Proactive Resource Provisioning for Voice over IP

Eric Chi, Michael Fu, and Jean Walrand
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
211 Cory Hall #1772, Berkeley, CA 94720-1772, USA
echi@eecs.berkeley.edu, mjfu@uclink.berkeley.edu, wlr@eecs.berkeley.edu

Keywords: Voice over IP, Differentiated Services, Quality of Service, Service Level Agreement.

ABSTRACT

This paper presents a dynamic Service Level Agreement (SLA) negotiation scheme between peer autonomous systems (ASes) that implement DiffServ per domain behaviors. For concreteness, we tailor our scheme to account for the needs of VoIP transport across multiple ASes. We present a heuristic but computationally simple and distributed scheme that uses traffic statistics to forecast the near-future demand. Our scheme provides a statistical guarantee on the renegotiation frequency and blocking probability. Simulations demonstrate that it achieves more efficient usage of the reserved bandwidth than would be accomplished by using the *de facto* static SLA negotiation schemes.

1. INTRODUCTION

Since its introduction in 1995, Voice over IP (VoIP) has enjoyed increasing popularity. Indeed, several privately managed VoIP networks have existed since as early as 1999, e.g. iBasis and ITXC are examples of currently active VoIP service. Developments in forwarding behavior such as Differentiated Services (DiffServ) provide the forwarding-plane building blocks necessary for supporting delay-sensitive applications like VoIP [1, 2, 3]. In this paper we consider privately managed IP networks that support, in addition to best effort, the Expedited Forwarding (EF) packet treatment which specifies that the service rate of an EF aggregate must exceed the arrival rate of the EF aggregate at a router [4].

Since an individual IP network often uses proprietary QoS mechanisms, it can only provide global connectivity if it has a globally comprehensive coverage of points of presence. The more scalable approach is to provide global coverage by peering with other privately managed IP networks. Thus a VoIP connection may be carried across multiple distinct parties participating in peer-to-peer arrangements. Each autonomous system (AS) determines the forwarding treatment of its packets.

To ensure desired end-to-end forwarding treatment, each AS has a logical entity called a bandwidth broker [5] which negotiates with brokers of its peer ASes to determine how much of an aggregate class of traffic an AS is willing to carry on behalf of its peer ASes. The bandwidth a broker *B* can allocate depends on the internal resources within *B*'s own AS and the amount of bandwidth *B*'s next hop ASes have agreed to provide for *B*'s traffic. The resulting bilateral business contracts are called Service Level Agreements (SLA).

To date, work on bandwidth brokers has focused more on implementation — signaling protocols and maintaining databases of network state [6, 7, 8, 9]. In accordance with the DiffServ philosophy, these mechanisms are designed for admission control and resource reservation of aggregate flows of traffic. While developing management mechanisms is critical, these mechanisms only ensure that once a session has been established, the quality of an ongoing session — e.g. low latency — will be adequately maintained. For some QoS applications, however, there are user expectations in session initialization.

Consider VoIP. Maintaining a low latency connection is necessary, but it does not achieve the QoS standards set by the Public Switched Telephone Network (PSTN). When a user picks up a phone, the user expects not only that the connection request will be granted with near certainty but also that the session will be established in negligible time. Such high performance is achieved with an architecture consisting of devices finely tuned to the sole task of serving voice connections.

Instead of optimizing the hardware throughout the IP network, to achieve comparable performance in an IP network, we propose separating admission control and resource provisioning — a pair of control decisions typically done together in circuit switched networks. By separate we mean the two control decisions occur asynchronously. We argue resource provisioning should be done in bulk, but it should be done in anticipation of resources needed in the near future. We maintain that admission control should be distributed to meet session initialization requirements. Then, when call requests eventually arrive, the required resources will either be reserved or not and an immediate admission decision may be made upon a request arrival at an ingress into a mesh of peering ASes.

Since connections have in-session QoS requirements, good admission decisions require some mechanism for such a distributed admission control to determine what resources have been provisioned within the network. Ensuring in-session QoS using distributed admission control is not the focus of this paper. This paper presents the design and evaluation of a distributed resource provisioning scheme engineered to complement a distributed admission control.

While statistics are collected to forecast near-future bandwidth demands, no specific traffic models are inherently integral to our heuristic scheme. Nonetheless, to quantitatively evaluate the performance of our heuristic, we consider our scheme under traditional teletraffic models and estimators.

