Probabilistic Bandwidth Reservation by Resource Priority Multiplexing *

Jeffery P. Hansen[1] Haifeng Zhu[2] Raj Rajkumar[2]

[1]Inst. for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, PA 15213

[2]Dept. of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Probabilistic bandwidth guarantees on variable bit-rate network flows offer significant improvements in throughput for only modest decreases in guarantee level. In this paper we present a probabilistic bandwidth reservation scheme for variable bit-rate flows called Resource Priority Multiplexing (RPM). The RPM algorithm uses packet marking on edge routers and selective dropping on core routers to provide per-flow specification of probabilistic QoS with guarantees ranging from best-effort to hard reservation. RPM also supports multiple metrics for probabilistic level of service specification. All flow tables are managed only by the edge routers enhancing scalability of the approach. Unlike existing statistical multiplexing approaches, RPM uses a time-multiplexed priority tagging algorithm which provides higher control over flows guarantee levels. The algorithm has been implemented in a live system as part of the IP protocol stack and experimental results are presented.

1 Introduction

Quality of Service (QoS) for multimedia applications has received considerable attention in recent years, and many approaches to QoS management have been reported in the literature. The majority of these approaches are based on hard reservations of resource requirements for each task. In many cases, this can lead to significant underutilization of resources when reservations must be made for variable bit-rate (VBR) flows, particularly when guarantees are required for all flows. For example, consider a video-conference flow operating at 30 frames-per-second (fps). Due to the compression algorithms used by the video encoding software, the bandwidth required to sustain this level of service can range from 0.5Mbps to 5Mbps. In order to provide guaranteed service, it would be necessary to reserve for the peak bandwidth of 5 Mbps. This means that that on a network with 100 Mbps links, we could provide guarantees for no more than 20 flows. If we are willing to relax the hard guarantee and provide statistical guarantees, we can significantly improve the resource utilization. With a guarantee level of 99% (i.e., on average 99% of packets are delivered) more than 40 of the above mentioned video flows can be admitted, doubling the reservation capacity of the network.

To provide the statistical guarantees, we propose a probabilistic approach similar to statistical multiplexing which we call Resource Priority Multiplexing (RPM). Like in statistical multiplexing, an admission control algorithm is used to determine which flows can be admitted, but unlike most statistical multiplexing methods, an active control algorithm is used to dynamically change the priority of flows providing fine grain control over the service provided to each flow. The basic idea is that each competing flow on a network link takes turn being vulnerable to packet loss, with the period of vulnerability being computed from the flow characteristics and the guarantee requirement for the flow. If an overload, due to many flows simultaneously being in a high-utilization state occurs, packets will be shed from the currently vulnerable flow.

RPM is part of an integrated approach to QoS resource management[8][9] incorporating algorithms for QoS optimization[7] and real-time queueing theory[10] for predicting lateness in stochastic systems. Our implementation of RPM is targeted at a medium-sized private networks such as those that might be found on next generation SC-21 (21st Century Surface Combatant) shipboard networks used to coordinate command and control efforts of a battle fleet.

1.1 Definitions

We define a flow as a stream of packets along a specific path from a source node to a destination node. Flows are modeled as a fluid. Probabilistic Level of Service (PLoS) for a flow is defined as any metric with a value in the range 0.0 to 1.0 (with 1.0 being best), describing a level of satisfaction with regard to lost or late packets.

*This work supported in part by contract DARPA N66001-97-C-8527.

PLoS metrics are expected values over an infinite duration. We define three metrics for PLoS. Time-based PLoS (TPLoS) is the fraction of time the flow is not obstructed (i.e., no packet loss), Loss-based PLoS (LPLoS) is the fraction of total fluid that is delivered (i.e., ratio of bytes received to bytes transmitted) and Instananeous-Loss-Based PLoS (IPLoS) is the average of the instantaneous fraction of fluid obstruction. We also define the following symbols which will be explained in greater detail later in this paper:

$r_{\max}$	Resource limit
n	Number of flows
m	Number of modes
N_i	Number of states in model for flow i
π_j	Priotity ordering of flows for mode j
π_{jk}	The kth flow in the ordering for mode j
α_j	Fraction of time RPM is in mode j
T_m	Time to rotate through all modes.
p_{ji}	PLoS for task i while in mode j
b_{li}	Bandwidth required by flow i in state l
q_{li}	Fraction of time flow i is in state l.
$f_i(r)$	pdf of bandwidth r on flow i
$g_{jk}(r)$	pdf of bandwidth r for flows with priority higher than k in mode j
A_i	PLoS requirement for flow i
$Q(r_h, r)$	PLoS metric function for flow requiring r when higher priority flows are using r_h.

1.2 Previous Work

Numerous mechanisms have been proposed for providing network QoS guarantees. One well-known mechanism is the bandwidth reservation protocol RSVP[1]. In RSVP, flows are identified by a flow label, and a fraction of link bandwidth can be reserved at routers along that flow. While reservations may not be made beyond the total bandwidth available at each link, any bandwidth unused by the reserving flow is available to other flows on a best-effort basis. Another example of a reservation-based approach is Darwin[2] which allows reservations on classes of flows in a hierarchy.

