

# Dynamic Edge Provisioning for Core IP Networks

Raymond R.-F. Liao and Andrew T. Campbell  
Dept. of Electrical Engineering, Columbia University  
530 W. 120th Street, New York, NY 10027, USA  
{liao, campbell}@comet.columbia.edu

**Abstract**—Effective edge capacity provisioning is an important architectural component of the emerging Differentiated Service Internet. We propose a set of dynamic provisioning algorithms, which are operational at the edge routers of a differentiated services core network. These edge mechanisms include: i) ingress dynamic link sharing, which augments class based queueing techniques with bandwidth utility functions so that dynamic link sharing can be used to distribute bandwidth among traffic conditioners located at edge routers; and ii) egress dynamic capacity dimensioning, which formulates bandwidth dimensioning at egress links to peering/transit networks taking into account measured core network traffic conditions. We demonstrate through analysis and simulation that the proposed edge provisioning architecture is efficient and effective at supporting user link sharing policies while taking into account the dynamics of traffic load measured in the core network.

## I. INTRODUCTION

Delivering effective capacity provisioning in a differentiated services (DiffServ) Internet is dependent on how intelligent edge routers are at providing service differentiation while keeping the core network mechanisms simple and the core state aggregated. Edge router mechanisms include ingress traffic conditioning [5] that shapes/polices incoming traffic to the core network, and egress capacity dimensioning that provisions bandwidth between peer/transit networks. Common wisdom suggests that static provisioning will suffice in delivering differentiated services on a large scale. However, applying these techniques to core networks is very challenging for the simple reason that the differentiated services architecture assumes the existence of coarse grained traffic profiles based on traffic aggregates and not per-flow information. By “static provisioning” we mean that the time scales over which traffic profiles can be updated (e.g., monthly or yearly) is many orders of magnitude greater than the time scales of traffic dynamics experienced by core network provisioning algorithms, which would typically operate over hundreds of milliseconds or seconds. Given this mismatch it is difficult to see how static traffic conditioning profiles can respond effectively to the onset of sustained congestion experienced by core networks. In a similar manner, statically provisioned egress links can become bottlenecks or significantly under-utilized.

We propose an alternative approach to static provisioning. We argue that capacity provisioning needs to be made dynamic to improve the efficiency of the core network and that this provisioning should be governed by a set of well understood “quantitative” rules. We envision a core network architecture where bandwidth utilization, granularity of provisioned services, ingress traffic conditioning and egress capacity dimensioning are responsive to network conditions over multiple time

scales.

In this paper, we propose an *edge provisioning architecture* for differentiated services. We assume the existence of core network algorithms that would operate in unison with the dynamic edge provisioning algorithms introduced in this paper. In related work we have proposed such core network mechanisms [11] that includes dynamic node and core provisioning algorithms. Briefly, node provisioning algorithms located at the core routers maintain a core traffic load matrix of measured network conditions. Core provisioning algorithms adjust bandwidth allocation at ingress traffic conditioners when persistent core congestion is detected. For the purpose of provisioning bandwidth on-demand we assume that service providers are willing to exploit dynamic peering relationships that allow dynamic changes to peering agreements between networks.

This paper is structured as follows. In Section II, we provide an overview of an edge provisioning architecture that interacts with core and node provisioning algorithms discussed in [11]. Following this in Section III, we present a utility function model that abstracts the bandwidth allocation and provisioning policies at the ingress and egress routers, respectively. In Section IV, we discuss two utility-based allocation policies for welfare-maximum and proportional utility-fair allocation. Following this, in Section V, we detail an ingress bandwidth distribution algorithm, which distributes core network provisioned bandwidth to traffic aggregates at the edge. In Section VI, we describe a dynamic egress provisioning algorithm, which dimensions the bandwidth at egress edge routers for peer/transit networks taking into account the measured core network traffic conditions and service level specifications. Finally, we present some concluding remarks in Section VII.

## II. EDGE PROVISIONING ARCHITECTURE

A number of research questions motivate the design of our edge provisioning architecture. First, how do we best distribute bandwidth among ingress traffic aggregates? Second, how do we determine the bandwidth needs at egress points for peering purposes? The answer to these questions provides insight into the more general issue of how to design or select service level specification (SLS) between a service provider and a customer, and between service providers themselves.

The dynamic edge provisioning architecture illustrated in Figure 1 comprises *dynamic ingress link sharing* and *dynamic egress capacity dimensioning*. In our architecture dynamic ingress link sharing is triggered by the core provisioning algorithms [11] (via a *Regulate\_Edge* signal) when core algorithms

determine that a change needs to be made to the current edge rate allocation for a particular service class. At the edge we introduce the  $U(x)$ -CBQ traffic conditioner algorithm, which uses aggregate bandwidth utility functions to adjust the link-sharing ratio of each flow aggregate based on class based queueing (CBQ) [6] techniques. This enforces link sharing among flow aggregates within a single service class (e.g., Expedited Forwarding (EF) Per-Hop-Behavior (PHB), or Assured Forwarding (AF) PHB Group [5]) at the ingress traffic conditioners.

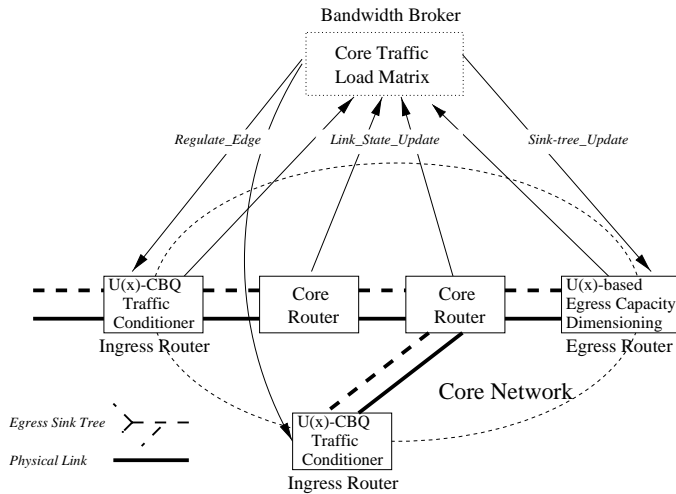


Fig. 1. Dynamic Edge Provisioning Architecture

Dynamic egress capacity dimensioning is invoked (*via a Sink-Tree\_Update* signal) by a core bandwidth broker when there is a significant change in rate allocation at ingress edge routers. We use the term “sink-tree” to refer to the topological relationship between a single egress link (representing the root of a sink-tree) and two or more ingress links (representing the leaves of a sink-tree) that contribute traffic to the egress point. Changes are detected using a traffic load matrix, which contains the state of every sink tree rooted at egress links. A capacity dimensioning algorithm, which is operational at each egress router, computes the ideal capacity between the egress link and the peering/transit network. This calculation reflects the internal core and ingress capacity bottlenecks and the measured traffic load distribution. An egress bandwidth utility function is formulated to drive the peering operation by using an aggregated utility function that takes into account ingress sessions.