While work has been done to determine what services should be requested in an SLA and how often the parameters of an SLA should be adjusted [10, 11, 12, 13], to the best of the authors' knowledge [14] is only one other scheme designed to make SLA

adjustments with low blocking probability of connections as an objective. That scheme, however, makes adjustments at regular time intervals; it does not allow for a dynamically adaptive update frequency. By relaxing a deterministic update frequency constraint to a statistical one, we allow occasional violations in a bound on the update frequency to accommodate atypical surges in demand while still limiting excessively frequent updates in the long run.

The rest of the paper is organized as follows. In Section 2, we formulate the VoIP provisioning problem between two peering ASes. In Section 3, we discuss features any good solution should have and, with these features in mind, construct our scheme. In section 4, we apply traditional telephony traffic models to our formulation. In Section 5, we describe simulation results. The primary motivation for simulation was to determine how well the scheme worked in a multi-AS network where each pair of ASes asynchronously managed their peering links. In section 6 we conclude with future directions.

2. VoIP PROBLEM FORMULATION

We assume that EF forwarding is achieved through priority queues [4]. Strict priority forwarding isolates voice traffic from non-voice traffic. Thus, we consider a network that carries only voice traffic. The core of the network consists of a mesh of DiffServ ASes. We use the term DiffServ AS to refer to an AS that provides DiffServ forwarding. Customer stub networks are connected to autonomous systems that serve as ingresses into the DiffServ core. Users in one stub network wish to establish voice sessions with users in another stub network. We assume that interdomain routing is fixed.

We assume that the only end-to-end requirement of voice connection is an effective bandwidth with EF forwarding. Consequently we make the simplification that brokers negotiate in units of connections. This assumption appears somewhat questionable since only aggregate forwarding equivalence classes may be discerned on an inter-AS link. Many current VoIP implementations follow the H.323 standards that have the option of performing PSTN functions such as link-by-link admission control through entities called gatekeepers [15]. One possibility is to require a VoIP call to register with an AS's broker after it has been admitted into the network and de-register when it leaves. Thus, a connection does not have to make itself known to all the ASes it traverses before it can begin transmitting packets. Moreover, note that the bandwidth broker need not keep a record of an individual call. The broker requires only maintaining the total number of active calls. The count can be incremented and decremented as calls come and go.

We do not explore the question of how many connections a transit AS should choose to transport. We assume that an AS employs an admission and intradomain provisioning strategy that ensures that whatever amount of traffic an AS agrees to carry, it will manage to provide the requested service with high likelihood. It is reasonable to assume this since the simple but inefficient over provisioning strategy, which is commonly used now, will work. Instead, as explained in the introduction, we explore the complementary question of how many connections should an AS request its downstream neighbor carry on its behalf.

Since we are designing an algorithm that maximizes the utilization of reserved resources, the client AS should always request resources it anticipates it will use in the near future whether or not its next-hop neighbor can lease the requested capacity. In other words, a client AS's increase in utility as a function of premium bandwidth is elastic up to its requested demand.

Additionally, maximizing utilization means sending as much premium traffic over reserved resources as possible. An AS can always serve best effort traffic with spare premium bandwidth; financial prudence, however, prescribes using as much premium bandwidth for premium traffic as possible.

To capture delays in exchanging SLA messages as well as delays due to resource allocation decisions, a deterministic amount of time τ elapses between the time when the bandwidth broker sends its request for more bandwidth and the time when the broker is notified that its request has been activated. Such a τ need not be the same for all SLA transactions. It is meant as a worst case bound on the time or a bound that holds true with some confidence. For simplicity, we assume the time τ is the same for all SLA transactions.

With this model, we seek to develop a dynamic SLA update strategy that accomplishes three goals:

1. The long-term average number of admitted connections is maximized
2. A connection is not admitted into an ingress AS, if the required bandwidth is not available end-to-end.
3. The strategy accomplishes the previous goals with the least over-provisioning.

3. ADJUSTING THE SLA

We now discuss criteria that we deem any reasonable strategy must meet in addition to the goals stated at the end of Section 2. For now, assume the arrival process is stationary with time; we address time inhomogeneity at the end of this section. We first consider two peering ASes and assume all VoIP calls are initiated in one AS and destined for the other. We consider how the client AS should adjust its SLA with its peering AS. The bi-directional case where the two ASes are transporting each other's traffic is a superposition of two instances of the previous case. The provisioning decisions over an entire network of ASes is the superposition of the provisioning decisions made in a two-peer scenario since inter-AS links are managed asynchronously.

