

# An Integrated Resource Negotiation, Pricing, and QoS Adaptation Framework for Multimedia Applications

X. Wang, H. Schulzrinne  
Columbia University

xwang@ctr.columbia.edu, schulzrinne@cs.columbia.edu

*Abstract—*

We study a dynamic, usage- and congestion-dependent pricing system in conjunction with price-sensitive user adaptation of network usage. We first present a Resource Negotiation and Pricing (RNAP) protocol and architecture to enable users to select and dynamically re-negotiate network services. In the second part of the paper, we develop mechanisms within the RNAP architecture for the network to dynamically formulate prices and communicate pricing and charging information to the users. We then outline a general pricing strategy in this context. We discuss candidate algorithms by which applications (singly, or as part of a multi-application system) can adapt their rate and QoS requests, based on the user-perceived value of a given combination of transmission parameters. Finally, we present experimental results to show that usage and congestion-dependent pricing can effectively reduce the blocking probability, and allow bandwidth to be shared fairly among applications, depending on the elasticity of their respective bandwidth requirements.

*Keywords—* Communication system economics, Adaptive systems, Resource management, Communication system signaling, Communication system traffic, Computer network management, Multimedia communication

## I. INTRODUCTION

The development and use of distributed multimedia applications are growing rapidly. These applications usually require a minimum Quality of Service (QoS) from the network, in terms of throughput, packet loss, delay, and jitter. Also, multimedia applications on the Internet commonly employ the UDP transport protocol, which lacks a congestion control mechanism. These applications can therefore starve TCP applications (which perform congestion control) of their fair share of bandwidth.

Different approaches have been considered to address these problems:

1. In order to guarantee a certain QoS to the application, researchers have proposed mechanisms such as network resource reservation [2][3], admission control [9], special scheduling mechanisms [10], and differentiated or prioritized service at network switches [13];
2. Adaptation protocols and algorithms have been proposed to dynamically regulate the source bandwidth according to the existing network conditions (a survey of this work is given in [1].)

If resource reservation is done statically (before transmission), resource reservation and provisioning tend to be conservative due to the lack of quantitative knowledge of traffic statistics. Moreover, the resource allocation is based on initial availability of resources and does not take into account changes in availability during an ongoing transmission. Many multimedia applications are long-lived, exacerbating the problem.

Users of rate-adaptive applications do not have any incentive to scale back their sending rate below their access bandwidth, since selfish users will generally obtain better quality than those that reduce their rate.

Pricing network services based on the level of service, usage, and congestion provides a natural and equitable incentive for applications to adapt their sending rates according to network conditions. Increasing the price during congestion gives the application an incentive to reduce its sending rate and at the same time allows an application with more stringent bandwidth and QoS requirements to maintain a high quality by paying more. Existing research work in this area is discussed briefly in Section II.

In this paper, we present work in two areas. In the first part of this paper (Sections III and IV), we present a Resource Negotiation and Pricing (RNAP) protocol and architecture, as a framework to enable a user to select from a set of available network services with different QoS characteristics, and enable the user and network to dynamically re-negotiate the contracted service parameters and price. RNAP has some features and goals in common with recent work on differentiated services [13] and RSVP [2]. However, our main goal is to study in detail how pricing mechanisms can be integrated with resource negotiation and reservation.

In the second part of the paper, we study a distributed, congestion sensitive price adjustment process, user adaptation in response to pricing, and the interaction between the two processes, within the RNAP framework. In Section V, we outline a pricing strategy which is volume- and congestion-sensitive, and can provide users the incentive to adapt. We also propose mechanisms to enable RNAP to formulate prices in a distributed manner, and communicate pricing information to the users. In Section VI, we discuss candidate algorithms by which applications can adapt their service requests so as to optimize user satisfaction under the constraint of a fixed budget. Finally, in Section VII, we present experimental results demonstrating important features of the price adjustment and user adaptation processes, using a simplified implementation of RNAP.

## II. RELATED WORK

In this section we briefly discuss related research work in three main areas: resource reservation and allocation mechanisms; adaptive applications; billing and pricing in the network.

### A. Resource Reservation and Allocation

Current research in providing QoS support in the Internet is mainly based on two architectures defined by the IETF: Per-flow

based *integrated services* (int-serv) [4], and class-based *differentiated service* (diff-serv) [13]. In both architectures, implementations should include a mechanism by which the user can request specific network services, and thus acquire network resources. Per-flow resource reservation in int-serv is generally implemented through the RSVP reservation protocol [2]. Implementation of resource reservation for diff-serv is a subject of ongoing research, and various approaches have been proposed [14]. In general, RSVP and the implementations of diff-serv lack integrated mechanisms by which the user can select one out of a spectrum of services, and re-negotiate resource reservations dynamically. They also do not integrate the pricing and billing mechanisms which must accompany such services.

Resource allocation schemes based on perceived-quality have been studied in [18][19][48]. These studies were limited to a local system, and did not address the interaction of the local system with a large network. Liao [42] allocates resources to achieve equal perceived quality. We argue that perceived quality does not directly represent the economic value of communications, as discussed below.

### B. Adaptive Applications

There has been a lot of recent research on adaptation of the sending rates of multimedia applications in response to available network resources [1], which relies on signaling mechanisms such as packet loss rates for feedback. The orientation of these methods is different from ours, since they assume no QoS support and no usage-sensitive pricing of network services. The frequent and passive rate adjustment can severely degrade multimedia quality, and sometimes can not guarantee that an application is able to maintain its minimum QoS requirement.

### C. Pricing and Billing in the Network

Microeconomic principles has been applied to various network traffic management problems. The studies in [16][18][20][21][24] are based on a maximization process to determine the optimal resource allocation such that the utility (a function that maps a resource amount to a satisfaction level) of a group of users is maximized. These approaches normally rely on a centralized optimization process, which does not scale. Also, some of the algorithms assume some knowledge of the user's utility curves and truthful revelation by users of their utility curves, which may not be practical.

In [15][17][22][23][?], the resources are priced to reflect demand and supply. The pricing model in these approaches is usage-sensitive. Some of these methods are limited by their reliance on a well-defined statistical model of source traffic, and are generally not intended to adapt to changing traffic demands.

The scheme presented in [23] is more similar to our work in that it takes into account the network dynamics (session join or leave) and source traffic characteristics (VBR). It also allows different equilibrium price over a different time period, depending on the different user resource demand. However, congestion is only considered during admission control. Our pricing algorithm has two congestion-dependent components - congestion due to excessive resource reservation (holding cost) and congestion due to network usage (usage cost).

In general, the work cited above differs from ours in that it does not enter into detail about the negotiation process and the network architecture, and mechanisms for collecting and com-

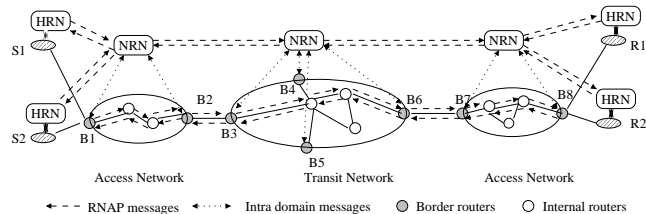


Fig. 1. RNAP-C architecture

municating locally computed prices. Our work is more concerned with developing a flexible and general framework for resource negotiation and pricing and billing, decoupled from specific network service protocols and pricing and resource allocation algorithms. Our work can therefore be regarded as complementary with some of the cited work.

In [27], a charging and payment scheme for RSVP-based QoS reservations is described. A significant difference from our work is the absence of an explicit price quotation mechanism - instead, the user accepts or rejects the estimated charge for a reservation request. Also, the scheme is coupled to a particular service environment (int-serv), whereas our goal is to develop a more flexible negotiation protocol usable with different service models.

## III. THE RNAP ARCHITECTURE

In this section, we define an architecture in which the customer and network service provider negotiate network services. A customer (sender or receiver) wishes to reserve network resources for multiple flows, for example, flows corresponding to audio, video and white-board applications in a video-conference. The customer negotiates with the network through a Host Resource Negotiator (HRN). The HRN negotiates only with its access network to reserve resources, even if its flows traverse multiple domains. It obtains information and price quotations for available services from the network. It requests particular services, specifying the type of service (guaranteed [5], controlled load [6] (CL), expedited forwarding [7], assured forwarding [8], best effort, etc.), parameters to characterize the user traffic (e.g., peak rate, average rate and burst size) and QoS requirements (e.g., loss rate and delay). The HRN can request a different service for each flow from network through RNAP. In addition to resource negotiation between the HRN and the network, the RNAP protocol is also intended for resource negotiation between two network domains.

For negotiations by the network service provider, we consider two alternative architectures, a centralized architecture, and a distributed architecture, described below.

### A. Centralized Architecture (RNAP-C)

The RNAP-C architecture is based on an underlying network divided into Autonomous Systems (AS). Each administrative domain negotiates through a Network Resource Negotiator (NRN) (Fig. 1). Protocol messages are sent between NRNs, or between HRNs and NRNs, and touch each AS once.

The NRN delivers price quotations for the different available service levels to customers, answers service requests from customers, and is also responsible for maintaining and communicating charges for a customer session.

The NRN may be an individual entity, or may be a complementary functional unit that works with other administrative en-

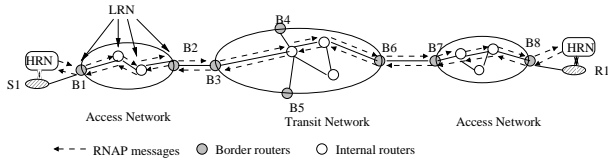


Fig. 2. RNAP-D architecture

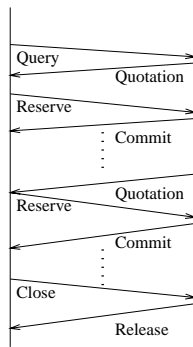


Fig. 3. RNAP messaging sequence between HRN and NRN.

titles. For example, the NRN can be part of (or function as) the Bandwidth Broker (BB) in the diff-serv model [12] and the PDP in the COPS architecture [28]. The NRN either has a well-known address, or is located via the service location protocol [34]. The NRN address of a neighboring domain can be pre-configured or obtained through DNS SRV.

