Edge-Based QoS Provisioning for Point-to-Set Assured Services

Satish Raghunath, Kartikeya Chandrayana, Shivkumar Kalyanaraman
Department of ECSE, Rensselaer Polytechnic Institute
rsatish, kartikc, shivkuma@networks.ecse.rpi.edu

Abstract

In this paper we propose an edge-based quality of service architecture aimed at site-to-site private networks over the Internet. We extend the traditional point-to-point service model to a point-to-set service model, assuming a finite, bounded set of destination sites. A point-to-set service allows a user to have a pool of premium tokens, which could be flexibly assigned to traffic going towards any destination within the set. The proposed point-to-set service provides statistical assurances and flexibility to users while allowing providers to obtain multiplexing gains. To realize a point-to-set service model, we introduce edge-based dynamic bandwidth tracking and provisioning schemes. The tracking algorithm predicts demand towards a given destination edge. This information is used by the provisioning algorithm, which uses a variant of the max-min approach, to efficiently allocate bandwidth towards the destinations in the set. We also define a metric to characterize the degree of flexibility allowed by a particular point-to-set assured service. Simulation results are presented to demonstrate the merits of the proposed architecture in terms of cost savings to the customer and efficient resource utilization to the provider. The scheme is shown to deliver gains of more than 50% *in terms of increased utilization to the provider while keeping the drop rates within* 0.65% *for the customer. Even higher gains for the provider is demonstrated with drop rates within* 1%*.*

1 Introduction

The best-effort (non-QoS) traffic in Internet is inherently of the *point-to-anywhere* nature, i.e., sources direct packets to any possible destination. In contrast, traditional quality-of-service (QoS) models set up premium services on a *point-to-point* basis (eg: virtual leased lines, frame-relay, ATM services, int-serv [4] etc). Recently, with the advent of IP differentiated services [3, 6, 5] there has been interest in expanding the *spatial granularity* of QoS models. Clark and Feng [6] proposed that a pool of "assured" service tokens could be allocated to a user or site with the flexibility to mark packets sent to any arbitrary destination with such tokens. While such a *"point-to-anywhere"* assured service model is very appealing to users, the large spatial granularity of the service makes efficient admission control and provisioning virtually impossible [5]. We consider the subset of this problem by examining assurances to a fixed set of destinations.

Consider a private network of sites A, B, C and D as shown in figure 1. The aggregate traffic from A is bounded by the "outgoing pipe" purchased from the provider. Site A's peak traffic to any node

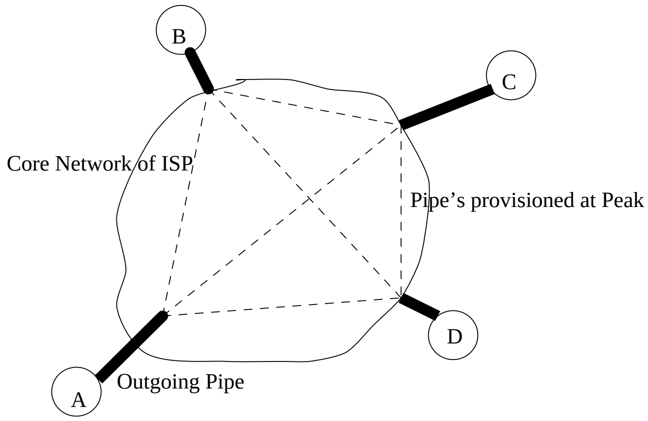

Figure 1: The Point-to-Set Architecture: Site A transmitting to {B,C,D}

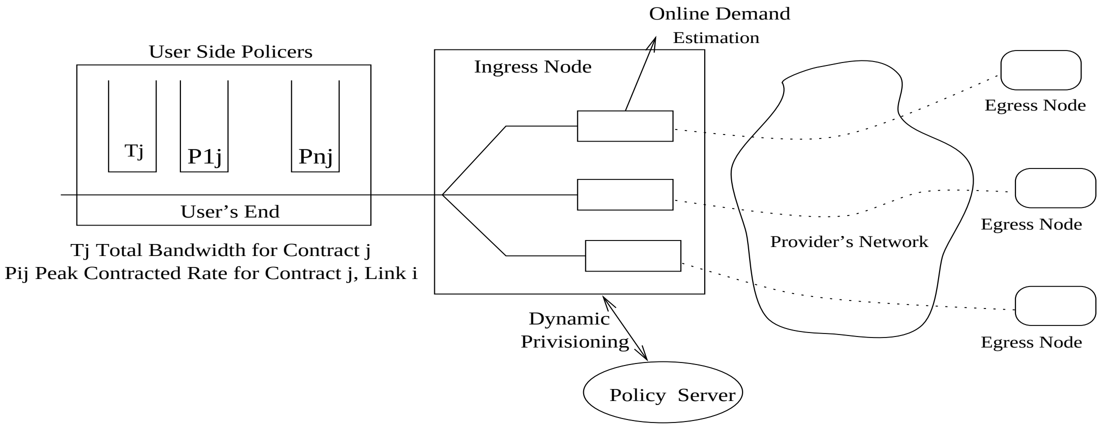

Figure 2: The Point-to-Set Architectural Model

in the private network, i.e. B, C or D (called the *set*) is bounded by "peak". Given the present infrastructure, the site A would require a link with capacity equal to "peak", to each destination in the *set* for an assured service towards the sites in the set. As such, the total purchased capacity from the provider (which is three times the peak here) exceeds "outgoing pipe" leading to wastage of resources. We propose a *point-to-set service* wherein a customer buys a bandwidth *equal* to his "outgoing pipe" (or a given total bandwidth), but is statistically assured that his traffic needs to any destination in the set are met. In other words, the customer buys bandwidth to a set of destinations, instead of purchasing point-to-point links to the destinations.

In this paper we develop an edge-based point-to-set QoS model for a site-to-site private network. Any node, which has data to be transmitted to other nodes in the network, is labeled as a *point* while the rest constitute the *set* and hence the name *point-to-set*. Instead of statically provisioning point-to-point links, at peak, to each source-destination pair, this architecture makes better use of the resources by allocating only as much bandwidth as the pair needs with suitable headroom to accommodate bursts. Depending on the traffic profile of the customer, the provider allocates bandwidth towards the various destinations in the set. Thus the provider network would appear as a set of variable-sized links connecting this edge to the destination set. In order to achieve this, there should be a demand estimation component and a provisioning component. We describe these components in succeeding sections.

In the previous example, if site A were to employ a point-to-set service, it would purchase bandwidth equal to the "outgoing pipe" towards the set {B, C, D}. Thus there is a cost saving in that the point-to-point links need not be leased from A to the set. For the provider, the paths connecting edge A to the set {B, C, D} can be multiplexed with other contracts. Simulations deploying the provisioning algorithm proposed here reflect these gains to the provider and customer as discussed above.