An AS has incentive to request bandwidth in bulk from its neighbors because it is reasonable to assume some fixed cost — processing time and messaging overhead — is incurred every time bandwidth is requested and allocated. If connection requests arrive frequently enough, it makes sense to pre-provision for the future arrivals by requesting resources prior to the arrivals. Thus, it is desirable to build into the strategy a means to control the frequency at which bandwidth agreements are updated. To be precise, an update should occur no sooner than some specified time T with some specified confidence p. On the other hand, more frequent but smaller bandwidth adjustment requests result in more cost-effective use of the allocated bandwidth.

Thus, the strategy should dynamically update a resource request schedule for each bandwidth broker based on anticipated demand. By schedule we mean a rule for deciding when a bandwidth broker should request more bandwidth or return bandwidth, as well as for deciding how much bandwidth to request or return. We call the observed number of active connections at which a bandwidth request is made an *update state*. We use the word ``state" for convenience; the true state of the system is not completely specified by the number of active connections. When the number of active calls equals *update state* then *new quota* calls are allocated, if that bandwidth is available. This quota assignment

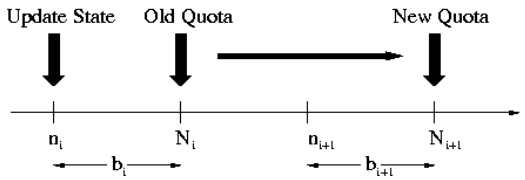

Figure 1. Schedule for Requesting Bandwidth

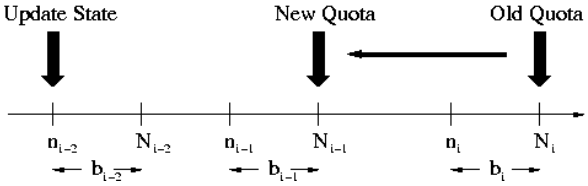

Figure 2. Schedule for Returning Bandwidth

rule holds true whether the number of active calls reaches *update state* from below or from above. This schedule should be such that it simultaneously minimizes processing time per connection and maximizes allocated bandwidth utilization.

The schedule should also account for the deterministic amount of time τ required to allocate new bandwidth. Connection admission requests do not cease to arrive, during this reconfiguration period. Thus, it is reasonable for an AS to request more bandwidth early enough so that there is low likelihood of running out of bandwidth before its neighbors can grant more bandwidth. To be precise, the schedule should be such that an AS requests for more bandwidth when the current number of admitted connections is such that in time τ with some specified confidence q the number of connection requests will not exceed the current number of allocated connections.

Figure 1 graphically represents a schedule. Let {Ni} denote the sequence of allocated quotas of bandwidths. The $i+1^{th}$ quota N_{i+1} is requested when the number of admitted connections reaches the i^{th} update state n_i. Note there is a buffer bandwidth, b_i, between the i^{th} update state and the i^{th} quota to account for the delay τ in making an SLA adjustment.

To minimize underutilization of allocated bandwidth, we also require a strategy for returning any unused bandwidth to a neighbor. If bandwidth is freed prematurely and enough new calls arrive, recently returned bandwidth will have to be re-requested. Recall we already have an update frequency defined by the parameters p and T. We choose update states at which to release bandwidth in a manner consistent with the desired update frequency.

Figure 2 graphically represents the schedule for returning bandwidth. If the last connection quota is N_i, we do not decrement the quota to N_{i-1} unless the number of connections falls to n_{i-2}. Recall n_{i-2} denotes the update point to request a new quota of N_{i-1}. The next update state is n_{i-1} and the quota requested for that state is N_i. Recall n_{i-1} was chosen so that starting from n_{i-2} we will reach n_{i-1} no sooner than in time T with confidence p. Thus, if the number of connections drops to n_{i-2} but begins to increase again, we will not request for bandwidth that was returned no sooner than in time T with confidence p. We refer to update states that correspond to an increase in quota to be *upgrade states* and update states that correspond to a decrease in quota to be *downgrade states*.