Resource reservation and admission decisions may be performed by the NRN or by other entities, such as the BB of the diff-serv model. If they are performed by other entities, the NRN communicates requests for services to them individually or in aggregate, and receives admission and pricing decisions from them. The implementation of resource reservation and admission control, and the associated communication with administrative entities, is closely related to specific Better than Best Effort (BBE) services, and is outside the scope of the RNAP protocol.

### B. Distributed Architecture (RNAP-D)

In this architecture, the RNAP protocol is implemented at each router, in the form of a Local Resource Negotiator (LRN) (Fig. 2). RNAP messages propagate hop-by-hop along the same path as customer data flows, from the first-hop LRN to the egress LRN, and in the reverse direction. We consider the messaging process in greater detail in Section IV-A.

The RNAP message format is independent of the architecture. Therefore, the two architectures can co-exist; for instance, a domain administered by a NRN can exchange RNAP messages with a neighboring domain which employs the distributed architecture. Also, a HRN does not need to know about the RNAP architecture of its local domain, since it receives and sends the same negotiation messages in either case.

## IV. THE RNAP PROTOCOL

The basic RNAP message sequence is as Fig. 3. Typically, the sequence of Fig. 3 repeats periodically, with a pre-defined *Negotiation Interval*. This allows the protocol to maintain soft-state - state information that expires in the absence of any RNAP *Reserve* message. It also allows the customer and the network to easily re-negotiate services.

The protocol messaging is briefly discussed in Section IV-A. The aggregation and de-aggregation of RNAP messages to make the protocol scalable in core networks is discussed in Section IV-B. A more detailed description of RNAP is given in [39].

### A. Protocol Messaging

We first consider how a customer reserves resources for a flow or group of flows **end-to-end**, to a particular destination address, assuming that the intervening domains implement RNAP-D.

1. The HRN sends a *Query* message to the first hop LRN (FHL), requesting a price quotation from the LRN for one or more services, for a flow or group of flows belonging to the customer. The HRN specifies a set of requirements (such as service time and QoS) with each service. The FHL forwards the *Query* message downstream to the last-hop LRN (LHL).

The LHL determines local service availability and a local price for each service, and initiates a *Quotation* message containing QoS specifications and price for each service. When a *Query* message does not specify any service, the LRN returns quotations for all available services using default values of unspecified service parameters.

The *Quotation* message is sent upstream towards the HRN, and each intermediate LRN verifies local availability of each service, and increments the price by the local price that it computes. The FHL returns the *Quotation* message to HRN.

In addition to asynchronously sending *Quotation* messages, the LHL also sends out *Quotation* messages periodically, containing price quotations for all services requested by the customer.

2. The HRN sends a *Reserve* message to the FHL to apply for services with specified service parameters for a flow or group of flows. This message is sent downstream similar to the *Query* message. A *Reserve* message is sent at the beginning of a session to request services for the first time, and thereafter, periodically or asynchronously to renew or change existing reservations.

In response to a *Reserve* message, the LHL initiates a *Commit* message stating the admissibility of the flow. The *Reserve* request may be admitted or denied, or admitted partially if network resources are scarce and the provider admits the service request with a lower QoS or sending rate than requested. If the flows are admitted, the *Commit* message also contains a local price for the contracted service, and for an on-going session, it also contains the accumulated local charge for a service. As the *Commit* message is forwarded upstream, the committed price and accumulated charge are incremented at each router.

When a customer flow traverses a domain implementing RNAP-C, with a controlling NRN, the flow of messages is identical to that considered earlier for RNAP-D, if each domain is considered to be equivalent to a single node, with the NRN corresponding to the LRN for that node. Accordingly, the NRN is responsible for collecting and communicating admission and pricing and charging information for the domain as a whole instead of for a single node. It is also possible that the flow traverses multiple domains some of which implement RNAP-C and others RNAP-D. In this case, the NRN of a RNAP-C domain would talk to the corresponding boundary LRN of an adjoining

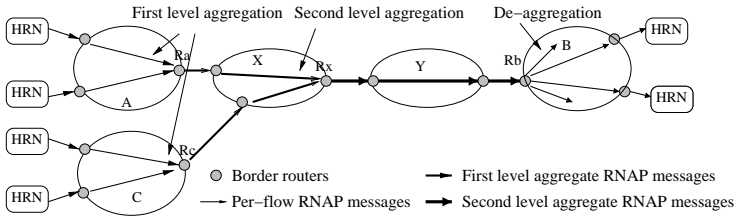


Fig. 4. Example RNAP-D message aggregation.

RNAP-D domain, and the messaging flow would be as before.

### B. State Aggregation

If end-to-end RNAP reservation is carried out for each customer flow, RNAP agents in the core network may potentially need to process RNAP messages for hundreds of thousands of flows, and maintain state information for each of them. In this section, we discuss how RNAP messages can be aggregated in the core of the network by allowing RNAP agents to handle reservations for flow-aggregates instead of individual flows.

#### B.1 Overview of the Aggregation Scheme of RNAP

All RNAP messages have an *Id* field identifying the corresponding data flow; it contains three sub-fields: *Flow Id*, *Aggregation Flag*, and *Aggregate Flow Id*. The merging point aggregates RNAP messages for user flows which request the same or similar services and have similar negotiation intervals.

We consider the aggregation of RNAP messages belonging to senders sharing the same destination network address, forming a “sink tree”. Sink tree based aggregation has also been discussed in [36], [37].

RNAP messages will be merged by the source domain and split again for each individual HRN at the border router (for RNAP-D) or NRN (for RNAP-C) of the destination domain. The merging point in the HRNs home network forwards two messages: one that travels directly to the destination network, without visiting any of the RNAP agents in between, and an aggregated-resource message that reserves resources and collects prices in the “middle” of the network.

The merged resource message have a resource request which is equal to the sum of all the branch resource requests further up in the sink tree. At each merging point, upstream flow arrivals, departures and reservation changes will trigger the update of the downstream merged request. To avoid frequent re-negotiation, the merging point may decide to reserve more resources than the sum of the upstream requests and add resources in larger increments if the current downstream allocation has been reached or is about to be reached. (BGRP [37] analyzes the trade-off in some detail.) We consider aggregation first for RNAP-D, and then for RNAP-C.

#### B.2 Aggregation and De-aggregation in RNAP-D

Fig. 4 illustrates how RNAP message aggregation works in a RNAP-D architecture. Consider the aggregation of *Reserve* messages (this also applies to *Query* messages). At access network A, the border router  $R_a$  creates an aggregate *Reserve* message, with the source address set to ‘a’, the interface address the aggregator  $R_a$ , and the destination network address set to the network address B. It also sets the *Aggregation Flag* to one in the *Id* structure, which marks the message as aggregate.  $R_a$  then forwards the aggregate *Reserve* message hop by hop as in

Section IV-A.  $R_a$  also turns off the router alert option of the incoming per flow messages and tunnels the per-flow *Reserve* messages down to the de-aggregation point ( $R_b$  in Fig. 4), so that per-flow reservation can be resumed in the destination network. In each per-flow *Reserve* message, the address of the aggregator will be included in the *Aggregate Flow Id* field, to enable proper mapping at the de-aggregation point. A per-flow *Reserve* message is encapsulated in an UDP packet with the destination network address set as B, and the port number set to a port reserved for RNAP, and forwarded.

A border router of a domain is a potential de-aggregation point for RNAP messages to that domain. Therefore filters are set up at border routers of a domain so as to intercept aggregate RNAP messages as well as tunneled per-flow RNAP messages. For instance, the border router  $R_b$  (Fig. 4) of domain B is set up to intercept UDP packets with destination address set to the network address B and port number set to the RNAP port. Once intercepted, aggregate *Reserve* messages and tunneled per-flow messages are sent up to the transport layer. The de-aggregation point will record the mapping between an aggregation flow and per flow messages, by checking the aggregation *Flow Id* field. The router alert option will be turned on for per-flow *Reserve* messages arriving at  $R_b$ , and the messages will be forwarded, allowing per-flow resource reservation within domain B. The aggregate *Reserve* message (identified as such by its *Aggregation Flag*) terminates at the de-aggregation router.

In response, a *Commit* message will be sent upstream for the aggregate *Reserve* message as well as each per-flow *Reserve* message. The de-aggregation point  $R_b$  will decide that the destination address for the per flow *Commit* message is ‘a’, by checking the mapping between the aggregate message and the per flow messages. Each per flow *Commit* message is then encapsulated in a UDP message with destination address ‘a’ and tunneled back to its aggregation point  $R_a$ . The aggregate *Commit* message will be forwarded hop by hop upstream until it reaches the aggregation point, and confirms the aggregate *Reserve* request sent by the aggregation agent. There is a similar message flow for RNAP *Quotation* messages in the upstream direction.

The aggregation entity on the source network side is also responsible for de-aggregation of RNAP response messages. It checks the mapping between an aggregate session and per-flow RNAP response messages. If it is the origination point for the corresponding aggregate session, it will map the aggregate-level pricing and charging (returned by the aggregate session *Quotation* and *Commit* messages) to the corresponding per-flow prices and charges for individual flows based on the local policy.

Multiple levels of aggregation can occur, so that aggregate messages are aggregated in turn, resulting in a progressively thicker aggregate “pipe” towards the root of the sink-tree. For a level two aggregation of several level one RNAP aggregate requests as shown in Fig. 4, node  $R_x$  in domain X forms a level two aggregate message with the source address in the *Flow Id* set to ‘x’. Node ‘x’ also records the level one requests, and terminates these messages instead of forwarding them. In response, the RNAP agent at the de-aggregation node  $R_b$  sends response messages for the level two aggregate towards point ‘x’. At point  $R_x$ , the level one response messages are formed by mapping the pricing and charge data from level two aggregate message to individual level one aggregate response messages to send towards  $R_a$  and  $R_c$ . All the per flow request messages are tunneled downstream to node  $R_b$ , and per-flow response mes-

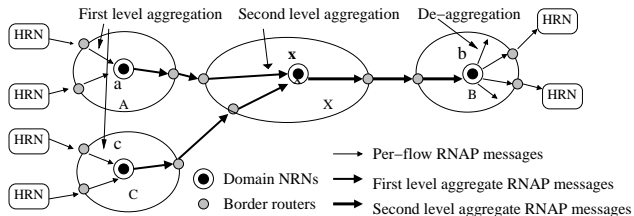


Fig. 5. Example RNAP-C message aggregation.

sages are tunneled from  $R_b$  directly either to  $R_a$  or  $R_c$ .

