incurs an acceptable updating overhead for the BB while maintaining a high utilization of the pipe. We study the performance of the threshold-based updating scheme and compare it with an ideal updating scheme. The ideal scheme is theoretically optimal as it uses future knowledge for periodically updating the pipe capacity. The performance of the threshold-based updating scheme is very close to the ideal updating scheme.

Keywords: Differentiated Services, End-to-End Guarantee, Expedited Forwarding, Internet, Pipe, Premium Services, Quality of Service

1 Introduction

The current Internet uses the best-effort service model. In this model the network allocates bandwidth to all the contending users as best as it can and attempts to serve all of them without making any explicit commitment to rate or any other service quality. With the proliferation of multimedia and real-time applications, it is becoming more desirable to provide certain Quality of Service (QoS) guarantee [1] for Internet applications. The Internet Engineering Task Force (IETF) [2] has recently proposed the Differentiated Services (DiffServ) model [4] as a scalable solution to provide Internet QoS.

Currently, IETF has defined one class for Expedited Forwarding (EF) [8] and four classes for Assured Forwarding (AF) [9]. EF was originally proposed by Jacobson in [10] as Premium Services. It is expected that premium traffic would be allocated only a small percentage of the network capacity and be assigned to a high-priority queue in the routers. Premium service is ideal for real-time applications such as IP telephony and video conferencing.

In order to maintain the service quality, each ISP domain need to control the amount of incoming traffic, which is negotiated through a service level agreement (SLA). An SLA from domain 1 to domain 2 specifies how much premium traffic and assured traffic could be sent from domain 1 to domain 2. Each domain has a bandwidth broker (BB) which manages the bandwidth resources within the domain and negotiate SLA with neighboring domains. Appropriate SLA negotiation is a very important aspect of DiffServ. If the SLA is oversubscribed, the traffic quality will be degraded. On the other hand, if the SLA is undersubscribed, a part of the bandwidth resources may be wasted. In the current proposals [5] [14], SLAs for assured services are usually static while SLAs for premium services are usually dynamic because they are more expensive.

Most of the current research works on DiffServ are focused on service specification, service architecture, and component definitions [11][12][7]. Few of them [6] discuss how resource management, especially dynamic SLA, could be implemented in the DiffServ environment. Without good resource management schemes, DiffServ itself would not provide any quality of service (QoS) assurances or guarantees. Usually, assured service only need a soft QoS guarantee. We agree that the SLA could be negotiated statically based on statistical estimation or through a learning process. Premium services, however, are more critical and may demand QoS guarantees. So the SLA negotiation should be more precise. As proposed in [14], a dynamic signaling process could negotiate the exact amount of SLA for the premium service traffic. In this scheme, when a new flow need to enter the domain, the BB will receive a signaling message. It will then check the resource database to see whether it can update the corresponding SLA. This will cause even worse scalability problem compared to RSVP [3] because in RSVP the signaling and reservation is distributed among all routers within the domain. In order to solve the scalability problem for BB, the SLAs for premium

services should not be updated upon the join/leave of each connection. A comprising approach can be used by aggregating the SLA updating requests and periodically signaling the BB for SLA updating. We can limit the amount of updating requests sent to BB. Since we update the SLA periodically, the resource utilization should be better than that of the static SLA scheme.

In this paper, we introduce the concept of "pipe" as a solution for resource management within a DiffServ domain. A pipe is a destination-aware SLA from the ingress router to a specific egress router. It specifies the amount of premium traffic that could flow from the ingress router to the egress router, hence forms a virtual channel with a certain bandwidth between an ingress and an egress router. When a new connection joins/leaves the pipe, it only signals the ingress router. BB only get involved when the pipe capacity needs to be updated. Thus, we offload and distribute parts of the resource management work to edge routers so that the BB can handle the scalability problem. We have reported the implementation details of pipe and have used voice traffic to evaluate its performance. The result shows that the use of pipe could greatly reduce the signaling overhead on BBs. Also, the utilization is comparable to the per-flow-based signaling schemes. In our simulations using IP telephony traffic, when the average number of calls in the pipe ranges from 100 to 1000, the updates would be only 10^{-2} to 10^{-4} times of the completely dynamic SLA scheme, while the utilization could still be maintained as high as 80 percent. Thus, without sacrificing any significant utilization we can greatly reduce the updating overheads.

The rest of the paper is organized as follows. Section 2 discusses the concept of pipe. Section 3 studies different implementation schemes of pipe. Section 4 discusses how to improve pipe utilization through updating. The conclusions are reported in Section 5.

2 Concept of Pipe

Although the deployment of Differentiated services (DiffServ) paradigm is simpler and scalable than the Integrated Services (IntServ), it makes end-to-end quality of service guarantee more challenging. The problem lies in the fact that DiffServ operates on the per-hop behavior (PHB) without any end-to-end guarantees. Unlike RSVP [3], it does not perform an end-to-end resource reservation before a connection is set up.

The expedited forwarding (EF-PHB) class of service reserves some fraction of the resources for premium services. Typically, real-time applications (e.g. voice over Internet, video over Internet) are going to be the users of this service. These applications mandates end-to-end resource reservation for proper operation.

In this paper we propose the use of "pipes" to establish end-to-end QoS guarantee in the DiffServ environment. The concept of pipe, as described earlier, can be used to guarantee ingress-to-egress bandwidth availability without the overheads of per-flow state maintenance. We have demonstrated

the implementation and effectiveness of the usage of "pipe" for IP telephony applications. IP telephony is used as a representative of real-time applications requiring premium services in the DiffServ Internet model.