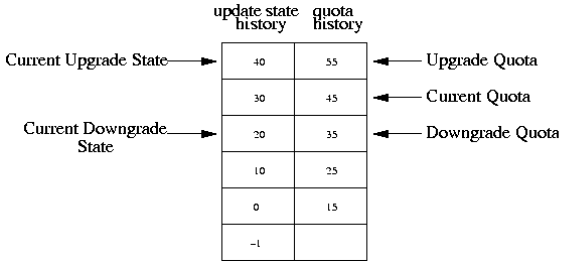

Figure 3. History Stack

To account for time inhomogeneity in the call arrival process we maintain a ``history stack'' of all downgrade states and corresponding downgrade quotas to use if the call arrival process were to stop. We do not keep the entire history. In the example stack shown in Figure 3 the current number of active calls is between 21 and 39. If the number of active calls reaches the current upgrade state of 40 calls, then a new upgrade state and corresponding quota is computed based on the current arrival statistics. This pair is pushed onto the top of the stack and the current upgrade and current downgrade state are appropriately re-designated. Similarly the current quota, the quota corresponding to the next upgrade, and the quota corresponding to the next downgrade are also re-designated. If the number of active connections drops to the current downgrade state, the current upgrade state and corresponding upgrade quota are popped off of the stack and the current upgrade state, downgrade state, current upgrade quota, current quota, and current downgrade quota are all re designated.

Thus in moving down the ``history stack'' the schedule re-applies previously computed update states and quotas. New information is incorporated in computing the next upgrade state and quota for only upgrade events. The five pointers shown in Figure 3 always maintain the same relative positions with respect to each other. The current upgrade state and quota always point to the entry at the top of the stack.

Note that in the example, the pair (10,15) corresponds to the initial upgrade state and initial quota. The update state of 0 is used so that if there are ever no active calls then the SLA should be reset to its initial value. The update state of -1 is used to prevent a downgrade in quota below the initial quota.

We decided to reuse previously computed update/quota pairs for downgrades based off of empirically measured daily demand profile of voice traffic which tends to be low in the morning; peak mid-day; and then decline thereafter. Using previous computations for downgrades yields a more conservative approach in returning bandwidth since the true arrival rate at the time of a downgrade is likely to be less than the arrival rate used to compute the subsequent update state after the downgrade. Upgrade events, on the other hand, probably indicate the arrival rate is increasing, and so the next update state should be computed based on the most current estimate of the arrival rate.

4. A MODEL FOR CALL SOURCES

For convenience we rely on traditional telephony models to numerically compute a schedule. Denote the number of active connections at time t as X_t. We model the connection arrival process as a Poisson arrival process with rate λ. Studies show that arrival rate of calls in the Common Channel Signaling Network is well described by a time-inhomogeneous Poisson process (e.g., [16]). We assume that although λ may vary with time that it does

not do so very rapidly and proceed to compute the schedule assuming that λ is fixed.

We wish to determine how much bandwidth to request to ensure a specified blocking probability and specified update frequency probability. We do not focus on blocks due to saturated downstream link capacity. That is a network dimensioning concern. We wish to minimize avoidable blocks — blocks that occur due to poorly scheduled update requests.

Holding times are i.i.d. exponentially distributed random variables with rate μ. The holding time of voice calls are effectively approximated by an exponential random variable with a mean of 3 to 5 minutes (e.g., [17]).

Thus, X_t evolves as an M/M/∞ queue. For now, we are not concerned with estimating λ and μ. We assume both vary slowly enough that they may be estimated accurately in deriving the schedule. We use the MLE estimator over small time intervals.

We continue with the simple case where the core DiffServ network consists of two peering ASes. Each AS serves one stub network, and there is demand for connections from one stub network to the other. We wish to calculate a schedule for the ingress AS to request and return bandwidth from its downstream neighbor given the traffic parameters μ and λ and the given parameters T, τ, p, and q as defined in Section 3.

We recursively calculate the update states n_i with $n_0 = 0$. Denote the first passage time to state m as T_m.

$$T_m = \inf\{t > 0 : X_t = m\}.$$

We want n_k to be the smallest state such that when we start at the last update state n_{k-1}, T_{nk} is greater than T with confidence p. For notational convenience let

$$P_k(A) = P(A \mid X_0 = k).$$

Then

$$n_k = \min\{m > n_{k-1} : P_{nk-1}(T_m > T) \geq p\}.$$

We take into consideration that updating an SLA takes non-negligible time τ. Hence, the bandwidth broker should request additional bandwidth so that, with high confidence, the number of net new calls in time τ will not exceed this buffer bandwidth. More precisely, when we start in state $X_0 = n_k$ we seek the smallest state $n_k + b_k$ such that T_{nk+bk} is greater than τ, with confidence q.

$$b_k = \min\{m > 0 : P_{nk}(T_{m+nk} > \tau) \geq q\}.$$

The schedule of update states is given by $\{n_i\}$. The schedule of bandwidth quotas is given by $\{N_i = n_i + b_i\}$.

We approximate the hitting time probability with the transition probabilities of an M/M/∞ queue. Justification of this approximation may be found in [18].