The main problem with hard reservations schemes is that while they are relatively strait-forward to implement and generally have simple admission control algorithms, many real-world applications produce variable bit rate flows. For example, due to the effects of video compression algorithms, a video-conference application might have a resource utilization that varies between 0.5 Mbps and 5 Mbps over the lifetime of the flow. Since we must reserve for the peak bandwidth, there can be substantial amounts of wasted resources when applications do not consume all of the bandwidth that was reserved for them. Since many applications, such as video-conferencing, can tolerate some packet loss without substantial reductions in performance, we feel that the hard reservation model is overly conservative and a probabilistic approach will results in a better cost/performance tradeoff.

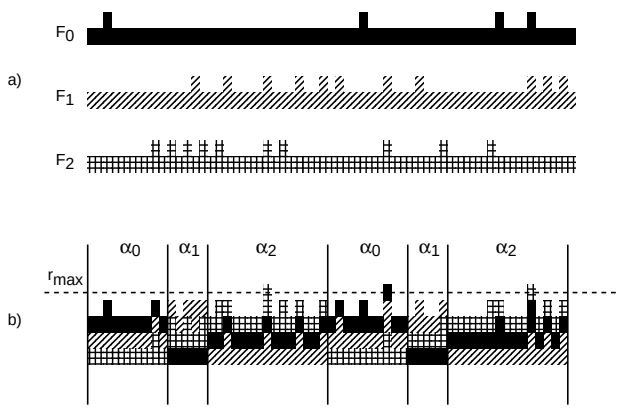

Figure 1: RPM Algorithm Example

One alternative to hard reservation schemes that has been proposed for variable bit-rate flows is statistical multiplexing [3][4][5][6]. Using statistical multiplexing, the system can be sized for the average case rather than the worst case. Applications can operate at higher QoS operating points (e.g., higher picture quality or higher frame rate) or more flows can be admitted by accepting a specified nominal rate of lost or late packets. Most statistical multiplexing schemes are based on using admission control policies to predict the level of service that can be expected from mixing a set of flows with specified traffic characteristics. If the service requirements (e.g., probability of packet loss) for any flows do not meet their requirement the flow is rejected, or a policy change is mandated. One method used for improving the performance of statistical method is to use priority levels and group flows having similar packet loss tolerances at the same level.

1.3 Our Approach

RPM is implemented as a set of time-multiplexed priority modes. Each mode is a total ordering of the flows in the set ranking them from least vulnerable to most vulnerable. The mode holding times (periods of vulnerability) are computed from the traffic characteristics, the QoS requirements of the set of flows, and the maximum capacity of the link or virtual path on which the flows are allocated. The mode holding times are chosen such that flows with high QoS requirements are vulnerable for a smaller fraction of the time.

As an example, consider the three flows shown in Figure 1a. Each of the three flows has a "high" resource consumption state and a "low" resource consumption state. The flows switch between the states independently and at random. RPM will allocate resources to each of the flows as shown in Figure 1b. Within each mode, resources are alloted from the high-priority flow

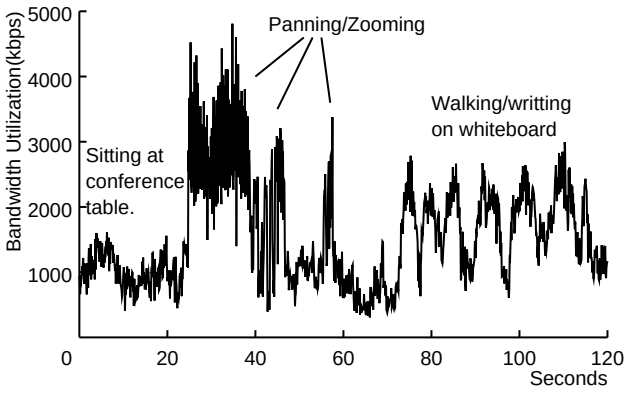

Figure 2: Sample Video Data

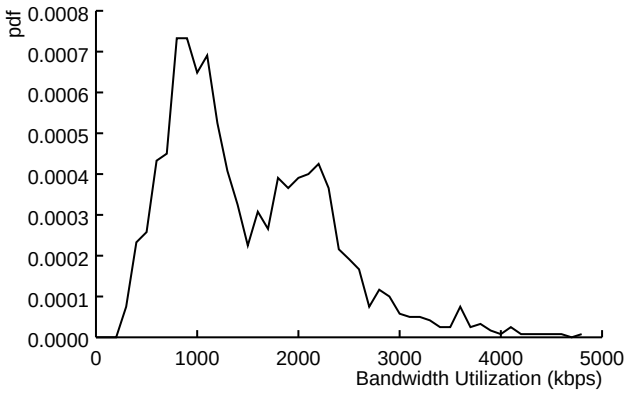

Figure 3: Sample Video PDF

until we exceed the available resources $r_{\max}$ (in the figure, the bottom flow is the highest-priority flow). As the mode changes, the order in which resources are allocated changes, with α_j representing the fraction of time RPM is in each mode j. In this example, we see that flow F_3 packets were dropped in both cycles of mode 2, and flow F_0 packets were dropped in the second cycle of mode 1.