2.1 Definition of a Pipe

A pipe is defined as a logical path between two end points on the network, having a predefined capacity. The choice of end points depends on many factors:

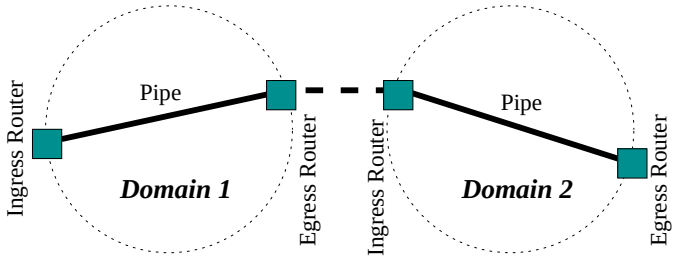


Figure 1: Pipes in DiffServ domains.

Service providers' point of presence: Technically, pipe could be within an ISP domain or extend through multiple domains as shown in Figure 1. However, it is preferable to have pipe within one ISP domain for convenience of management. Usually a pipe starts from an ingress router and end at an egress router.

Traffic between regions: The purpose of pipe is to reduce the signaling and computation overhead of BB. Usually, for the same type of traffic (eg. voice traffic), the aggregate traffic gets smoother as the total amount of traffic increases. Depending on the average traffic amount between an ingress router and an egress router pair, it may or may not be necessary to construct a pipe between this pair. If the total traffic between the two points exceed a certain threshold, we can construct a pipe. Since the aggregate traffic is relatively smooth, pipe capacity need not to be updated frequently. So the BB need not get involved frequently, thus minimizing its load and thereby enhancing scalability. If a connection need to cross an ingress-egress router pair without a pipe, it has to use the dynamic signaling process through the BBs.

Dynamism in the traffic pattern: The choice of end points may also depends on the dynamism of the traffic pattern. The smoother the aggregate traffic between the end points, the higher is the benefit obtained by constructing a pipe between the two points.

2.2 Relationship between Pipe and SLA

Pipe is just a destination-aware SLA. According to the definition in [4], the scope of SLA could be one of the three situation:

1. all traffic from ingress point A to any egress point
2. all traffic between ingress point A and egress point B
3. all traffic from ingress point A to a set of egress points

Pipe belongs to the second category. All of the pipes' configurations within the domain are stored in the domain's BB. BB could also store the topology of the domain and the link capacity between each pair of nodes. When a pipe needs to update its capacity, it talks to the BB and the BB decides whether the request should be granted or rejected. If the request is granted, the BB will notify the ingress router to reconfigure its traffic conditioner (TC), and thereby the SLA is updated.

2.3 Establishments of Pipes

Pipe is usually set up by the network administrator based on the traffic provisioning information. The initial set up information includes ingress router, egress router, and initial bandwidth. The information is sent to the BB of the domain from the ingress router. Based on the network topology information and the resource utilization within the network, BB will either admit or reject the pipe set up request. If the request is admitted, the ingress router will configure a traffic conditioner (TC) based on the initial bandwidth. Once the pipe is set up, it could update its capacity based on the actual utilization. Details about the updating mechanism will be discussed in Section 4.

2.4 Using Pipes to Construct an End-to-end QoS Guaranteed Connection

When an end host need to make a connection with another end host using premium services, it contacts all the pipes it need to use in its path one by one. If the connection are admitted by all of the pipes, it will be set up successfully. Note that it only needs to signal those ingress routers. If one or more of the pipe do not have enough capacity to admitted the new connection, the ingress router could have two choices:

1. reject the connection
2. contact the corresponding BB to increase the pipe capacity to admit this connection, which in turn could be accepted or declined.

By concatenating multiple pipes, an end-to-end QoS guaranteed connection can be established.

3 Implementation of Pipe

In this section we study how a pipe can be implemented within the constraints of current Internet architecture and the DiffServ paradigm. Since expedited forwarding should not be delayed (or experience bounded small delay) per-hop, it is desirable to have it implemented on a priority queuing environment.

Unlike the virtual channel in ATM, a pipe in the DiffServ is just a destination-aware SLA. It is configured in the markers and policy enforcers at the edge routers only for admission control purpose. Once a packet enters the domain core, we cannot tell which pipe it comes from. The BB of the domain stores the pipe information. So BB knows whether the domain is able to increase a pipe capacity or construct a new pipe. There is no per-flow or per-pipe state information stored in the domain core routers. Depending on the topology of the domain, multiple pipe may share some common paths. The forwarding behavior in the core routers is only determined by the DiffServ Code Point (DSCP) [7] of the packet, which retains the per-hop behavior in the core. The service quality of the premium services is ensured through the following methods:

1. premium service packets are queued separately and treated preferentially. The premium service queue could be implemented as a high priority queue over the RIO queue [12]. It could also be implemented as a WFQ with the RIO queue and the premium queue could be given higher weight compared to its actual traffic.

2. Pipe would ensure that the aggregate traffic through it would not exceed its capacity. Since each pipe is destination oriented, BB could make sure that for each link, the aggregate traffic of all the pipes through it would not exceed a certain percentage of the link capacity. (Usually, premium services would not occupy more than 20 percent of the total capacity of a link.) So even there is no per-pipe level service guarantee in the core routers, the QoS of each pipe could still be ensured.

We set up the following experiment to show the service quality of the aggregate scheme under different queuing disciplines. We use the *ns* [13] simulator to implement different queuing disciplines which include priority queuing (PQ) and different WFQs. Figure 2 shows the network topology that is simulated. Four nodes: n0, n1, n2, n3 are part of the network core. The link bandwidth of n0-n1, n1-n2, n2-n3 are 10Mbps, 5Mbps, and 10Mbps, respectively. A pipe with capacity of 10 voice streams exists between n0 and n3. A pipe with capacity of 15 voice streams also crosses link n0-n1. Similarly, a pipe with capacity of 3 voice streams and a pipe with capacity of 15 voice streams cross link n1-n2, n2-n3, respectively. These three pipes (n0-n1, n1-n2, and n2-n3) are used as cross traffic. All of the voice traffic use the premium services through pipes. The remaining capacity of

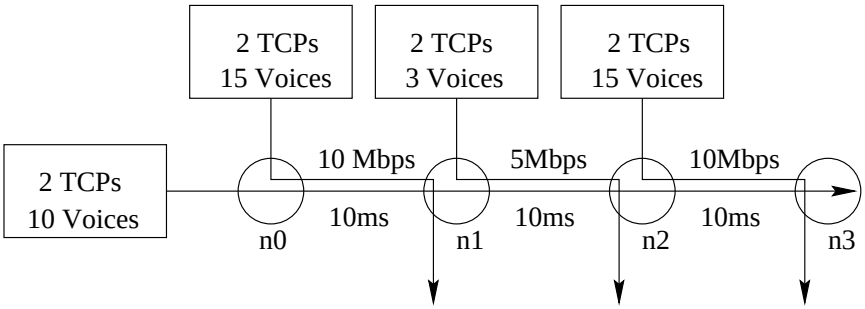


Figure 2: Simulation topology.

the links are used by the best-effort TCP traffic. The premium service queue are implemented in all of the core routers n0, n1, n2, and n3.