We again recursively calculate the update states $\{n_i\}$ with $n_0 = 0$, this time using transition probabilities. We want n_k to be the smallest state such that when we start at the last update state n_{k-1} with confidence p, X_T is strictly less than n_k.

$$n_k = \min\{m : P(X_T < m \mid X_0 = n_{k-1}) \geq p\}.$$

We choose the buffer bandwidth b_k for the k^{th} update state to be the smallest state such that when we start at the update state n_k, with confidence q, X_τ is strictly less than $n_k + b_k$.

$$b_k = \min\{m : P(X_\tau < m + n_k \mid X_0 = n_k) \geq q\}.$$

To perform the above computations, we require the transition probabilities of an M/M/∞ queue. Recall that X_t denotes the number of active calls at time t.

Given the number of active calls $X_{t'}$ at time t', we can express the number of active calls at time $t+t'$ as the sum of two independent random variables, $N(t,t')$ and $Y(t,t')$. $N(t,t')$ denotes the number of the calls out of the $X_{t'}$ that are still active after the time t, and $Y(t,t')$ denotes the number of calls that arrived between the time of t' and $t+t'$ that are still active at time $t'+t$. $N(t,t')$ is distributed binomially with parameters $X_{t'}$ and exp(-μt). $Y(t,t')$ is Poisson with rate λ/μ(1−exp(-μt))

To simplify calculations we approximate both the Binomial and Poisson components with Gaussian random variables. Furthermore, we take τ to be orders of magnitude smaller than μ^{-1}, thus we keep only the first order terms of the Taylor expansion of the terms $e^{-\mu\tau}$. Applying these simplifications we obtain the following recursive update schedule

$$\mu_Y = \lambda/\mu(1-)$$
$$\mu_{nk} = n_k\mu(1-\mu\tau)$$
$$\sigma^2_{nk} = n_k\mu\tau(1-\mu\tau)$$
$$n_{k+1} = [\mu_Y + \mu_{nk} + Q^{-1}(1\text{-}p)(\mu_Y + \sigma^2_{nk})^{1/2}]$$
$$b_k = [\lambda\tau - n_k\mu\tau + Q^{-1}(1\text{-}q)[\lambda\tau + n_k\mu\tau)^{1/2}].$$

Q denotes the Q-function and [y] denotes the smallest integer greater than y.

As stated earlier, it is a simplification to assume that the arrival process is stationary. From an estimated arrival rate an upgrade point could be computed adaptively as the estimated arrival rate changes in time as described in Section 3. We shall refer to the scheduling algorithm as proactive resource provisioning (PRP) for the remainder of the paper.

5. SIMULATION RESULTS

We describe the simulation scenarios used to evaluate the performance of the PRP scheme. Call durations were modeled as i.i.d. exponentially distributed random variables with a mean of 2 minutes. The call arrival process for each source-destination stub pair was modeled as a Poisson arrival process with a right continuous step intensity function, λ (t). We present two sets of simulations.

The first set considers provisioning a single OC-3 link between two peer ASes which each serve one stub network. All calls are initiated in one stub and are destined for the other. The arrival rate increases and then decreases in a manner that mimics empirically measured arrival rates. The update frequency and blocking rate statistics are examined, and the time average utilization is compared to that which would have been observed if provisioning is done statically applying the Erlang B formula to the busiest hour.

The second set applies PRP to a network of six ASes and five links each consisting of two OC-3 links. A naive distributed admission control strategy is used in conjunction with the provisioning provided by PRP. The purpose of this simulation set is to determine how much of the average utilization gained in using PRP over static dimensioning is lost when global knowledge of the link provisioning is unavailable at the admission points into the core network.

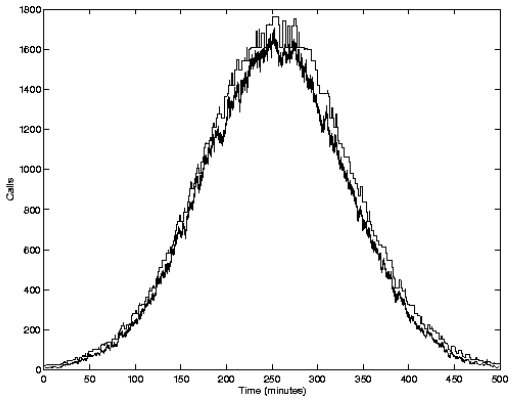

Figure 4. Single Link: T = 1 minute

We find that PRP achieves higher link utilization than static schemes while maintaining prescribed stochastic bounds on update frequencies.