The point-to-set service is provisioned over a single provider (or cooperating multi-provider) IP inter-network. The service is negotiated with the provider as contracts which are admitted using an admission control scheme. The proposed scheme can be *completely provisioned and managed at the network edges* in coordination with a centralized admission control component. The point-to-set contracts are weaker than traditional QoS contracts as they cannot offer strict guarantees and only offer statistical assurances. In spite of this limitation, we believe that our point-to-set model is a first step in realizing Clark et al's [6] goal of going beyond point-to-point services and providing flexibility to users, while at the same time realizing multiplexing gains for providers.

LIRA [5] considers the problem of large spatial granularity. But the focus of [5] is not on VPN resource management. Duffield et al [13] propose a framework for VPN resource management. They introduce the idea of a "hose" as a resizeable access link for a VPN node. This is similar to the per-path dynamic provisioning presented in this paper. However, [13] does not consider the problem of provisioning bandwidth among contracts. Further, [13] uses a fixed measurement window based estimator while we present a new demand estimation algorithm based on correlation horizon of the underlying traffic process (refer section 3.3). Our initial model considered in this paper differs from the simple assured service model [6] in the following respects: a) we assume a simple FIFO internal network, b) we assume the availability of a network map with bandwidths at a policy server (possible through OSPF extensions [2]), and c) all admitted traffic in the network is in-profile and traffic waits at the edge till it obtains tokens.

The rest of the paper is organized as follows. Section 2 outlines the point-to-set architectural model. Section 3 summarizes the building blocks of the model. The building blocks are detailed in Section 3.3, 3.4 and 3.5 which focus on online demand estimation, dynamic provisioning and admission control respectively. In Section 4 we present the simulation results and compare our model with point-to-point allocation resource model. Section 5 summarizes and concludes the paper.

2 Point-to-Set Service Model

The *point-to-set service* model (Figure 2) allows a site j to claim a pool of premium tokens T_j per unit time. Here, T_j refers to the total contracted bandwidth of the contract. Let any destination in site j's set be addressed as i, where $i = \{1, 2, .., N_j\}$ and N_j is the cardinality of Site j's set. Then the point-to-set service model allows the site j to flexibly use these tokens for the traffic towards it's set, subject to the limit that on each path ji the *peak rate* is less than or equal to p_{ji}. Observe that the user has the flexibility to use up to the peak rate p_{ji} on any single path, but pays the provider only based upon the aggregate pool of premium tokens T_j per unit time. This is more economical for the user if $T_j < \sum_i p_{ji}$ because unlike point-to-point leased lines or frame-relay CIR[1] it need not pay for unused committed rates. The essential data plane component is a traffic conditioner which now consists of a set of token buckets: one for the overall pool of tokens with rate T_j and one for each per-path peak assured rate limits p_{ji} (Figure 2). We assume that bucket depths are minimal for smoothness. In this paper we consider only premium traffic. However, in a real implementation we can allocate capacity for the premium traffic and leave the rest for the best effort traffic.

From the provider perspective, however, this scheme does not yield multiplexing gains compared to peak-rate provisioning *unless a dynamic measurement and re-provisioning strategy is used.* That is, in the absence of dynamic re-provisioning, the provider will have to have per-path policers (at the ingress edge block in Figure 2) operating at the peak rates p_{ji}. To achieve multiplexing gains, we develop the following components. First, we develop an online demand estimation module which monitors the varying per-path traffic demands (and includes adjustments for queue backlogs at the provider-side per-path policer). It accounts for potential long-range dependence in traffic by exploiting the concept of *correlation horizons* [10], i.e., the existence of an estimation time-scale over which second-order statistics hold. In particular, the module detects the current correlation horizon dynamically and uses the second-order statistics of demand estimated on that horizon. Second, these per-path demand estimates d_{ji} are used in a dynamic provisioning module (d_{ji} is the estimate for site's traffic from j to destination i in the set). This module operates in two phases: a) per-path provisioning phase and b) per-contract provisioning phase. The two phases can be explained as follows. On a bottleneck link, there could be many contending Contracts. The sum of the estimates for each Contract on that link may exceed the link capacity, i.e., $\sum_j d_{ji} > C_i$, where C_i is the capacity of the constrained path ji. Therefore for this constraint, we need to do resource allocation at the "Path Level". Let us assume, each contract was allocated y_{ji} on the path ji. Now, when decisions at Path level have been made, then we need to do a dynamic provisioning decision at each "Contract Level". This is needed because the sum of the allocations made at each link, for a particular Contract, may exceed the Contract's total bandwidth T_j, i.e., $\sum_i y_{ji} > T_j$. Finally it enforces per-path rate limits z_{ji} at the provider-side policers for each path ji such that $\sum_i z_{ji}$ is close to T_j. This means that the provider can attain multiplexing gains because

[1]CIR = Committed Information Rate parameter of frame-relay service

$T_j << \sum_i p_{ji} \Rightarrow z_{ji} \leq p_{ji}$. However, the potential random shift of traffic from one path to another implies that each dynamic provisioning decision or admission control decision is subject to error. The queue backlogs due to errors in a provisioning period are accounted in the next period. The rare case of consistent queuing due to errors in admission control and persistent shifts in traffic may be dealt with by forced renegotiation of peak assured rates p_{ji} only on the affected paths i. Buffer overflows during such phases lead to packet loss, i.e., degradation of assured service which is a cost of attaining multiplexing gains.

The choice of provisioning time-scales is important since it decides the responsiveness of the provisioning algorithm to traffic changes. We assume that online measurement intervals Section 3.3 are much more fine-grained than provisioning time-scales. The provisioning timescale is set based according to the roundtrip times of between the edge routers. Our simulations show that provision timescales of less than twice the roundtrip between the edges suffices the requirements. This is discussed in more detail in Section 4.

3 Components of the Point-to-Set Architecture

The architecture contains the following components: contract definition, data plane components (user-side and provider-side shapers), online demand estimation component, per-path dynamic provisioning (at the policy server), per-contract dynamic provisioning (at the ingress edge) and an admission control component. These topics are elaborated in the following sub-sections.

3.1 Contract Definition

The contract for user j is defined as:

- The finite, known set of destinations S_j with cardinality N_j.
- T_j, is the aggregate traffic to the set, S_j, i.e. if t_{ji} is the traffic generated towards destination $i \in S_j$, then $\sum_i t_{ji} \leq T_j$.
- The per-path peak rate, p_{ji} on path ji.
- Duration of the contract. Not explicitly leveraged in this paper.