We pick up one voice stream from the pipe between n0 and n3 to study the voice packet delay and jitters. In this experiment, we used several queuing disciplines:

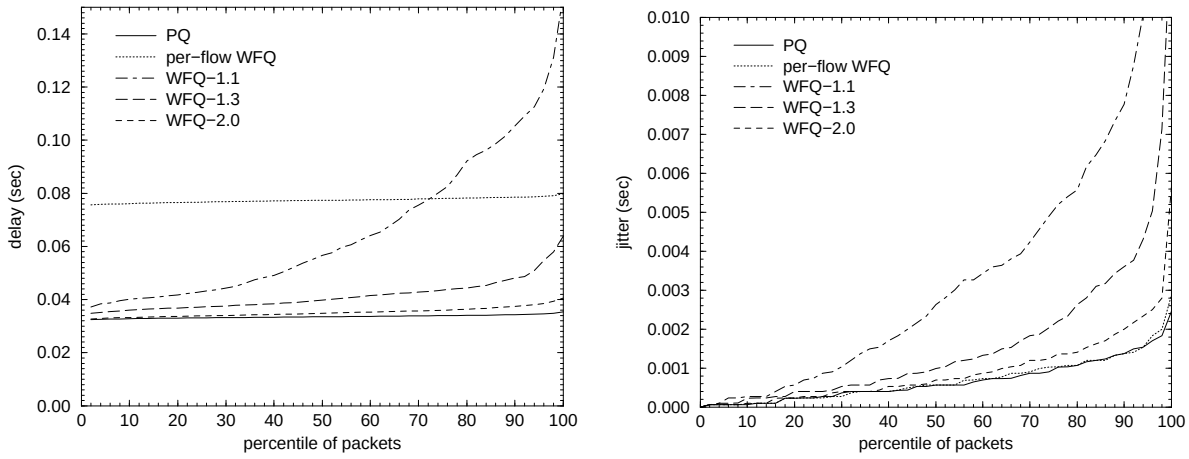
1. *Priority Queue (PQ):* There are two queues in the router, one for the premium services and the other for best-effort services. The premium service queue has higher priority over the best-effort service queue. Thus, all the packets of the premium service queue are served ahead of the best-effort service queue.

2. *Weighted Fair Queue (WFQ):* Same as PQ except that the two queues are weighted fair queues. In order to ensure the quality of premium services, the weight assigned to the premium service queue is more than the actual amount of traffic. For example, WFQ-2.0 means if the average premium service traffic is 1Mbps, we assign a weight equal to 2Mbps capacity to the premium service queue.

3. *Per-flow WFQ:* Each micro-flow has its own queue and is given the weight equal to the peak rate of the voice stream. This queuing scheme is not advisable for DiffServ. We use the result to compare with the first two schemes used in DiffServ, and show how well the aggregate schemes work.

In this simulation, we use the ITU G.711 PCM [15] VoIP traffic as our voice source. The bandwidth of each voice during talk spurt is 87.2 kbps, including the relative protocol headers. Each packet size is 218 bytes. The average burst time is 0.4 second. Average idle time is 0.6 second. It is an exponential ON/OFF model. The packet size of TCP flow is 1000 bytes. The TCP flows are simulated traffic of FTP transferring large size (unlimited) of bulk data.

The simulation result in terms of delay and jitter for one of the voice stream from n0 to n3 is shown in Figure 3. Figure 3(a) is the plot of delay variation and Figure 3(b) is the plot for



(a) Delay (b) Jitter

Figure 3: Delay and jitter in a pipe.

jitters. Observe from the graph that PQ has the best quality, and WFQ-2.0 has almost the same performance as PQ. The performance of WFQ-1.3 is also acceptable. In term of delay, PQ, WFQ-2.0, and WFQ-1.3 work even better than per-flow WFQ. The reason for this could be intuitively explained as follows. Per-flow WFQ could be deemed as a service in which each voice stream uses a thin link and the thin links are separated from each other. PQ, WFQ-2.0 and WFQ-1.3 use a fat link for all of the voice streams. The thin link will stretch individual packet, hence introduce longer delay. The fat link could transmit individual packet much faster, even though the link utilization are similar in both cases. However, per-flow WFQ has almost a fix delay. So the jitter of per-flow WFQ is as good as PQ. The following conclusions could be drawn from these results:

1. Premium services could be implemented with a PQ or an aggregate level WFQ if the relative weight is given higher than 1.3. The performance is comparable to per-flow level WFQ.

2. With appropriate policing at the entrance of pipe, aggregating multiple pipes would not have much side effect on the quality of premium services.

