

Fair Bandwidth Sharing Among Virtual Networks: A Capacity Resizing Approach

Rahul Garg
IBM India Research Lab,
Indian Institute of Technology, Delhi.
Email: grahul@in.ibm.com

Huzur Saran
Department of Computer Sc. and Engg.,
Indian Institute of Technology, Delhi.
Email: saran@cse.iitd.ernet.in

Abstract—Virtual Private Networks (VPN) and link sharing are cost effective way of realizing corporate intranets. Corporate intranets will increasingly have to provide Integrated Services for voice and multimedia traffic. Although packet scheduling algorithms can be used to implement Integrated Services in link sharing [1] and virtual private networks [2], their statistical multiplexing gains are limited. Recently capacity resizing of Virtual Leased Links (VLLs) has been proposed to obtain better statistical multiplexing gains. Carrying out capacity resizing naively may result into unfair distribution of capacities among different VPNs. This is a serious problem especially because the VPN customers who pay for the service expect a well defined grade of service and protection from other misbehaving customers sharing the same infrastructure.

We propose a new scheme called Stochastic Fair Sharing (SFS) to carry out fair link sharing and fair sharing among VLLs. In the link sharing environment, capacities allocated to different classes are adjusted dynamically as sessions arrive (or depart). The SFS admission control algorithm decides which sessions to accept and which to reject depending upon the current utilizations and provisioned capacities of the classes. SFS gives protection to classes with low session arrival rate against classes with high session arrival rates by ensuring them a low blocking probability.

In case of multi-hop VLLs, capacity resizing requests are sent in the service providers's network which are admission controlled using SFS. Our simulations indicate that using SFS, equivalent capacity of virtual links converge to their max-min fair capacity, with a fairness index of 0.97 in extreme situations. The average signaling load of the protocol was found to be reasonable.

The scheme is simple to implement, efficient, and robust. The potential applications of SFS are fair and efficient resource sharing in telecommunication networks, ATM networks, virtual private networks (VPN) and integrated services or differentiated services based IP networks.

Keywords: Link Sharing, Trunk Reservation, QoS, VPN, Loss Networks, Integrated Services Networks, SFS, Erlang's Formula, Call blocking probability.

I. INTRODUCTION

Link sharing [3] is an important concept using which service providers lease portions of their link to several organizations. To implement link sharing, the link capacity is statically partitioned into classes and assigned to the organizations. Once a link is partitioned, the capacity allocations remain fixed irrespective of the actual usage by the organizations. Thus, if one of the organization is not fully using its capacity, the free link capacity cannot be used by other organizations to accept new real-time sessions. For example, in telecommunication networks, once a service provider has leased a T1 line to an organization, it will not carry more than 24 calls of the organization through its line even if the trunk has required spare capacity. A better alternative is to fairly redistribute the free capacity by resizing (and possibly charge for it) depending upon the current usage of different organizations sharing the link.

In the natural extension of link sharing to virtual private networks (VPN) [4], instead of leasing bandwidth on a single link, the service provider leases bandwidth on a path to an organization. The organization tunnels its VPN traffic through this path as if it was a single *virtual link*. Such Virtual Leased Links (VLL) may be realized by tunnels [5] and label switched paths in IP networks, and virtual paths in ATM networks. The notion of fair sharing of residual capacity naturally maps to the max-min fair sharing [6] in the service provider's network¹.

Use of hierarchical packet fair queueing algorithms [1] to efficiently implement link sharing has been proposed in literature. These algorithms ensure that each traffic class gets its requisite share of the link capacity, and carry out fair redistribution of the residual unused bandwidth of a traffic class to other classes. The hierarchical packet fair queueing algorithms work at the scale of packet transmission time. Although they redistribute the free capacity of inactive classes to active classes, they cannot ensure, on their own that the redistributed capacity will be available for a predictable period of time. This residual capacity goes back to its respective class as soon as the sending rate of the class is increased. As a result, real-time sessions, which require a guaranteed Quality of Service (QoS) for their lifetime cannot utilize this excess capacity. Even the best effort sessions with minimum bandwidth requirement cannot get their minimum bandwidth guarantee from redistributed capacity. Therefore, capacity sharing using packet scheduling algorithms only, is limited to the *elastic* portion of the best effort traffic.

In order to support Integrated Services in VPNs, the multi-hop virtual links will have to provide tight bandwidth and delay guarantees to their traffic. We have shown in [2] that an inherent limitation of tunneling real-time traffic without explicit QoS marking, is loss of statistical multiplexing. The traffic cannot be sent on a virtual link at a rate higher than its allocated capacity, even if the underlying provider network has large spare capacity. Thus the resource sharing in provider's network cannot be achieved without dynamically changing the capacity of virtual links.

In [7] a dynamic resizing approach to resource management in VPNs has been proposed and is shown to result into significant (factor 1.5 to 3) statistical multiplexing gain. Similarly in [8] a pipe resizing approach for AT&T switched network is shown to result into a gain of 2. However the issue of fairness and protection has not been discussed in these ap-

This work was done when Rahul Garg was pursuing PhD at IIT Delhi.

¹In this paper we only discuss max-min fairness. However, the same scheme can be naturally extended to other revenue based measures of fairness

proaches. Without fairness guarantees, a few VLLs may take most of the bandwidth of the network resulting into starvation of other VLLs.

We propose a scheme called Stochastic Fair Sharing (SFS) to implement fair sharing among VLLs by dynamically modifying their capacities. Virtual link capacities are increased upon session arrivals and are decreased as sessions complete. As a result, the redistributed capacities, which are available for session lifetime, can be used by the real-time traffic as well as the best-effort traffic with minimum bandwidth requirement. To ensure fairness and protection, a SFS admission control algorithm decides whether increasing virtual link capacity is acceptable. If session arrivals are random and non bursty, and session holding times are small, then most of the virtual link capacities quickly converge close to their fair share.

In case of a single link, the proposed SFS technique maps to simply extending the admission control algorithm at the link. The SFS admission control algorithm sorts the classes in increasing order of their normalized usage (current reservation divided by allocated capacity). A new session of a class is accepted only if the free capacity after accepting the session is greater than or equal to the sum of trunk reservation of classes with lower normalized usage. For simplicity of exposition, we assume that QoS requirement of a session is expressed using a single bandwidth parameter². Once a session is accepted, the corresponding weights in underlying hierarchical fair packet scheduler (if any) are adjusted to reflect the change. In this way, this scheme attempts to equalize the normalized usage of classes by giving higher priority to classes with low normalized usage. Thus, SFS accounts for long term fluctuations in usage by carrying out capacity redistribution over time scale of session holding times, while the underlying hierarchical packet fair queueing algorithms handle short term fluctuations in traffic load by redistributing free capacities over the scale of packet transmission times. SFS ensures that while the free capacity of a class has been redistributed to other classes, it has low session blocking probability and can quickly regain its share of capacity as its session arrival rate increases.

In case of a multi-hop virtual links of virtual networks, the current capacity of a virtual link may be changed by sending an *increase* or *decrease* request in the provider's network. The increase request is admission controlled using SFS at each link of the provider's network. After processing these requests, the corresponding scheduling weights in provider network are adjusted accordingly. In case of overload, the provider network generates *reduce* messages to request down-sizing of over-allocated virtual link capacities. With this scheme we show that, the unbottlenecked virtual links achieve throughput equal to their current demand, and the equivalent capacity of 94% of bottlenecked virtual links is within 90 – 105% of their weighted max-min fair capacity (weighted by provisioned capacity).