### B.3 Aggregation and De-aggregation in RNAP-C

In the RNAP-C architecture of Fig. 5, the aggregation and de-aggregation entity are NRNs. Once again, we consider the aggregation of *Reserve* messages. At an aggregating NRN ‘a’, the aggregate *Reserve* message will be formed and sent domain by domain towards the destination domain NRN ‘b’, as in Section IV-A. In addition, the destination domain NRN is located through DNS SRV [35], and the aggregating NRN encapsulates the per flow *Reserve* messages in UDP packet headers and tunnels them directly to the destination domain NRN ‘b’.

The destination domain NRN sends a *Commit* messages “hop by hop” (each hop is one domain) upstream towards ‘a’ in response to an aggregate *Reserve* message. It will also receive the encapsulated per flow *Reserve* messages from ‘a’, process them to perform per-flow reservation in the destination domain, and determine from the *Aggregate Flow Id* field that per-flow response messages are to be encapsulated and tunneled back to ‘a’. There is a similar message flow for RNAP *Quotation* messages in the upstream direction. The mapping of pricing and charging information from aggregate session to per flow message is similar to that in RNAP-D.

### B.4 Overhead Reduction due to Aggregation

As a result of the aggregation of RNAP messages, the message processing overhead and the storage of the RNAP state information are greatly reduced in the core network. Since per flow messages need to be tunneled to the destination network, so the RNAP message transmission bandwidth is not reduced, and actually slightly increased because of the extra aggregation messages. But since RNAP messages are updated with a relatively long interval, this is not a major concern compared with the bandwidth that will be consumed by the data flows.

## V. PRICING AND CHARGING

The main RNAP messages, *Query*, *Reserve*, *Quotation* and *Commit*, all contain a common *Price* structure, used to convey pricing and charging information. In the first part of this section, we discuss how the service provider formulates prices and charges for this purpose. We first briefly describe the *Price* field used in RNAP messages. We then describe end-to-end price and charge formulation in RNAP-D, and in RNAP-C architectures. Finally, in section V-C, we propose a specific strategy for pricing a BBE service at a single network point. This lies outside the scope of the RNAP protocol and architecture, but taken together with the global pricing and charging mechanisms, it constitutes a complete and viable pricing system.

### A. Price Structure in RNAP Messages

The *Price* structure carried by RNAP messages consists of the following fields: *New Price*, *Current Charge*, *Accumulated Charge*, and *HRN Data*. There is a *Price* structure corresponding to each service being negotiated by a RNAP message. The *New Price* field contains the price quoted by the network provider to the negotiating HRN for the next negotiation period. The *Current Charge* field contains the amount charged by the network provider for the preceding negotiation period. The *Accumulated Charge* field contains the total amount charged by the network provider since the beginning of the negotiation session and is carried to protect against the loss of *Commit* messages.

### B. Arriving at Price and Charge

In the previous section, we discussed how price and charge information are communicated to the HRN through RNAP messages. We now consider the issue of arriving at the contracted price to be quoted for a flow receiving a particular service in a given negotiation period, and computing the charge for the service at the end of the period.

#### B.1 Price and Charge Formulation in RNAP-D

In the RNAP-D (distributed) architecture, each router-LRN maintains charging state information for the flows passing through it, based on prices computed at the router. At the beginning of a negotiation period (and also in response to a *Query* message), the last hop LRN originates a *Quotation message*. The *Quotation* message is sent hop-by-hop back towards the first-hop LRN. At each LRN, the *Price:New Price* fields in the message are incremented according to the current *New Price* computed for the corresponding service at the LRN. In Section V-C, we discuss a specific local pricing strategy in which a set of prices is computed for each service. In this case, some mapping behavior may have to be defined to obtain a single increment for the quoted *New Price*. When the *Quotation* message arrives at the negotiating HRN, it carries the total quoted price for each service.

Similarly, *Commit* messages originate at the last-hop LRN, and are sent hop-by-hop back to the first-hop LRN. In this case, the *New Price*, *Current Charge*, and *Accumulated Charge* fields are all incremented at each router-LRN on the way.

#### B.2 Price Formulation in RNAP-C

When the centralized negotiation architecture is used, the local charging state information for a domain is maintained by the NRN. The price formulation strategy is a much more open-ended problem. Various alternatives may be considered, and different domains may apply different local policies. The NRN may compute a price based on the service specifications alone. The price could be fixed, or modified based on the time of day, etc. In general, if the price charged to a flow needs to depend on the network state and the flow path, we consider the following three approaches:

1. The NRN makes the admission decision and decides the price for a service, based on the network topology, routing and configuration policies, and network load. In this case, the NRN sits at a router that belongs to a link-state routing domain (for example an OSPF area) and has an identical link state database as other routers in the domain. This allows it to calculate all the routing tables of all other routers

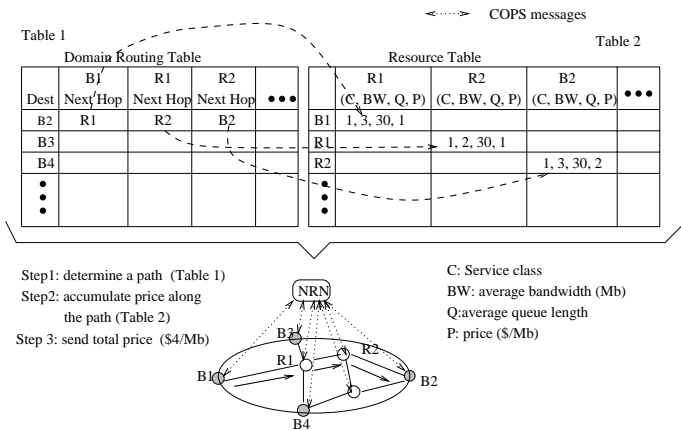


Fig. 6. Price formulation in RNAP-C

in the domain using Dijkstra’s algorithm. A similar idea has been explored in [36] in a different context.

The NRN maintains a domain routing table which finds any flow route that either ends in its own domain, or uses its domain as a transit domain (Fig. 6). The domain routing table will be updated whenever the link state database is changed. A NRN also maintains a resource table, which allows it to keep track of the availability and dynamic usage of the resources (bandwidth, buffer space). In general, the resource table stores resource information for each service provided at a router. The resource table allows the NRN to compute a local price at each router (for instance, using the usage-based pricing strategy described in Section V-C). For a particular service request, the NRN first looks up the path on which resources are requested using the domain routing table, and then uses the per-router prices to compute the accumulated price along this path. The resource table also facilitates monitoring and provisioning of resources at the routers. To enable the NRN to collect resource information, routers in the domain periodically report local state information (for instance, average buffer occupancy and bandwidth utilization) to the NRN. A protocol such as COPS [28] can be used for this purpose.

To compute the charge for a flow, ingress routers maintain per-flow (or aggregated flow from neighboring domain) state information about the data volume transmitted during a negotiation period. This information is periodically transmitted to the NRN, allowing the NRN to compute the charge for the period. The NRN uses the computed price and charge to maintain charging state information for each RNAP session.

- Prices are computed at the network boundary, and communicated to the NRN. For price calculation, there are two alternatives.

One alternative is that the ingress router periodically computes a price for each service class and ingress-egress pair. The calculation is based on service specifications and local per-service demand at the ingress router; internal router states along the flow path are not taken into account.

The other alternative allows internal router load to be taken into account. Probe messages are sent periodically from an egress router to all ingress routers. A probe message carries per-service *Price* structures which accumulate prices hop-by-hop at each router in a similar manner to Section V-B.1.

In both of the above cases, the ingress router maintains per-flow state information that includes the per-flow price (the price charged to the service class the flow belongs to), as well as the per-flow data volume entering the domain. This information is transmitted every negotiation period to the NRN, which computes the charge and is responsible for the messaging.

- Price formulation takes place through a intra-domain signaling protocol. If resource reservation for a particular service in a domain is performed through a dynamic resource reservation protocol, such as RSVP or YESSIR[3], the price information is collected through the periodic messages of the reservation protocol, and stored at the ingress router. For example, the RSVP PATH message and RTCP [38] messages in YESSIR can collect pricing information. If the ingress router is responsible for sending the price information to the NRN, the price accumulated from a domain will be send back to ingress router along with the RSVP RESV message. Such an implementation, utilizing RSVP, is described in VII. Communication between the ingress router and NRN occurs as discussed in the first scenario.

### C. Pricing Strategy

In the previous sections, we assumed the existence of specific pricing strategies or rules for the negotiated service. As discussed earlier, specific pricing strategies are outside the scope of the RNAP protocol itself. However, for completeness, we consider a pricing strategy that could work with the RNAP protocol.

We propose a simple pricing algorithm to determine a price for a particular kind of forwarding service from the router based on the competitive market model [25]. The price computation is performed periodically, with a price update interval  $\tau$ , which is generally independent of the negotiation interval of the services supported by the router. The price within each negotiation interval is kept constant, to provide predictability to users.

We assume that routers support multiple services and that each router is partitioned to provide a separate link bandwidth and buffer space for each service, at each port. We consider one such logical partition. The competitive market model defines two kinds of agents: consumers and producers. The routers are considered as the producers and own the link bandwidth and buffer space for each output port. The flows (individual flows or aggregate of flows) are considered as consumers who consume resources. The total demand for link bandwidth is based on the aggregate bandwidth reserved on the link for a price computation interval, and the total demand for the buffer space at an output port is the average buffer occupancy during the interval. The supply bandwidth and buffer space need not be equal to the installed capacity; instead, they are the targeted bandwidth and buffer space utilization. We decompose the total charge computed at a router into three components: *holding charge*, *usage charge*, and *congestion charge*.