4 Improving Pipe Utilization Through Updating

Depending on the burstiness of the aggregate traffic through the pipe, pipe capacity could be set as static or dynamic. If it is static, then all of the admission controls are done at the entrance of the pipe, BB will not receive any updating message from the pipe. However, in order to limit the rejection rate, we will have to reserve the peak of the traffic as the pipe capacity. This, of course, will make the utilization very low given the fact that the traffic could vary greatly during different time of a day and probably vary during different days. On the other hand, we can make the pipe

capacity completely dynamic, that is, upon each new call arrival, we update the pipe capacity. In this case, the utilization could be up to 100 percent. However, BB will receive one updating message upon each call arrival/departure, which increases the load and thus defies the benefit of building pipes. So there is always a tradeoff between the utilization and the updating overhead. One possible solution is, instead of using static or completely dynamic pipe, a hybrid scheme can be used by updating the pipe capacity periodically. The period is set to several seconds or minutes so that the updating overhead is negligible or acceptable. At the same time, the utilization could still be kept at a high level. In the remaining parts of this section, we explore several updating methods and show how well they work.

4.1 Traffic model

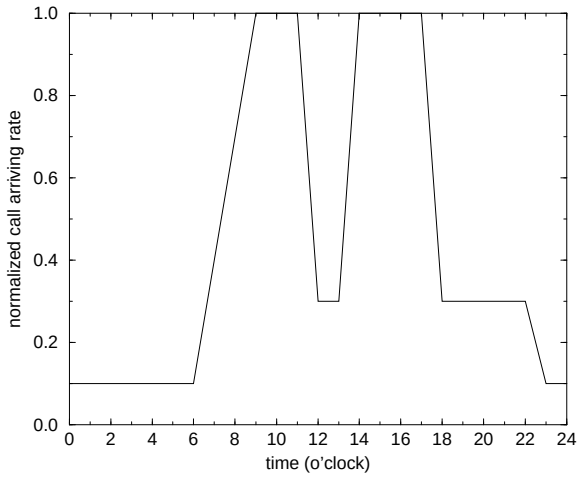
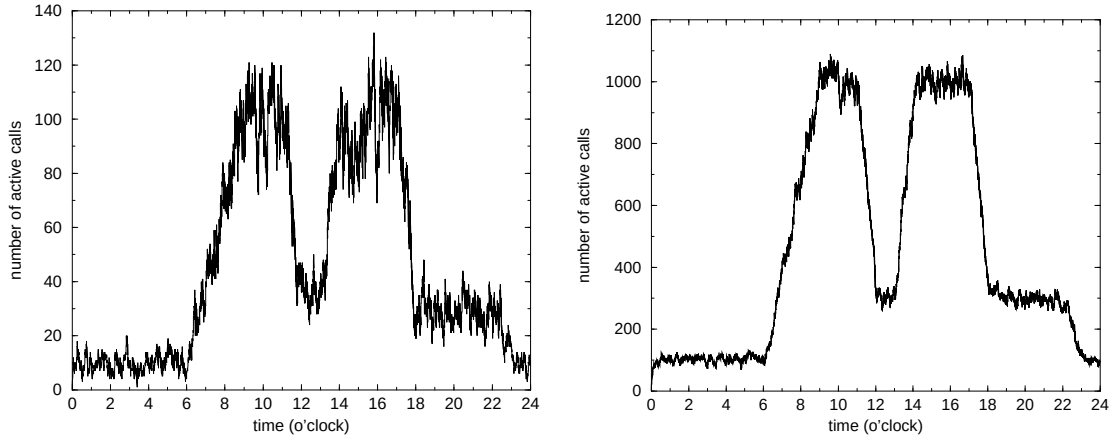


Figure 4: Call arrival distribution during a day.

Premium services are designed for real-time traffic like VoIP, Video Conference, etc. Here we select voice traffic as a representative of real-time traffic for our study. VoIP is most likely to become the first user of pipe because of its popularity and the bandwidth limitations of the Internet. The intensity of telephone calls vary greatly during different time of a day. The British Telecommunications (BT) published the average call numbers in their network. Figure 4 shows the call arrival pattern where the peak rate (during 9am-11am and 2pm-5pm) is set as 1.0. Actually, the call arrival during a certain period, say, 9am-11am, is not flat as we show in Figure 4. In short term, the call arrival could be approximated as a Poisson process. In order to show the detail traffic and see how well different updating methods works for pipe, we build a simple simulator to mimic this arrival/departure process. The length of each call is exponential with an average of 5 minutes. Figure 5 is the traffic during a day from the output of our traffic simulator. In Figure 5(a), the

peak arrival rate is set as 20 calls/min so that on average there are 100 calls in the pipe during the peak period (i.e., 9am-11am, 2pm-5pm). In Figure 5(b), the peak arrival rate is set as 200 calls/min so that on average there are 1000 calls in the pipe during the peak period. In the rest of this paper, for convenience, we simply say that the pipe with traffic shown in Figure 5(a) has 100 calls and the pipe with traffic shown in Figure 5(b) has 1000 calls, although 100 or 1000 calls are only the average number of calls in the pipes during the peak period. From the graph we can observe that the relative burstiness of Figure 5(a) is larger than that of Figure 5(b). For a Poisson arrival/departure process, the variance decreases as the average number of calls increases. Our goal is to find a good prediction method to update the pipe capacity so that we can have both high utilization and acceptable updating overhead.



(a) 100 calls (b) 1000 calls

Figure 5: Number of active calls in the pipe during different time of a day.

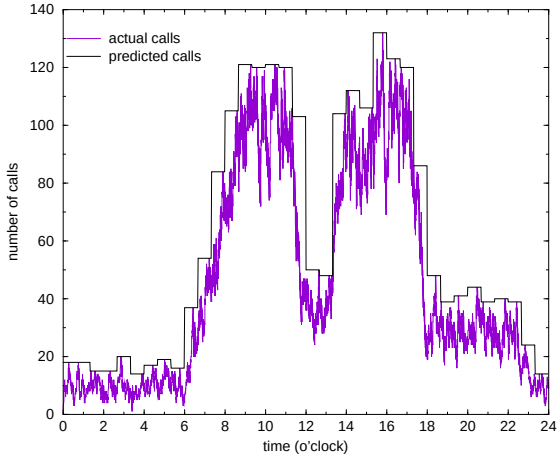
4.2 Prediction Model

The variance of the number of calls is subject to two main factors: a long term factor which is caused by different usage of phone during different time slot; a short term factor which is caused by the Poisson call arrival process. The long term factor may be due to the variation during a day, or week, or month, or season. The short term factor is due to the independent behavior of users. Without updating, a pipe has to reserve its capacity according to the usage during the peak time. Here we propose a threshold-based prediction scheme to update the pipe capacity. In order to see how well this scheme performs, we first study the ideal case which we call as ideal prediction.