SFS can be used by service providers to ensure that bandwidth distribution is fair even in case of high and uneven load. SFS provides fairness by ensuring that every virtual link can in-

crease its capacity up to its fair share within an interval of two mean session holding time. This also ensures that well behaving customers are protected from misbehaving ones (those who asked for low capacity paths and resize it to higher capacities).

A similar scheme to carry out *virtual partitioning* of a single link has been proposed in [9]. This scheme categorizes each class as underloaded or overloaded. The underloaded classes are given priority over the overloaded classes using the technique of trunk reservation. While this scheme ensures that the residual capacity of a class may be utilized by other classes, it does not attempt to do a fair redistribution of this free capacity. While using virtual partitioning, a class having a high session arrival rate may take the residual capacity of all the classes. The hierarchical virtual partitioning [10] suffers from the same drawback and is not applicable to multi-hop virtual links.

The proposed SFS scheme is simple, robust and can be implemented in large networks without significant overheads. In our simulations, the bandwidth penalty for using SFS was less than 2% and its average signaling load was less than 100 messages per second per router.

SFS may be used in telecommunication networks to achieve better sharing and lower call blocking probabilities. It may be used in the context of ATM networks to dynamically carry out fair bandwidth allocation to virtual paths [11]. Use of SFS to carry out dynamic bandwidth allocation to virtual links in a virtual network may result into better network utilization and fair resource sharing. In the context of differentiated services [12] in IP based networks, SFS may be integrated with the proposed bandwidth broker architecture to achieve fair sharing of portions of network among several administrative domains.

This paper is organized as follows. We describe our model in Section II and the SFS admission control in Section III. We discuss the case of multi-hop virtual links in Section IV. Single link simulations of SFS are discussed in Section V and simulations of SFS on a network are discussed in Section VI. Conclusions are presented in Section VII.

II. MODEL

We first describe the single link case. We assume that a link of capacity C is to be partitioned into N logical links (or classes) of capacities C_i , such that $\sum_{i=1}^N C_i \leq C$. We assume that real-time sessions with bandwidth requirement arrive randomly as a stochastic process. Bandwidth is reserved upon session arrivals and is released upon session completion (using a protocol like RSVP [13], PNNI [14] or OpeNet [15]).

There is an admission control entity at the link which decides whether the link has adequate free capacity to accept the reservation requests of sessions. In case the reservation request of a session can't be accepted, we say that the session is blocked. An important performance parameter, in the context of telecommunication networks is the call blocking probability. In these networks, the link capacities are provisioned to keep the call blocking probabilities below a specified value (say 0.02). The session blocking probabilities of sessions having different bandwidth requirement may be different. Therefore it is more appropriate to analyze the bandwidth blocking probability [16], which is the weighted average of session blocking probability weighted by session bandwidth. The other performance parameter is average

²For VBR traffic the concept of equivalent bandwidth may be used, for delay sensitive traffic the delay parameter may be translated to a bandwidth parameter (assuming that rate allocating schedulers are used at this link), and for best-effort traffic the minimum bandwidth requirement of the session may be used.

throughput.

Let r_i denote the current bandwidth reservation of logical link i . If the link was statically partitioned, then the admission control procedure could simply check if sum of current reservation (r_i) and the reservation requested by new session (r), is less than or equal to the logical link capacity (C_i). The two main drawbacks of static partitioning approach are inability to do capacity sharing and loss of statistical multiplexing.

Given a partitioning of a link into several logical links, the Stochastic Fair Sharing (SFS) admission control decides which session reservation requests to accept and which to block. The decision process is designed to achieve the following³:

Partitioning The link capacity is partitioned into logical links.

Fair Sharing If some of the logical links are lightly loaded, their free capacity should be fairly redistributed to other logical links. The notion of fairness used in packet scheduling algorithms [18] can be generalized as follows. The average throughput of logical links which have blocking probability greater than a threshold (say 0.05) should be proportional to their equivalent capacity.

Statistical Multiplexing Since the traffic of all the logical links is carried on the same physical link, the bandwidth blocking probabilities while using SFS should be less as compared to static partitioning.

Isolation For sessions of a logical link it should appear as if the logical link capacity has been dedicated to them. A high session arrival rate on other logical links should not significantly increase the steady state bandwidth blocking probability (or reduce average throughput) of the logical link. The bandwidth blocking probabilities of a logical link using SFS should be comparable to that using static partitioning in the worst conditions.

If all the sessions require equal bandwidth and session arrival rate is a Poisson process and session holding time is exponentially distributed, then the session blocking probability for a link can be obtained using the well known Erlang's formula [19], [20].

$$E(\lambda, C) = \frac{\lambda^C / C!}{\sum_{i=0}^C \lambda^i / i!}$$

where λ is the mean session arrival rate multiplied by the mean session holding time (λ is also called arrival rate in Erlangs) and C is the link capacity (in units of number of sessions it can carry). Note that as λ and C are increased in equal proportion, the blocking probability is reduced because of better statistical multiplexing.

Let Th_i and p_i be the mean throughput and blocking probability on logical link i . The mean arrival rate λ_i on the logical link is given by $\lambda_i = \frac{Th_i}{1-p_i}$. The equivalent capacity (C_i^e) of the logical link is defined as the capacity at which the arrival rate $\frac{Th_i}{1-p_i}$ results into a blocking probability of p_i .

$$C_i^e = C : E\left(\frac{Th_i}{1-p_i}, C\right) = p_i \quad (1)$$

Note that C_i^e is unique for a given p_i and Th_i and is always greater than Th_i . As p_i approaches 1, C_i^e approaches Th_i .

³In this paper we discuss stochastic fair sharing in the context of link sharing, but the technique is generic enough to achieve fair sharing of a resource in a loss network [17].

According to the fairness criteria, the equivalent capacity of logical links with significant blocking probability ($p_i > p_t$, $p_t = 0.05$ say), should be proportional to their provisioned capacities (C_i). Formally,

$$\frac{C_i^e}{C_i} = \frac{C_j^e}{C_j} \forall i, j : p_i, p_j > p_t \quad (2)$$

The term $\frac{C_i^e}{C_i}$ represents the factor by which the capacity of logical link i is increased because of unused capacity of other logical links. This quantity is always equal to 1 in case of static link partitioning, and is expected to be greater than one (representing the degree of sharing) for SFS. The variation of this quantity across the logical links (with $p_i \geq p_t$) represents a measure of unfairness of this sharing.

In a more general setting, the closed form expression for call blocking probability is very difficult to obtain and simulations are used for its estimation.

SFS achieves lower bandwidth blocking probability when the arrival rate is high, at the expense of slightly higher bandwidth blocking probability when the arrival rate is low. When the arrival rate at a logical link is low, some of its free capacity is fairly redistributed to other logical links, which marginally increases its bandwidth blocking probability. However, the links with heavy load experience lower bandwidth blocking probability because of better statistical multiplexing and redistributed capacity. As a result, the overall throughput is increased significantly.