The work on dynamic bandwidth distribution for ingress traffic conditioners is strongly related to CBQ link sharing techniques. The  $U(x)$ -CBQ traffic conditioner algorithm is built on CBQ and supports sharing policies that are functions of time-varying link bandwidth. The functional form of the sharing policies is based on bandwidth utility functions. The formulation of bandwidth utility functions for traffic aggregates is an open research issue, however. We approach the problem by categorizing applications into groups, and develop out aggregate utility functions for each application group, including TCP, and small and large size video/audio applications. A policy that maximizes total utility with non-concave utility functions is computationally intensive (i.e., NP-hard). We also observe that it is difficult to aggregate the utility function states for flow aggregates.

We resolve these problems by defining piece-wise linear utility functions, and present fast algorithms realizing utility maximization and fairness, as well as operations that support utility aggregation for scalability.

Egress capacity dimensioning discussed in this paper leads on from our previous work [16] on the dynamics of peering among different core networks. In that work, a core network is abstracted into a single bottleneck capacity shared by all the ingress/egress peering links, whereas each link takes its individual portion of the bottleneck capacity. This model has the benefit that it is tractable but it over-simplifies a network with multiple bottleneck links. In this paper, we formulate an aggregated utility function for each service class at each egress peering point with consideration of utility functions at ingress routers and bottleneck capacities inside the core network. The negotiation of peering agreements at different egress links is based on the egress utility function, and the calculation of bandwidth values is coordinated to avoid provisioning excess bandwidth that can not be utilized due to bottleneck constraint at other parts of the core network. In addition, an aggregated egress utility function formulated at an egress link of a core network serves as an input to the  $U(x)$  - CBQ traffic conditioner of the peering network that the link is connected to. This formulation allows recursive application of utility-based ingress conditioning and egress dimensioning from one core network to another, hence it results in the same treatment to SLS from end-users and peering/transit core networks.

### III. BANDWIDTH UTILITY FUNCTION

The concept of utility function describes a user’s relative preference to an amount of received goods or service, and in the case of bandwidth utility function, the allocated bandwidth. Bandwidth utility functions have been used as a conceptual abstraction of the application’s valuation of available bandwidth in analyzing some fundamental Internet design questions [17], [3]. Breslau and Shenker [3] use several types of bandwidth utility functions to investigate the merits of bandwidth reservation for adaptive and rigid applications.

The wide adoption of bandwidth utility functions has been impeded, however, by the absent of efficient formulation of application-specific utility functions. In this paper, we treat bandwidth utility function as part of the service level specifications (SLS) solely defined by the service providers for different service classes. The pricing component of a bandwidth utility function remains unclear as the service providers are still searching for the appropriate pricing structure for the Internet: flat rate pricing or congestion-based pricing. Therefore, without the loss of generality, we focus on the price-independent version of utility function and leave pricing as an additional augment to the formulation of utility function.

Even though there remains a significant challenge to formulate generic utility function for a wide range of applications, progress has been made in utility function modeling for individual types of applications. For audio/video applications using UDP, the bandwidth utility function is related to the distortion-rate function [13] used in information theory. In this case, the utility of a given bandwidth is the reverse of the quality distortion under this bandwidth due to information loss at the encoder

(i.e., rate-controlled encoding) or inside network (i.e., media scaling). Because distortion-rate functions are usually content and encoder dependent, a practical approach to utility generation for video/audio content is to measure the distortion given different amount of scaled-down bandwidth. Distortion could be measured using subjective metrics such as the 5-level mean-opinion-score (MOS) test [7] to construct utility function off-line [14], or using objective metrics such as the Signal-to-Noise Ratio (SNR) whose simplicity allows online utility generation [10]. Figure 2 shows an example of utility function generated for an MPEG-1 video trace [10] using an on-line method. Each curve is constructed as the envelope of the per-frame rate-distortion function for the previous generation interval. The per-frame rate-distortion function is obtained by a dynamic rate shaping mechanism [4] which regulates the rate of MPEG traffic by dropping the minimal-distortion coefficients from the MPEG frames.

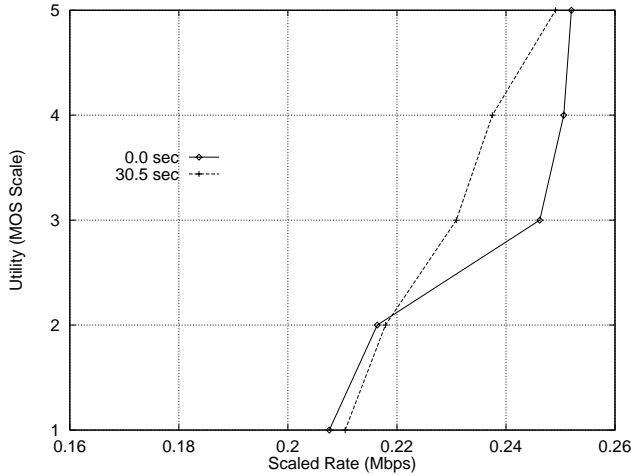


Fig. 2. Results for on-line utility function generation for a 30 second generation interval. Each curve is labeled by the generation time instant. Utility values are scaled using the 5-level MOS test.

Extending utility formulation from individual application to flow aggregates needs to resolve the issue of utility aggregation, which relates to specific utility-based allocation policies. There are generally two types of allocation policies: maximizing the sum of the utility (i.e., welfare-maximum) and fairness. In this paper, we consider the *welfare-maximum* and *proportional utility-fair* allocation policies. Proportional utility-fair allocates bandwidth to each flow (or flow aggregate) such that the proportion of resulting utility over the corresponding maximum utility will be the same for all flows (or flow aggregates). These allocation policies are discussed in Section 4.

For TCP-like reliable transport protocols, the effect of packet drops does not cause information distortion but loss of goodput due to retransmissions and congestion avoidance algorithms. Therefore, a distortion based bandwidth utility function is not applicable in this case. Rather, we choose to model the packet loss effect on TCP goodput. We define a normalized utility function for TCP as  $U(x) = 1 - \frac{\text{goodput}}{\text{throughput}} \approx 1 - \frac{p \cdot x}{x} = 1 - p$ , where  $p$  is the packet loss rate. This optimistic approximation of utility valuation is based on selective acknowledgement (SACK) TCP steady-state behavior under the condition of light to moderate packet losses, which is a reasonable assumption for a core network with provisioning. For the aggregation of TCP flows

experiencing the same order of packet losses, the normalized aggregated utility function is  $U_{agg\_TCP}(x) = 1 - \frac{\sum \text{goodput}}{\sum \text{throughput}} \approx p \cdot \frac{\sum x}{\sum x} = 1 - p$ , which is the same as the individual utility function. The value of  $p$  can be derived from TCP steady-state throughput-loss formula given by the inequality (4) in [12] as  $x < \left(\frac{MSS}{RTT}\right) \frac{1}{\sqrt{p}}$ , where  $MSS$  is the maximum segment size and  $RTT$  is the round trip delay. If we denote  $b_{min}$  the minimum bandwidth for TCP flow (aggregate) with a non-zero utility valuation,  $b_{min} = n \frac{MSS}{RTT}$ , where  $n$  is the number of active flows in the aggregate, then we have an upper bound on loss rate:  $p < \frac{b_{min}^2}{x^2}$ , and

$$U_{agg\_TCP}(x) = 1 - \frac{b_{min}^2}{x^2}. \quad (1)$$

In the DiffServ service profile,  $b_{min}$  can be defined as part of the service plan taking into consideration the service charge, the size of flow aggregate ( $n$ ) and the average round trip delay ( $RTT$ ). For example, one can have two types of utility functions modeling TCP sessions passing only one or more than one core network, respectively. The latter case can have a  $b_{min}$  one third of the first one assuming that on average a session passes three core networks if it passes more than one.