The per-path peak rates are assumed to have been determined in a characterization phase before the contract is arrived up on. In this paper, we also use a concept of minimum per-path assured service expectation which states that the user will get a minimum provisioning of $\min T_j/N_j$ on any path at any time. This constraint is enforced by admission control. Beyond that, the provisioning

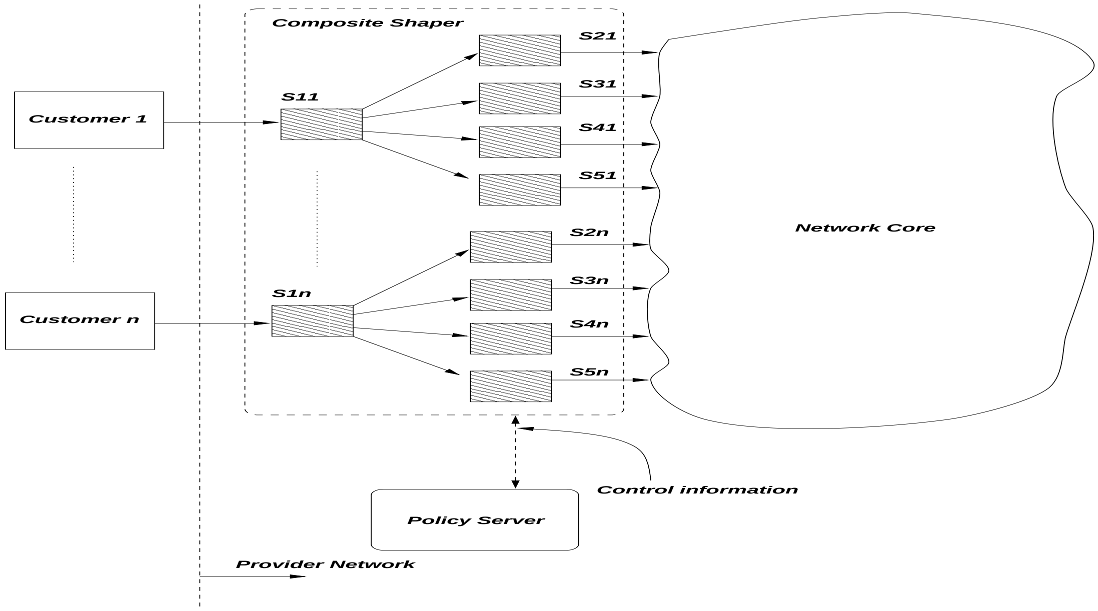

Figure 3: Composite Shaper

may depend upon the demand estimation and multiplexing characteristics of the path. Errors in provisioning or admission control in this regime manifest as delays (on which there is no assurance), packet losses, and/or involuntary downward renegotiation of peak rates on specific paths. One can imagine a regime without such a minimum service expectation, which could lead to higher multiplexing gains, but also lead to higher multiplexing costs due to provisioning/admission control errors.

3.2 Traffic Conditioners: An Integrated Implementation

Figure 2 had illustrated a simple data-plane model with user-side and provider-side policers/shapers. On each side, the model needs one aggregate shaper per contract and also a shaper per destination in the set specified in the contract. Hence the total number of shapers for contract with user j is $N_j + 1$. Instead of multiple shapers handling this need, a single entity, a *composite shaper*, can be envisioned to achieve the same function. Figure 3 illustrates this concept.

3.3 Online Demand Estimation

The online demand estimation component interfaces with the composite shaper discussed in the previous section to compute demand estimates d_{ji} for each path ji. This demand estimation has to also be conducted carefully. Experimental evidence [11] suggests that network traffic exhibits properties of self-similarity and long range dependence (LRD). In order to track demand patterns

Algorithm 1 Adaptive calculation of *EWMA* weight parameter

$mindeviation = MAXNUM;$
$index = 0;$
for $weight = 0.05$ to 0.99 **do**
  {Update the mean deviation using latest sample in arrivals and meanarrivals as previous prediction}
  $meandev[index] = meandev[index] + abs(meanarrivals[index] - arrivals);$
  {To find the weight with minimum deviation}
  if $meandev[index] < mindeviation$ **then**
    $mindeviation = meandev[index];$
    $optimalwt = index;$
  end if
  {Update the moving average for arrivals}
  $meanarrivals[index]* = (1 - weight);$
  $meanarrivals[index]+ = weight * arrivals;$
  {For calculation of deviation}
  $sqrmeanarrivals[index]* = (1 - weight);$
  $sqrmeanarrivals[index]+ = weight * arrivals * arrivals$
  $index + +;$
  $weight+ = 0.05;$
end for
{Using *optimalwt* calculate prediction}
$std\_deviation = sqrt(sqrmeanarrivals[optimalwt] - (meanarrivals[optimalwt])^2);$
$prediction = meanarrivals[optimalwt] + 2 * std\_deviation$

it is essential to address issues due to long-range dependence.

The immediate question to be answered is that of the amount of correlation information and the appropriate timescale to be considered for demand estimation. One would expect that traditional p-Markov models would not be sufficient in modeling such traffic. However, recent work [10] analysing finite buffer queues with LRD input indicates that only a finite amount of correlation need be considered for certain queuing performance measures (e.g., loss rate). The amount of correlation that needs to be taken into account depends not only on the correlation structure of the source traffic, but also on time scales specific to the system under study. However, the paper establishes the existence of a correlation horizon at any point in time. In particular, it is shown that the impact of correlation in the arrival process on loss rates becomes negligible beyond a time-scale, referred to as the *correlation horizon*. This implies that we may choose either self-similar models or models with finite memory as long as it captures correlation up to the correlation horizon for the system.

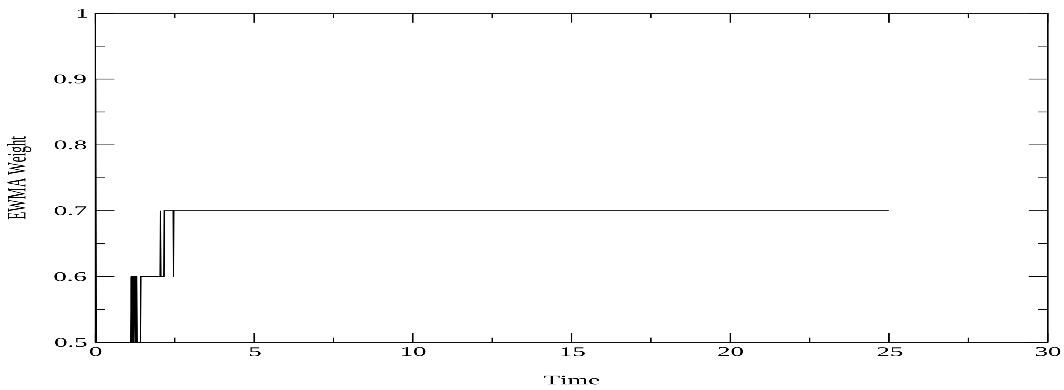

Figure 4: EWMA weight α vs Time

We therefore choose a simple Exponentially weighted moving average (EWMA) predictor. The weight parameter of the EWMA predictor is changed dynamically to suit the traffic characteristics depending upon the current correlation horizon. This missing piece is to detect the correlation horizon. The demand estimation scheme hence maintains a *vector of EWMA predictors* for different weight parameters ($\alpha = [0.001, 0.9]$, though the pseudo-code below implies a smaller range). At the end of a measurement window, the weight parameter *corresponding to the minimum deviation from the observed sample for the previous interval* is chosen. The minimum deviation here implies that the corresponding weight parameter reflects underlying correlation horizon of the traffic. The details of the algorithm can be found in the pseudo code. The predictor then provides the average (μ) and variance (σ^2) of the tracked process. The estimate, e, is calculated as