III. THE SFS ADMISSION CONTROL

Let r_i be the amount of this capacity already reserved (read used) by logical link i . The normalized usage of logical link i is given by $n_i = r_i / c_i$. A logical link with normalized usage less than 1 is underutilized whereas a logical link with normalized usage greater than 1 is using more than its allocated capacity. If the given link is statically partitioned, n_i is always less than or equal to 1. In case of stochastic fair sharing, if some logical link is underutilized, other logical links may make use of its unused capacity and have a normalized usage of greater than 1.

Let the logical links be relabelled in increasing order of their normalized usage (link 1 has lowest normalized usage and link N has the highest). The link with lowest normalized usage is given the highest priority. A new session for logical link 1 is accepted if the free capacity of the physical link is greater than or equal to the bandwidth requirement of the session. A new session for logical link 2 (which has the second lowest normalized usage) is accepted only if enough free capacity (t_1) would be left in the physical link, for logical link 1 after accepting the session. This reserved capacity is called the trunk reservation for logical link 1 (a term adopted from telecommunications networks) [20]. In general, a session of logical link i is accepted if the free capacity after accepting the session is at least equal to the sum of the trunk reservations of the logical links with lower normalized usage. Formally, a new session of logical link i , with bandwidth request of r is accepted if and only if:

$$\sum_{j=1}^N r_j + r + \sum_{j < i} t_j \leq C \quad (3)$$

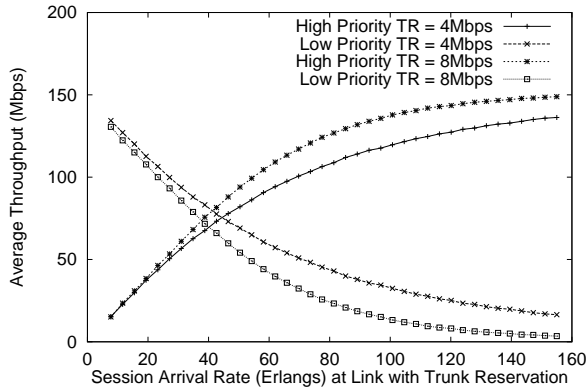


Fig. 1. Average throughput of two links in presence of trunk reservation

A session from the most heavily loaded logical link is accepted only if $\sum_{j < N} t_j$ capacity remains free after accepting this session. This scheme gives higher priority to sessions of logical links having low normalized usage and thus attempts to equalize the normalized usage of different logical links.

It has been shown in the context of telecommunications networks that even a small amount of trunk reservation gives almost absolute priority to one class of calls over the other [21]. If there are only two types of calls arriving on a link and certain amount of trunk reservation (equivalent to that needed by 1-10 calls) is set aside for high priority calls, then as call arrival rate increases, the high priority calls start occupying most of the link. However, if the arrival rate of high priority calls is not large enough to saturate the link, the low priority calls occupy rest of the link. This is shown in Figure 1. Sessions with bandwidth requirement of 2Mbps and mean holding time of three minutes arrive at a link of 155Mbps. There is a trunk reservation for high priority sessions. The arrival rate of low priority sessions is kept high at 310 Erlangs and the arrival rate of high priority sessions is varied. The graph shows that even small amount (4 Mbps) of trunk reservation gives almost absolute priority to high priority sessions.

In the proposed SFS approach, sessions in the link having the least normalized usage are given priority over rest of the sessions. Therefore, its usage is expected to increase. As soon as its usage becomes larger than that of next link, its priority is reduced. Similarly, the link having the highest normalized usage is given the lowest priority, allowing it to only make use of the leftover capacity of all other logical links. As a result, this approach attempts to equalize the normalized usage of the logical links.

Consider a link partitioned into two logical links of equal capacity. The state of the link at any given time is represented by the current bandwidth reservation of the two logical links. Figure 2(a) shown the state space diagram of the system. The x and y axis represent the current reservation of the first and the second logical link respectively. The current state of the link can be represented by a point in the state space diagram. The shaded region represents the possible states where the system may be present at any given time. Clearly, the sum of the reservations can not be more than the link capacity as shown by the negatively sloped line on the right.

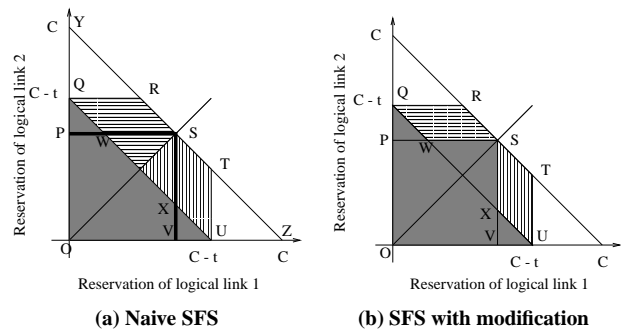


Fig. 2. State space diagram for a link

When a new session on first logical link is accepted, the system moves to the right in the state space diagram. Similarly, upon acceptance of a new session on second logical link, the system moves up. When sessions of first or second logical link complete, the system moves towards left or downwards. Assume that trunk reservation for both the links is same. As long the total free capacity is more than the trunk reservation, the system may move in any direction (assuming that the bandwidth requirement of sessions is small). The area shaded as gray represents this region. In the region where the available capacity is less than the trunk reservation (the area between the two slanting lines), session of the first logical link can be accepted only if the current reservation of the first logical is less than that of the second logical. Thus, the system can move left, right or down but not upwards. This region is represented by horizontal shading. Similarly, the region shaded by vertical lines represent state space where the system can freely move upwards but not towards the right.

If the link was statically partitioned, then its state space is limited to the rectangular region OPSV and is free to move in any direction in this region. If there is no partitioning, then the state space of the system would be the region OYZ and the system would be free to move in any direction in this region. The stochastic fair sharing approach is between these two extreme cases.

Note that WSX region of the state space is characterized by roughly equal and moderate to high load on both the logical links. In static partitioning approach, the system is free to move in either direction in this region, whereas in SFS approach its movement is restricted. This may result into higher blocking probabilities for SFS if the system is expected to be in this region most of the time. Thus the performance of the SFS approach may become worse than that of static partitioning. The situation is worse if the trunk reservation is more than half the link capacity. We therefore propose a minor modification to our basic scheme described earlier.

If the normalized usage of a logical link is close to its fair share, then it is not necessary to have a large value of trunk reservation for the logical link. In this case, we reduce the trunk reservation of the logical link. Logical link i has a static trunk reservation parameter \hat{t}_i which is the maximum value of trunk reservation for the link. The fair share (f_i) of capacity for each logical link is dynamically computed. If the difference between the fair share and current reservation of a logical link is less than its static trunk reservation, then trunk reservation for the link is

set to the difference. Otherwise the trunk reservation is set to the static value. Formally, $t_i = \min[\hat{t}_i, f_i - r_i]$.

The fair share of logical link 1 is given by $f_1 = \frac{C_1}{\sum_{j=1}^N C_j} C$.