To simplify operation, we quantize a utility function into a continuous and monotonically increasing piece-wise linear function. The  $k$ th segment of a piece-wise linear utility function  $U_{\cdot,k}(x)$  is denoted as

$$U_{\cdot,k}(x) \triangleq \eta_{\cdot,k}(x - b_{\cdot,k}) + u_{\cdot,k}, \quad \forall x \in [b_{\cdot,k}, b_{\cdot,k+1}), \quad (2)$$

where  $\eta_{\cdot,k} > 0$  is the slope.

For TCP utility functions, because  $U(x) \rightarrow 1$  only when  $x \rightarrow \infty$ , we approximate the maximum bandwidth by setting it to a value corresponding to 95% of the maximum utility, i.e.,  $b_{\cdot,K} = b_{min}/\sqrt{0.05}$ .

We denote the piece-wise linear utility function as a vector of its first-order discontinuity points such that:

$$\left\langle \left( \begin{array}{c} u_{i,1} \\ b_{i,1} \end{array} \right) \cdots \left( \begin{array}{c} u_{i,K_i} \\ b_{i,K_i} \end{array} \right) \right\rangle \quad (3)$$

and from Equation 1, we have the vector representation for TCP aggregated utility function as:

$$\left\langle \left( \begin{array}{c} 0 \\ b_{i,min} \end{array} \right) \left( \begin{array}{c} 0.2 \\ 1.12b_{i,min} \end{array} \right) \left( \begin{array}{c} 0.4 \\ 1.29b_{i,min} \end{array} \right) \left( \begin{array}{c} 0.6 \\ 1.58b_{i,min} \end{array} \right) \right. \\ \left. \left( \begin{array}{c} 0.8 \\ 2.24b_{i,min} \end{array} \right) \left( \begin{array}{c} 1 \\ 4.47b_{i,min} \end{array} \right) \right\rangle \quad (4)$$

Figure 3 illustrates an example of bandwidth utility function and its corresponding piece-wise linear approximation for TCP aggregate with  $b_{min} = 1$  Mb/s.

For individual non-adaptive applications, the bandwidth utility function will be of convex functional form. The following proposition supports the generalization for convex utility aggregation.

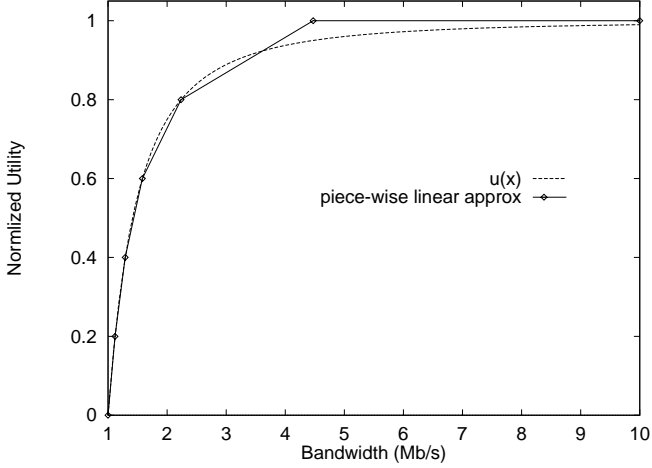


Fig. 3. Example of Normalized  $U(x)$  for TCP,  $b_{min} = 1$  Mb/s

*Proposition 1:* For flows with convex utility functions, welfare-maximum allocation is equivalent to sequential allocation; that is, the allocation will satisfy one flow to its maximum utility before assigning available bandwidth to another flow.

**Proof:** The proof is based on the well known convex analysis result that for convex functions, the maximum solution lies at the extreme points of the convex hull ([15], Section 32).  $\square$

When a flow aggregate contains a large number of these non-adaptive applications with convex bandwidth utility functions, under welfare maximization, from Proposition 2, the aggregated bandwidth utility function is a cascade of individual convex utility functions. The aggregated bandwidth utility function can be approximated as a linear function with the slope of  $u_{max}/b_{max}$  between the two points  $(0, 0)$ , and  $(nb_{max}, nu_{max})$ , where  $n$  is the number of flows, where  $b_{max}$  is the maximum required bandwidth and  $u_{max}$  is the corresponding utility of each individual application. In other words,

$$\begin{aligned} U_{agg\_rigid}(x) &= U_{single}\left(x - \left\lfloor \frac{x}{b_{max}} \right\rfloor\right) + \left\lfloor \frac{x}{b_{max}} \right\rfloor u_{max} \\ &\approx \left(\frac{u_{max}}{b_{max}}\right)x, \quad \forall x \in [0, nb_{max}] \end{aligned} \quad (5)$$

In summary, we generalize the aggregation of bandwidth utility functions based on the following application categories:

- TCP-based application aggregates: Formulation 4;
- UDP-based audio/video application aggregates, where each application consumes small amount of bandwidth in comparison to the link capacity: Equation 5; and
- UDP-based audio/video application with large bandwidth consumption in comparison to the link capacity: measured distortion-rate function

Service providers could use the above methods to decide the shape of a bandwidth utility function for a specific service class, and then control the parameters  $b_{min}$ ,  $b_{max}$ , and  $u_{max}$  based on the administrative policy, pricing plan, and the requirement for service differentiation.

## IV. UTILITY-BASED ALLOCATION POLICIES

### A. Welfare-Maximum Allocation

Welfare-maximum allocation distributes the link capacity  $C$  into per-flow (aggregate) allocations  $\mathbf{x} = (x_1, \dots, x_n)$  to maximize  $\sum_{k=1}^n U_k(x_k)$  under constraint  $\sum_{k=1}^n x_k = C$  and  $x_k \geq 0$ .

This maximization problem with target functions that are not always concave is an NP-hard [9] problem. In the case of convex utility functions, the optimal solution lies at the extreme points of the convex hull with the optimal solution only being found by enumerating through all the extreme points. In the Q-RAM project [9], various algorithms have been investigated to reduce the complexity of aggregating utility functions. In the following, we improve the Q-RAM method by exploiting the structure of piece-wise linear utility functions and reducing the algorithm's searching space.

One direct result from the Kuhn-Tucker [8] necessary condition for maximization is that:

*Proposition 2:* At the maximum-utility allocation  $(x_1^*, \dots, x_n^*)$ , the allocation to  $i$  belongs to one of the two sets: either  $i \in \mathcal{D} \triangleq \{j \mid U'_j(x_j^* -) \neq U'_j(x_j^* +)\}$ , namely  $x_i^*$  is at a first-order discontinuity point of  $U_i(x)$ ; or otherwise,  $\forall i, j \in \overline{\mathcal{D}}$ ,  $U_i(x_i^*)$  and  $U_j(x_j^*)$  have the same slope:  $U'_i(x_i^*) = U'_j(x_j^*)$ . In addition, the slope has to meet the condition that

$$U'_j(x_j^* -) \geq U'_i(x_i^*) \geq U'_j(x_j^* +), \quad \forall i \in \overline{\mathcal{D}} \text{ and } j \in \mathcal{D} \quad (6)$$

The intuition behind the formulation is that for  $i, j \in \overline{\mathcal{D}}$ , they must have the same slope, otherwise total utility can be increased by shuffling bandwidth from the one with lower slope to the other. By the same argument, the slope of  $U_i(x_i^*)$ ,  $i \in \overline{\mathcal{D}}$  has to be no greater than the slope of  $U_j(x_j^* -)$ , but no smaller than that of  $U_j(x_j^* +)$ , for  $j \in \mathcal{D}$ .