For all simulations we took τ to be 1 minute. Parameters *p* and *q* were taken to be 5% and 1% respectively in all simulations except for the multi-link simulation where *p* was taken to be 1%. The arrival rate is estimated with the MLE over sliding five-minute. The mean holding time is assumed to be known.

Single Link

Figures 4 and 5 depict typical realizations for a prescribed threshold update period *T* of 1 minute and 5 minutes respectively. The daily variability used here is consistent with measurements of the number of calls placed at Stanford University's telephone exchange during every hour of a weekday averaged over a period of six months in 1995 [19]. The same trial was repeated 1000 times for different *T*. The relevant statistics are shown in Tables 1 and 2.

The random variable T_{obs} is the time interval between quota update request events. $P^*(T_{obs} > T)$ and E^*T_{obs} are the empirical tail distribution and sample mean of T_{obs} respectively. U^* is the time average utilization of the provisioned resources.

PB^*P is the sample mean of maximum of observed ratios of blocked calls to arrived calls within a τ time interval over a single simulation run, and B^*P is the sample mean of the observed ratio of blocked calls to arrived calls over an entire simulation run.

Table 1. Single Link: Update and Utilization Statistics

T (min.)	$P^*(T_{obs} > T)$	E^*T_{obs} (min.)	U^*
1	0.997	1.74	0.908
5	0.996	22.8	0.654
10	0.996	25.5	0.633
∞	-	-	0.194

Table 2. Single Link: Blocking Rate Statistics

T (min.)	PB^*P	B^*P
1	0.048	5.6×10^{-5}
5	0.011	4.9×10^{-5}
10	0.012	5.5×10^{-5}
∞	-	0.01

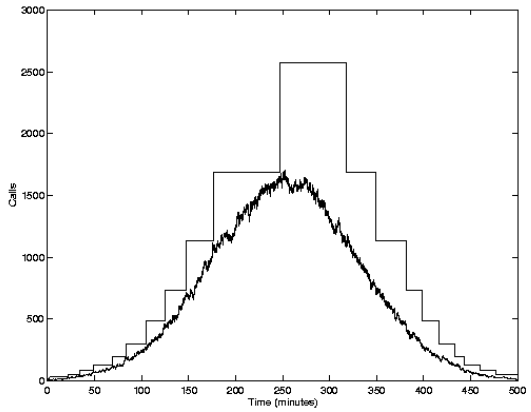

Figure 5. Single Link T = 5 minutes

The last row in Table 1 and 2, corresponding to $T = \infty$, was calculated using the Erlang B formula to determine the provisioning required to achieve a blocking probability of 0.01, which equals our prescribed parameter *q* for the simulations, for the peak arrival rate prescribed over the 500 simulated minutes. Note all prescribed inequalities are satisfied while the average utilization of provisioned resources compares favorably to the average utilization achieved through static worst-hour provisioning.

Not surprisingly, decreasing the value of *T* increases bandwidth efficiency. The sharp drop in average utilization shown in Table 1 when *T* is increased beyond 1 minute can be explained by PRP's conservative return bandwidth policy. In Figure 5 at about 250 minutes into the simulation the bandwidth quota is increased perhaps a bit too proactively, and PRP waits over an hour for the demand to drop significantly prior to returning the bandwidth.

Multiple links

PRP specifies only a resource provisioning strategy. An admission control scheme is not prescribed here but is necessary to complement PRP. Recall distributed admission control at the edges of the DiffServ core expedites admission decisions. Here we demonstrate PRP's effectiveness in a multi-link network where spare capacity of all inter-AS links are not known at all times to admission control points at the edge of the DiffServ core. Each AS applies PRP provisioning on each of its outgoing links in the network shown in Figure 6.

Each AS broadcasts to all its neighbors its spare capacity to each given stub network. An AS receiving the broadcast from an upstream AS knowing its own spare capacity on the inter-AS link from which it received the broadcast takes the minimum of that spare capacity and the spare capacity in the broadcast message.