#### Usage Charge:

The usage charge is determined by the actual resources consumed, the level of service guaranteed to the user, and the elasticity of the traffic. For example, on a per-byte basis, best-effort traffic will cost less than reserved, non-preemptable CBR traffic. The usage price ( $p_u$ ) will be set such that it allows a retail

network to recover the cost of the purchase from the wholesale market, and various static costs associated with the service. The usage\_charge  $c_u(n)$  for a period  $n$  in which  $V(n)$  bytes were transmitted is given by:

$$c_u(n) = p_u * V(n) \quad (1)$$

### Holding Charge:

The holding charge can be justified as follows. If a particular flow or flow-aggregate does not utilize the resources (buffer space or bandwidth) set aside for it, we assume that the scheduler allows the resources to be used by excess traffic from a lower level of service. The holding charge reflects revenue lost by the provider because instead of selling the allotted resources at the usage charge of the given service level (if all of the reserved resources were consumed) it sells the reserved resources at the usage charge of a lower service level. The holding price ( $p_h$ ) of a service class is therefore set to be proportional to the difference between the usage price for that class and the usage price for the next lower service class. The holding price can be represented as:

$$p_h^i = \alpha^i * (p_u^i - p_u^{i-1}), \quad (2)$$

where  $\alpha^i$  is a scaling factor related to service class  $i$ . The holding\_charge  $c_h(n)$  when the customer reserves a bandwidth  $R(n)$  is given by:

$$c_h(n) = p_h * R(n) * \tau \quad (3)$$

where  $\tau$  is the duration of the period. The  $R(n)$  can be estimated from the traffic specification and QoS request of the customer, for example, an effective bandwidth [11].

Defining a usage charge and a holding charge separately allows the customer to reserve resources conservatively, without penalizing him excessively for unused resources.

### Congestion Charge:

The congestion charge is imposed when congestion is deduced, that is, the resource request or average usage for a partition (in terms of buffer space or bandwidth) exceeds supply (the targeted buffer space or bandwidth). The congestion price for a service class is calculated as an iterative tâtonnement process [25]:

$$p_c(n) = \min\{p_c(n-1) + \sigma(D, S) * (D - S)/S, 0\}^+, p_{max}\}, \quad (4)$$

where  $D$  and  $S$  represent the current total demand and supply respectively, and  $\sigma$  is a factor used to adjust the convergence rate.  $\sigma$  may be a function of  $D$  and  $S$ ; for example, it is higher when congestion is severe. The router begins to apply the congestion charge only when the total demand exceeds the supply. Even after the congestion is removed, a non-zero, but gradually decreasing congestion charge is applied until it falls to 0, to protect against further congestion. The maximum congestion price is bounded by the  $p_{max}$  parameter so that the total price for a service class does not exceed that for a higher level of service. When a service class needs admission control, all new arrivals are rejected when the price reaches  $p_{max}$ . If  $p_c$  reaches  $p_{max}$  frequently, it indicates that more resources are needed for the corresponding service and new configuration for local resources

may be needed. For a period  $n$ , the total congestion charge is given by

$$c_c(n) = p_c(n) * V(n). \quad (5)$$

Based on a price formulation strategy such as the one we have discussed, a router arrives at a price structure for a particular RNAP flow or flow-aggregate at the end of each price update interval. The total charge for a session is given by

$$session\_charge = \sum_{n=1}^N (c_h(n) + c_u(n) + c_c(n)) \quad (6)$$

where  $N$  is total number of intervals spanned by a session.

## VI. USER ADAPTATION

Although dynamic re-negotiation and pricing are integral features of RNAP, it is compatible with applications with different capabilities and requirements. Applications may choose services that provide a fixed price, and fixed service parameters during the duration of service. Alternatively, if they are not constrained by a fixed user budget, they may use a service with usage-sensitive pricing, and maintain a constant QoS level, paying a higher charge during congestion. Generally, the long-term average cost for fixed-price service is higher since the network provider will add a risk premium. Applications may also be *adaptive*, that is, operate with a budget constraint, and adjust their service requests in response to price increases during congestion.

In this section, we discuss how a set of user applications performing a given task (for example, a video conference) adapt their sending rate and quality of service requests to the network in response to changes in service prices, so as to maximize the benefit to the user.

### A. The Utility Function

We consider a set of user applications, required to perform a task or *mission*, for example, audio, video, and white-board applications for a video-conference. The *Reserve* request from the user specifies certain transmission parameters for each application. In general, the transmission parameters are the sending rate, as well as QoS parameters, usually loss and delay. The user must define quantitatively, through a *utility function*, the value provided by the corresponding network resource allocation towards completing the mission. The utility function is therefore a function in a multi-dimensional space, with each dimension representing a single transmission parameter allocation for a particular application.

#### A.1 Utility as Perceived Value

Clearly, the utility of a transmission depends on its quality as perceived by the user. However, since the user is paying for the transmission, it appears reasonable to define the utility as the *perceived value* of that quality to the user. An audio transmission requiring a certain sending rate and certain bounds on the end-to-end delay and loss rate may be worth 10 cents/minute to the user. The perceptual value is strongly correlated to the perceptual quality, but is not exactly the same. A pair of audio transmissions encoded identically and with the same transmission QoS parameters also have the same perceived quality, but their perceived values may differ according to the application requirements.

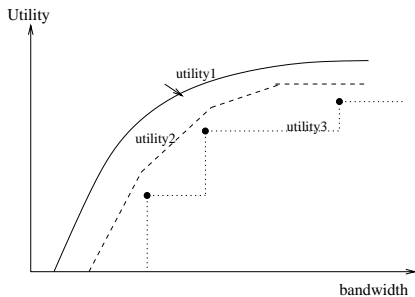


Fig. 7. Some example utility functions

The measurement of subjective quality of multimedia transmissions has been reported by a number of researchers. Generally, these experiments were intended to derive the Mean Opinion Score (MOS), which is measured as an average perceptive quality across a number of test subjects. However, in our framework, perceived value very strongly reflects individual user preferences, and the application task being performed. We therefore consider it likely that an user application will have one or more of the following features:

- allow user to customize utility function(s);
- allow user to define “scenario”-specific utility functions; a particular scenario may be selected by the user during a session, or may be deduced by the application based on user actions;
- allow user to specify a certain time-dependence of utility function.

#### A.2 Utility as a Function of Bandwidth

It is likely that only a few alternative services will be available to a multimedia application on the Internet - at the current stage of research, some possible services are guaranteed [5] and controlled-load service [6] under the int-serv model, Expedited Forwarding (EF) [7] and Assured Forwarding (AF) [8] under diff-serv. A particular user application would be able to choose from a small subset of the available services. Each such service would probably provide some qualitative or quantitative guarantee for loss and delay. It seems likely, therefore, that the user would develop an utility function as a function of the transmission bandwidth (which in turn would depend on specific encoding parameters such as frame rate, quantization, etc.), at different discrete levels of loss rate and delay.

We can make some general assumptions about the utility function as a function of the bandwidth, at a fixed value of loss and delay. The application has a minimum transmission bandwidth, and the utility is zero for bandwidth below this threshold. Also, user experiments reported in the literature suggest that utility functions typically follow a model of diminishing returns to scale, that is, the marginal utility as a function of bandwidth diminishes with increasing bandwidth and eventually goes to zero, defining a maximum QoS requirement.

Fig. 7 shows some possible utility vs. bandwidth curves. Utility1 is a smooth function. User experiments for deducing the utility function would be performed at discrete bandwidths, and some form of interpolation, such as a piecewise linear function (utility2), can be used to approximate the utility function. In addition, in some multimedia applications, only discrete bandwidths are feasible. For example, audio codecs can only operate at certain bandwidth points (Utility3).

#### A.3 Effect of Scaling and Shifting Utility Function

We also studied how the bandwidth adaptation is influenced by linear operations on the utility function - an offset applied uniformly to the utility over all bandwidths, and a multiplicative scaling of the function. Such linear operations could be used, for example, to reflect an evolution with time of the value of a particular information stream (which will be presented in more detail in Section VI-A.4) or the evolution of relative importance of individual applications in a system. We discuss the operation qualitatively here, and present some experimental results in section VII-B.4.

A multiplicative scaling of the utility function by a factor greater than one tends to increase its bandwidth share since it reduces the demand elasticity of the application. The opposite effect is observed when the scaling factor is less than one.

Alternatively, a constant offset to the utility function will not influence the resource distribution as long as the valuation of a bandwidth is higher than its cost. This is because the utility function represents the relative preference of the user for different bandwidths. But it changes the minimum perceived value, which represents the user’s willingness to pay to just keep the application alive.

#### A.4 Time Dependence of Utility

For a particular application, the value of the information may vary with time. An user may perceive a higher value initially after the connection is established, and a lower value after a certain duration (typically, a phone call is very important to the user in the first one minute, compared to one that has lasted 30 minutes), or the reverse (for a movie, the ending is usually more interesting than the introduction). The relative importance of individual applications in a system may also evolve with time.

The evolution with time of the application utilities may be defined based on various user-defined scenarios. A simple way of representing the time evolution is to represent the multiplicative scaling and additive offset in Section VI-A.3, with a pair of time dependent parameters,  $\alpha$  and  $\beta$ , so that the time-dependent utility can be represented as  $\alpha_j(t) * U_j(\cdot) + \beta_j(t)$ , where  $j$  represents a task performed at a time  $t$ .

#### B. Application Adaptation

Consumers in the real-world generally try to obtain the best possible “value” for the money they pay, subject to their budget and minimum quality requirements; in other words, consumers may prefer lower quality at a lower price if they perceive this as meeting their requirements and offering better value. Intuitively, this seems to be a reasonable model in a network with QoS support, where the user pays for the level of QoS he receives. In our case, the “value for money” obtained by the user corresponds to the surplus between the utility  $U(\cdot)$  with a particular set of transmission parameters (since this is the perceived value), and the cost of obtaining that service. The goal of the adaptation is to maximize this surplus, subject to the budget and the minimum and maximum QoS requirements.

We first consider the adaptation strategy of a single application when its utility is a function only of bandwidth (at a fixed loss and delay). We then discuss the adaptation strategy when the utility is function of multiple transmission parameters (bandwidth, loss and delay). Finally, we consider the problem of maximizing the *mission-wide* utility of a system comprising multiple

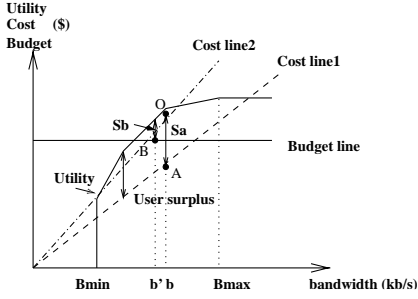


Fig. 8. A perceived value based rate adaptation model

applications performing a certain task. We assume the applications belong to a single user.