Inequality (6) is important to improve the efficiency of the `combine_and_merge()` and `resource_sieve()` methods proposed in [9]. When aggregating two piece-wise linear utility functions  $U_i(x)$  and  $U_j(x)$ , the aggregated utility function is composed from the set of shifted linear segments of  $U_i(x)$  and  $U_j(x)$ , which can be represented by  $\{U_{i,l}(x - b_{j,m}) + u_{j,m}, U_{j,m}(x - b_{i,l}) + u_{i,l}\}$  with  $l = 0, 1, \dots, K(i)$ , and  $m = 0, 1, \dots, K(j)$ . From Inequality (6), we can remove at least one of  $U_{i,l}(x - b_{j,m}) + u_{j,m}$  and  $U_{j,m}(x - b_{i,l}) + u_{i,l}$  from the set because they can not both satisfy the inequality. In addition, when  $U_i(x)$  is convex, all  $U_{j,m}(x - b_{i,l}) + u_{i,l}$  except  $l = 0$ , or  $K(i)$  will be removed. This will significantly reduce the operating space for the `combine_and_merge()` operation discussed in [9].

### B. Proportional Utility-Fair Allocation

The utility-fair allocation algorithm presented in [2] aims to maximize total utility but under the constraint that all flows get the same utility value. However, because this algorithm requires the utility value to be normalized, it does not inter-operate with welfare-maximum allocation, nor can it accommodate service classes that prefer different  $u_{max}$ .

We modify [2] to support *proportional utility-fair*. Instead of giving the same absolute utility value as in [2], proportional utility-fair gives the same proportion of the maximum utility

value to each flow (aggregate). In addition, we generalize the formulation by removing the constraint given in [2] that all individual utility functions need to have the same discrete utility levels (i.e.,  $u_{i,k} = u_{j,k}$ ).

We denote the normalized discrete utility levels of a piecewise linear function  $u_i(x)$  as a set  $\left\{\frac{u_{i,k(i)}}{u_i^{max}}\right\}$ . Then the aggregated utility function  $u_{agg}(x)$  will have a set as the union of each individual set  $\bigcup_i \left\{\frac{u_{i,k(i)}}{u_i^{max}}\right\}$ . We rename the members of the set sorted in ascending order as  $\psi_k$ .

Under this policy, the aggregated utility function becomes:

$$U_{agg}(x) = \frac{(\psi_{k+1} - \psi_k)u_{agg}^{max}}{b_{agg,k+1} - b_{agg,k}}(x - b_{agg,k}), \quad (7)$$

$\forall x \in [b_{agg,k}, b_{agg,k+1})$ , where  $u_{agg}^{max} = \sum_i u_i^{max}$ , and  $b_{agg,k} = \sum_i U_i^{-1}(\psi_k u_i^{max})$ .

Given a link capacity  $C$ , the resulting allocation  $x_i$  and utility value  $u_i$  to each flow (aggregate) is:

$$u_i = \frac{U_{agg}(C)}{u_{agg}^{max}} u_i^{max}, \text{ and } x_i = U_i^{-1}(u_i). \quad (8)$$

**Remark:** When all the  $U_i(x)$  are the same, the outcome of the algorithm is equal allocation, which is the same result as the max-min allocation for a single link.

### C. Aggregated Utility Functions

The aggregated utility function under proportional utility-fair allocation contains the state for all individual utility functions. When a utility function is removed from the aggregated utility function, the reverse operation of Formula 7 does not involve other individual utility functions.

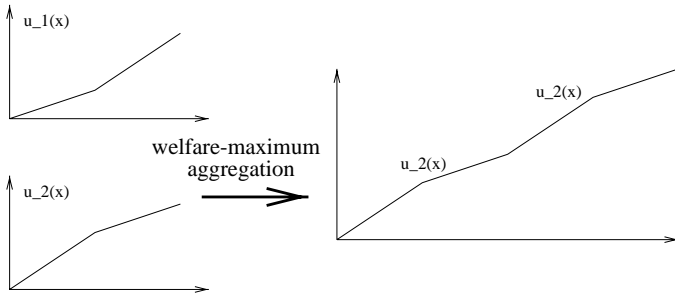


Fig. 4. Example of Utility Aggregation under Welfare-Maximum Policy

However, this is not the case for the welfare-maximum policy. As shown in Figure 4,  $u_1(x)$  is convex and  $u_2(x)$  is concave. The aggregation of these two functions only contains information of the concave function  $u_2(x)$ . When  $u_2(x)$  is removed from the aggregated utility function, there is insufficient information to reconstruct  $u_1(x)$ . In this sense the utility function state is not scalable under welfare-maximum allocation. Because of this reason and complexity, welfare-maximum allocation should not be used for large numbers of flows (aggregates) with convex utility.

## V. INGRESS DYNAMIC LINK SHARING

Dynamic provisioning algorithms in the core network need to react to persistent network congestion. This naturally leads

to time-varying rate allocation at the edge traffic conditioners. This poses a significant problem for link sharing under time-varying link capacity. Link sharing algorithms such as CBQ [6] focus on differentiating bandwidth distribution under static link capacity. CBQ employs administrative rules to guide sharing. When the link capacity is time-varying, the distribution policy needs to dynamically adjust the bandwidth allocation for individual flows. Consequently, quantitative distribution rules based on bandwidth utility functions are desired to dynamically guide the distribution of bandwidth.

### A. $U(x)$ -CBQ Link-Sharing Traffic Conditioner

In this section, we present the architecture of our  $U(x)$ -CBQ traffic conditioner. As illustrated in Figure 5, the  $U(x)$ -CBQ traffic conditioner regulates all users' traffic sharing the same network service class at an ingress link to a core network. The CBQ link sharing structure comprises two levels of policy-driven weight allocations. At the upper tier, each CBQ agency corresponds to one DiffServ service profile subscriber. The "link sharing weights" are allocated by the proportional utility-fair policy to enforce fairness among users subscribing to the same service plan. Because each aggregated utility function is truncated to  $b_{max}$ , users subscribing to different plans (i.e., different  $b_{max}$ ) will also be handled in a proportional utility-fair manner.

At the lower tier, within each agency, the sharing classes are categorized by application types with respect to their associated utility function characteristics. In Figure 5, we use the three application types discussed in Section III, namely TCP aggregates, aggregate of a large number of small-size non-adaptive applications, and individual large-size adaptive video applications. The TCP aggregates can be further classified into two aggregates for intra- and inter-core networks, respectively.