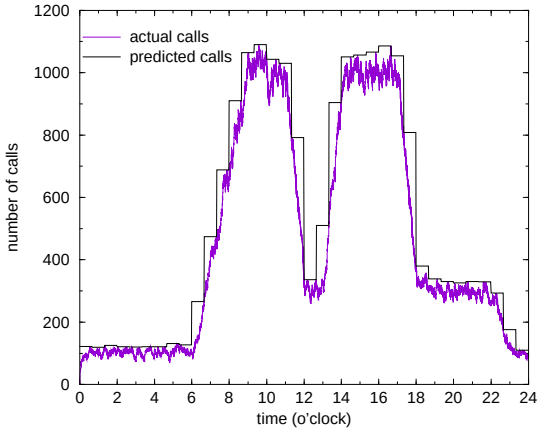
4.2.1 Ideal Prediction

In this scheme, the pipe capacity is updated periodically. Assume that upon each updating point, we can precisely predict the peak bandwidth of the next period so that we can set the pipe capacity

to that value. In this scheme, we can ensure that no call gets rejected and the utilization is as high as possible. However, the utilization is not 100 percent because, during the period between two neighboring updating points, the traffic still varies. Ideal prediction cannot be implemented in real world since we are not aware of the future events. However in our simulation we first count the number of calls during different times and use that information as our "prediction", thus using the future knowledge. Figure 6 shows the updating processes. The solid line is the pipe capacity and the doted line is the actual used bandwidth. The updating interval is 40 minutes. From the graph we can find, for the same updating interval, Figure 6(b) has higher utilization than Figure 6(a). The reason is: Figure 6(b) has 10 times more calls than Figure 6(a), so the traffic in Figure 6(b) is smoother than that in Figure 6(a).



(a) 100 calls



(b) 1000 calls

Figure 6: Ideal prediction (Update Interval=40min).

Since we choose the peak rate of the period as the pipe capacity, using smaller update interval could increase the pipe utilization. This result is observed in Figure 7. From Figure 7, we can also observe that the pipe utilization could be higher than 80 percent as the update interval is less than 30 minutes. However, if there is no update during the 24 hours, the utilization would be as low as 40 percent. So updating scheme could improve the utilization by a factor of 2 or more. This result coincides with the result shown in [16] where the traffic trace from AT&T telephone network was used for a similar analysis. For the same update interval, the pipe with 100 calls always have lower utilization than the pipe with 1000 calls. So more the traffic between two end points, more the benefit we can get by building a pipe between them.

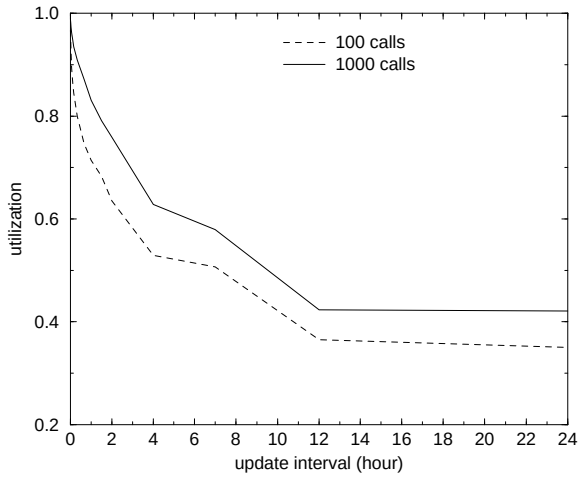


Figure 7: Ideal Predication: Utilization vs. Update Interval.

4.2.2 Threshold-Based Prediction

Ideal prediction is not realistic since we cannot know the future events. Our goal is to find a realistic prediction algorithm which has a performance comparable to the ideal prediction. A linear prediction is not good for this case because of the Poisson call arrival process. Here we propose a simple prediction scheme called *threshold-based prediction*. The motivation for the threshold-based prediction scheme is very simple: if we use the per-flow updating scheme, we need to increase the pipe capacity by one for each call arrival and decrease the pipe capacity by one for each call departure. This could make sure that the pipe has the highest resource utilization, but will incur very high updating overhead. However, if we increase the pipe capacity by δ ($\delta > 1$) when a new call arrives and the pipe is full, then we do not have to update the pipe capacity for every incoming arrival as long as the total number of calls does not exceed the new pipe capacity. By reserving more than we actually need right now, we loose a bit utilization, but we may be able to save a lot of

updates, given the fact that the total number of calls in the pipe will not change drastically during a short period. The value for δ is selected based on the tradeoff analysis between overhead and utilization. When the number of calls in the pipe drops below a threshold, we can decrease pipe capacity in order to increase the utilization. But we do not decrease the pipe capacity to the exact amount of calls. We can keep it higher than the actual number of calls so that the new coming calls will not trigger a new updating. The algorithm can be stated as follows:

1. Upon a call arrival, if the number of calls reaches the *pipe_capacity*, then *pipe_capacity* is increased by δ.

2. Upon a call departure, if the number of calls is under $pipe\_capacity - 2 \times \delta$, then *pipe_capacity* is decreased by δ.