$$e = \mu + 2\sigma$$

.Figure 4 shows a typical graph for the variation of the weight parameter with time, for a specific simulation run. The weight parameter initially varies, but later on stabilizes around a single value. The estimates produced by the predictor are plotted against the actual samples, in Figure 5. The estimates can be seen to be effectively tracking the samples. The final estimate of demand is referred to as d_{ji} in subsequent sections.

3.4 Per-Path Dynamic Provisioning

As mentioned earlier the dynamic provisioning decision is made in two phases, and uses the demand estimate d_{ji} to finally yield the rate limits for the shapers z_{ji}. The per-path decision is made with the knowledge of demand estimates for competing contracts on every link in the edge-to-edge path. The decision is then returned to the ingress edge which uses it in the second phase (per-contract provisioning) before finally enforcing the rate-limits. Let us define a_j, p_j, d_{ji}, and y_{ji} as the average rate and peak rate for the Contract j, resource demanded (estimate) for the present timescale,

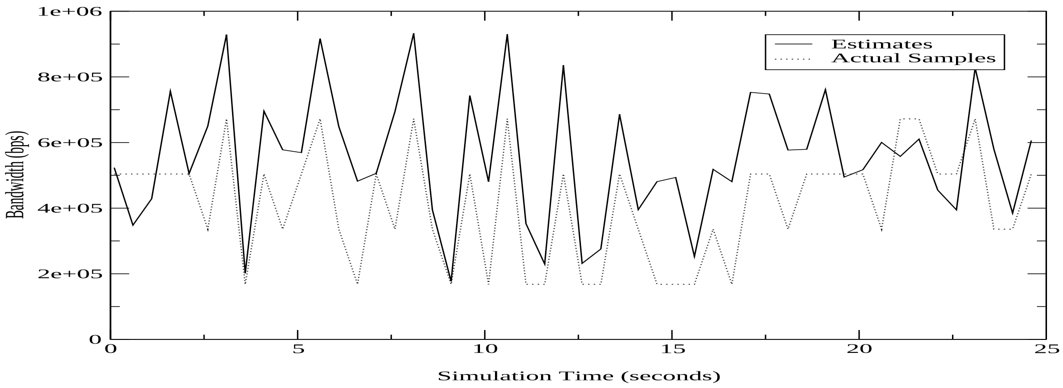

Figure 5: Online Estimates of Traffic vs Time

and the final allocation made for Contract j on Link i respectively. Also define that there are N contracts on the link. Then the per-path provisioning proceeds as follows:

- If the sum of per-path provisioning estimates (over all contracts) on a link ($\sum y_{ji}$) is less than the link capacity C, then allocate estimate y_{ji} to each shaper. Otherwise:
- For all the contracts where $d_{ji} \leq a_j$ allocate d_{ji}, i.e., $y_{ji} = d_{ji}$.
- For the remaining contracts, minimize the given equation, according to constraints stated.

$$Minimize \ \max_j (y_{ji} - a_j) \ subject \ to \tag{1}$$

$$\sum_{j=1}^{N} y_{ji} \ \leq \ C, \tag{2}$$

$$y_{ji} \ \leq \ d_{ji}, \forall j \tag{3}$$

$$\sum_{j=1}^{N} a_j \ < \ C, \tag{4}$$

$$y_{ji} \ \geq \ 0, \forall j \tag{5}$$

where

$$a_j = \frac{T_j}{N_j} \tag{6}$$

The solution to the above minimization is suggested in Algorithm 2. Assume that the contracts are maintained, in a list, in the ascending order of the difference of estimate and average, i.e., $(d_{ji} - a_j)$.

In the case where, $d_{ji} \leq a_j$, another allocation strategy could have been always provisioning a_j. Though this assures a contract of a minimum guaranteed bandwidth, it will be at the cost of greater

Algorithm 2 Per-Path Provisioning Decision

For all $d_{ji} \leq a_j$ allocate d_{ji}. Delete these contract from the List. Calculate available resource A, as the difference of allocations made and the link Capacity C.

For the rest of the contract allocate a_j, where a_j is the average rate for the Contract j. Define $x_{ji} = a_j$ and recalculate A.

while $A > 0$ **do**

  Define *diff* = Minimum$(d_{ji} - x_{ji})$ over all j, M as the number of contracts for which provisioning decision has to be made.

  if *diff* $\geq A/M$ **then**

    Define *increment* $= A/M$

    Allocate resources equal to *increment* to each contract.

    $A = 0$

  else

    increment = *diff*

    Allocate resources equal to *increment* to each contract.

    Update x_{ji} as x_{ji} += *increment*

    A -= *diff* * M

    Delete the contracts for which $(d_{ji} - x_{ji})$=0 from the List (of contracts for which provisioning has to be made).

    Update M.

  end if

end while

losses if the contract makes excursions to peak. Assuming that a contract will make excursions above and below the contracted average, a_j, by allocating bandwidth only equal to the estimate (even though it is below the contracted average, a_j) we distribute losses uniformly over time.

Figure 6 describes the per-path provisioning policy. We first allocate resources to all the contracts whose estimate is less than their average and eliminate them from the provisioning list. Figure(6) shows all the remaining contracts. Here we first allocate the corresponding average *avg* to each contract. Then, the minimum difference between the estimate and the difference is calculated $d1$ and is allocate to each contract, and the available bandwidth on the link is updated. With this allocation, we find that the first contract, Contract 1, demands has been met and subsequently it is removed from the list. Again, the new minimum difference, $d2$ between the estimate and the allocated (till now) is calculated and dispersed amongst all the remaining contracts, eliminating yet another contract from the list. This process is repeated recursively until, we run out of the available bandwidth on the link. This is similar to a max-min algorithm.