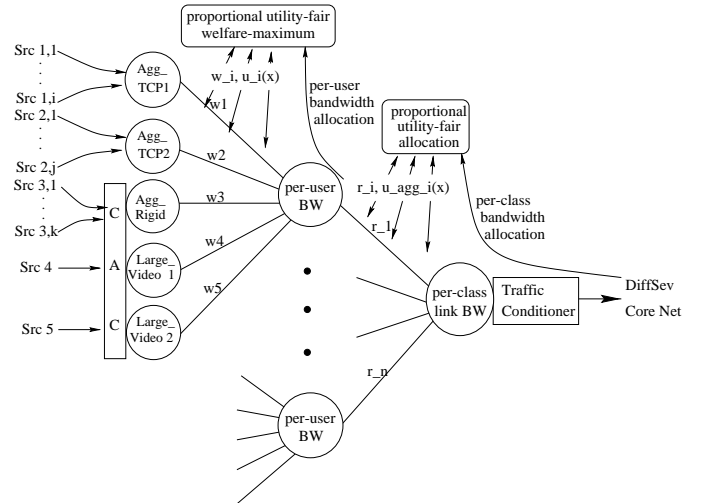


Fig. 5.  $U(x)$ -CBQ based Traffic Conditioner

We use the CBQ formal link-sharing guidelines as described in [6]. We choose the weighted round robin (WRR) algorithm as the scheduler for CBQ because the service weight of each class provide a clean interface for our utility-based allocation algorithms to program CBQ.

CBQ was originally designed to support packet scheduling rather than traffic shaping/policing. When CBQ is used as a traf-

fic policer instead of traffic shaper, the scheduling buffer should be removed. In our simulation, we choose a small buffer size (1-2 packets) for every leaf class just to ensure the correct operation of the CBQ WRR scheduler. We use the same priority for all leaf classes of a CBQ agency because priority in traffic shaping/policing does not reduce traffic burstiness.

In CBQ, the link sharing weights control the proportion of bandwidth allocated to each class. Therefore administering sharing weights is equivalent to allocating bandwidth. In this section, we introduce a *hybrid allocation* policy for CBQ sharing weights that represents a hybrid of constructed from proportional utility-fair and welfare-maximum. The hybrid allocation policy is motivated by the different behavior of adaptive and non-adaptive applications.

At the highest level, we use proportional utility-fair policy to administer sharing weights based on each user’s service profile and monthly charge. At the lowest level (i.e., utility aggregation level), adaptive applications with homogeneous concave utility functions (e.g., TCP) are aggregated under proportional utility-fair policy. In this case (see Proposition 2) proportional utility-fair and welfare-maximum are equivalent. In the case of non-adaptive applications with convex utility functions, they only need to be aggregated under the welfare-maximum policy. Otherwise, a bandwidth reduction will reduce the utility of all the individual flows in a significant manner due to the convex nature of the individual utility function. For this reason we introduce an admission control (CAC) module as illustrated in Figure 5. The role of admission control is to safeguard the minimum bandwidth needs of individual video flows that have large bandwidth requirements, and the bandwidth needs of non-adaptive applications at the ingress link. These measures help avoid the random dropping/marking of non-adaptive traffic aggregate by traffic conditioners that can affect all the individual flows within an aggregate. One can limit the impact to a few individual flows and hence maintain the welfare-maximum allocation using measurement-based admission control.

At the middle level, we have the flexibility to use either of the allocation policies to distribute sharing weights among different flows (aggregates) for the same user. Mixing policy in this manner causes no conflict due of the link sharing hierarchy. There is no clear argument, however, that one policy is strictly better than the other. The welfare-maximum policy has clear economic meaning and can provide incentive compatibility for applications to cooperate. When bandwidth changes occur, it usually only adjusts the allocation to one flow, rather than all the flows under proportional utility-fair allocation. As discussed in Section IV, the welfare-maximum suffers from complexity and scalability problems. In the case of our framework we leave the choice of policy for the middle level to user and service profile providers.

## B. Simulation Results

We evaluate our algorithms using the ns simulator with its built in CBQ [6] and DiffServ modules [18]. Unless otherwise specified, we use the default values in the standard ns release for simulation parameters.

The goal of our simulation is to evaluate the effect of welfare-maximum and proportional utility-fair policies in handling a

combination of adaptive and non-adaptive applications. Because the packet level performance of CBQ has been widely studied in the literature, our simulation results will focus on how to adjust the sharing weights under time-varying link capacity.

The simulation topology is a simplified version of the one shown in Figure 5; that is, one access link shared by two agencies. The access link has DiffServ AF1 class bandwidth varying over time. The maximum link capacity is set to 10 Mb/s. Each agency represents one user profile. Agency A has a maximum bandwidth quota  $b_{A,max} = 8Mb/s$ , which is twice as much as  $b_{B,max} = 4Mb/s$ . This does not necessarily translate into a doubled bandwidth allocation for user A because the exact allocation depends on the shape of the aggregated utility function. This is a nice feature of utility-based allocation, which is capable of realizing complex application-dependent and capacity-dependent allocation rules.

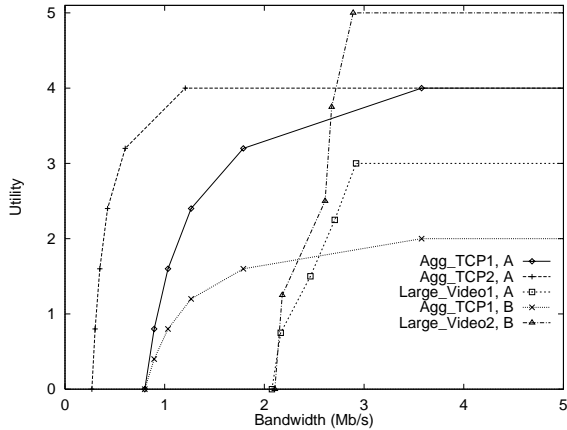
The leaf classes for agency A are Agg\_TCP1, Agg\_TCP2, and Large\_Video1, and the leaf classes for agency B are Agg\_TCP1 and Large\_Video2. We do not explicitly simulate the admission control module and the Agg\_Rigid leaf class because their effect on bandwidth reservation can be incorporated into the  $b_{min}$  value of the other aggregated classes.

Following the same approach as [6], we concentrate on the study of link sharing policies free from extraneous factors. A single constant-bit-rate source for each leaf class is used where each has a peak rate higher than the link capacity. The packet size is set to 1000 bytes for TCP aggregates and 500 bytes for video flows.

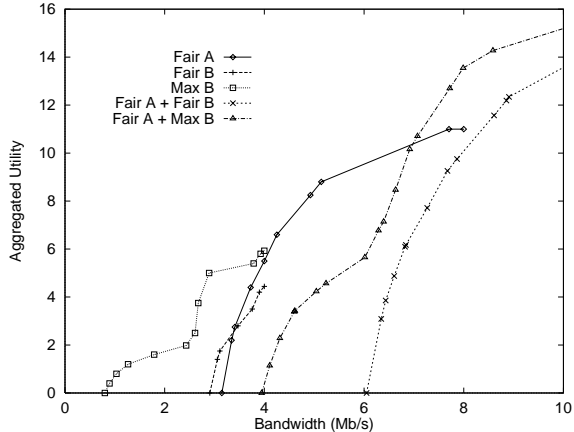
We use the formula from Equation 4 to set the utility function for Agg\_TCP1 and Agg\_TCP2, where  $b_{min}$  for Agg\_TCP1 and Agg\_TCP2 is chosen as 0.8 Mb/s and 0.27 Mb/s, respectively, to reflect a 100 ms and 300 ms *RTT* in intra-core and inter-core cases. In both cases, the number of active flows in each aggregate is chosen to be 10, and *MSS* is set to 8 Kb. We specify the maximum utility value  $u_{max}$ : for agent A, we set  $u_{max}$  to be 4 for Agg\_TCP1 and Agg\_TCP2, and for agent B,  $u_{max} = 2$ , so that agency A has a higher grade service profile than agency B both in terms of  $b_{max}$  and  $u_{max}$ . The two utility functions for Large\_Video1 and Large\_Video2 are measured from the MPEG1 video trace discussed in Section III.