If $\delta = 1$, then this scheme is same as the per-flow updating scheme. Usually, δ is selected larger than 1. The larger δ is, the less frequently the pipe capacity is updated. By doing this, we could ensure:

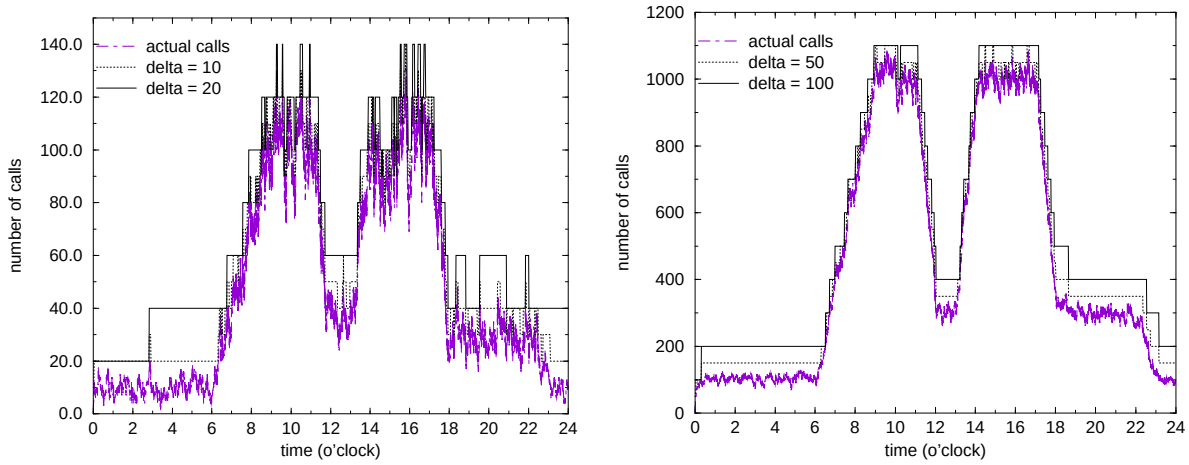
$$pipe\_capacity - 2 \times \delta \leq number\_of\_calls < pipe\_capacity \tag{1}$$

So, the utilization has a lower bound:

$$utilization \geq \frac{pipe\_capacity - 2 \times \delta}{pipe\_capacity} \tag{2}$$

The results of the threshold-based updating are shown in Figure 8. Figure 8(a) is the plot for the pipe with 100 calls. Here the gray line is the actual number of calls in the pipe. The black solid line is the pipe capacity for $\delta = 20$ and the black dotted line is the pipe capacity for $\delta = 10$. With the smaller δ, we have higher utilization however the updating is also more frequent. Figure 8(b) is the plot for the pipe with 1000 calls. The pipe capacity for $\delta = 100$ and $\delta = 50$ are shown there. We can observe that the pipe capacity predictions in Figure 8(b) are closer to the actual traffic compared to that in Figure 8(a) because the traffic are smoother. Instead of updating the pipe capacity periodically, we only update the pipe capacity when the actual number of calls changes out of the region between the upper and lower thresholds. So in Figure 8, the updates are less frequent during 0am to 6am, but are more frequent during 6am to 9am.

From equation (2) we infer that the utilization could be controlled through δ. Figure 9 shows the relationship between utilization and δ. For the pipe with 100 calls (shown in 9(a)), if we select $\delta = 10$, the utilization is about 80 percent. For the pipe with 1000 calls (shown in 9(c)), if we select $\delta = 100$, the utilization is about 80 percent. For both of them, utilization increases as δ decreases. The corresponding updating overheads are also shown in the right side. Here we use the normalized updates as the updating overhead. As we know, if we choose $\delta = 1$, i.e., upon each call



(a) 100 calls (b) 1000 calls

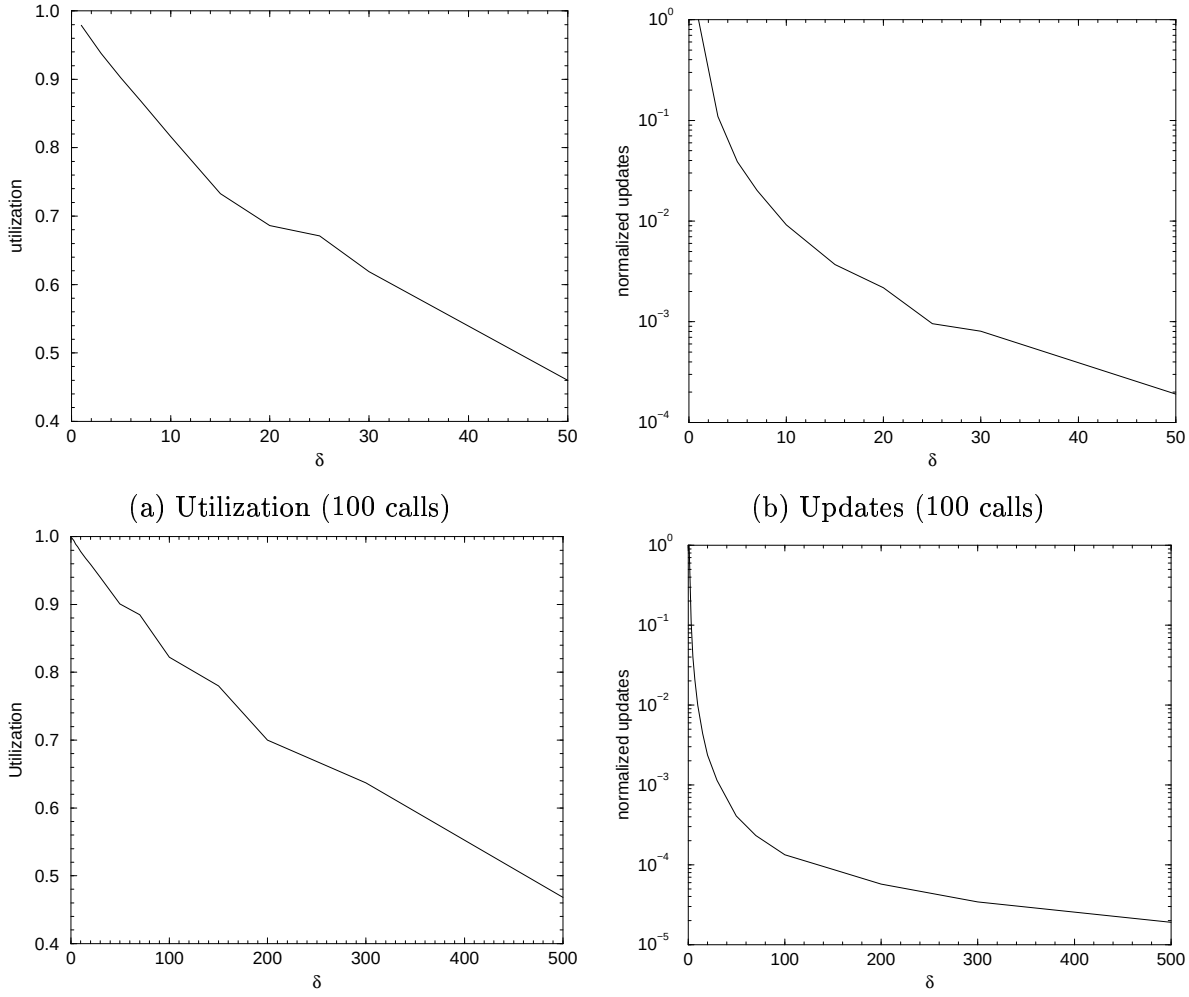
Figure 8: Threshold prediction (δ=delta).