2 Flow Model

RPM uses a Markov Modulated Fluid Flow model of application bandwidth requirements. This model is based on observations of the bandwidth requirements of real-world multimedia applications. For example, consider the two minute segment taken from a 45 minute video-conference show in Figure 2 of bandwidth utilized by the "vic" video conference tool using the h261 encoding at 30 fps. The bandwidth in this case ranges from a low of about 1 Mbps for low-motion scenes such as a group of people sitting at a conference table, to a medium level of about 3.8 Mbps for medium-motion scenes such as people walking about a room, to a high of 5 Mbps for high-motion scenes such as camera zooming or panning. These variations in bandwidth utilization are due to variations in inter-frame and intra-frame compression ratios as the complexity or amount of movement in the scene changes. Notice that the fraction of time that the peak bandwidth requirement of the flow is at the maximum level of 5Mbps is very small. This is further evident from the pdf of bandwidth over the clip shown in Figure 3.

In the Markov Modulated Fluid Flow model, the bandwidth requirements for a flow i is given by a set of N_i states. Associated with each state is the fraction of time q_{li} the flow is in that state, and the bandwidth b_{li} required while in that state. In the rest of this paper, we will represent this model by its pdf expressed as a weighted sum of delta functions:

$$f_i(r) = \sum_{l=0}^{N_i-1} q_{li}\delta(r - b_{li}) \quad (1)$$

3 RPM Admission Control

We define a mode π_j, to be a total ordering of the set of n flows. In this paper, we will assume that the number of modes m is equal to the number of flows and that:

$$\pi_j = \{j \bmod n, j+1 \bmod n, \ldots, j+n-1 \bmod n\} \quad (2)$$

The priority of flow π_{jk} is considered to be higher than the priority of flow $\pi_{jk'}$ when $k > k'$.

We consider two special cases when the PLoS guarantee A_i is equal to 1 (hard guarantees) or 0 (best effort). Flows with $A_i = 1$ guarantees are statically assigned the highest priority in all modes and the number of modes m is reduced by number of such flows. Flows with $A_i = 0$ are simply ignored for the purposes of the model and are treated as non-RPM traffic. The number of flows n and the number of modes m are both decreased by the number of these best-effort flows.

The RPM admission control program (rpmacp) is invoked for each flow admission and departure to determine if QoS requirements can be met and to compute the fraction of time α_j in which RPM must be in each mode j. The input to the algorithm is the flow model $f_i(r)$ and probabilistic level of service requirement A_i for each of the flows. The four possible outcomes of the admission control algorithm are:

Always Guaranteed - Worst case requirements of all flows can be satisfied (light load).

Always Satisfied - All QoS requirements can be met with basic statistical multiplexing (medium load).

Multiplexing Required - QoS requirements can be met only with RPM multiplexing (heavy load).

Unsatisfiable - QoS requirements can not be met even with RPM multiplexing (overload).

The "Always Guaranteed" case corresponds to the level of traffic that can be handled using hard bandwidth guarantees where the sum of the worst case requirements

for all flows is less than the capacity of the bottleneck link. The "Always Satisfied" cases corresponds to the level of traffic that can be handled using basic statistical multiplexing. In the "Multiplexing Required" case, a set of mode holding time coefficients α_j will also be computed. The coefficients represent the the fraction of time that the system should be in each mode.

3.1 Mode Holding-Time

Computation of the mode times is based on computing the delivered probabilistic level of service for each flow i in terms of the mode holding times α_j, and solving for the α_j which satisfy the constraints $\sum_{j=0}^{m-1} p_{ji}\alpha_j \geq A_i$ for each flow i and the constraint $\sum_{j=0}^{m-1} \alpha_j = 1$. The left-hand-side of each of the inequality constraints is the delivered Probabilistic Level of Service for flow i. It is computed as the sum of the delivered Probabilistic Level of Service for each mode weighted by the fraction of time spent in each mode.

Because most of the coefficients in the constraints are close to 1, solving with the equality relation as a constraint tends to cause numerical instability. For this reason, we formulate the problem as the linear program:

$$\begin{array}{ll} \min & \sum_{j=0}^{m-1} \alpha'_j \\ \text{s.t.} & \forall i : \sum_{j=0}^{n-1} p_{ji}\alpha'_j \geq A_i \end{array}$$

and solve it using an off-the-shelf linear program solver. The α'_j are unnormalized mode holding times. If the sum of the α'_j is greater than 1.0, then the constraint problem is infeasible and it is not possible to admit all of the flows and meet their probabilistic QoS requirements. We can make this statement since decreasing an α'_j value will always decrease the left-hand-side of one or more inequality constraints.

The normalized mode holding times α_j are computed by distributing the slack $(1 - \sum \alpha'_j)$ over all of the non-zero unnormalized holding times. Adding the slack in this way will maximize the tolerance for error in the actual mode switching times obtained by the RPM mechanisms.