The fair share of the other logical link is computed by redistributing the free capacity of logical links with lower normalized usage as follows:

$$f_i = \frac{C_i}{\sum_{j=i}^N C_j} (C - \sum_{j=0}^{i-1} (r_j + t_j))$$

The above expression is a natural extension of the fairness criteria [18] used in packet schedulers. The first term of the product in the above expression represents the fraction of residual capacity which should be given to logical link i . The second term represents the residual capacity after taking out the usage of logical links which are more lightly loaded than logical link i . Thus, if the usage of logical link is less than its fair share, then the free capacity is fairly redistributed among heavily loaded links after keeping aside a small trunk reservation.

The state space diagram for this approach is shown in Figure 2(b). Now the system is free to move towards achieving utilization of its entire capacity as long as normalized usage of both the logical links is close to each other.

Note that the goals of isolation and sharing are somewhat contradictory in nature. For perfect isolation, the link has to be statically partitioned, resulting into no sharing. The static trunk reservation parameter (simply called trunk reservation in rest of the paper) can be used to balance isolation and sharing. If the trunk reservation is set to zero, then the entire link capacity is shared and there is no isolation. As the trunk reservation is increased, the sharing is reduced and isolation is improved. We have derived an approximate Markov chain based model of the system which we used to determine the trunk reservation parameter. Due to space constraints, we omit the model in this paper.

IV. SFS FOR MULTI-HOP VIRTUAL LINKS

In case of virtual networks, the virtual links are realized by paths in the service provider's network. Each virtual link has a provisioned capacity allocated at the time of establishment of the link. This provisioned capacity is determined based on the long term expectation of the traffic in the virtual network and is included in the service level agreement (SLA) with the provider. In addition, each virtual link also has a current capacity which can be changed by sending *increase* or *decrease* requests in the provider network. These requests travel along the path of the virtual link. At each node in the provider network, SFS state corresponding to each virtual link traversing the node is maintained. Variable r_i of SFS state represents the current capacity of virtual link i and C_i represents its provisioned capacity.

Upon the receipt of an increase request of amount r , SFS admission control test (Eq. 3) is evoked to decide if the request can be accepted. If the test succeeds, SFS state at the node is updated (r_i is increased), and the request is forwarded to the next node. If the test fails, a failure indication is sent back along the same path. While this failure indication travels back, each link in the path decreases its respective r_i . If the increase request reaches the other end of the virtual link, a success message is sent back. While this message travels the reverse path,

the scheduler weights are adjusted. For a decrease request, r_i 's are decreased and scheduler weight are adjusted in the forward path.

A virtual link could simply send an increase or decrease request in the provider's network upon session arrival or completion. This could flood the provider's network with signaling messages. To limit the signaling load, we set a minimum step size (C_{min}) by which the virtual link capacities can be changed. In order to prevent rapid oscillations in virtual link capacity around integer multiples of C_{min} we introduce hysteresis in sending increase and decrease requests. The decrease request is sent only if free virtual link capacity is at least $1.5C_{min}$.

The minimum step size restriction creates the following problem. The SFS algorithm for single link critically relies on the fact that session holding times are finite. If the usage of a logical link is above its fair share (because of low traffic on other logical links in the recent past), new sessions on that logical links will not be accepted, and ongoing sessions will eventually complete and release the bandwidth. However, if the minimum step size is fixed (say C_{min}) and bandwidth requirements of sessions is small as compared to C_{min} , then, instead of a decrease request, a session completion will generate some free capacity in the virtual link. At this stage, if a new session arrives it can be accepted without going through SFS admission control. Thus the virtual links may not automatically reduce their capacity even while they are operating at usage higher than their fair share.

To counter this problem, we introduce a network initiated *reduce* message. After processing an increase request at a link, the node finds out virtual links using more than their fair share ($i : \sum_{j=1}^N r_j + \sum_{j=1}^{i-1} t_j > C$). It then sends reduce request to these virtual links. After introducing hysteresis, the condition to generate reduce request for virtual link i becomes ($\sum_{j=1}^N r_j + \sum_{j=1}^{i-1} t_j > C + C_{min}$). Upon the receipt of reduce request, the virtual link stops admitting new sessions until it sends a decrease request. This ensures that capacity can be quickly reclaimed from virtual links using more than their fair share.

Finally, some virtual links may still overwhelm the provider network by quickly and repeatedly sending increase requests most of which fail. We therefore impose a constraint that after an increase request fails, a virtual link is not allowed to send another increase request for a fixed interval (taken as 1 sec. for simulations).

The key ideas needed to extend SFS to multi-hop virtual links are summarized as follows.

- *Increase* and *decrease* requests in provider network to dynamically resize virtual link capacity.
- Use of SFS admission control at each provider link for *increase* requests.
- A minimum resize capacity to limit signaling load.
- Network initiated *reduce* request to signal highly loaded virtual links to cut down their traffic.
- Hysteresis in algorithms generating *increase*, *decrease* and *reduce* requests.

V. SINGLE LINK SIMULATIONS

SFS critically relies on the assumptions that (a) individual session bandwidth requirement is small as compared to the link

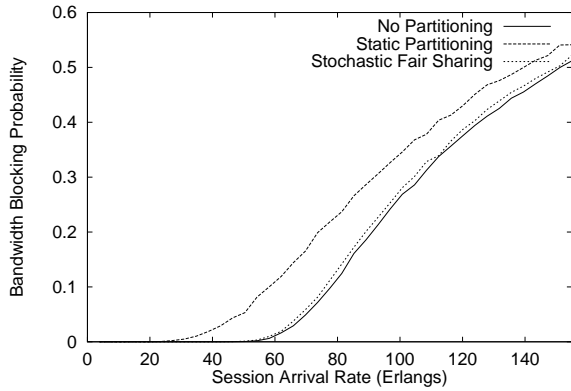


Fig. 3. Bandwidth blocking probabilities under same arrival rate on all logical links

capacity, (b) most of the sessions have small holding time, and (c) the session arrival process is not very bursty.

Accurate models for the arrival process of sessions with QoS requirements are not available. The only models available are for call arrivals in telecommunications networks where researchers have extensively used Poisson call arrival model (with arrival rate as a function of time of the day) with exponential holding time (with mean of about 3 minutes). The assumptions required for SFS are definitely true in telecommunications networks.

These assumptions also seem reasonable for future Integrated Services Networks. For instance, the bandwidth requirement of MPEG video stream is between 1–6 Mbps, which is reasonably small as compared to link speeds of 155 Mbps to 2.4 Gbps. The average throughput of a typical web connection is also small (4 Kbps to 128 Kbps) [22] as compared to the link speeds. Though the holding time for web connections is shown to be heavy tailed [23], most of the connections are still short-lived. The session arrival process is also not as bursty as the data arrival process.

We therefore simulated using the most simplistic model. We assume that sessions arrive as a Poisson process having exponentially distributed holding time with a mean of three minutes. The bandwidth requirement of these sessions vary uniformly at random from 250Kbps to 3.750Mbps in increments of 250Kbps. The mean session bandwidth is 2Mbps. Though these assumptions do not accurately model the session arrival process, we do expect them to give a glimpse into the SFS performance. We assume a OC3 link of 155 Mbps, partitioned into logical links.