arrival or departure, we update the pipe capacity, then for each call we have two updates. If we choose $\delta > 1$, we will have less updates. The normalized updates is defines as the actual number of updates during the simulation period divided by double the number of calls, i.e.,

$$normalized\_updates = \frac{number\_of\_updates}{2 \times number\_of\_calls} \tag{3}$$

Equation (3) could be used to evaluate how much updating overhead can be saved by using pipe. For example, in Figure 9(b), if we select $\delta = 10$, the normalized updates is 10^{-2}. So we removed 99% of the updating overhead. In Figure 9(d), if we select $\delta = 100$, the normalized updates is 10^{-4}, which means we removed 99.99% of the updating overhead! Thus, by sacrificing a little utilization, we could save a lot of updating overhead, especially when the aggregation degree in the pipe is high (e.g. for the pipe with 1000 calls).

Figure 10 shows the relationship between normalized updates and utilization directly so that we can evaluate the performance of the threshold-based updating more clearly. It also makes it possible for us to compare the performance with that of the ideal prediction. The dashed line is the ideal prediction and the solid line is our threshold-based prediction. Obviously, the ideal prediction performs better than our threshold prediction. However, the difference is not large. Given the fact that the threshold-based prediction is very simple and could be easily implemented, it is a very good scheme. In Figure 10(a), if we want to keep the utilization to be 80%, for both of the prediction schemes, the normalized updates overhead is about 10^{-2}. In Figure 10(b), if we want to keep the utilization to be 80%, for both of the prediction schemes, the normalized updates overhead is about 10^{-4}. The actual number of calls in the later case is ten times of the former case. So the absolute number of updates in the later one is only about 10% of the former one. The number of updating



(a) Utilization (100 calls)

(b) Updates (100 calls)

(c) Utilization (1000 calls)

(d) Updates (1000 calls)

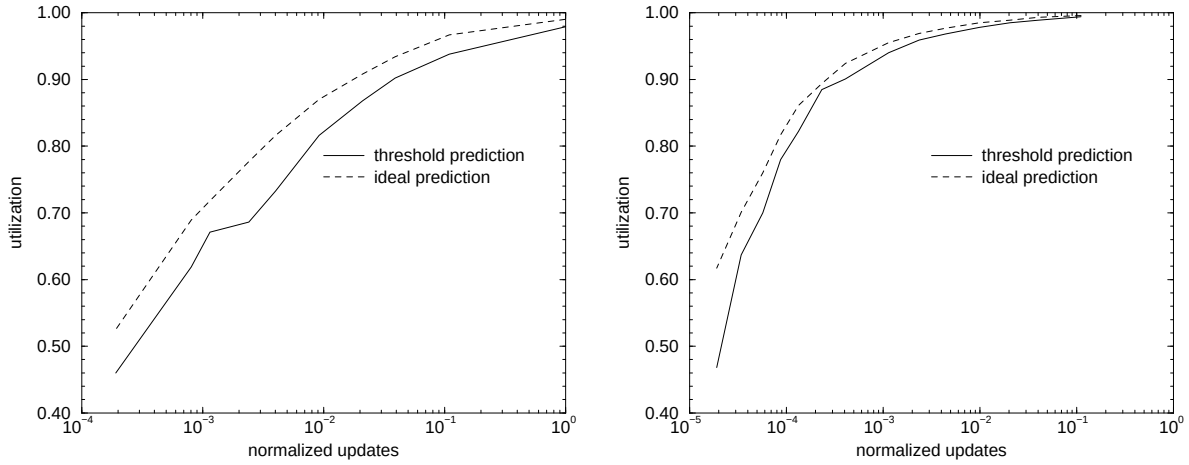
Figure 9: Relationship between Utilization, Updates and δ.

messages receive by BB decreases as the total number of calls increases. However, without the proposed pipe implementation, upon each call arrival/departure, the BB will receive an signaling message. The signaling message overhead received by BB is proportional to the number of calls, which may inhibit scalability. This scalability problem is solved by using pipes.