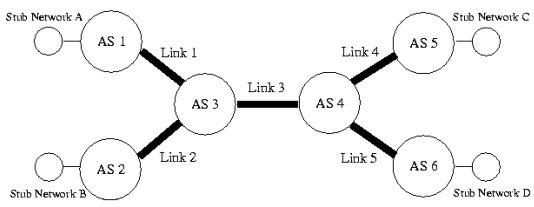

Figure 6. Multiple Links Toplogy

Table 3. Multiple Link Network: Update and Utilization Statistics

Link	$P^*(T_{obs} > 10$ min)	E^*T_{obs} (min.)	U^*	U_s^*
1	1.0	57.4	0.622	0.161
2	0.99	84.6	0.585	0.162
3	0.95	75.5	0.617	0.287

Table 4. Multiple Link Network: Blocking Rate Statistics

Source/Destination Pairs	PB^*P	B^*P
AC	0.006	5.0×10^{-5}
BD	0.004	3.2×10^{-5}

It then broadcasts this minimum to all inter-AS edges other than the one it received the broadcast. This is reminiscent of distance vector routing where nodes announce their distances to a given sink but not the routes. Broadcasts are made both periodically with a specified inter-broadcast time and whenever an update in quota occurs on any link in the network. We acknowledge this scheme fails to scale, but we require some distributed admission control scheme to evaluate PRP in the network. More scalable distributed admission control schemes such as the one described in [20] which uses Explicit Congestion Notification do exist and may be used with PRP.

Broadcasts occur every 5 minutes. T was taken to be 10 minutes. Calls arrive for two different source-destination pairs — stub A to stub C and stub B to stub D. The arrival rate for stub A to stub C increases to a peak rate at 250 minutes and then decreases symmetrically before settling down to a constant rate of 100 calls per minute. The arrival rate for stub B to stub C is the reflection about the 350-minute mark of the arrival rate for stub A to stub C calls. Note that the second goal stated at the end of Section 2 may be violated when using a distributed admission control policy. Here the broadcast frequency was chosen high enough so that at no time did the number of active calls exceed the allocated quota on any link for all 1000 runs.

Update and utilization statistics are shown in Table 3, and blocking rate statistics are shown in Table 4. Again the specified stochastic bounds are met. The last column in Table 3 shows the average utilization achieved by statically provisioning for the worse case arrival rate with the prescribed 0.01 probability of call blocking.

6. Conclusion

Our key contributions are: 1) provisioning resources to meet a statistical guarantee on the SLA renegotiation frequency in addition to connection blocking probabilities; 2) demonstrating that our AS pair scheme run asynchronously for all AS pairs in a multi-AS network efficiently uses reserved resources.

We emphasize that PRP and distributed admission control complement and do not take the place of the other. PRP as it is presented here does not announce what resources have been provisioned. It is up to the admission control scheme to determine what resources have been provisioned within the network by PRP. At the same time, distributed admission control schemes at best only ensure in-session QoS requirements such as low latency but have no control over call blocking probability. The latter QoS requirement can be met only by judicious SLA provisioning.

In conjunction with a distributed admission control scheme, PRP facilitates fast session initialization without sacrificing blocking probability or network utilization while controlling the rate at which SLA adjustments are made.

Additionally, PRP enables any given AS to make provisioning adjustments independently of the states and actions of all other ASes. The downside of having each AS pair renegotiate asynchronously is that no end-to-end guarantees can be made on end-to-end resource availability. In this sense PRP provides best-effort end-to-end resource allocation. The simulation results, however, suggest that stringent coordination of provisioning of AS peering links is not necessary. Indeed, despite a lack of end to-end coordination among ASes, increases and decreases along particular end-to-end paths are correlated since reservation adjustments are made based on measured load levels. Thus, although each AS-pair makes provisioning adjustments asynchronously with respect to other pairs, end-to-end like provisioning is roughly met.

Furthermore, these adjustments do not require flooding the IP network with control messages. Again, allocations occur independently and moreover asynchronously without need for ASes to broadcast allocation decisions to each other. By design only changes in traffic along a given inter-AS link initiate an adjustment along that inter-AS link which limits the communication overhead making PRP a scalable provisioning strategy. Together with a good distributed admission control scheme, PRP requires marginal overhead within the core of the network.

Given how encouraging the simulation studies in multi-AS networks were, developing models that predict the multi-AS simulation results is a natural next step that we plan to pursue. We feel, however, that the most interesting future direction is to make PRP robust against modeling error. PRP could be fashioned to observe the objectives of interest — update frequency, utilization, and blocking rate — and make adjustments that move the observed objectives in the desired direction. This approach presents new challenges. While utilization and SLA update frequency may be monitored locally at an AS, blocking probability is only observable at the edges of a network where admission decisions are made. Adapting PRP to make decisions locally at an AS based on information from the edges is critical to making PRP practically applicable to all delay-sensitive applications with session initialization requirements.

ACKNOWLEDGEMENTS

The authors thank Dragan Petrović for helpful discussions in the early development of this work as well as reviewers for their constructive comments.