A. Statistical Multiplexing

Figure 3 and 4 show how SFS can achieve more statistical multiplexing. The physical link is partitioned into five logical links. The session arrival rate in each of the link is kept the same and the total session arrival rate is increased till 155 Erlangs (two times the rate required to saturate the link). The bandwidth blocking probability, which is same for each logical link, is measured and plotted for different schemes. “No partitioning” refers to the scheme in which a new session is accepted if the physical link has enough free capacity. In “Static partitioning”, a new session is accepted only if free capacity of its logical link is greater than or equal to its bandwidth requirement. The

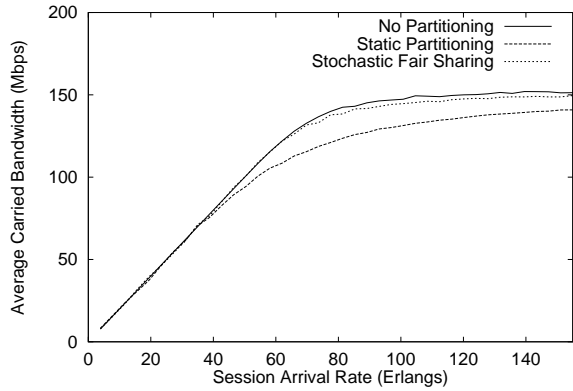


Fig. 4. Average throughput under same arrival rate on all logical links

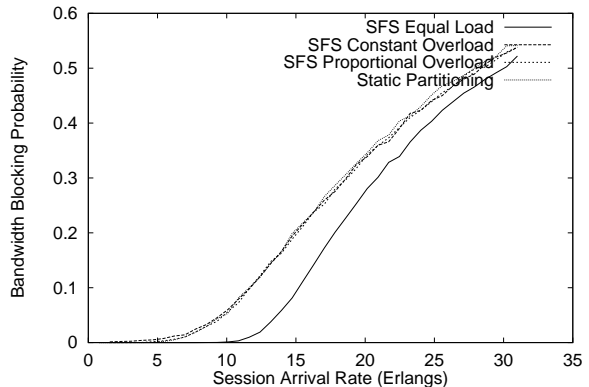


Fig. 5. Bandwidth blocking probability of lightly loaded link, when other links are overloaded

bandwidth blocking probability and the average throughput of SFS is close to that of the “no partitioning” approach which is the best achievable. Therefore, SFS regains most of the statistical multiplexing lost by static partitioning.

B. Isolation

The performance parameters of a logical link should not degrade in the presence of overload on other logical links. To see this, we have simulated two overload scenarios. We partition the physical link into five logical links and assign high session arrival rates on four of the links. The session arrival rate on other logical link (which is lightly loaded) varies from 0 at 31 Erlangs (two times the rate needed to saturate the logical link). In “constant overload” scenario, the session arrival rate on overloaded logical links was kept to 310 Erlangs, whereas in “proportional overload” scenario the arrival rate on the overloaded links was kept ten times the arrival rate on the lightly loaded link. If there is adequate isolation, the bandwidth blocking probabilities of the lightly loaded link would not be much worse than that of static partitioning approach.

Figure 5 plots the bandwidth blocking probability of lightly loaded link as its session arrival rate increases. Note that the blocking probability in presence of overload is very close to that of static partitioning. This shows that the SFS approach achieves isolation even in the presence of overload in other links. The same graph is shown in log scale in Figure 6. Note that

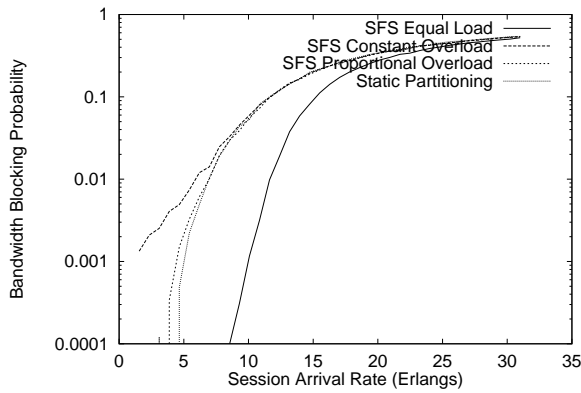


Fig. 6. Bandwidth blocking probability of lightly loaded link, when other links are overloaded

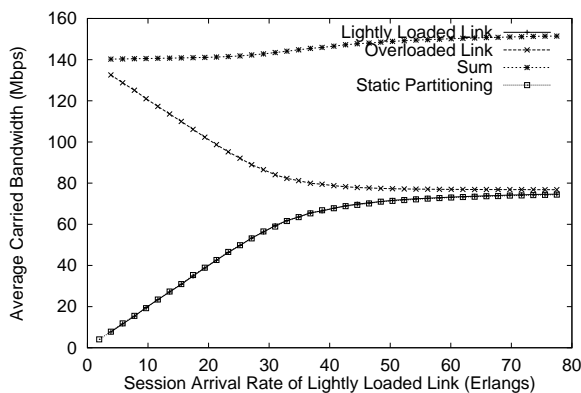


Fig. 7. Average throughput of the two logical links

when the arrival rates are very low, and the blocking probability is 0.01 (the target blocking probability) or less, then blocking probability of SFS approach does not reduce as fast as in static partitioning. In this range of operation, the free capacity of lightly loaded link is redistributed to overloaded links resulting into higher blocking probability. However, if the arrival rates on all the links is equal, then blocking probability is reduced.

C. Fair Sharing

In order to illustrate sharing, we assume that the physical link is divided into two logical links of equal capacity. Again, one of the link is lightly loaded whereas the other link is overloaded. The overloaded link has a constant session arrival rate of 310 Erlangs, which is eight times the rate required to saturate the physical link. The arrival rate in other logical link varies from 0 to 77.5 Erlangs. Figure 7 shows the variation in the average throughput as the arrival rate in lightly loaded link increases. At zero session arrival rate, most of the free capacity is used by the overloaded link. However as the session arrival rate increases, the average throughput of the lightly loaded increases. The graph showing average throughput of the lightly loaded link overlaps with the graph showing the same if static partitioning was used, indicating that there is virtually no loss of throughput at the lightly loaded link even if its capacity is redistributed to other links. The total average throughput while using static partitioning varies from 77 Mbps to 155 Mbps. The total av-

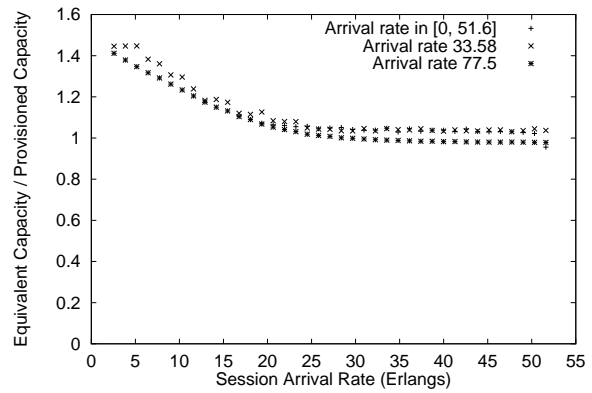


Fig. 8. Graphs showing fair sharing of residual capacity while using SFS

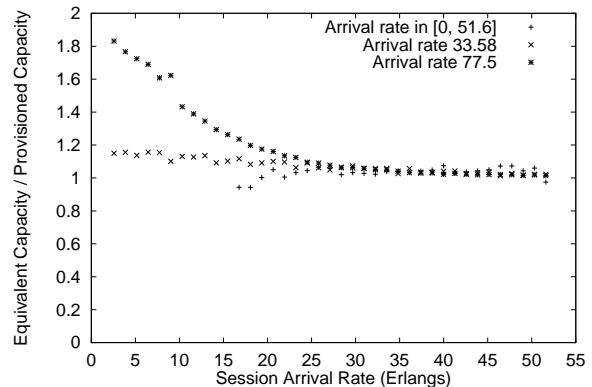


Fig. 9. Sharing of residual capacity while using virtual partitioning

erage throughput while using SFS increases from 140Mbps to 155Mbps, indicating that the trunk reservation of the lightly loaded link is also utilized as its arrival rate increases.