The normalized updates shown in Figure 10 is the average value over the 24 hours' simulation. The average update interval could be calculated as the following:

$$average\_update\_interval = \frac{24\ hours}{normalized\_updates \times number\_of\_calls \times 2} \tag{4}$$

Since the threshold-based prediction does not use periodical updating, the update interval during the worst case should be much shorter than the average update interval. The average value is important because one BB may be in charge of many pipes. It is unlikely that all of the pipes



(a) 100 calls (b) 1000 calls

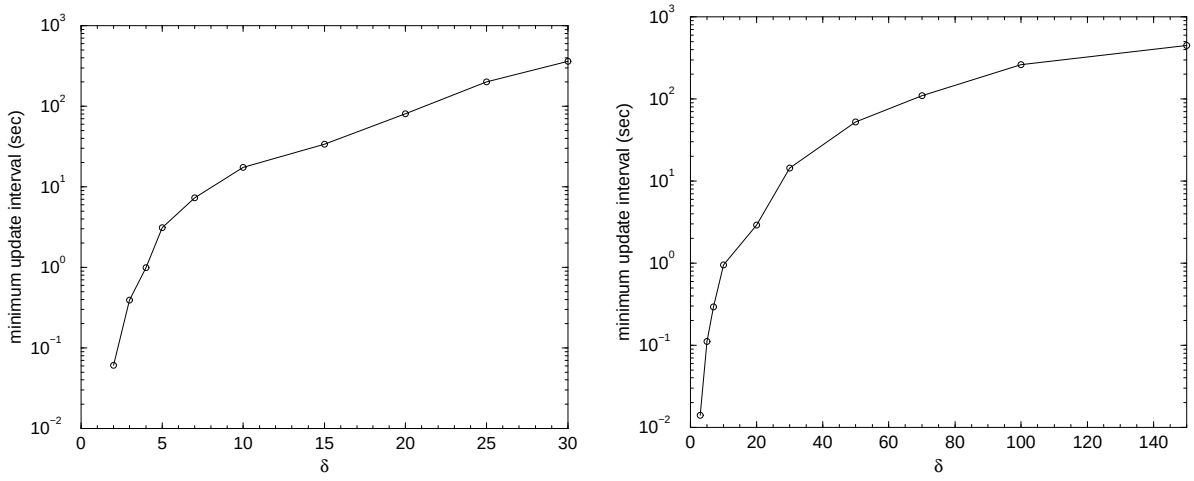
Figure 10: Performance comparison of threshold-based prediction and idea prediction.

will be in the worst case at the same time. Statistically, when one pipe is frequently updated during a period, other pipes may be in a stable state and need not be updated often. However, our simulation shows that even in the worst case, the updating interval is pretty long. The result is shown in Figure 11. For the pipe with 100 calls during the peak period in Figure 11(a), if we choose $\delta = 10$, then minimum update interval is about 20 seconds. The utilization corresponding to $\delta = 10$ for this pipe is about 80% (see Figure 9(a)). For the pipe with 1000 calls during the peak period in Figure 11(b), if the we choose $\delta = 100$, then minimum update interval is about 300 seconds. The utilization corresponding to $\delta = 100$ for this pipe is about 80% (see Figure 9(c)). The updating interval bounds are long enough. The updating overhead is acceptable even in the worst case.

The proposed threshold-based updating uses a fix δ. In our simulation, we use the current number of calls in the pipe as a trigger of the pipe capacity updating. In the real implementation, we may use the pipe utilization as a trigger instead of using the number of calls. This will make it easier to implement in the DiffServ environment because the DiffServ edge router may or may not keep the per-flow state information. The utilization could be achieved through counting the wasted tokens in the leaky bucket. Then we do not have to keep the soft state of each flow at the edge routers (i.e., entrance of the pipe).

5 Concluding Remarks

In this paper we introduced the concept of "pipe" as a solution for resource management of premium services in the differentiated services environment. Pipe could be implemented in an aggregate level



(a) 100 calls (b) 1000 calls

Figure 11: Minimum update interval for different δ.

as a destination-aware SLA. We use VoIP traffic as an example and show the QoS of a voice steam under different queuing schemes of premium services. Since pipe is relative static compared to the per-flow completely dynamic resource management scheme, it greatly reduces the signaling overhead on bandwidth brokers. In order to improve the utilization of pipe, we proposed the threshold-based updating scheme. Through simulation, we have shown that this updating scheme incurs very little overhead while providing high utilization. The threshold-based updating scheme could provide a performance comparable to the ideal prediction scheme, which uses the knowledge of future events.

References

[1] P.Ferguson and G. Huston, *"Quality of Service,"* John Wiley & Sons, 1998.

[2] IETF Home Page, *"http://www.ietf.org/"*.

[3] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, *"Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification,"* RFC 2205, Sept. 1997.

[4] Y.Bernet, J. Binder, S. Blake, M. Carlson, B. E. Carpenter, S. Keshav, E. Davies, B. Ohlman, D. Berma, *"A Framework for Differentiated Services,"* Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-diffserv-framework-02.txt, Feb. 1999.

[5] X. Xiao and L. M. Ni, *"Internet QoS: the Big Picture,"* IEEE Network, Mar. 1999.

[6] A. Terzis, L.Wang, J. Ogawa, L. Zhang, "A Two-Tier Resource Management Model for the Internet," IEEE Global Internet'99, Dec. 1999.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, *"An Architecture for Differentiated Services,"* RFC 2475, Dec. 1998.

[8] V. Jacobson, K. Nichols, K. Poduri, "An Expedited Forwarding PHB," RFC 2598, IETF, Jun. 1999.

[9] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Assured Forwarding PHB Group," RFC 2597, IETF, Jun. 1999.

[10] V. Jacobson, *"Differentiated Services Architecture,"* Talk in the Int-Serv WG at the Munich IETF, Aug. 1997.

[11] D. D. Clark, "Adding Service Discrimination to the Internet," Telecommunication Policy, 20:169-181, 1996.

[12] D. D. Clark, and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," IEEE/ACM Transactions on Networking, 6(4):362-373, Aug. 1998.

[13] UCB/LBNL/VINT Network Simulator-ns(version 2), *"http://www-mash.cs.berkeley.edu/ns/".*

[14] K. Nichols, V. Jacobson and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet," RFC 2638, July 1999.

[15] International Telecommunication Union (ITU), *"http://www.itu.int/".*

[16] P. Goyal, A. Greenberg, C.R. Kalmanek, W.T. Marshall, P. Mishra, D. Nortz, K.K. Ramakrishnan, *"Integration of Call Signaling and Resource Management for IP Telephony,"*IEEE Network Magazine, VOl 13, No. 3, May/June 1999, pp-24-32.