Two other special cases are possible when formulating the linear program. If all of the p_{ji} are 1.0, then the "Always Guaranteed" condition applies meaning that the worst case resource requirements for all flows can be met simultaneously. If the lowest value of p_{ji} for each flow i is greater than or equal to the probabilistic requirement A_i for each flow, then the requirements can be met for any set of mode holding times and the "Always Satisfied" condition holds.

To compute per-mode PLoS values p_{ji} we first define $g_{jk}(x)$ as the pdf of bandwidth utilization for flows of priority level k and higher. For the highest priority level $k = n - 1$, this is simply the unity weighted delta function:

$$g_{j(n-1)}(x) = \delta(x) \tag{3}$$

since there are no higher priority flows. For all other flows, $g_{jk}(x)$ is the convolution:

$$g_{jk} = g_{j(k+1)} * f_{\pi_{j(k+1)}} \tag{4}$$

The convolution represents the pdf of the sum of the resource requirements for flows of priority higher than $k+1$ and the resource requirements for the flow at priority $k+1$.

We can then write the probabilistic level of service for flow i in mode j as:

$$p_{ji} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_i(r')g_{ji}(r-r')Q(r', r-r')dr'dr \tag{5}$$

This equation represents the convolution of f_i and g_{ji} integrated from $-\infty$ to ∞ with the addition of a metric function $Q(r, r_h)$ added as a weighting factor. The metric function maps the resources r needed by some flow and the resources r_h needed by all flows of a higher priority to a Probabilistic Level of Service value for the flow under those conditions. In this paper we consider three metric functions.

The Time-Based Probabilistic Level of Service (TPLoS) metric function:

$$Q_T(r, r_h) = \begin{cases} 1 & \text{if } r_h + r \leq r_{\max} \vee r = 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

is 1 for a flow i when the resources remaining after subtracting out the instantaneous resources r_h consumed by all higher-priority flows meets the instantaneous resource requirements r of that flow. Using this metric will result in a p_{ji} metric which represents the fraction of time the flow is not experiencing any packet loss.

The Loss-Based Probabilistic Level of Service (LPLoS) metric function:

$$Q_L(r, r_h) = \frac{1}{\overline{r}_i}\begin{cases} r & \text{if } r_h + r \leq r_{\max} \\ 0 & \text{if } r_h > r_{\max} \\ r_{\max} - r_h & \text{otherwise} \end{cases} \tag{7}$$

where $\overline{r}_i = \int_{-\infty}^{\infty} r f_i(r) dr$ is the average consumed bandwidth. Using this metric function will result in a PLoS metric representing the fraction of packets delivered. While this metric function may be greater than 1 for some r, the integral in Equation (5) will still be bounded between 0 and 1.

The Instantaneous-Loss-Based Probabilistic Level of Service (IPLoS) metric function:

$$Q_I(r, r_h) = \begin{cases} 1 & \text{if } r_h + r \leq r_{\max} \vee r = 0 \\ 0 & \text{if } r_h > r_{\max} \\ \frac{r_{\max} - r_h}{r} & \text{otherwise} \end{cases} \tag{8}$$

is similar to the Q_L metric function except it represents the average over time of the instantaneous packet loss rate.

3.2 Optimizations

Numerous techniques can be used to optimize the performance of the admission control algorithm. Flow resource requirement models are quantized by representing them in terms of multiples of some specified slot size. For example, if the slot size is 100 kbps, then a 1 Mbps bandwidth requirement would be represented as 10 slots. We can then represent the pdf for flows as a sorted list of slot count and probability density values. A further optimization can be made by merging all of the probability density for resource demands greater than the link capacity into a single delta function. By doing this, the worst case time for computing the convolution of two pdfs is $O(s^2)$ where s is capacity of the constraining link in slots.

3.3 Admission Control Example

Consider three flows F_0, F_1 and F_2. Flow F_0 has a IPLoS requirement of 0.99 and transmits at 500 Kbps 95% of the time and 2 Mbps 5% of the time. Flow F_1 has a IPLoS requirement of 0.99 and transmits at 200 Kbps 85% of the time and 1.8 Mbps 15% of the time. Flow F_2 has a IPLoS requirement of 0.98 and transmits at 700 Kbps 90% of the time and 1.2 Mbps 10% of the time. Also assume that they are to be transmitted on a link with a capacity of 3 Mbps, and that we use the modes $\pi_0 = \{2,1,0\}$, $\pi_1 = \{0,2,1\}$, and $\pi_2 = \{1,0,2\}$. Also assume that we use a slot size of 100 Kbps. The pdfs of each of the flows are:

$$f_0(r) = 0.95\delta(r-5) + 0.05\delta(r-20)$$
$$f_1(r) = 0.85\delta(r-2) + 0.15\delta(r-18)$$
$$f_2(r) = 0.90\delta(r-7) + 0.10\delta(r-12)$$

Using Equation (5) we can compute the coefficient:

$$\begin{aligned} p_{20} &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(0.95\delta(r'-5) + 0.05\delta(r'-20)) \\ &\quad (0.90\delta(r-r'-7) + 0.10\delta(r-r'-12)) \\ &\quad Q_I(r', r-r')dr'dr \\ &= 0.995 \end{aligned} \tag{9}$$