### B.1 Adaptation of Single Application over Fixed Transmission Quality

If the quality of transmission is fixed (a particular delay and loss), the application utility (that is, the user-perceived value) increases monotonically with the bandwidth. Hence the maximization problem for the user can be written as:

$$\begin{aligned} \max & [U(x) - C(x)] \\ \text{s. t.} & C(x) \leq b \\ & x_{min} \leq x \leq x_{max}, \end{aligned} \quad (7)$$

where  $x$  is the bandwidth under consideration,  $C(x)$  is the cost for the requested bandwidth,  $b$  is the budget of user,  $x_{min}$  is the minimum bandwidth requirement, and  $x_{max}$  represents the maximum bandwidth requirement. Note that  $U$ ,  $b$  and  $c$  are in units of money/time.

One way of carrying out this optimization is to fit the utility function to a closed form function. The optimal solution is then obtained by using Kuhn-Tucker conditions for a maximum subject to inequality constraints.

As mentioned earlier, the application utility is likely to be measured by user experiments and known at discrete bandwidths. In this case, it is convenient to represent the utility as a piecewise linear function, as shown in Fig. 8. The figure also assumes a constant unit bandwidth cost  $C$ , so that the cost-vs-bandwidth is a straight line with slope equal to  $C$ . The budget is shown as a horizontal line passing intercepting the cost/utility axis. From the figure, it is evident that the optimal bandwidth is **either** the segment end-point with the highest surplus, if this end-point meets the budget constraint ( $b$  in Fig. 8 case A) **or else** the bandwidth corresponding to the intersection point of the cost line with the budget line ( $b'$  in Fig. 8 case B).

### B.2 Adaptation of Single Application over Multiple Transmission Parameters

We now consider the maximization of the application surplus over a set of transmission parameters (usually, the bandwidth, loss rate and delay). The objective function is as shown earlier in equation 7, but  $x$ ,  $x_{min}$  and  $x_{max}$  are now vectors corresponding to the set of transmission parameters. If a complete quality of service parameter space is considered, the searching cost can be prohibitive. As briefly explained however, we believe it is likely that the application utility will take the form of a small set of utility versus transmission bandwidth functions, each at a different level of loss rate and delay, corresponding to

a particular service. In this case, the optimization routine is as follows:

1. For each available service, use the corresponding utility versus bandwidth function to determine the optimal bandwidth, as in Section VI-B.1.
2. Select the service which gives the highest surplus at its optimal bandwidth.

### B.3 Simultaneous Adaptation of Multiple Applications corresponding to Single Task

We now consider the simultaneous adaptation of transmission parameters of a set of  $n$  applications performing a single task. The transmission bandwidth and QoS parameters for each application are selected and adapted so as to maximize the mission-wide “value” perceived by the user, as represented by the surplus of the *Total Utility*,  $\hat{U}$  over the total cost  $C$ . We can think of the adaptation process as the allocation and dynamic re-allocation of a finite amount of resources between the applications.

A number of researchers have noted the interaction between the perception of the different component media in a multimedia system, such as a video conference [44][45][46][47]. For example, an investigation of video phone systems indicated that any increase in visual representation of the speaker increases the viewer’s tolerance to audio noise [44]. To take into account the interdependencies among applications, the utility of the  $i_{th}$  application should, in general, be written as  $U^i(x^1, \dots, x^i, \dots, x^n)$ , where  $x^i$  is the transmission parameter tuple for the  $i_{th}$  application. The total utility function of a system consisting of  $n$  individual application streams can be represented in general as  $\hat{U}(U^1(\cdot), \dots, U^n(\cdot))$ , where  $U^i(\cdot)$  represents the utility of stream  $i$ . Since we consider utility to be equivalent to a certain monetary value, we can write the total utility as the sum of individual application utilities :

$$\hat{U} = \sum_i [U^i(x^1, \dots, x^i, \dots, x^n)] \quad (8)$$

and the optimization of surplus can be written as

$$\begin{aligned} \max & \sum_i [U^i(x^1, \dots, x^i, \dots, x^n) - C^i(x^i)] \\ \text{s. t.} & \sum_i C^i(x^i) \leq b \\ & x_{min}^i \leq x^i \leq x_{max}^i, \end{aligned} \quad (9)$$

where  $x_{min}^i$  and  $x_{max}^i$  represent the minimum and maximum transmission requirements for stream  $i$ , and  $C^i$  is the cost of the type of service selected for stream  $i$  at requested transmission parameter  $x^i$ .

The general approach to solving this problem is to represent each utility  $U^i(\cdot)$  as a continuous function of the entire space of transmission parameters of all  $n$  applications, and solve the Kuhn Tucker equations so as to maximize the total surplus.

We make the simplifying assumption that for each application, a utility function can be defined independently and is a function only of the transmission parameters of that application -  $U^i(\cdot) = U^i(x^i)$ . This is a reasonable assumption since  $U^i(\cdot)$  would normally depend strongly mainly on the vector  $x^i$ .

As earlier, we can decompose a single utility function  $U^i(x^i)$  into a set of service-specific utility functions which are functions only of bandwidth, each corresponding to a particular delay and

loss provided by a particular service. Clearly, several combinations of services (and hence, service-specific utility functions) are possible. We first consider one particular combination of service-specific utility functions. Let the utility of an application  $i$  be defined at  $L^i$  bandwidth levels. The utility at each level is  $u_l^i$  ( $l = 1, 2, \dots, L^i$ ), and the utility function is piece-wise linear. Segment  $l$  (the straight line between levels  $l$  and  $l + 1$ ) has a slope  $k_l^i$ . The optimal transmission parameter set for a particular combination of service-specific utility functions is then determined as follows:

1. From the utility function for each application  $i$ , determine the segment end-point  $l_{opt}(l = 1, 2, \dots, L^i)$ , with bandwidth  $B_{opt}^i$ , at which the surplus (utility minus cost) is maximized for that application. Let the cost of the targeted bandwidth be  $C_{opt}^i(B_{opt}^i)$ .
2. If the total expenditure needed for the system,  $\sum_i C_{opt}^i(B_{opt}^i)$ , exceeds the total system budget, go to step 3, else stop.
3. From all the applications that receive service at level  $l_{opt} > l_{min}$ , find the application  $i_{victim}$  with the smallest slope in the surplus ( $u_l^i - C_l^i$ ) from level  $l_{opt}$  to  $l_{opt} - 1$  (this corresponds to the smallest sensitivity of application surplus to a reduction in bandwidth). Reduce the current bandwidth allocation for this application to the next lower bandwidth level ( $l_{opt} = l_{opt} - 1$ ).
4. If the total system expenditure remains greater than the system budget, go back to step 3. If there is excess budget, allocate the excess budget to the current victim application (from step 3) to acquire as much bandwidth as permitted by the budget.

The above algorithm is repeated for each possible combination of service-specific utility functions; each time, an optimal transmission parameter set is obtained. The transmission parameter set with the highest total surplus is then selected.

#### B.4 Stability of the Pricing and User Adaptation Processes

Applications will re-negotiate network services when a price quoted by the network changes or when the media traffic format changes, resulting in different bandwidth requirements. In addition, a new application entering the network or an existing application leaving the network will also lead to resource re-allocation. We show the stability of the process as applied to our pricing algorithm in [40].

In Section VII-B, we will discuss a situation in which frequent bandwidth adaptation by users sharing the same link bandwidth leads to oscillatory behavior. However, this is seen to be alleviated by introducing a damping factor into the rate adaptation algorithm, so that the bandwidth request is only changed by small increments. In the core network, oscillatory behavior can be minimized by aggregating RNAP requests, reducing the frequency with which the RNAP agent re-allocates resources and adjusts the price.

## VII. EXPERIMENTS

In this section, we describe preliminary experimental results demonstrating some of the important features of our work, using a simplified implementation of RNAP. The implementation was based on an extension of the RSVP signaling protocol, and carried out on a test-bed consisting of two nodes connected by a

single 10 Mb/s link. An RNAP agent (LRN) was implemented at each node. Two types of service were implemented - the traditional best-effort service, and the int-serv Controlled Load service.

Although our implementation was simplified, it allowed us to demonstrate several features: the periodic RNAP negotiation process including resource negotiation and pricing and charging; the stability of the usage-sensitive pricing algorithm and its effectiveness in controlling congestion; the adaptation of user applications in response to changes in network conditions and hence in the service price; and the effect of user utility functions on user adaptation and resource allocation.

The protocol implementation and test-bed setup are discussed in Sections VII-A and VII-A.1, and results are presented and analyzed in Section VII-B.

### A. Protocol Implementation

The RNAP *Quotation*, *Reserve* and *Commit* messages were implemented as embedded messages in the RSVP *Path*, *Resv* and *ResvErr* messages. The RNAP *Query* message was not implemented; this was not critical, since only a single service was available to the user. RNAP *Quotation*, *Reserve* and *Commit* information were embedded in RSVP *Path*, *Resv* and *ResvErr* messages. Since *Commit* messages could not easily be sent periodically in this implementation framework, the *Quotation* message carried periodic charging information (in the *Price* field) instead of the *Commit* message. The RNAP negotiation period was set to be the same as the RSVP refresh period, 30 seconds.

The sequence of messages was as follows:

1. RSVP *Path* messages, with embedded RNAP *Quotation* information are sent periodically from the sender-LRN towards the receiver-LRN. As a *Path* message passes each node, the *Price* field is updated to add the price computed at the local node and the incremental charge for the previous period.
2. The HRN at the receiver receives the *Path* message and sends a RSVP *Resv* request, with embedded RNAP *Reserve* information. The *Price* received from *Path* is copied into the *Price* field of the *Resv* message, with the *Price:HRN Data* field updated to indicate receiver information.
3. When a RSVP *Resv* request is rejected, an RSVP *ResvErr* message is sent to the receiver HRN, with embedded *Commit* information. This information includes “bandwidth available” information in the *Price:HRN Data*  $\rightarrow$  *Maximum Rate* field.

*rsvpd* version 4 from ISI [31] was extended to support RNAP. Resource reservation on a link was performed using Class-Based Queueing (CBQ) [32], as part of the ALTQ package [33].

Pricing was done as follows. A RSVP *Policy Element*, called the *Price Element*, was defined to hold the RNAP *Price* structure. As with other *Policy Elements*, the *Price Element* was opaque to RSVP and only understood by policy peers. The *Price Element* was embedded within the *POLICY\_DATA* objects [30][29] of *Path* messages, *Resv* messages and *ResvErr* messages.