Figure 6 illustrates all the utility functions used in the simulation. Figure 6(a) shows the individual utility functions, while 6(b) displays the aggregate utility functions under proportional utility-fair for agency A and B, under welfare-maximum for B, and under proportional utility-fair at the top level. Here we verify that the proposed proportional utility-fair and welfare-maximum formula defined in Section IV can be applied to complex aggregation operations of piece-wise linear utility functions with different discrete utility levels,  $u_{max}$ ,  $b_{min}$  and  $b_{max}$ .

In the following simulations, we set up two test scenarios. In the first scenario, proportional utility fair policy is used at all link sharing levels. In the second scenario, welfare-maximum policy is adopted for agency B only. The assigned link capacity to this service class starts from 90% of the link capacity and then reduces to 80% and 70% at 20 and 35 seconds, respectively, before finally increasing to 100% of the physical link capacity. This sequence of changes invokes the dynamic link sharing algorithms to adjust the link sharing ratio for individual classes.



(a) Individual Utility Function



(b) Aggregated Utility Function

Fig. 6. Bandwidth Utility Function used in the Simulation

Our simulation results are shown in Figure 7 and 8. The three plots represent traces of throughput measurement for each flow (aggregate). Bandwidth values are presented as relative values of the ingress link capacity.

Figure 7 presents the link sharing effect with time-varying link capacity. The first observation is that the hybrid link-sharing policies do not cause any policy conflict. The difference between the aggregated allocation under the first and second scenario are as a result of the different shape of aggregated utility functions for agency B as illustrated in Figure 6(b) where one is aggregated under proportional utility fair and the other under welfare maximum. Other than this, the top level link sharing treats both scenarios equally.

We have argued that bandwidth utility functions provide a rich programming tool to implement complex link sharing policies. We can appreciate this by studying the effectiveness of controlling  $b_{A,max}$  and  $b_{B,max}$ . Since we limit  $b_{B,max}$  to 4 Mb/s, the two aggregated utility functions of agency B are truncated at 4 Mb/s as shown in Figure 6(b). This equally limits the allocation of agency B below 4 Mb/s, which is verified by the bottom two traces in Figure 7.

We also observe the steep rise of agency A's allocation when the available bandwidth is increased from 7 to 10 Mb/s. The reason for this is that agency B's aggregated utility function rises sharply toward the maximum bandwidth, while agency A's

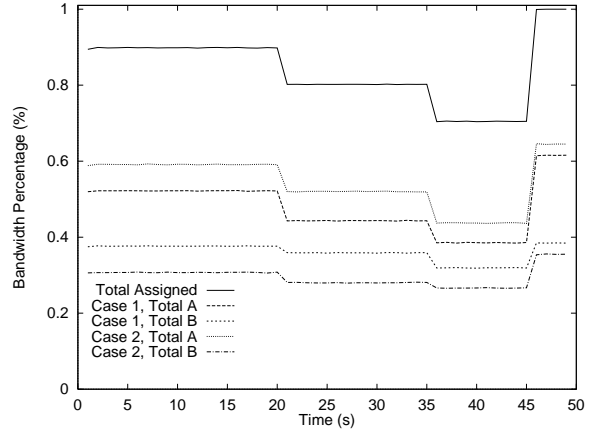


Fig. 7. Top-tier Link Sharing

aggregated utility function is relatively flat as shown in Figure 6(b). Under conditions where there is an increase in the available bandwidth, agency A will take a much larger proportion of the increased bandwidth with the same proportion of utility increase.

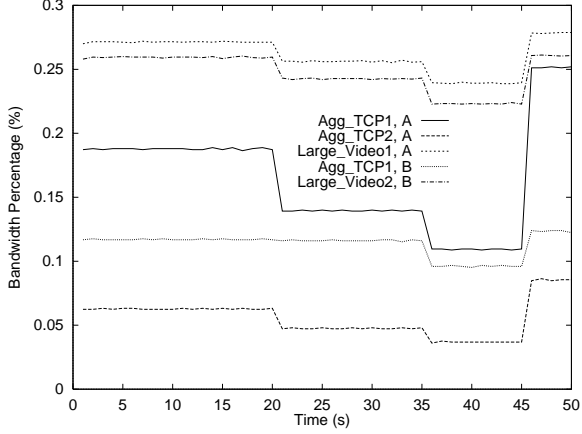
In Figure 8(a) and (b), we present the lower-tier link sharing results within the leaf classes of agency A and B, respectively. In both figures, we observe the effect of using  $u_{max}$  to differentiate bandwidth allocation. As shown in Figure 6(a), within agency B, we choose a large  $u_{max} = 5$  for the Large\_Video2 flow while at the same time a small  $u_{max} = 3$  for the Agg\_TCP1 flow aggregate. The differentiation in bandwidth allocation is visible for the first scenario of proportional utility-fair mainly from the large  $b_{min}$  of the Large\_Video2 flow. However, this allocation differentiation is increased in the second scenario of welfare-maximum allocation. In fact, Agg\_TCP1 is consistently starved as shown at the bottom of Figure 8(b), while the allocation curve of Large\_Video2 appears at the top of the plot.

Through simulation, we have verified the effectiveness of our U(x)-CBQ algorithm. We have also identified several control parameters that can be adjusted to offer differentiated service. These include the maximum subscribed bandwidth at the agency level, the maximum utility value of a bandwidth utility function, the minimum and maximum bandwidth of a utility function, and most importantly, the bandwidth utility function itself.

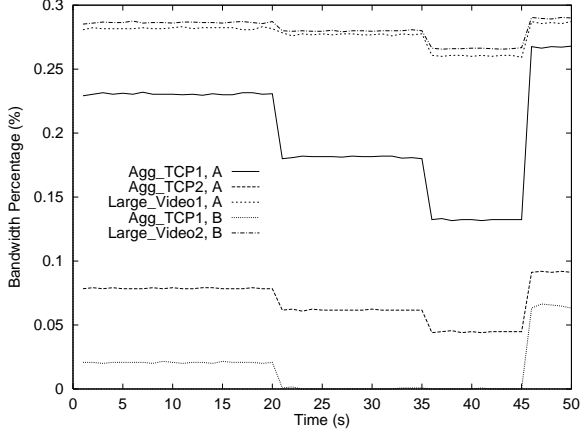
## VI. EGRESS DYNAMIC CAPACITY DIMENSIONING

Egress provisioning faces two challenges. First, the capacity calculation for the egress link needs to reflect the bottleneck capacity inside the core and at the ingress of the network. Second, for a core network with multiple egress links to peering/transit networks, capacity negotiation for egress links need to be coordinated.