To demonstrate fairness of sharing, we assume that the physical link is partitioned into three logical links of equal capacity. One of the logical link is overloaded with an arrival rate of 77.5 sessions per minute (three times the nominal load), other link is loaded with an arrival rate of 33.58 Erlangs (1.3 times the nominal load) and the arrival rate at the third link is varied from 0 to 51.6 Erlangs (two times the nominal load). The ratio of equivalent capacity to provisioned capacity for links having blocking probability greater than 0.05 is plotted in Figure 8. The same graph is plotted for virtual partitioning scheme [9] in Figure 9. Points with zero ratio indicate that the bandwidth blocking probability is smaller than 0.05. In case of virtual partitioning, the free capacity is taken almost entirely by the logical link having high arrival rate. When the arrival rate on one of the logical link is low, there is large free capacity. In this range, the equivalent capacity of the link having arrival rate of 77.5 Erlangs, is 1.8 times its provisioned capacity whereas the equivalent capacity of the other link with arrival rate 33.58 Erlangs is only 1.15 times its provisioned capacity. As the arrival rate on one of the link increases, the free capacity decreases and eventually becomes equal to zero. In this region of the graph, equivalent capacity of virtual links approach their provisioned capacity. The graphs for SFS show that the ratio of equivalent capacity to provisioned capacity is very close to each other for the entire range

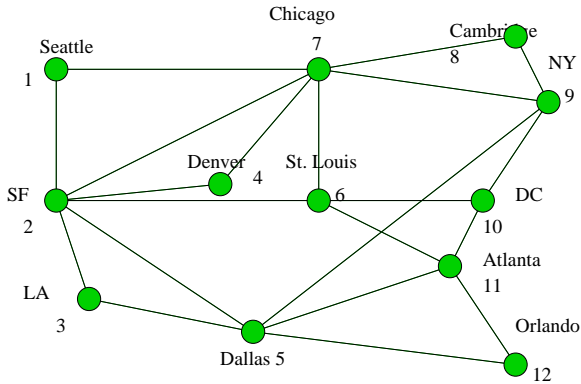


Fig. 10. Topology of simulated network

of arrival rate on one of the logical link. This indicates that SFS redistributes the residual capacity fairly among the logical links in accordance with our fairness criteria.

VI. SIMULATIONS FOR A NETWORK

We report the results for simulations carried out on twelve node approximate AT&T Worldnet topology (also used in [7]) as shown in Figure 10. Each link is assumed to be of capacity 2.4 Gbps and propagation delay 20ms. We setup a VLL between every pair of these nodes. Therefore, there are 132 unidirectional virtual links. A fixed trunk reservation of 10 Mbps and a minimum increase/decrease step size of 2 Mbps was used.

A. Max-min Fairness

We assume that all the 132 virtual links have same provisioned capacity. We assume, as in Section V, that sessions of rate uniformly distributed between 0.25 Mbps and 3.75 Mbps in increments of 0.25 Mbps, randomly arrive (as a Poisson process) at these virtual links. The session holding time is exponentially distributed with a mean of three minutes ($1/\mu = 3\text{min.}$). A load of 1 on a virtual link implies a session arrival rate of $f\mu/r$, where f is its max-min fair capacity, r is its mean session bitrate. In the (0.0-2.0) scenario, the load on each virtual link is set uniformly at random between 0 and 2. In the (0.5-1.5) scenario, the load on virtual links is set uniformly at random in the range 0.5 and 1.5. In the 5% misbehaving scenario, 5 percent of the virtual links misbehave with a load of 10. The other virtual links have a load of 1.

We compared SFS with the simplest first come first serve (FCFS) scheme. In FCFS, each virtual link sends increase and decrease requests as in SFS, but the admission control procedure in the provider network grants these requests in first come first serve order (i.e. as long as there is free capacity on the link).

Simulations were run for 2000 simulated seconds. In the end, each virtual link was classified as bottlenecked or non-bottlenecked, depending upon whether its session blocking probability was more than a given threshold ($p_{th} = 0.05$) or not. The average throughput of non-bottlenecked virtual links was subtracted from the link capacities in its path. For rest of the virtual links, weighted max-min fair share was computed using the residual capacities. Now, using the Erlang's formula, the throughput of non bottlenecked virtual links was adjusted up-

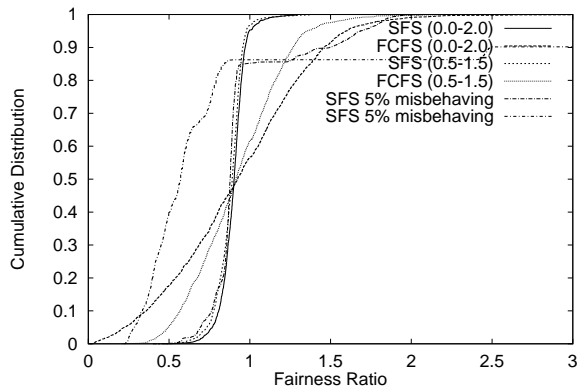


Fig. 11. Cumulative distribution of fairness ratio for multi-hop virtual links, with and without using SFS

wards, depending upon their session blocking probabilities, to reflect their equivalent capacity. The *fairness ratio* represents the ratio of equivalent capacity to max-min fair capacity. The experiment was repeated a number of times with different random seeds.

Figure 11 shows the cumulative distribution function of fairness ratio of all the virtual links for different loading scenario. From the figure, it is evident that SFS converges reasonably well to the weighted max-min fair share in the provider network. For 90% of the cases, the fairness ratio of SFS is between 0.75 and 1.00, for (0.0-2.0) and (0.5-1.5) scenarios. Note that the Erlang's formula is only applicable for constant bandwidth sessions. Therefore computing the equivalent capacity using Erlang's formula even in the presence of variable bandwidth sessions, introduces a small error. A significant portion of the spread in fairness ratio can be attributed to this error. If constant bandwidth sessions were used in simulations the fairness ratio remains between 0.90 and 1.05 for about 95% of the cases.

As the variation in load increases from (0.5-1.5) to (0.0-2.0), the spread in fairness ratio for FCFS increases, whereas for SFS this spread remains nearly the same. The fairness ratio ranges from 0.09 to 3.02 for FCFS and from 0.43 to 1.38 for SFS.

In case of 5% misbehaving virtual links, the FCFS allocates large capacities to the misbehaving virtual links, resulting into a significantly lower fairness ratio of most of the well behaving virtual links. In case of SFS, the fairness ratio of well behaving virtual links remains nearly same as before, indicating that they are well protected from the misbehaving virtual links. The range of fairness ratio for FCFS and SFS for this scenario are 0.22 to 5.23 and 0.40 to 2.08 respectively.