The LRN at a node was implemented as part of the Local Policy Decision Point (LPDP) proposed in the COPS architecture [28][29]. The RNAP agent periodically computed a set of prices (for the CL service) based on traffic through a link, by monitoring the CBQ states. It also maintained state information for each

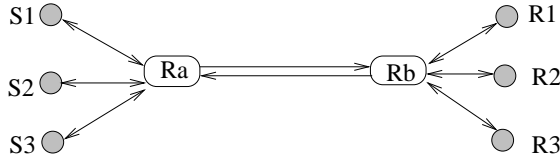


Fig. 9. Testbed setup

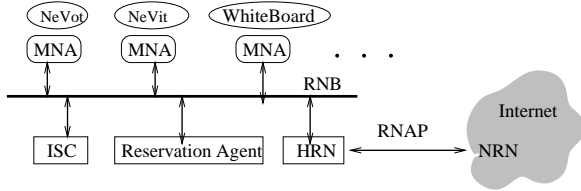


Fig. 10. The architecture of the extended MINT system

RNAP session at the node. Congestion charge was levied based on the total link usage relative to the total link bandwidth.

Since the system offers only a single class of service, namely CBQ, we assume that the utility depends only on the bandwidth. In that class, delay depends on the allocated bandwidth and there is no congestion-induced packet loss.

### A.1 Experimental Setup and Parameters

The test setup consisted of 2 routers (Ra and Rb) connected by a 10 Mb/s link, schematically represented in Fig. 9.

Three RNAP sessions were established end to end, and shared the same output interface of the link. To create different levels of network load, a simple data source model was used in each session to continuously send UDP packets. The packet generation rate was tunable to allow user adaptation.

Out of the total capacity of 10 Mb/s, 4 Mb/s was configured for CL service. The congestion threshold was set to 70% of the CL capacity (2.8 Mb/s). Background traffic was also sent using best effort service.

In addition to experiments using the simple source model to generate traffic, one set of experiments was performed using traffic generated by a multimedia application - the Multimedia Internet Terminal (MINT) [41] system. The audio and video application components of MINT, NeVoT and NeViT, support rate adaptation. We extended the MINT system to couple the rate adaptation process to RNAP negotiation. Each application was represented by a Media Negotiation Agent (MNA). The MNA communicated application requirements and changes in requirements to the HRN over a Resource Negotiation Bus (RNB). The HRN was responsible for RNAP negotiation with the LRN, as well as allocation of resources (sending rates) to the MNAs using the adaptation algorithm discussed in Section VI-B.3.

## B. Experimental Results

We now describe a set of experiments which address the following issues: (i) the sharing of bandwidth between competing adaptive applications with identical utility functions; (ii) the sharing of bandwidth between competing applications with utility functions reflecting different amounts of elasticity in bandwidth requirements; (iii) distribution of bandwidth among applications belonging to a single-user multimedia system so as to maximize mission-wide value; (iv) the influence of specific changes in the utility function on the bandwidth adaptation; (v)

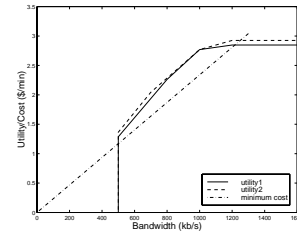


Fig. 11. Utility functions used in the experiments of section VII-B.1 and VII-B.3

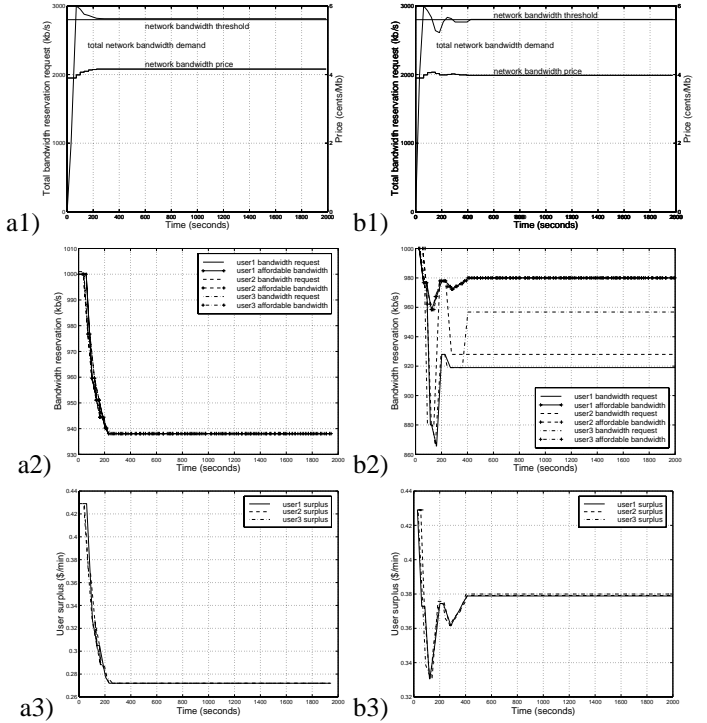


Fig. 12. Allocation of bandwidth and surplus for three competing users sharing a link. a1, a2, and a3 show the results when the users all have the Utility 1 function from Fig. 11, and b1, b2, and b2 show corresponding results when the users have the Utility 2 function from the same figure

adaptive behavior of audio and video applications belonging to the MINT system.

In each experiment, we study the behavior of the price in response to bandwidth demand, the influence of the price in driving adaptation of user bandwidth requirements, and the “benefit” gained by the applications in terms of the surplus (or perceived value of the service relative to its cost). We ascertain that a stable and equitable distribution of bandwidth is reached in each case.

### B.1 Bandwidth Sharing between Users

In the first experiment, we study the adaptive behavior when applications having the same utility function and belonging to different users compete for network resources. The same experiment is performed with two different utility functions, Utility 1 and Utility 2, shown in Fig. 11.

Fig. 12-a1, 12-a2, and 12-a3 show different aspects of adaptive behavior when Utility 1 is used. Initially, in response to the initial price, each user determines that the optimal bandwidth (giving the maximum surplus) is 1000 kb/s. Since the total reservation of 3000 kb/s made by the three users is higher than the

congestion threshold of 2800 kb/s, the network imposes an additional congestion price, resulting in a gradual increase in the price.

Fig. 12-a1 shows the initial increase in price, from 3.9 cents/Mb, until it stabilizes at 4.2 cents/Mb after about 150 seconds (corresponding to 5 negotiation periods). Fig. 12-a1 also shows the variation with time of the total bandwidth reservation, and Fig. 12-a2 shows the variation with time of the individual bandwidth reservations, and the maximum per-user bandwidth that the user budget permits. As the price increases, each user is constrained by its budget to decrease its sending rate in response. As a result, the reserved bandwidth decreases smoothly, until the link becomes un-congested, and the price stabilizes. Fig. 12 a3 shows a gradual decrease in the surplus obtained by each user until the price stabilizes. All users are observed to have nearly identical adaptation traces.

The second experiment uses Utility2 in Fig. 11. Utility2 differs from Utility1 in that the optimal bandwidth (at the initial un-congested link price) of 1000 kb/s differs only slightly from the next sub-optimal bandwidth of 700 kb/s with respect to the perceived surplus.

In Fig. 12b, the adaptation traces are observed to be different from that shown in Fig. 12a. When the price increases, the applications are constrained by their budget to reduce their bandwidths initially. When the price increases to a certain value, the optimal bandwidth requirement for all the users (calculated at slightly different times) shifts to 700 kb/s, since the increase in cost for a larger bandwidth is higher than for a smaller bandwidth. Since the two optimal points in our example are very far apart in bandwidth, and the perceived surplus of the two bandwidths are very close, an oscillation between 2100 kb/s and 3000 kb/s was observed in the total bandwidth when this experiment was performed.

To avoid this problem, a proportional plus derivative (PD) controller [49] was used to reduce the oscillation. During each negotiation period, instead of letting the requirement jump to a new optimal bandwidth, the user shifts to a bandwidth between the current one and the optimal one, resulting in temporarily sub-optimal operation. The PD control law regulates the bandwidth request as follows:

$$r_{i+1} = \begin{cases} r_i - \alpha_0(r_i - r^*) - \alpha_1(r_i - r_{i-1}), & \text{if } \frac{|SP(r^*) - SP(r_i)|}{SP(r_i)} > \theta \\ r_i, & \text{otherwise} \end{cases} \quad (10)$$

where  $r^*$  is the desired optimal rate,  $r_i$  is the rate requested for negotiation period  $i$ , and  $SP(x)$  represents the surplus obtained by obtaining bandwidth  $x$ . Quicker convergence is attained by making  $\alpha_0$  large, while the overshoot is minimized by making  $\alpha_1$  large. In addition to the PD control, the bandwidth was allowed to be adjusted only if the new bandwidth led to an increase in surplus of at least  $\theta$  %. In the experiment,  $\alpha_0$ ,  $\alpha_1$  and  $\theta$  are set separately as 0.4, 0.6, and 2%, which led to the quick convergence without large overshoot.

Fig. 12b-2 shows that the bandwidth requirement of all three users stabilized within seven negotiation periods, with different users having different bandwidth shares. This is partly because of the asynchronous user negotiation behaviors, and partly because of the possible sub-optimal bandwidth (within  $\theta$  % of optimal) choice of some users. All three users end up with final

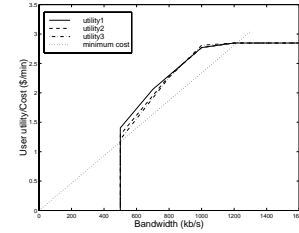


Fig. 13. Utility functions with different bandwidth sensitivity

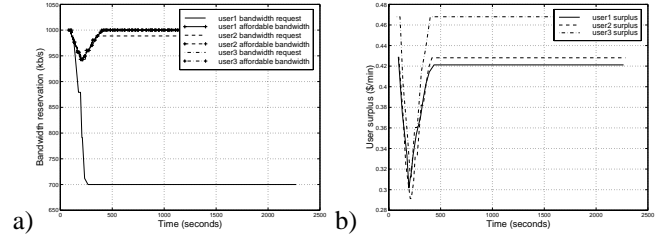


Fig. 14. Bandwidth reservation a) and perceived surplus b) when all users have different demand elasticity

surplus values very close to each other (within 2 %). This is important since we consider the perceived surplus, rather than the bandwidth, as a measure of the user satisfaction.