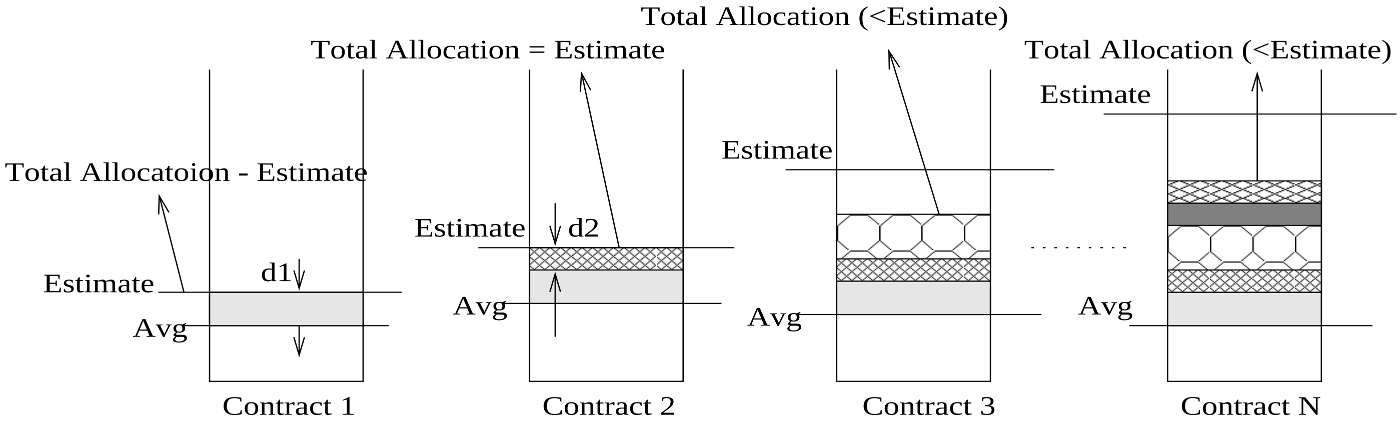

Figure 6: Per-Path Reprovisioning Policy

3.4.1 Per-Contract Reprovisioning

At the end of the Per-Path Reprovisioning, the allocations conveyed by the Policy Server to the agents on the Ingress edges are y_{ji}. This means that the per-path allocation is y_{ji} to contract j on path ji. However, the total suggested allocations for the contract may exceed his total contracted bandwidth T_j. So the agent at the edge needs to do another level of reprovisioning amongst different paths in a contract to obey this SLA and also to conform to the suggested allocations by the Policy Server. This reprovisioning can be again expressed as an optimization problem. Let z_{ji} be the final allocation made (by agent) to Contract j on path ji and N_j be the number of destinations in the Contract j's set. Then the problem is stated as:

$$Minimize \max_i z_{ji} \;\; subject \; to \tag{7}$$

$$\sum_{i=1}^{N_j} z_{ji} \le T_j, \tag{8}$$

$$z_{ji} \le y_{ji}, \forall\, i \tag{9}$$

$$z_{ji} \ge 0, \forall\, i \tag{10}$$

The solution to the above equations can again be computed using the algorithm 2 (i.e. a max-min allocation), albeit taking care of the new variables and the constraints.

3.5 Measurement-Based Admission Control

A Policy Server makes all the Admission Control decisions. The Policy Server is assumed to have a map of the network including link bandwidths (available through OSPF extensions [2]) and a picture of the current set of contracts. Similarities in the Policy Server placement can be drawn from the Rendezvous Point in the PIM/SM implementation of the Multicast. When the user passes his requirement to the Policy server, associate the contract to a path and calculate the most constrained (by resources) link on that path. We assume that the routing is stable over longer timescales and hence the path remains constant throughout the life time of the contract. Here a path is defined as the set of link, starting with the edge's link, to a destination in the Set. The path which includes the link (at any edge) with the minimum unused resources (bandwidth) is defined as the most constrained path. The admission control decision can then be made at the edge, which has been communicated the unused capacity of most constrained link on the path.

The admission control policy is defined as follows. The admission decision is based on the most constrained link, i of total capacity C with N contracts currently using that path. Let the new contract needs a Total resource of T_{new} with a peak p_{new} and average $a_{new} = T_{new}/n$ where n is the number of destinations in the contract.

Define a measure flexibility, f_{ik} which characterizes the allowed variation (by the point-to-set service) of traffic profile to the Contract k on Link i. Let C_i be the total capacity on a link and the user's destination set consists of links $i = 1 \ldots n$. Assume at the beginning of the provisioning timescale, the estimate for the Contract j on the Link i is d_{ji} and N contracts use the link i and their usage is y_{ji}. The unused capacity U_i on the link i can then be calculated as

$$U_i = C_i \; - \sum_{j=1, j \neq k}^{N} y_{ji} \tag{11}$$

Then, we define the flexibility f_{ik} as:

$$f_{ik} \; = \; 1 \, if \, \frac{U_i}{y_{ik}} \; \geq \; 1 \, and \, \frac{y_{ik}}{p_j} \; \leq \; 1 \tag{12}$$

$$= \frac{U_i}{y_{ik}}, \, if \, \frac{U_i}{y_{ik}} \; \leq \; 1 \tag{13}$$

A flexibility of 1 implies that a contract obtains whatever bandwidth it demands. Thus flexibility gives a measure of the degree of multiplexing on the constrained link. Finally, the contract's flexibility on a constrained link is an EWMA of the f_{ik}'s. On a constrained link i, the flexibility for a contract j, f_{ji} is called f_j. Then the admission control decision is specified as:

If a_j, f_j are the average rate and the flexibility of a Contract j on the constrained path i, then admit a new contract if,

$$\sum_{i=1}^{N} a_i + B/n \; \leq \beta \, C \;, where \, \beta \leq 1 \; and \tag{14}$$

$$(f_1 * f_2 * \ldots * f_N)^{1/N} \; \geq \; F_{min} \tag{15}$$

where F_{min} is the minimum flexibility needed on any link. This parameter F_{min}, is defined by the provider to maintain a specified grade of service.