REFERENCES

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

[2] Brim, S.; B. Carpenter, F. Le Faucheur. 2000. ``Per Hop Behavior Identification Codes." RFC 2836, IETF, May.

[3] Nichols, K.; S. Blake; F. Baker; D. Black. 1998. ``Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers." RFC 2474, IETF, Dec.

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

[5] Nichols, K.; V. Jacobson; L. Zhang. 1997. ``A Two-bit Differentiated Services Architecture for the Internet.'' Internet Draft, IETF, Nov.

[6] Teitelman, B. and P. Chimento. ``QBone Bandwidth Broker Architecture.'' Internet2 QBone Draft homepage, http://qbone.internet2.edu/bb/bboutline2.html.

[7] Engel, T.; H. Granzer; B.F. Koch; M. Winter; P. Sampatakos; I.S. Venieris; H. Hussmann; F. Ricciato; S. Salsano. 2003. ``AQUILA: Adaptive Resource Control for QoS Using an IP-based Layered Architecture.'' *IEEE Communications Magazine*.Vol. 41, No. 1, Jan:46-53.

[8] Cortese, G.; R. Fiutem; P. Cremonese; S. D'antonio; M.\ Esposito; S.P. Romano; A. Diaconescu. 2003. ``CADENUS: Creation and Deployment of End-User Services in Premium IP Networks.'' *IEEE Communications Magazine.* Vol. 41, No. 1, Jan:54-60.

[9] Flegkas, P.; P. Trimintzios; G. Pavlou. 2002 ``A Policy-Based Qualifty of Service Management System for IP DiffServ Networks.'' *IEEE Network*. Vol. 16, No. 2, March-April:50-6.

[10] Chuah, C.; L. Subramanian; R. Katz; A. Joseph. 2000. ``QoS Provisioning Using a Clearing House Architecture.'' In *Proceedings of IEEE/IFIP Eighth International Workshop on Quality of Service* (Pittsburg, PA, Jun.).IEEE,Piscataway, NJ, 115-24.

[11] Kamienski C. and D. Sadok. 2001. ``Chameleon: An Architecture for Advanced End-to-End Services.'' In {\em Proceedings of the Second IEEE Latin American Network Operations an Management Symposium} (Belo Horizonte - MG, Brazil, Aug.).

[12] Günter M. and T. Braun. 1999. ``Evaluation of bandwidth Broker Signaling.'' In *Proceedings of IEEE Seventh International Conference on Network Protocols*. (Toronto, Canada, Oct.).IEEE Computer Society,Los Alamitos, CA, 145-52.

[13] Dermler, G.; M. Günter; T. Braun; B. Stiller. 2000. ``Towards a Scalable System for Per-flow Charging in the Internet.'' In *Proceedings of the Applied Telecommunication Symposium* (Washington D.C., Apr. 17-19.),SCS, San Deigo, CA, 182-7.

[14] Hampshire, R.; W. Massey; D. Mitra; Q. Wang. 2002. ``Provisioning of Bandwidth Sharing and Exchange.'' In *Telecommunications Network Design and Economics and Management: Selected Proceedings of the 6th INFORMS Telecommunications Conferences* (Boca Raton, FL, Mar. 10-13). Kluwer Academic Publishers, Boston/Dordrecht/London, 207-226.

[15] Open H323 Project: http://www.openh323.org

[16] Duffy, D.; A. McIntosh; M. Rosenstein; W. Willinger. 1993. ``Analyzing telecommunications traffic data from working common channel signaling subnetworks.'' In *Proceedings of the 25th Interface* (San Diego, CA). Computing Science and Statistics, Vol. 25, Interface Foundation of North America, 156-165.

[17] Doshi, B.T.; S. Dravida; D. Jeske, K. Murti; B. Samadi. 1999. ``Performance Analysis of Switch Access Systems.'' *Bell Labs Technical Journal*, Vol. 4, No. 2, April-June: 197-208.

[18] Chi, E. Proactive Resource Provisioning for VoIP. UC Berkeley, Department of EECS M.S. Thesis (Dec. 2001).

[19] Lam, D.; J. Jannink; D.C. Cox; J. Widom. 1996. ``Modeling Location Management in Personal Communications Services.'' In *Proceedings of the IEEE 5th International Conference on Universal Personal Communication*, (Cambridge, MA, Sep.).IEEE, New York, NY, 596–601.

[20] Kelly, F.P.; P.B. Key; S. Zachary. 2000. ``Distributed Admission Control.'' *IEEE Journal on Selected Areas in Communications*, Vol. 18., No. 12, Dec:2617-28.