## B.2 Bandwidth Sensitivity and Demand Elasticity

In this experiment, we study the effect of different elasticities in user demand on user bandwidth sharing and adaptation, using different utility functions (Fig. 13) for different users. An utility function with a smaller slope reflects a higher elasticity in the bandwidth requirement of the user. Fig. 14-a shows that the user with the more elastic requirement is more sensitive to price changes and reduces his resource requirement faster when the network price increases. Correspondingly, Fig. 15-a shows that as a reward for elastic behavior, the average network charge for the more elastic user is lower, while the three users have similar perceived surplus (Fig. 14-b).

Thus, users with less stringent bandwidth requirements express this flexibility through a less bandwidth-sensitive utility function, and bear a greater share of reductions in bandwidth for congestion-control. Users with more bandwidth-sensitive requirements have to pay a higher charge during congestion to maintain their bandwidths at current levels.

## B.3 Adaptation Across Media

In this section, we look at how utility functions guide the distribution of bandwidth across different media which are part of a multimedia system belonging to a single user. The results of

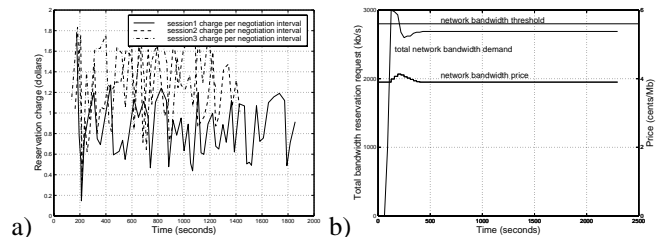


Fig. 15. Network charges for different users a) and the total network bandwidth demand and price b) when the users have different demand elasticity

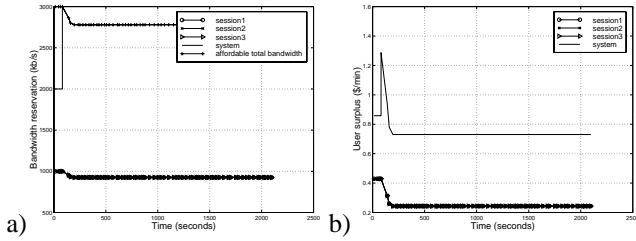


Fig. 16. Bandwidth reservation a) and perceived surplus value b) for adaptation across media sessions in a system, all sessions having the same utility

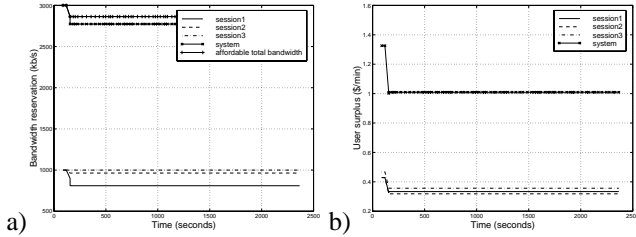


Fig. 17. Resource reservation a) and perceived surplus value b) among sessions of a system with different bandwidth sensitivity

two experiments are presented.

In the first experiment, the system consists of three media sessions, all of which have the same utility function, Utility1 shown in Fig. 11. When the system budget is exceeded due to congestion, the HRN adjusts the application bandwidths downwards according to the adaptation algorithm described in Section VI-B.3. Since all the applications have identical utilities, the total system bandwidth is equally distributed between them at all times, as seen in Fig. 16a.

The second experiment is similar except that the three media sessions have different utility functions shown in Fig. 13. Fig. 17a shows that when the total optimal bandwidth requirement for all the media sessions in the system exceeds the system budget, the media session with the more elastic resource demand will be assigned relatively less bandwidth so as to maximize the overall perceived value. This is a similar result to that obtained in section VII-B.2 for multiple competing user applications. In effect, the system regards a media session with more elastic requirements as being more able to absorb bandwidth reductions, and “borrows” bandwidth from this session to give to other sessions.

#### B.4 Linear Operations on Utility Functions

In section VI-A.3, we qualitatively discussed how the shape of the user utility functions influences bandwidth selection and distribution. We now experimentally study the impact of two linear operations on the utility function, multiplicative scaling

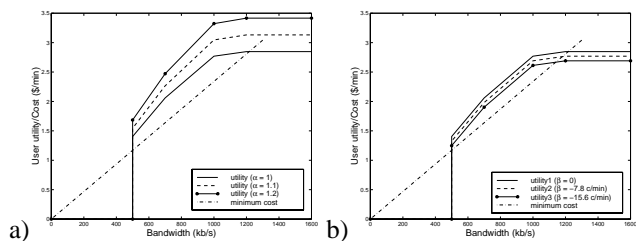


Fig. 18. Equivalent utilities under multiplicative scaling a) and additive shifting b)

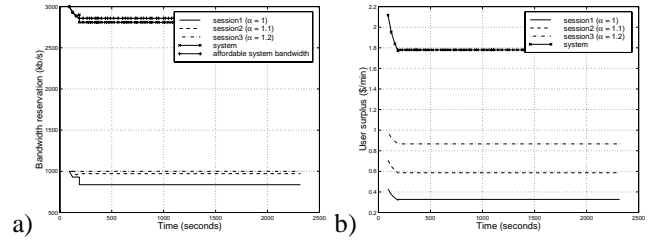


Fig. 19. Bandwidth reservation and perceived surplus for utilities scaled multiplicatively by different amounts

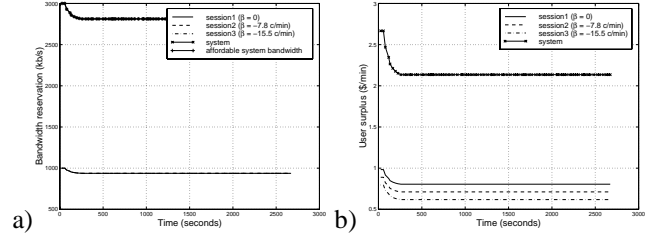


Fig. 20. Bandwidth reservation and perceived surplus for utilities shifted additively by different amounts

by a weight  $\alpha$ , and additive or subtractive shifting by an amount  $\beta$ . The experiment studies bandwidth distribution between multiple sessions in a system belonging to a single user, though similar results have also been observed with applications belonging to different users.

Consider three media sessions belonging to a system, all with the same basic (un-scaled) utility function (we use utility1 of Fig. 13). Sessions 1, 2, and 3 are assigned scaling factors of 1, 1.1, and 1.2 respectively. The resulting scaled utilities are shown in Fig. 18a.

Fig. 19 shows the variation of individual and system bandwidth allocations and perceived surpluses. Expectedly, when the adaptation is constrained by the system budget, an application with a higher  $\alpha$  gets a larger bandwidth share because of its lower elasticity of demand.

We now consider the effect of an offset applied uniformly to the utility over all bandwidths. In Fig. 20b, the utility1 function (which is the same as utility1 in Fig. 13a) is shifted downwards and forms utility2 and utility3. Three different sessions are assigned different utility functions.

The results shown on Fig. 20a show that all three sessions are allocated the same bandwidth though Fig. 20b shows that the allocation results in different values of perceived surplus. This is because utility function represents the relative preference of the user for different bandwidths. The absolute value of the utility is not important - the adaptation algorithm only searches for the bandwidth with the maximum perceived value relative to its cost.

#### B.5 Adaptation in MINT

Finally, we examine the adaptive behavior of the audio (NeVoT) and video (NeViT) applications in the MINT video conference system. The utility functions for the audio and video applications are shown in Fig. 21a.

At the un-congested link bandwidth price, the optimal audio bandwidth for MINT is 64 kb/s, and the optimal video bandwidth is 384 kb/s. The MINT applications compete for bandwidth with three single media applications belonging to different users. The applications use the utility functions of Fig. 13.

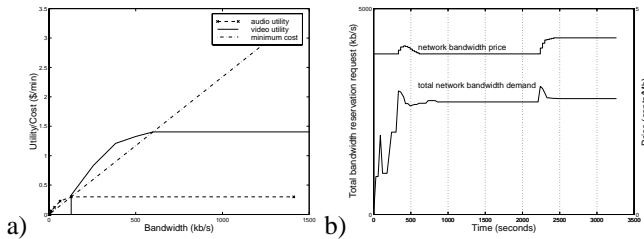


Fig. 21. a) Audio and video utility functions used for adaptation by MINT b) Price and total bandwidth variation in the same experiment

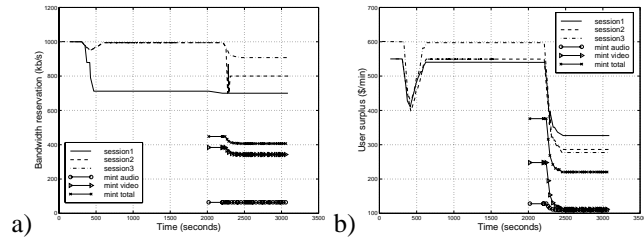


Fig. 22. Individual bandwidth reservations and perceived surplus in the adaptation of Mint applications

The three user applications are started first, and reach stability at time 630 seconds with bandwidth allocations of 712 kb/s, 994 kb/s, and 994 kb/s respectively.

At time 2000 seconds, the MINT video conference system is started, and it first requests optimal bandwidth allocation (64 kb/s + 384 kb/s). The total requested bandwidth exceeds the link congestion threshold, forcing the price up. It is observed that the NeVoT bandwidth remains unchanged, and the NeViT bandwidth is reduced to 342 kb/s. The bandwidth share of the three competing user application drops to 700 kb/s, 800 kb/s and 907 kb/s respectively. User 1 has the most elastic bandwidth requirement between 700 kb/s and 1000 kb/s, and therefore initially gets a smaller share. But it is less elastic above 700 kb/s, and after the MINT applications are started, user 2, which has a relatively greater elasticity near its current allocation, reduces its requirement the most. The above experiment demonstrates the efficacy of the adaptation framework in allowing new sessions to join gracefully even when the network is highly loaded.

## VIII. SUMMARY

The overall objective of this paper has been to study a dynamic, usage and congestion dependent network pricing system in conjunction with price-sensitive user adaptation. In addition to this objective, we have also developed RNAP as a dynamic resource negotiation platform for multiple delivery services and environments. Our main focus in developing RNAP has been to integrate service negotiation with network pricing.