We use $f_0(r)$ as the pdf for the $f_i(r)$ term in Equation (5) and $f_2(r)$ as $g_{ji}(r)$ term representing the pdf for higher priority flows. Similarly computing each of the other p_{ji} parameters, we can use them to derive the linear program needed to solve for the unnormalized mode holding times.

$$\begin{aligned} \min \quad & \alpha'_0 + \alpha'_1 + \alpha'_2 \\ \text{s.t.} \quad & \alpha'_0 + 0.974\alpha'_1 + 0.995\alpha'_2 \geq 0.99 \\ & 0.9925\alpha'_0 + \alpha'_1 + 0.974\alpha'_2 \geq 0.99 \\ & 0.974\alpha'_0 + \alpha'_1 + \alpha'_1 \geq 0.98 \end{aligned}$$

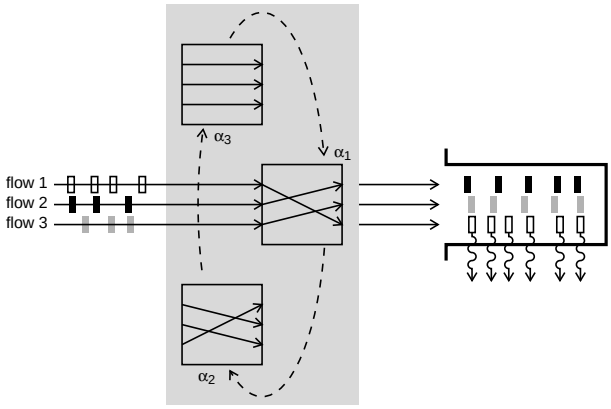

Figure 4: RPM Tagging Algorithm

Solving this linear program, we get the unnormalized mode holding times: $\alpha'_0 = 0.65730$, $\alpha'_1 = 0.25671$, and $\alpha'_2 = 0.08308$. Since the sum of these is less than 1.0, we know that it is possible to admit all of the flows using RPM. We next compute the slack by subtracting the sum of the α'_j from 1.0 getting a value of 0.00291. Dividing this by three and adding the result to each of the mode holding times we get the normalized mode holding times: $\alpha_0 = 0.658$, $\alpha_1 = 0.258$, and $\alpha_2 = 0.840$. Finally, we can substitute these α_j values back into the constraint equations to compute the guarantees actually assigned to each of the flows as $p_0 = 0.993$, $p_1 = 0.995$, and $p_2 = 0.990$.

4 RPM Mechanisms

The RPM mechanisms are comprised of three main components: a mode change control component, a tagging component and a selective-dropping queue component. The basic operation of these mechanisms are shown in Figure 4. Packets for each of the flows pass through the tagging mechanism and are tagged according to the current mode. Each mode is maintained for a holding time $T_m\alpha_j$, after which RPM activates the next mode in the cycle (T_m is the system configurable time for a complete cycle through all modes typically on the order of seconds to tens of seconds). The tagged packets are inserted into a common queue. If the queue is full, and the priority of all packets in the queue is higher than the arriving packet, then the arriving packet is dropped. If any packets in the queue have a lower priority than the arriving packet, then a packet at the lowest priority level is dropped, and the arriving packet is inserted into the queue.

The tagger and the selective-dropping queue are implemented in the kernel for each of the routers (although the tagger is only used on edge routers), while the mode change controller is implemented as a user-level daemon running on each of the edge routers. In addition to these modules there is also a utility program `rpmctl` which al-

lows flows to be registered, mode tables to be created, etc. There is also a RPM control library which can be linked with a resource manager to allow it to access the RPM mechanisms. Control messages are passed using a reliable protocol implemented on top of UDP. Our reference implementation is running under the FreeBSD 2.2.7 and FreeBSD 4.1 kernels.

4.1 Tagger

The tagger operates in the kernel of all edge routers in the network. The tagger intercepts packets as they propagate up to the IP layer before being passed to the IP forwarding mechanism. Once tagged, a packet keeps its tag as it propogates through the network. The tagger maintains a table of registered flows, where each flow is identified by an RSVP-style[1] flowlabel. The flowlabel consists of the source and destination IP addresses and port numbers (masks may be used to specify ranges of address and port numbers). The tagger also maintains a flow id for each flow and a current tag value. Note that only the edge routers need to maintain this flow table. One tag value is reserved to indicate that a flow should not be tagged. After a flow has been registered, the mode change control mechanisms can change the tags for registered flows by specifying the flow IDs and tag values. Packets already having an RPM tag, or not registered as RPM flows are unmodified by the tagger.

The tags are implemented as 4-byte IP options with 16-bits available for a tag value, although currently only 256 tag values are supported. Tags are added to packets on the first hop router. The `mbuf` chain representation for packets in FreeBSD allows us to insert the options field quickly and efficiently. The insertion is performed by allocating an additional `mbuf` object and linking it to the beginning of the chain, copying the packet header to the new `mbuf`, appending the IP options to the end of the new header, and updating the old header `mbuf` to exclude the old header information.