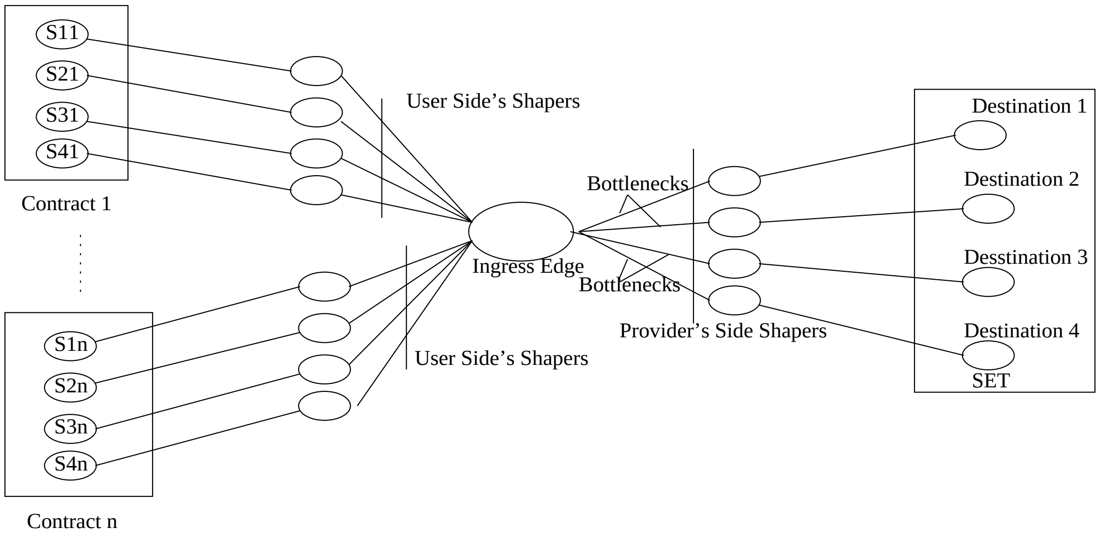

Figure 7: Topology used for Simulation

4 Results and Discussion

The topology shown in Figure 7 was simulated on NS [9]. All the contracts have an equal total contracted bandwidth, T, of 2 Mbps, a peak rate of 0.75 Mbps and have a common set, of 4 destinations. The constrained links are of 25 Mbps and the one way end-to-end propagation delay is 30ms. All the links other than the constrained links are of 10 Mbps. The source traffic is shaped by static (customer side) shapers to ensure conformity with the contract, in terms of the peak rates. The provider network features the dynamically provisioned shapers which adapt to the user's traffic distribution with respect to his destinations.

The point-to-set model was tested in the following settings:

- **Traffic simulated by agents provided by NS:** Constant Bit Rate traffic sources (CBR) with rates varying randomly with time were employed. A *uniform* random variable was sampled to obtain the rates for the CBR agents every 0.15 seconds. The traffic was generated such that it is distributed uniformly across destinations. Thus it is equally likely that a particular rate is observed towards a destination. Note that the dynamically provisioned shapers adapt to the traffic profile of the customer and are not in anyway aware of this arrangement. So even if a traffic profile features sources biased to certain elements of the destination set, the shapers will adapt appropriately.

- **Trace driven simulation:** MPEG encoding of the Star Wars movie, converted into the NS trace format, is available at [12]. The "Application/Traffic/Trace" agent in NS was employed to generate video traffic from this trace. The source was attached to the NS UDP agent.

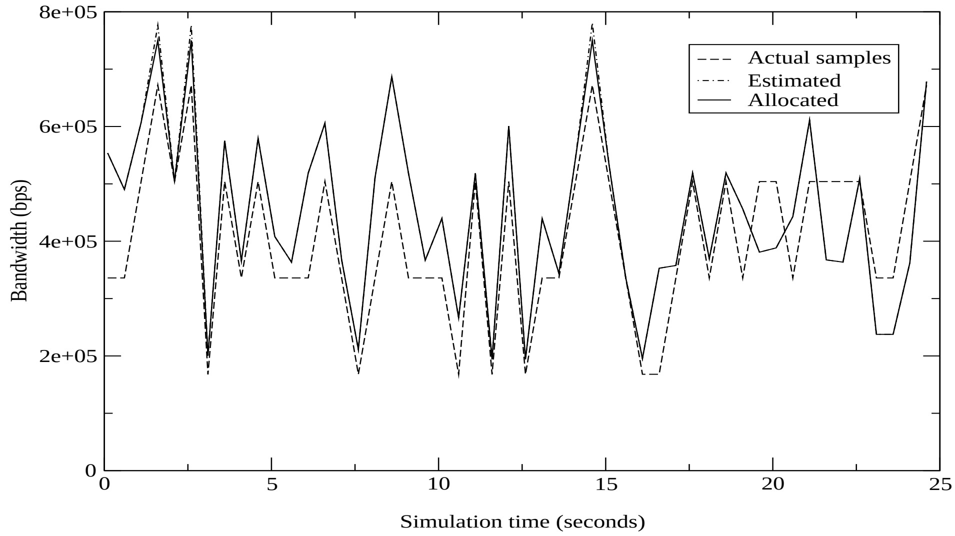

Figure 8: Allocated, Estimated bandwidth and Actual Traffic Samples Vs Time

Every source picks a random start point in the trace file [9].

In the following paragraphs, the performance of the provisioning algorithm is first examined and then the gains of the proposed architecture are presented.

4.1 Performance of the provisioning scheme

In Section 3.4 we presented a strategy to handle per path bandwidth provisioning. The performance of this strategy decides the costs to the customer. Figure 8 shows the plots for the actual allocation made for a contract along with the traffic samples and estimates versus time. The Allocation curve is almost coincident with the Estimate curve. This is indicative of the fact that most of the time the contract was allocated what it demanded. Since the customer's traffic is loosely policed, his peak rate at some time instants exceed the negotiated peak, i.e. 0.75MBps and in such cases the allocation is clipped to the contract's peak. In effect the plot shows that the contract requirements are met.

4.2 Multiplexing Gains

The proposed model exploits the fact that customer does not offer peak load towards every destination. As against a point-to-point model, where the per-path peak would have been reserved on each path, this model only needs the contracted per path average to be set aside. This means, the sum of peaks of contracts on a link can add up to a number greater than the link capacity. The

Num Contracts	$p_j/T_j = 0.5$	$p_j/T_j = 0.6$	$p_j/T_j = 0.7$	$p_j/T_j = 0.8$
10	0.7425	2.2410	4.5084	6.5784
12	0.8202	2.3092	4.2187	6.7317
15	0.8812	2.2225	4.5198	7.0512

Table 1: Simulated Traffic: Drop rates (%), with 25M constrained link, total bandwidth per contract = 5M and per path peak varying from 2.5M to 4M

advantage of this is that the provider can offer a cheaper and more flexible service. If the user buys a total bandwidth (towards the set of destinations) that is smaller than the sum of peaks (towards these destinations) he stands to gain. This would typically satisfy the needs of a customer whose average usage is low as compared to the peak usage. The trade off here is the statistical nature of the assurances. Unlike the point-to-point model, there is a cost in terms of dropped packets. Here we consider the cost of multiplexing contracts in the simulation scenario described above in terms of drop rates.

4.3 Customer gain with simulated traffic

To examine the customer gain, we employ the ratio of the per-path peak, p_{ji}, to the total contracted bandwidth for the set of destinations, T_j. Observe that the customer is paying for T_j, that is T_j/N towards each destination in the set, while being allowed to send p_{ji} towards destination j. Thus, higher the ratio p_{ji}/T_j, higher the gain for the customer. However, it is important to note that a higher value of this ratio implies *greater strain* on the provisioning algorithm and a higher trade-off in terms of drops. This is because, the provider has to maintain a "zero sum game" in terms of the bandwidth allocated to this contract towards the destinations in its set. Also, the tracking algorithm has to deal with higher variability of offered load towards a destination.