A pair of alternate protocol architectures has been described. The RNAP-D architecture is based on a distributed, per-node model, while the RNAP-C architecture concentrates the negotiation functionality at a centralized entity, the NRN. The architectures provide mechanisms for incremental price computation at a single point in the network, collation of local prices in order to compute end-to-end prices along different routes, and communication of prices and charges to the client. Several price and charge collation mechanisms have been described for the distributed and centralized architectures, and end-to-end pricing and charging across several administrative domains has also been discussed. An algorithm for local pricing at a router has

been discussed in detail, but the pricing and charging mechanisms in the protocol are independent of the specific pricing algorithm used.

We have proposed mechanisms for rate and QoS adaptation by an application or multi-application system, based on the utility (defined as user-perceived value) of a given combination of transmission parameters, relative to the cost of obtaining the corresponding service from the network. The adaptation system takes into account constraints imposed by the minimum application requirements and the budget specified by the user, and responds actively to changes in price signaled by the network by dynamically adjusting network resource usage by the application. In a multi-application system such as a video-conference application, the framework allows the system budget to be distributed among the component media so as to maximize the overall perceived value relative to cost. Some heuristics are discussed to simplify this process. The system budget is dynamically re-distributed among applications in response to changes in price, as well as changes in the relative utilities with time or under different application scenarios.

Experiments based on a prototype implementation of the important RNAP functionalities have been described. It is observed that the usage-sensitive pricing can effectively reduce the blocking rate at call admission time. Experimental results also show that perceived value based adaptation allows bandwidth to be shared among competing users fairly. At the onset of congestion, the bandwidth share of users with more elastic demands (less bandwidth-sensitive utilities) is reduced, and the bandwidth shares of those with inelastic demands (or non-adaptive applications) remains fairly constant. However, users with elastic requirements continue to receive a fair level of perceived surplus (perceived value relative to cost). The distribution of system bandwidth among multiple sessions belonging to a multimedia system is also demonstrated. The effect of a PD control law is shown in minimizing oscillations and abrupt transitions in the bandwidth adaptation. The effect of changes in the utility functions on resource distribution has been examined.

## IX. ACKNOWLEDGMENTS

The authors would like to thank Tony Eyers of University of Wollongong for the constructive comments on the paper.

## REFERENCES

- [1] X. Wang, H. Schulzrinne, "Comparison of adaptive Internet multimedia applications," *IEICE Transactions on Communications*, pp. 806-818, June, 1999.
- [2] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation protocol (RSVP) - version 1 functional specification," RFC 2205, Sept. 1997.
- [3] P. Pan and H. Schulzrinne, "YESSIR: A simple reservation mechanism for the Internet", *International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'98)*, Cambridge, England, July 1998.
- [4] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview," RFC 1633, Internet Engineering Task Force, June 1994.
- [5] S. Shenker, C. Partridge, and R. Guerin, "Specification of guaranteed quality of service," RFC 2212, Internet Engineering Task Force, Sept. 1997.
- [6] J. Wroclawski, "Specification of the controlled load quality of service," RFC 2211, Sept. 1997.
- [7] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding PHB," RFC 2598, Internet Engineering Task Force, June 1999.
- [8] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," RFC 2597, Internet Engineering Task Force, June 1999.

- [9] S. Jamin, S. J. Shenker, and P. B. Danzig, "Comparison of measurement-based admission control algorithms for controlled-Load service," *Proc. IEEE INFOCOM'97*, April 1997.
- [10] H. Zhang and S. Keshav, "Comparison of rate-based service disciplines," *Proc. ACM SIGCOMM'91*, Zurich, Switzerland, Sept. 1991.
- [11] R. Guerin, H. Ahmadi and M. Naghshineh, "Equivalent capacity and its application to bandwidth allocation in high-speed networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 7, pp. 968-981, Sep. 1991.
- [12] K. Nichols, V. Jacobson, and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet," RFC 2638, Internet Draft, Internet Engineering Task Force, July 1999.
- [13] K. Nichols and S. Blake, "Differentiated services operational model and definitions," Internet Draft, Internet Engineering Task Force, Feb. 1998. Work in progress.
- [14] Internet 2 Bandwidth Broker Information, <http://www.merit.edu/working.groups/i2-qbone-bb>.
- [15] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang, "Pricing in computer networks: Motivation, formulation, and example," *IEEE/ACM Transactions on Networking*, vol. 1, pp 614-27, Dec. 1993.
- [16] J. F. MacKie-Mason and H. Varian, "Pricing Congestible Network Resources," *IEEE J. Select. Areas Commun.*, vol. 13, no. 7, pp 1141-9, Sept. 1995.
- [17] N. Anerousis and A. A. Lazar, "A framework for pricing virtual circuit and virtual path services in ATM networks", *ITC-15*, pp. 791 - 802, 1997.
- [18] A. Hafid, G. V. Bochmann and B. Kerherve, "A quality of service negotiation procedure for distributed multimedia presentational applications," *Proceedings of the Fifth IEEE International Symposium On High Performance Distributed Computing (HPDC-5)*, Syracuse, New York, 1996.
- [19] T. F. Abdelzaher, E. M. Atkins, and K. Shin, "QoS negotiation in real-time systems and its application to automated flight control," To appear in *IEEE Transactions on Software Engineering*, 1999.
- [20] H. Jiang and S. Jordan, "A pricing model for high speed networks with guaranteed quality of service," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Fransisco, California), Mar. 1996.
- [21] S. Low and P. Varaiya, "An algorithm for optimal service provisioning using resource pricing," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Toronto, Canada), June 1994.
- [22] D. F. Ferguson, C. Nikolaou, and Y. Yemini, "An economy for flow control in computer networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Ottawa, Canada), pp. 110-118, IEEE, Apr. 1989.
- [23] E. W. Fulp, D. S. Reeves, "Distributed network flow control based on dynamic competitive markets," *Proceedings International Conference on Network Protocol (ICNP'98)*, Austin Texas, Oct. 13-16, 1998.
- [24] F. P. Kelly, A.K. Maulloo and D.K.H. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society* 49 (1998), 237-252.
- [25] H. Varian, "Microeconomic Analysis," Third Edition, 1993. W.W. Norton & company.
- [26] Hahn F (1982). Stability. In: Arrow KJ and Intriligator MD (eds), "handbook of Mathematical Economics", Volumn II. Noth-Holland, Amsterdams, pp 745-793.
- [27] M. Karsten, J. Schmitt, L. Wolf, and R. Steinmetz, "An embedded charging approach for RSVP," *The Sixth International Workshop on Quality of Service (IWQoS'98)*, pp 91-100, Napa, California, USA.
- [28] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, A. Sastry, "The COPS (Common Open Policy Service) Protocol," RFC 2748, Internet Engineering Task Force, Jan., 2000.
- [29] R. Yavatkar, D. Pendarakis, and R. Guerin, "A framework for policy-based admission control," RFC 2753, Internet Engineering Task Force, Jan. 1998.
- [30] S. Herzog, "RSVP extensions for policy control," RFC 2750, Internet Engineering Task Force, Jan. 2000.
- [31] RSVP software release, <ftp://ftp.isi.edu/rsvp/release>.
- [32] Floyd, S., and Jacobson, V., "Link-sharing and Resource Management Models for Packet Networks", *IEEE/ACM Transactions on Networking*, Vol. 3, No. 4, pp. 365-386, August 1995.
- [33] K. Cho, ALTQ: Alternate Queueing for FreeBSD.
- [34] J. Veizades, E. Guttman, C. Perkins, and S. Kaplan, "Service location protocol," RFC 2165, Internet Engineering Task Force, June 1997.
- [35] A. Gulbrandsen, P. Vixie, "A DNS RR for specifying the location of services (DNS SRV)," RFC 2052, Oct. 1996.
- [36] O. Schelen, S. Pink, "Resource reservation agents in the Internet," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pp 153-156, July, 1998.
- [37] P. Pan, E. Hahne, H. Schulzrinne, "BGRP: A Tree-Based Aggregation Protocol for Inter-Domain Reservations," in *Journal of Communications and Networks*, June, 2000.
- [38] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," RFC 1889, Jan. 1996.
- [39] X. Wang and H. Schulzrinne, "RNAP: A Resource Negotiation and Pricing Protocol", *International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'99)*.
- [40] X. Wang and H. Schulzrinne, "Incentive-Compatible Adaptation of Internet Real-Time Multimedia," Columbia University Technical Report CUCS-009-00, Apr. 2000.
- [41] D. Sisalem and H. Schulzrinne, "The multimedia Internet terminal (MINT)," *Journal of Telecommunications*, vol. 9, pp. 423-444, 1998.
- [42] G. Bianchi, A.T. Campbell, and R.R.-F. Liao, "On utility-fair adaptive services in wireless networks," *6th International Workshop on Quality of Service (IEEE/IFIP IWQoS'98)*, Napa Valley, CA, May 1998.
- [43] ITU-T P.800, "Methods for subjective determination of transmission quality".
- [44] O. Ostberg, B. Lindstrom, and P.-O., Renhall, "Contribution of display size to speech intelligibility in video-phone systems", *International Journal of Human-Computer Interaction*, 1(1), pp 149-159, 1989.
- [45] A. H., Anderson, E. G. Bard, C. Sotillo, A. Newlands, G. Doherty-Sneddon, "Limited visual control of the intelligibility of speech in face-to-face dialogues," in *Perception and Psychophysics*, 59(4), 580-592., 1997.
- [46] A. Watson and M. A. Sasse, "Evaluating audio and video quality in low-cost multimedia conferencing systems," *Interacting with Computers*, Vol. 8 (3), pp. 255-275, 1996.
- [47] E. A. Isaacs and J. C. Tang, "What video can and cannot do for collaboration: A case study," *Multimedia Systems*, vol. 2, pp. 63-73, 1994.
- [48] C. Lee, J. Lehoczky, R. Rajkumar and D. Siewiorek, "On Quality of Service Optimization with Discrete QoS Options," *Proceedings of the IEEE Real-time Technology and Applications Symposium*, June 1999.
- [49] R. Vaccaro, "Digital control, a state space approach", McGraw Hill, New York, 1995