Another measure of fairness has been proposed in [24]. The fairness index is defined as $I = (\sum_{i=1}^n f_i)^2 / (n \sum_{i=1}^n f_i^2)$, where f_i is the fairness ratio of virtual link i and n is the number of virtual links. A fairness index of 1 implies perfect fairness and fairness index 0 implies gross unfairness. The ranges fairness index of SFS for the three scenarios are (0.989 – 0.997), (0.993 – 0.998), (0.963 – 0.974). These ranges for the FCFS are (0.938–0.985), (0.869–0.973), (0.644–0.658). According to this index of fairness, SFS achieves close to perfect fairness for all the scenarios, whereas FCFS performs significantly worse in the 5% misbehaving scenario.

| Signaling load (messages / sec) | Percentage of Low Bandwidth Traffic | | | |
|------------------------------------|-------------------------------------|--------|--------|--------|
| | 0 | 30 | 60 | 90 |
| Avg. (out) v-links | 1.36 | 1.33 | 1.33 | 1.60 |
| Max. (out) v-link | 5.66 | 6.00 | 6.50 | 8.42 |
| Avg. (VPN) v-link | 2.13 | 21.59 | 41.03 | 59.9 |
| Max. (VPN) v-link | 9.41 | 94.23 | 178.62 | 263.18 |
| Avg. over routers | 76.20 | 74.24 | 74.09 | 88.48 |
| Max. over routers | 120.37 | 118.06 | 118.57 | 144.71 |
| Addnl. setup latency (ms) | | | | |
| Avg. over v-links | 29.27 | 2.78 | 1.32 | 0.92 |
| Max over v-links | 76.77 | 7.63 | 4.13 | 2.96 |

TABLE I

SIGNALING LOAD AND ADDITIONAL CALL SETUP LATENCY FOR DIFFERENT MIX OF HIGH-BANDWIDTH AND LOW-BANDWIDTH TRAFFIC

All these results indicate that SFS achieves its primary objective of protection and max-min fair sharing in a generic network.

B. Signaling Load

Some fraction of low bandwidth traffic (64Kbps sessions) was mixed with the high bandwidth traffic, and the signaling load on virtual links and routers was measured. Table I shows the results for one simulation run of 2000 seconds. On each virtual link, the number of *increase*, *decrease* requests sent and *reduce* requests received was measured. The table indicates that the average rate of these signaling requests remains fairly constant in the range 1.33 to 1.60, even if the fraction of low bandwidth sessions in the VPN increases. The number of average VPN signaling requests processed by virtual links increases to up to 59.9 as the percentage of low bandwidth sessions increase. However, most of these requests are filtered by the virtual links and do not result into a signaling message in the provider network. The rate of signaling message in provider network remains fairly independent of the traffic mix in the VPN. The maximum signaling requests were usually generated by the Denver-Chicago (4-7) virtual link (between 5.66 and 8.14 messages per second for different traffic mix). The table indicates that the average signaling load on provider routers using SFS is also fairly constant over different traffic mix (between 74.09 and 88.48), and is quite manageable⁴. The maximum signaling load was on Chicago and SF routers (up to 144.71 messages per second) which also have maximum number of links connected. This indicates that using SFS in a large network is quite feasible and does not require a large signaling overhead for the increase, decrease and reduce requests.

C. Other Parameters

Use of SFS may introduce additional session setup latency in VPN because the setup request may trigger an increase request on virtual link and then the setup request has to wait till the response of increase request comes back. Table I also shows the average and maximum of additional session setup latencies. If the traffic mix consists of high bandwidth traffic only, then

⁴Note that SFS admission control is evoked on less than 1/4th of these messages (only on increase-fwd out of increase-fwd, increase-bwd, decrease-fwd, decrease-bwd and reduce).

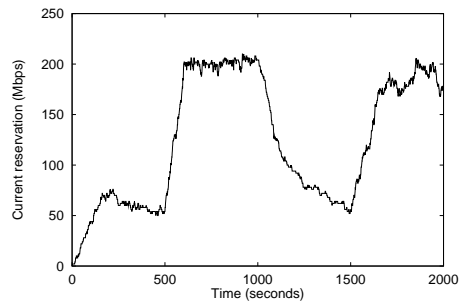


Fig. 12. Variation of current reservation on a virtual link, when arrival rate was changes at 500, 1000 and 1500 seconds

approximately two session arrivals on a virtual link result into one increase request. So, the additional setup latency is about half the round-trip time on virtual links. As the fraction of low bandwidth session increases, this setup latency goes down significantly. Only a very small fraction of low bandwidth sessions generate a increase request and therefore have low average latency.

Adaptability is another important parameter while using SFS in a network. What happens if session arrival rate on one of the virtual link changes? How quickly can SFS adapt to change in traffic patterns? Figure 12 gives the answers. It shows the variation of reserved bandwidth on Seattle-Orlando virtual link with time. We simulate with only high-bandwidth sessions in the network, and randomly change the session arrival rate on the Seattle-Orlando virtual link at time $T = 500, 1000$ and 1500 sec. The initial arrival rate on the virtual link was less than that needed to saturate its max-min fair capacity. Therefore, the reservation level steadily increased to 70 Mbps and then varied randomly till $T = 500$. The blocking probability was 0 in this interval. At $T = 500$ the arrival rate became large enough to saturate the max-min fair capacity. So the average reservation increases rapidly and stabilizes around max-min capacity of 200Mbps. The blocking probability is 0.3 in this interval. At $T = 1000$, the arrival rate becomes small again. Now the average reservation reduces exponentially to 56Mbps in till $T = 1500$. The blocking probability in this region is 0.025. At $T = 1500$, the arrival rate is increased again, but is insufficient to saturate the max-min capacity. Therefore the reservation varies between 170Mbps and 200Mbps. The blocking probability in this region is again 0. If the arrival rate is increased, it takes about 180 seconds (the mean call holding time) for the reservation level to reach close to its steady-state value. If the arrival rate is reduced, the decay in reservation is slow and exponential (which can be attributed entirely to the exponential holding time of sessions). Thus, the rate of adaptation of SFS primarily limited by the mean session holding time.

The trunk reservation used by SFS may lead to some bandwidth wastage. To estimate this, we measured the total number of bits carried while using SFS and compared it with FCFS (no partitioning, increase request served in FCFS order). The average throughput per virtual link of SFS for scenarios (0.5-1.5), (0.0-2.0) and "misbehaving" were 404 Mbps, 373 Mbps and 432 Mbps respectively. The average throughputs of FCFS for these scenarios were 409 Mbps, 379 Mbps and 429 Mbps respectively.

For “misbehaving” scenario, SFS outperforms FCFS whereas in the other two scenarios the average throughput of SFS is within 98% of FCFS. Thus, the bandwidth wastage in SFS is small as compared to the potential resizing gains (around 2 as reported in [7]), while the protection and fairness benefits are large which is essential for any resizing approach to work in practice.