4.2 Mode Change Control

Mode changes are handled by a daemon running on each of the edge routers. The daemons manage the RPM mode table, cycling through the modes round robin and updating the RPM tags on managed routes at each mode change by using an `ioctl()` interface on a special `/dev/rpm` pseudo-device. Since mode changes on the routers must be synchronized, we use NTP to ensure that the clocks of the routers are synchronized and compute the absolute times at which mode changes should occur relative to the Unix Epoch (January 1, 1970). Since mode changes occur on the order of seconds, the level of synchronization provided by NTP is generally sufficient. The RPM daemon also receives remote requests for changes and queries to the mode table and the RPM kernel module managed route table. This interface can be used by a resource manager to set up RPM flows, or by the `rpmctl` control utility.

4.3 Selective-Dropping

Selective dropping in RPM is implemented by replacing the IP output queue in the kernel with a custom RPM queue. The queue allows a packet already in the queue to be dropped when a packet of higher priority arrives and the queue is full. The queue supports 256 priority levels and implements $O(1)$ enqueue, dequeue and drop operations. Each packet in the queue is inserted into two doubly linked lists, a primary list of all the packets in the queue and in a secondary list of all packets with the same priority tag value. Untagged best-effort packets are inserted into the lowest-priority list. An enqueue operation is implemented by inserting the packet at the end of the primary list, and at the end of the secondary list corresponding to its tag value. A dequeue operation is implemented by removing the packet from the head of the primary list and the head of the secondary in which it was inserted. A drop operation is implemented by checking each of the 256 secondary lists from the lowest-priority up until we find a non-empty list. We then remove the packet at the head of this secondary list, and then remove it from the primary list.

4.4 Super Reservations

The RPM priority mechanism can also be used to implement a "super reservation" to pre-reserve priority levels for critical tasks. For example, consider a computer network on a battleship where we wish to provide statistical guarantees for routine communication, but allow critical communication to supersede the routine communication in the event of an emergency such as when responding to an attack. This can be implemented by using different disjoint sets of priority tags. Each of the two classes of flows (routine and critical) can be treated together as separate RPM admission control problem with the priority tags for the routine flows being assigned at a lower priority level. For example, normally dormant critical flows A,B and C might be assigned priority levels 1,2 and 3, while normally active flows D,E and F might use priority levels 4,5 and 6. If an emergency situation arises, the critical flows need simply begin sending packets and the they will automatically supersede the lower priority flows.

5 Experimental Results

Experiments with RPM were conducted on a test-bed consisting of two Windows NT machines and a FreeBSD-based RPM router linked in series. One of the NT machines was used as a traffic source and the other was a sink. The link between the traffic source and the FreeBSD machine was operating at 100Mbps, and the

	RPM				best-effort		
Flow	kb sent	kb drp.	asng. delv.	meas. delv.	kb sent	kb drp.	meas. delv.
1	58724	1794	96.1%	97.0%	47625	2284	95.4%
2	53795	1707	94.9%	96.9%	50677	2096	96.0%
3	49017	2553	93.2%	95.1%	49023	2006	96.1%
4	52226	3447	91.7%	93.8%	49743	2486	95.2%
5	48965	3874	90.1%	92.7%	53010	2345	95.8%
6	45114	4330	88.4%	91.2%	47705	2141	95.7%
7	46000	3167	88.3%	93.6%	48883	2274	95.6%
8	47942	4656	88.3%	91.2%	48023	1996	96.0%
9	47755	3567	88.3%	93.1%	44892	2174	95.4%
10	45493	4220	88.3%	91.5%	51480	2208	95.9%
11	49935	5656	88.3%	89.8%	48846	2044	96.0%
12	44945	4069	88.3%	91.7%	49309	2107	95.9%
13	46726	4703	88.4%	90.9%	50749	2594	95.1%
14	43474	4801	88.4%	90.1%	50481	2281	95.7%
15	42782	3825	88.4%	91.8%	46654	2088	95.7%
16	44169	3401	88.4%	92.9%	54206	2514	95.6%
17	43953	4376	88.4%	90.9%	41507	1650	96.2%
18	48943	4810	88.4%	91.1%	47525	1832	96.3%
19	47973	3968	88.4%	92.4%	50851	2012	96.2%
20	53092	4009	88.4%	93.0%	52738	2402	95.6%
21	477418	3004	99.4%	99.4%	401339	23617	94.4%