In the simulations, p_{ji} values were assumed to be equal (say, P_j) Figure 9 and table 1 depict the drop rate with increasing number of contracts for various values of P_j/T_j, for simulated traffic. For a given number of contracts the drop rate increases with the ratio P_j/T_j. However, we observe from the graph (and from table 1) that the drop rates are within 0.8% even for a peak to total ratio of 0.5. This implies that with a low drop rate, the customer can offer a peak rate of $N\Sigma_i p_{ji}$, (N=4 in the simulations) while paying for $2P_j$ (assuming the same peak towards all destinations, $P_j = p_{ji}$ for all i).

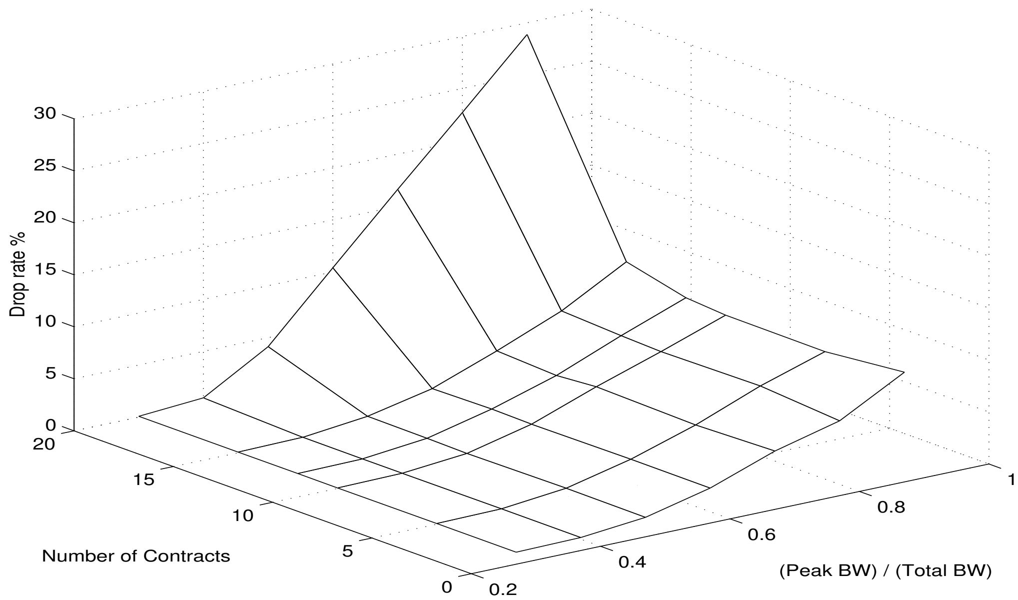

Figure 9: Drop Rates (%) against number of contracts and Peak/Total ratio

4.4 Customer gain with trace-driven simulations

In this section we shall examine the same parameters as those in the previous subsection, the difference being the source of traffic. The results here are pertaining to packets generated from traces of actual traffic. In figure 10 the drop rates (%) are plotted against the number of contracts and the ratio p_j/T_j. The plot and the data in table 2 indicate that even with a peak to total ration of 0.6 the drop rates are within 0.8%. This reinforces the fact that there is cost saving for the customer in opting for the point to set model, if the specified error rates are tolerable.

Num Contracts	$p_j/T_j = 0.5$	$p_j/T_j = 0.6$	$p_j/T_j = 0.7$	$p_j/T_j = 0.8$
10	0.8007	0.7863	0.7834	0.7833
12	0.8167	0.8042	0.8013	0.8012
15	0.8769	0.8676	0.8654	0.8654

Table 2: Star Wars trace: Drop rates (%), with 25M constrained link, total bandwidth per contract = 5M and per path peak varying from 2.5M to 4M

4.5 Provider gain with trace-driven simulations

In the case of static provisioning, the provider sets aside the peak contracted bandwidth causing resources to be underutilized. However, with dynamic provisioning of resources, the provider sets

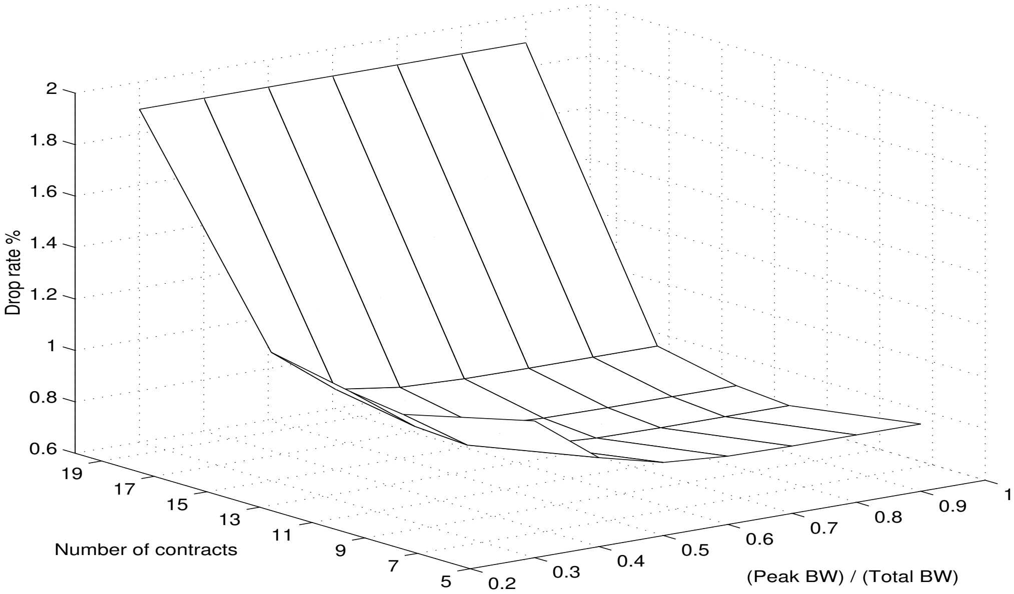

Figure 10: Star Wars trace: Drop Rates (%) against number of contracts and Peak/Total ratio

aside as much bandwidth as is demanded by the customer traffic. If the average demand seen for a contract j is $\bar{d}_{ji}$ towards destination i, then the ratio of peak P_j to $\bar{d}_{ji}$ quantifies the average gain for the provider. Taking note of this fact, the table 3 presents relevant data. Each row of the table gives data for a particular range of gain. Thus in the simulation with 12 contracts, there were 3 contracts for which the gain was in the range $(1.0, 1.5)$ and the average gain was 1.14. From table 3 we see, that higher gains (for the provider) are accompanied by a higher drop rate (for the customer). But the average gain with drop rates within 0.65% was 65%. This means the provider can increase his resource utilization by more than 50% while keeping drop rates within 0.65%. Also, the table indicates that gains of up to 200% can be achieved with drop rates of less than 1%, by multiplexing 15 contracts.