VII. CONCLUSIONS

In this paper we presented a new scheme called stochastic fair sharing (SFS) to carry out fair link sharing and max-min fair sharing among virtual leased links of VPNs. We argue that fair scheduling algorithms like (WFQ) carry out excess capacity redistribution over time-scales of packet transmission time and therefore fair sharing gains can only be utilized by “elastic” non real-time traffic. In case of virtual links of virtual networks, even the non real-time traffic may not be able to make full use of all the free capacity available on physical links. The proposed SFS scheme carries out fair sharing by dynamically resizing virtual link capacity at the granularity of session arrivals. The redistributed capacity is available for session lifetime, and thus the real-time sessions can make use of it by appropriately adjusting the weights in the underlying schedulers.

As a result of sharing, the unused capacity of lightly loaded virtual links is redistributed to heavily loaded virtual links, resulting into higher aggregate throughput (and possibly higher revenues). This capacity redistribution is carried out in such a way that blocking probability of lightly loaded logical link is not significantly affected. Therefore, the lightly loaded logical links can quickly regain their share of link capacity if their session arrival rate is increased. This ensures adequate protection to lightly loaded virtual links, which enables them to release their unused capacity for reallocation by the network.

The key components of the SFS scheme are SFS admission control, which decides whether a capacity resizing request should be accepted, the algorithm to generate explicit reduce request to request down-sizing of some virtual links, and the network level signaling protocol which carries out capacity resizing with minimal overheads.

Using simulations, we illustrate the sharing, statistical multiplexing and isolation (protection) properties of SFS when used for link sharing. SFS results into improved throughput, and lower blocking probabilities because of better statistical multiplexing and sharing. Simulations of SFS on a network indicate that it achieves max-min fair sharing. 94 percent of virtual links achieve an equivalent capacity within -10 to $+5$ percent of their max-min fair capacities. The signaling load to implement SFS is shown to be small (between 75 and 90 messages per second per router) and relatively insensitive to the traffic mix. The bandwidth penalty for using SFS was found to be less than 2%.

SFS is ideal for achieving fair link sharing in telecommunication networks and ATM networks. It may be used to carry out dynamic bandwidth allocation of virtual paths in ATM networks. In addition to achieving fair link sharing, SFS may be used in virtual networks to carry out dynamic bandwidth allocation of virtual links. Thus, SFS can provide fair resource sharing in virtual networks. In the context of differentiated service in IP based networks, SFS may be integrated with bandwidth brokers to implement fair resource sharing in networks. The SFS tech-

nique is generic enough to carry fair sharing of resource in a generic loss network. A simple extension of SFS can carry out the resource partitioning hierarchically.

REFERENCES

- [1] Hui Zhang and Jon C. R. Bennett, “Hierarchical packet fair queueing algorithms,” in *Proceedings of ACM SIGCOMM*. ACM, September 1997, pp. 143–156.
- [2] Rahul Garg and Huzur Saran, “Scheduling algorithms for bounded delay service in virtual networks,” Jan. 1999, Submitted for publication. Available at <http://www.cse.iitd.ernet.in/rahul/mypapers/vpn-paper.ps>.
- [3] Sally Floyd and Van Jacobson, “Link sharing and resource management models for packet networks,” *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, August 1995.
- [4] B. Gleeson, A. Lin, J. Heinanen, and G. Armitage, “A framework for IP based virtual private networks,” Sept. 1998, Internet Draft <draft-gleeson-vpn-framework-00.txt>.
- [5] W. Simpson, “IP in IP tunneling,” Oct. 1995, RFC 1853.
- [6] Jeffrey M. Jaffe, “Bottleneck flow control,” *IEEE Transactions on Communications*, vol. COM-29, no. 7, pp. 954–961, July 1981.
- [7] N. G. Duffield, Pawan Goyal, Albert Greenberg, K. K. Ramakrishnan, and Jacobs E. van der Merwe, “A flexible model for resource management in virtual private networks,” in *Proceedings of SIGCOMM*, Aug. 1999.
- [8] Pawan Goyal, Albery Greenberg, Chuck Kalmanek, Bill Marshal, Partho P. Mishra, Doug Nortz, and K. K. Ramakrishnan, “Integration of call signaling and resource management for IP telephony,” *IEEE Network*, vol. 13, no. 3, pp. 24–32, May-June 1999.
- [9] Sem C. Borst and Debasis Mitra, “Virtual partitioning for robust resource sharing: Computational techniques for heterogeneous traffic,” *IEEE Journal on Selected Areas of Communications*, vol. 16, no. 5, pp. 668–678, June 1998.
- [10] Debasis Mitra and Ilze Ziedins, “Hierarchical virtual partitioning: Algorithms for virtual private networking,” in *IEEE Global Telecommunications Conference*, 1997, pp. 1784–1791.
- [11] Ariel Orda, Giovanni Pacifici, and Dimitrios Pendarakis, “An adaptive virtual path allocation policy for broadband networks,” in *Proceedings of INFOCOM*, San Francisco, USA, Mar. 1996, vol. 1, pp. 329–336.
- [12] Steven Blake, David Black, Mark Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss, “An architecture for differentiated services,” Oct. 1998, Internet Draft <draft-ietf-diffserv-arch-02.txt>.
- [13] Lixia Zhang, S. Deering, D. Estrin, S. Shenker, et al., “RSVP: A new resource reservation protocol,” *IEEE Network*, vol. 7, no. 5, pp. 8–18, September 1993.
- [14] “Private network to network interface specification 1.0,” ATM Forum, Prentice Hall, Mar. 1996.
- [15] I. Cidon, A. Gupta, T. Hsiao, A. Khamisy, A. Parekh, R. Rom, and M. Sidi, “OPENET: an open and efficient control platform for ATM networks,” in *Proceedings of INFOCOM*, San Francisco, USA, March-April 1998.
- [16] A.S. Maunder and P.S. Min, “Investigation of rate control in routing policies for B-ISDN networks,” in *International Teletraffic Congress - ITC 15*, June 1997.
- [17] Frank P. Kelly, *Loss networks*, University of Cambridge, Cambridge, England, 1989.
- [18] A. K. Parekh and Robert G. Gallager, “A generalized processor sharing approach to flow control - the single node case,” in *Proceedings of INFOCOM*. IEEE, May 1992, pp. 915–924.
- [19] Dimitri P. Bertsekas and Robert G. Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, NJ, USA, 1992.
- [20] Richard J. Gibbens and Frank P. Kelly, “Network programming methods for loss networks,” *IEEE Journal on Selected Areas in Communications*, invited paper for special issue on *Advances in the Fundamentals of Networking*, vol. 13, no. 7, pp. 1189–1198, 1995.
- [21] Debasis Mitra, Richard J. Gibbens, and Baosheng D. Huang, “State-dependent routing on symmetric loss networks with trunk reservation - I,” *IEEE Transactions on Communications*, vol. 41, no. 2, pp. 400–411, Feb. 1993.
- [22] Hari Balakrishnan, Srinivasan Seshan, Mark Stemm, and Randy H. Katz, “Analyzing stability in wide-area network performance,” in *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, June 1997.
- [23] Mark E. Crovella and Azer Bestavros, “Self-similarity in world wide web traffic: Evidence and possible causes,” *ACM/IEEE Transactions on Networking*, vol. 5, no. 6, pp. 835–846, Dec. 1997.
- [24] Raj Jain, Arjan Durresti, and Gojko Babic, “Throughput fairness index: An explanation,” ATM Forum Contribution: AF/99-0045, Feb. 1999.