Table 1: All ON/OFF Traffic

	RPM				best-effort		
Flow	kb sent	kb drp.	asng. delv.	meas. delv.	kb sent	kb drp.	meas. delv.
1	56541	171	99.9%	99.7%	56210	1864	96.8%
2	58668	280	99.9%	99.5%	45864	1676	96.5%
3	45353	257	99.9%	99.4%	61893	1774	97.2%
4	73923	453	99.9%	99.4%	51292	2122	96.0%
5	57126	857	99.9%	98.5%	47754	1668	96.6%
6	55888	1209	99.8%	97.9%	50507	1527	97.1%
7	59589	1793	99.4%	97.1%	50011	1168	97.7%
8	69208	2379	98.6%	96.7%	46075	1479	96.9%
9	46603	2271	97.8%	95.4%	51177	1443	97.3%
10	52338	2512	96.8%	95.4%	54890	2167	96.2%
11	55165	4379	95.3%	92.7%	59055	1646	97.3%
12	44982	4891	93.5%	90.2%	55757	1333	97.7%
13	42850	4491	90.6%	90.5%	48234	1852	96.3%
14	53699	6268	87.9%	89.6%	41572	1271	97.0%
15	40087	4300	87.9%	90.3%	53204	1763	96.8%
16	51096	3185	92.2%	94.1%	58219	1580	97.4%
17	46807	4569	90.0%	91.1%	59400	1428	97.7%
18	39810	3344	87.9%	92.3%	54320	1762	96.9%
19	45708	4984	87.9%	90.2%	46639	1530	96.8%
20	51472	4312	88.0%	92.3%	59277	2346	96.2%
21	394075	1146	99.9%	99.7%	349140	17771	95.2%

Table 2: High Priority Video and Low Priority ON/OFF

link between the FreeBSD machine and the sink machine was a 10Mbps link to allow us to create a resource bottleneck on the FreeBSD output port (In a real network the bottleneck would be created due to flows on multiple input ports passing to the same output). Two types of flows are generated on the source machine. ON/OFF flows are synthetically generated flows using a two state Markov model in which one state represents a high bandwidth state and the other represents a low bandwidth state. In addition to the synthetic flows we also use an actual video flow generated by vic[11] under the h261 video format. The examples shown in Figures 2 and 3) are excepts from the flow. A VCR is connected to the source machine through a frame grabber and a tape of an actual video conference is used as a video source for vic. The use of a video tape ensures that the exact same video stream is used in all experiments.

In the following sections, results from the four experiments we performed are presented. Each of the experiments compares the assigned delivery rate (guarantee) of the packets with the actual measured delivery rate. Note that the assigned guarantee, determined by the linear program solution, may be higher than the specified guarantee.

5.1 All ON/OFF Traffic

In this experiment, 21 ON/OFF flows are multiplexed using the RPM algorithm. All flows are comprised of constant-length 1000 byte packets. Flows 1 through 20 have a high bandwidth state of 1 Mbps with an average holding time of 2 seconds, and a low bandwidth state of 100 Kbps with an average holding time of 6 seconds. The holding times are exponentially distributed yielding state occupancy probabilities of 0.25 and 0.75 respectively. Flow 21 has a high bandwidth state of 5 Mbps with an average holding time of 2 seconds and a low bandwidth state of 500 Kbps with an average holding time of 3 seconds (also exponentially distributed). The peak bandwidth requirement is 25 Mbps, well over the capacity of the 10 Mbps link on the output of the FreeBSD router so packet loss can be expected. Flow 21 is designated a high-priority flow with a guarantee of 99%, while the other flows are assigned guarantees of at least 88% (the actual guarantees are assigned by the admission control algorithm).

Results for a 1000 second run of the synthetic traffic in an RPM router and a non-RPM router (best-effort) are shown in Table 1. The first column is the ID number of the flow, the second and third columns are the total number of kilo-bytes sent and dropped for each flow under RPM, the fourth column is the probabilistic guarantee for the fraction of packets delivered, and the fifth column is the actual fraction of packets delivered. Columns six through eight show the actual kilo-bytes transmitted and received, and the fraction of packets delivered in the best-effort case. Differences in transmitted bytes between the RPM case and the best-effort case are due to random variations in traffic. We can see that with RPM active, the target guarantee of 99.4% for flow 21 was satisfied, while it only received a packet delivery rate of 94.4% under best effort. Flows 1-20 sustained a slight drop in their packet delivery rate under RPM, but are still near or above the contracted level. This shows that RPM can be used to control which flows receive preferential use of resources.

5.2 High Priority Video with Low Priority ON/OFF Traffic

This experiment is similar to the previous experiment except that flow 21 is the actual vic video stream described above. Vic was run using the h.261 format at 30 fps with the highest quality setting and is modeled with a three state model having 1 Mbps, 2.7 Mbps and 5 Mbps states with occupancy probabilities of 0.55, 0.37 and 0.08 respectively. The synthetic ON/OFF flows have high bandwidth states of 1050 Kbps with average