It is important to observe here that with 15 contracts, although the sum of the peak rates (37.5M) exceeds the link capacity the drop rates are low. This means that as compared to static provisioning, point-to-set service delivers considerable gains to the provider. However if the number of contracts being multiplexed increased to 20, the drop rates go beyond 1% since the sources are more likely to overwhelm the link capacity. Due to space constraints, the table corresponding to simulated traffic is not presented here.

4.6 Effect of provisioning timescale on performance

The provider-side shapers are dynamically reprovisioned periodically every T seconds. The choice of T affects the performance of the provisioning algorithm. This is because, T decides the respon-

Num Contracts	Gain Range	Avg Gain $P_j/\bar{d}_j$	Drop Rate (%)	No Contracts seeing gain
12	(1.0,1.5)	1.14	0.04	3
	(1.5,2.0)	1.65	0.66	3
	(2.0,2.5)	2.03	0.40	1
	(2.5,3.0)	2.86	1.40	2
	(3.5,4.0)	3.15	1.00	2
	(4.0,4.5)	4.45	2.20	1
15	(1.0,1.5)	1.09	0.13	3
	(1.5,2.0)	1.70	0.65	4
	(2.0,2.5)	2.39	0.40	1
	(2.5,3.0)	2.77	0.50	1
	(3.5,4.0)	3.53	1.47	3
	(4.0,5.5)	4.80	1.60	3
20	(1.0,1.5)	1.15	0.88	5
	(1.5,2.0)	1.69	1.26	6
	(2.0,2.5)	2.19	1.00	1
	(3.5,4.0)	3.30	2.84	5
	(4.0,4.5)	4.42	3.47	3

Table 3: Star Wars trace: Provider gain with 25M constrained link, total bandwidth per contract = 5M and per path peak = 2.5M

siveness of the decision process to changes in the input traffic profile. If the the response is quick, the drop rates are reduced. In the above simulations the timescale was set to 1.5 times the edge to edge propagation delay (0.1 seconds). If the timescale is varied to different multiples of RTT, we observe the drop rates as in figure 11. As the provisioning timescale increases with respect to the edge to edge propagation delay, the drop rates increase.

The rate changes will occur at multiples of the end-to-end RTT. In order to be responsive to rate changes, the provisioning timescale must be of the order of the RTT. However, the end-to-end RTT is not known to the provider. Hence the edge to edge propagation delay is a valid choice.

5 Summary and Conclusions

This paper developed the building blocks for an efficient implementation of point-to-set assured service capabilities. The contributions include the development of a simple architecture, contract model, online demand estimation, dynamic provisioning and suggestions for simple admission con-

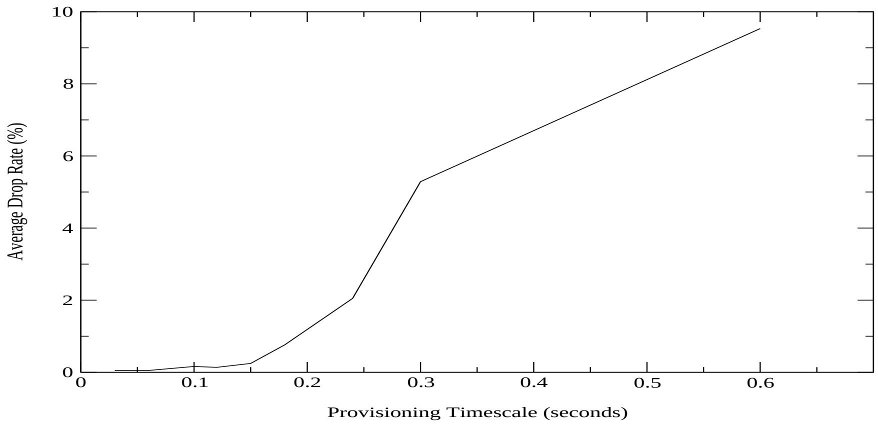

Figure 11: Drop Rates (%) against different provisioning timescales

trol techniques. Simulation results were presented to demonstrate the gains to the customer and provider due to the scheme. The scheme was shown to deliver gains of more than 50% in terms of increased utilization to the provider while keeping the drop rates within 0.65% for the customers. Even higher gains for the provider are shown to be possible with drop rates within 1%. The customers on the other hand stand to gain in terms of cost savings in that only the required total bandwidth needs to be purchased. Future work will focus on quantifying the errors in admission control, estimation and provisioning.

References

[1] D. Harrison, S. Kalyanaraman, S. Ramakrishnan, "Overlay Bandwidth Services: Basic Framework and an Edge-to-Edge Closed-Loop Building Block," preprint. http://www.ecse.rpi.edu/Homepages/shivkuma/OverlayQoS.ps

[2] G.Apostolopoulos, R. Guerin, S. Kamat, A. Orda, T. Przygienda, D. Williams, "QoS Routing Mechanisms and OSPF Extensions," *RFC 2676,* August 1999

[3] S Blake etal, "An Architecture for Differentiated Services", *IETF RFC 2475,* December 1998.

[4] R. Braden, D. Clark, Scott Shenker, "Integrated Services in the Internet Architecture: An Overview", *IETF RFC 1663,* June 1994.

[5] Ion Stoica, Hui Zhang, " LIRA: An Approach for Service Differentiation in the Internet",*Proceedings of NOSSDAV, Cambridge,* England, pp. 115-128d, July 1998.

[6] D. Clark, W. Fang, "Explicit Allocation of Best-Effort Packet Delivery Service", *IEEE/ACM Trans. on Networking,* 6 (4), pp. 362-373, August 1998.

[7] Floyd, S., and Jacobson, V. "Random Early Detection gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking,* Vol.1 No.4, p. 397-413, August 1993.

[8] G. Iannaccone, M. May, C. Diot. "Aggregate Traffic performance with Active Queue Management and Drop from Tail", *Sprint Technical Report TR01-ATL-012501,* Sprint ATL, January 2001.

[9] UCB/LBLN/VINT "Network Simulator - ns (version 2)", http://www.isi.edu/nsnam/ns, 1997.

[10] Matthias Grossglauser, Jean-Chrysostome Bolot. "On the Relevance of Long-Range Dependence in Network Traffic," *SIGCOMM'96,* August 1996.

[11] Will E Leland, Murad S Taqqu, Walter Willinger and Daniel Wilson. "On the self-similar nature of Ethernet traffic (Extended version)," *IEEE/ACM Trans. on Networking*, 2(1):1-15, February 1994

[12] "Traffic Trace for ns", http://www.research.att.com/~breslau/vint/trace.html

[13] N.G. Duffield et al, "A Flexible Model for Resource Management in Virtual Private Networks", *SIGCOMM'99.*