The solution that addresses the first challenge lies in the traffic load matrix illustrated in Figure 1, which formulates the traffic load dependency of all egress links on ingress links. The relation between ingress link capacity and egress link capacity is modeled by a sink tree rooted at an egress link with leaves at the ingress links contributing traffic to egress links. In what follows, we outline the core traffic load matrix discussed in [11] and describe its application to egress capacity dimensioning.



(a) Proportional Utility-Fair for A and B



(b) Welfare-Maximum for B only

Fig. 8. Lower-tier Link Sharing

### A. Traffic Load Matrix

Let us consider a core network with a set  $\mathcal{L} \triangleq \{1, 2, \dots, L\}$  of link identifiers of *per-class* unidirectional links. Then one physical link is modeled as  $2J$  unidirectional links. Let  $c_l$  be the finite capacity of link  $l, l \in \mathcal{L}$ . Similarly, let the set  $\mathcal{K} \triangleq \{1, 2, \dots, K\}$  denote the set of *per-class* nodes in a core network, and specifically, the set of *per-class* edge nodes is denoted as  $\mathcal{E}, \mathcal{E} \subset \mathcal{K}$ .

A core network traffic load plays an important role in our model and consists of a matrix  $\mathbf{A} = \{a_{l,i}\}$  that models the *per* DiffServ user traffic distribution on links  $l \in \mathcal{L}$ , where  $a_{l,i}$  indicated the fraction of traffic from user  $i$  passing link  $l$ . Let the link load vector be  $\mathbf{c}$  and user traffic vector be  $\mathbf{u}$ , then we have:

$$\mathbf{c} = \mathbf{A}\mathbf{u}. \quad (9)$$

Without loss of generality, we can rearrange the columns of  $\mathbf{A}$  into  $J$  sub-matrices, one for each class. Then we have:  $\mathbf{A} = [\mathbf{A}(1) : \mathbf{A}(2) : \dots : \mathbf{A}(J)]$  and  $\mathbf{u} = [\mathbf{u}(1) : \mathbf{u}(2) : \dots : \mathbf{u}(J)]^T$ .

The construction of matrix  $\mathbf{A}$  is based on the measurement of its column vectors  $\mathbf{a}_{\cdot,i}$ , each represents the traffic distribution of one user  $i$ . There are several methods to compose the matrix  $\mathbf{A}$  from distributed traffic measurement. In this paper we assume that the matrix is updated in a timely manner by using the methods proposed in [11].

The interdependence of egress and ingress link capacity provisioning can also be modeled by using the traffic load matrix  $\mathbf{A}$ .

We can rearrange the rows of  $\mathbf{c}$  and  $\mathbf{A}$  so that  $\mathbf{c} = \begin{bmatrix} \mathbf{c}_{core} \\ \dots \\ \mathbf{c}_{out} \end{bmatrix}$ ,

which represents the capacity of internal links of the core network and egress links, respectively, and  $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{core} \\ \dots \\ \mathbf{A}_{out} \end{bmatrix}$ . The relationship between ingress link and egress link capacity then becomes:

$$\mathbf{c}_{out} = \mathbf{A}_{out}\mathbf{u}. \quad (10)$$

Figure 9 illustrates an example of the relationship between egress and ingress link capacity. Each row of the matrix  $\mathbf{A}_{out}$ , i.e.,  $\mathbf{a}_{i,\cdot}$ , represents a sink-tree rooted at egress link  $c_i$ . The leaf nodes of the sink-tree represented ingress user traffic aggregates  $\{u_j \mid a_{i,j} > 0\}$ , which contributes traffic to egress link capacity  $c_i$ .

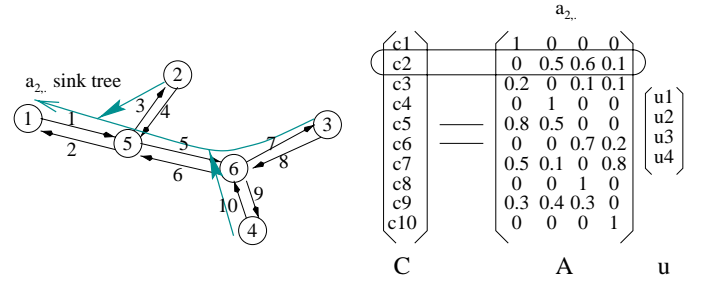


Fig. 9. Example of a Egress-to-Ingress Sink-Tree

### B. Coordinated Egress Provisioning

We approach the second challenge of coordinating the capacity negotiation of multiple egress links using dynamic programming. The partition of  $\mathbf{c} = \mathbf{A}\mathbf{u}$  into  $\mathbf{c}_{out} = \mathbf{A}_{out}\mathbf{u}$  and  $\mathbf{c}_{core} = \mathbf{A}_{core}\mathbf{u}$  forms the basis for dynamic programming. In what follows, we discuss the operations involved in egress provisioning.

#### B.1 Calculating Ideal Egress Capacity

First, we calculate the ideal egress link capacity by assuming all the egress links are not bottlenecks. With the traffic load matrix, the resulting optimal bandwidth allocation at ingress links will give the ideal capacity dimension at the egress links.

By assuming  $\mathbf{c}_{out} = \infty$  in  $\mathbf{c} = \mathbf{A}\mathbf{u}$ , the matrix equation constraint becomes equivalent to  $\mathbf{c}_{core} = \mathbf{A}_{core}\mathbf{u}$ . Then under the constraint of  $\mathbf{A}_{core}\mathbf{u} \leq \mathbf{c}_{core}$ , with a modified max-min fair allocation<sup>1</sup>, we have the optimal ingress bandwidth allocation  $\hat{\mathbf{u}}(n)$ . Consequently, we have one sample of the ideal egress link capacity:  $\hat{\mathbf{c}}_{out}(n) = \mathbf{A}_{out}\hat{\mathbf{u}}(n)$ .

The actual capacity vector  $\hat{\mathbf{c}}_{out}$  used for capacity negotiation is obtained as a probabilistic upper-bound on  $\{\hat{\mathbf{c}}_{out}(n)\}$  for control robustness. The bound can be obtained by using the techniques employed in measurement based admission control (e.g., the Chernoff bound).

<sup>1</sup>The algorithm is modified from the standard max-min fair algorithm [1]. The detection of the most congested link is changed to take into consideration the tree structure of a DiffServ traffic aggregate rather than a single pipe. See [11] for full details.

## B.2 Constructing Egress Utility Functions

Using the same approach, we can construct egress bandwidth utility functions for use at the ingress traffic conditioners of peering networks. The utility function  $U_i(x)$  at egress link  $i$  is calculated by aggregating all the ingress aggregated utility functions  $\{U_j(x) \mid a_{i,j} > 0\}$  under the proportional utility fair formula of Equation (7). In addition, each  $U_j(x)$  is scaled in bandwidth by a multiplicative factor  $a_{i,j}$  because only  $a_{i,j}$  portion of ingress  $j$  traffic passes through egress link  $i$ . Because of the property of proportional utility-fair allocation, the egress-aggregated utility function will have  $u_i^{max} = \sum_{j:a_{i,j}>0} u_j^{max}$ . This property that aggregated utility value is equal to the sum of individual utility value is important in DiffServ because traffic conditioning in DiffServ is for flow aggregates. The bandwidth decrease at any one egress link will cause the corresponding ingress links to throttle back even though only a small portion of traffic may be flowing through the congested egress link.