	RPM				best-effort		
Flow	kb sent	kb drp.	asng. delv.	meas. delv.	kb sent	kb drp.	meas. delv.
1	55992	42	99.9%	99.9%	56210	1864	96.8%
2	64593	65	99.9%	99.9%	45864	1676	96.5%
3	51591	58	99.9%	99.9%	61893	1774	97.2%
4	73169	97	99.9%	99.9%	51292	2122	96.0%
5	57507	211	99.9%	99.6%	47754	1668	96.6%
6	43624	97	99.9%	99.8%	50507	1527	97.1%
7	61075	348	99.9%	99.4%	50011	1168	97.7%
8	55286	439	99.9%	99.2%	46075	1479	96.9%
9	53472	818	99.9%	98.5%	51177	1443	97.3%
10	62110	1349	99.9%	97.9%	54890	2167	96.2%
11	57961	1355	99.9%	97.7%	59055	1646	97.3%
12	58808	2000	99.9%	96.7%	55757	1333	97.7%
13	57102	3157	99.9%	94.8%	48234	1852	96.3%
14	58005	3326	99.8%	94.6%	41572	1271	97.0%
15	50873	6271	99.6%	89.0%	53204	1763	96.8%
16	38117	3742	99.3%	91.1%	58219	1580	97.4%
17	48667	7545	98.8%	86.6%	59400	1428	97.7%
18	42970	5395	95.9%	88.9%	54320	1762	96.9%
19	48578	5610	94.0%	89.7%	46639	1530	96.8%
20	40958	9851	91.6%	80.6%	59277	2346	96.2%
21	326078	17938	81.8%	94.8%	349140	17771	95.2%

Table 3: Low Priority VIC and High Priority ON/OFF

holding times of 7 seconds and low bandwidth states of 100 Kbps with average holding times of 21 seconds. The results for the RPM and best-effort case are shown in Table 2. We can see that the Vic session (Flow 21), achieved a measured guarantee of 99.7%, very close to its target guarantee of 99.9%, the other flows are also close to their assigned guarantees. Note that it is possible for a flow to receive less than the contracted guarantee as the contracted guarantee represents an expected value over an infinite duration flow.

5.3 Low Priority Video with High Priority ON/OFF Traffic

In this experiment, we use the same flow characteristics as in the previous experiment, but specify 99.9% guarantees for flows 1 through 20, and an 80% guarantee for the video flow. The experimental results are shown in Table 3. We see that RPM was able to move resources to the high priority flows and that most flows have measured delivery rates close to the guaranteed values.

5.4 Scalability Experiment

The scalability of RPM algorithm is demonstrated by mixing the Vic video with 200 competing ON/OFF synthetic flows. The ON/OFF flows have a high bandwidth state of 64 Kbps with an average holding time of 10 seconds, and a low bandwidth state of 6 Kbps with an average holding time of 3 seconds. This is similar to typical telephone traffic. An assigned guarantee of 99% was given to the vic flow yielding an actual measured delivery rate of 99.6% compared to 75.2% in the best-effort case.

6 Conclusion

In this paper a new algorithm for probabilistic bandwidth reservation was presented. It was shown that probabilistic guarantees can be provided with fine grain control over the guarantees provided to individual flows. It was also shown that the method could be scaled to large numbers of flows. By adding policing mechanisms such as those in [3][6], the techniques shown here can be applied to adversarial flows as well as conforming flows.

References

[1] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, "RSVP: A New Resource ReSerVation Protocol", From *IEEE Network*, vol. 7, no. 5, p. 8-18, September 1993

[2] P. Chandra, A. Fisher, C. Kosak, T. S. Eugene Ng, P. Steenkiste, E. Takahashi, H. Zhang, "Darwin: Resource Management for Value-Added Customizable Network Service", From *6th IEEE International Conference on Network Protocols (ICNP'98)*, October 1998

[3] M. Salamah, H. Lababidi, "BLLB: A Novel Traffic Policing Mechanism for ATM Networks", From *8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, p. 411-415, August 2000

[4] I. Elhanany, D. Sadot, "A Prioritized Packet Scheduling Architecture for Provision of Quality-of-Service in Tbit/sec WDM Networks", From *Proceedings of IEEE International Conference on Communications*, pg. 696-700, June 2000

[5] E.W. Knightly, "H-BIND: A New Approach to Providing Statistical Performance Guarantees to VBR Traffic", From *Proceedings IEEE INFOCOM '96 Conference on Computer Communications*, pg. 1091-1099, March 1996

[6] E.W. Knightly, "Enforceable Quality of Service Guarantees for Bursty Traffic Streams", From *Proceedings IEEE INFOCOM '98 Conference on Computer Communications*, pg. 635-642, March 1998

[7] C. Lee, J.P. Lehoczky, R. Rajkumar, J.P. Hansen, "A Scalable Solution to the Multi-Resource QoS Problem", From *Proceedings of the 20th IEEE Real-Time Systems Symposium*, December 1999

[8] C.L. Hoover, J.P. Hansen, P. Koopman, S. Tamboli, "The Amaranth Framework: Policy-Based Quality of Service Management for High-Assurance Computing", Accepted for publication in *International Journal of Reliability, Quality, and Safety Engineering*, November 1999

[9] C.L. Hoover, J.P. Hansen, P. Koopman, S. Tamboli, "The Amaranth Framework: Probabilistic, Utility-Based Quality of Service Management for High-Assurance Computing", From *Proceedings of the 4th IEEE International High-Assurance Systems Engineering Symposium (HASE '99)*, November 1999

[10] J.P. Lehoczky, "Scheduling communication networks carrying real-time traffic", In *Proceedings Real-Time Systems Symposium*, pg. 470-479, December 1998

[11] UCB/LBNL Video Conferencing Tool, http://www-nrg.ee.lbl.gov/vic