We can also use the same techniques to obtain a probabilistic bound  $\hat{U}_i(x)$  on the samples of  $\{U_i(x, n)\}$ . One such algorithm is developed in [10]. Because we use proportional utility-fair for allocation, the probabilistic bound is a lower-bound on utility which translates into an upper-bound on allocated bandwidth.

## B.3 Peer Network Coordination

With  $\hat{c}_{out}$ , egress links can negotiate with peering/transit networks with or without market based techniques (e.g., auction). When the peer network supports the  $U(x)$ -CBQ traffic conditioner,  $\hat{U}_i(x)$  becomes essential to create a scalable bandwidth provisioning architecture. The egress link  $i$  can become a regular subscriber to its peering network by submitting the utility function  $\hat{U}_i(x)$  to the  $U(x)$ -CBQ traffic conditioner. A peer network does not need to treat its network peers in any special manner because the aggregated utility function will reflect the importance of a network peer via  $u_{max}$  and  $b_{min}$ .

The outcome from bandwidth negotiation/bidding is a vector of allocated egress bandwidth  $\mathbf{c}_{out}^* \leq \hat{\mathbf{c}}_{out}$ . Because inconsistency can occur in this distributed allocation operation, to avoid bandwidth waste, a coordinated relaxation operation is required to calculate the accepted bandwidth  $\tilde{\mathbf{c}}_{out}$  based on the assigned bandwidth  $\mathbf{c}_{out}^*$ . One approach is proportional reduction:

$$\tilde{c}_{out} = \gamma \hat{c}_{out}, \text{ where } \gamma = \min_i \left\{ \frac{c_i^*}{\hat{c}_i} \right\}.$$

This is the method used in our prior work [16] which assumes a single bottleneck in each core network. However, when a core network has multiple bottleneck links, proportional reduction is over conservative. Rather, we put  $\mathbf{c}_{out}^*$  in  $\mathbf{c} = \mathbf{A}\mathbf{u}$  to calculate  $\tilde{\mathbf{u}}$  by a modified max-min fair algorithm. Subsequently,  $\tilde{\mathbf{c}}_{out} = \mathbf{A}_{out}\tilde{\mathbf{u}}$ .

Because an egress capacity dimensioning module will interact with peer/transit networks in addition to its local core network, we expect that egress capacity dimensioning will operate over slower time scales than ingress capacity provisioning to improve algorithm robustness to local perturbations. The sequence of operations presented in this section is a single provisioning adjustment. The interaction among different core networks can cause a series of provisioning adjustments that may never

converge. In [16], we identify the convergence conditions for single-bottleneck core networks. Applying the same approach to a general core network setting is much more challenging. We are currently addressing this issue in our research.

## VII. CONCLUSION

In this paper, we have proposed an edge architecture for differentiated services which includes ingress and egress provisioning algorithms. The ingress algorithm supports utility-based dynamic link sharing policies under time-varying link capacity; and the egress algorithm is capable of delivering efficient capacity provisioning to peering networks with respect to the measured bandwidth demand from the core internal bottlenecks and ingress utility functions.

The ingress link sharing algorithm extends the work on CBQ [6] to cover time-varying link capacity with bandwidth utility functions. We present a general approach to formulate utility functions for TCP aggregates, non-adaptive application aggregates, and large size adaptive applications. Efficient algorithms are derived to calculate aggregated utility functions which are essential to bandwidth allocation. Two different sharing policies were investigated and a hybrid policy proposed which combines the benefit of both. The algorithms are designed to respond to changes caused by the arrival/departure of sessions whose time scales are orders of magnitude larger than the time scale of packet forwarding. Over this time scale, the overhead introduced by utility-based operations is negligible.

The edge capacity provisioning algorithm uses measured internal bottleneck bandwidth demands and edge utility functions to estimate the overall network utility function at the peering points, hence providing a mathematical tool capable of provisioning bandwidth on-demand. In addition to the core provisioning algorithms discussed in [11], our proposed edge provisioning architecture can help us move toward a more quantitative understanding of engineering the differentiated service Internet. We are currently implementing these algorithms using programmable routers.

## ACKNOWLEDGMENTS

This work is funded in part by a grant from the Intel Corporation for "Signaling Engine for Programmable IXA Networks".

## REFERENCES

- [1] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, Englewood Cliffs, NJ, 1992.
- [2] G. Bianchi, A. T. Campbell, and R. R.-F. Liao. On utility-fair adaptive services in wireless networks. In *IEEE IWQOS'98*, Napa Valley, CA, May 1998.
- [3] L. Breslau and S. Shenker. Best-effort versus reservations: a simple comparative analysis. In *Proc. ACM SIGCOMM*, September 1998.
- [4] A. Eleftheriadis and D. Anastassiou. Dynamic rate shaping of compressed digital video. In *Proc. of 2<sup>nd</sup> IEEE Intl. Conf. on Image Processing*, Arlington, VA, October 1995.
- [5] S. Black et. al. An architecture for differentiated services. IETF RFC 2475, December 1998. <ftp://ftp.isi.edu/in-notes/rfc2475.txt>.
- [6] S. Floyd and V. Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Trans. Networking*, 3(4):365–386, August 1995.
- [7] Recommendation ITU-R BT.500-7. Methodology for the subjective assessment of the quality of television picture, October 1999. ITU-R Recommendations.

- [8] H. W. Kuhn and A. W. Tucker. Non-linear programming. In *Proc. 2<sup>nd</sup> Berkeley Symp. on Mathematical Statistics and Probability*, pages 481–492. Univ. Calif. Press, 1961.
- [9] C. Lee, J. P. Lehoczky, R. Rajkumar, and D. Siewiorek. On quality of service optimization with discrete qos options. In *Proc. IEEE Real-time Technology and Applications Symposium*, June 1999.
- [10] R. R.-F. Liao, P. Bouklee, and A. T. Campbell. Online generation of bandwidth utility function for digital video. In *Proc. of PacketVideo'99*, New York City, April 26-27 1999.
- [11] R. R.-F. Liao and A. T. Campbell. A dynamic provisioning model for Differentiated Service. Technical report, Columbia University, 2000. Submitted for publication.
- [12] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *ACM Comput. Commun. Review*, 27, 1997.
- [13] A. Ortega and K. Ramchandran. Rate-distortion methods for image and video compression. *IEEE Signal Processing Magazine*, pages 23–50, November 1998.
- [14] D. Reininger and R. Izmailov. Soft quality of service with VBR<sup>+</sup> video. In *Proc. Int. Workshop on Audio-Visual Services over Packet Networks*, pages 207–211, Aberdeen, Scotland, UK, September 15-16 1997.
- [15] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.
- [16] N. Semret, R. R.-F. Liao, A. T. Campbell, and A. A. Lazar. Peering and provisioning of differentiated internet services. In *Proc. IEEE INFOCOM*, March 2000.
- [17] S. Shenker. Some fundamental design decisions for the future internet. *IEEE J. Select. Areas Commun.*, 13:1176–1188, 1995.
- [18] UCB/LBNL/VINT. Network simulator - ns.