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

X. Wang, H. Schulzrinne
xwang@ctr.columbia.edu, schulzrinne@cs.columbia.edu

Abstract— **In this paper, we present a Resource Negotiation and Pricing (RNAP) protocol and architecture to enable users to select and request network services based on their requirements, and dynamically re-negotiate services in response to changes in service prices or user requirements. The protocol also provides mechanisms for the network to dynamically formulate prices and communicate pricing and charging information to the users. We outline a general pricing strategy in this context. RNAP provides a framework in which rate-adaptive applications can adapt their sending rates and QoS requirements in response to price increases during network congestion, or changes in user requirements. We discuss candidate algorithms by which applications can carry out this adaptation. We also present experimental results based on a simplified implementation of RNAP.**

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 back-off 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 a Resource Negotiation and Pricing (RNAP) protocol and architecture with the following main properties:

1. enable a user to select from a set of available network services with different characteristics and offering different levels of QoS, based on user requirements and budget;
2. enable the user and network to dynamically re-negotiate a contracted service, allowing price and transmission parameters to be adjusted according to changing network conditions and user requirements;
3. provide mechanisms for the network to dynamically formulate prices in a distributed manner, compute a total price for a service, and communicate pricing and billing information to the customer.

The proposed protocol and architecture and pricing mechanism are intended to co-exist with current Internet QOS schemes (e.g. those proposed within the int-serv and diff-serv frameworks), and work in a scalable manner over a variety of network architectures. We present RNAP as a stand-alone protocol, but it is also possible to implement some components of RNAP as a layer on top of RSVP or other hop-by-hop reliable signaling protocols.

RNAP is intended for use by both adaptive and non-adaptive applications. Non-adaptive applications may choose services that offer a static price, or absorb any changes in price while maintaining their sending rate. Adaptive applications adapt their sending rate and/or choice of network services in response to changes in network service prices. RNAP provides a framework within which an application can adapt so as to obtain the best value from the network. In this paper, we also discuss candidate algorithms by which applications can carry out this adaptation so as to optimize user satisfaction under the constraint of a fixed budget.

The paper is organized as follows. Section II contains a brief discussion of related research. In sections III and IV we discuss the RNAP protocol, and also propose two alternative modes in which RNAP can be implemented: a distributed architecture suitable for per-flow (RSVP-like) reservations, with control messages following the data path; and a centralized architecture suitable for per-class (diff-serv-like) reservations, in which control messages touch each administrative domain only once. Pricing and charging mechanisms in RNAP are discussed in Section V. Section VI presents some candidate adaptation algorithms. Finally, in Section VII, we present experimental results demonstrating some of the important features of our work, 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 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][54]. These studies were limited to a local system, and did not address the interaction of the local system with a large network. Liao [47] 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 the multimedia quality, and sometimes an application is even not 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][27], the resources are priced to reflect demand and supply. The pricing model in these approaches is usage-sensitive - it has been shown that usage-sensitive pricing results in higher utilization than traditional flat (single) pricing [15]. 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 communicating 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.

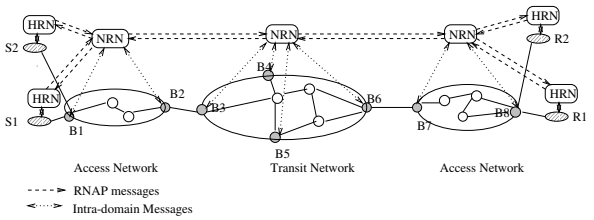


Fig. 1. RNAP-C Architecture

In [28], 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. 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. An access domain A may receive requests for a service in a certain direction passing through a neighboring transit domain B from one or more users, and use RNAP to reserve resources for the flow or flow-aggregate in domain B.

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 entities.

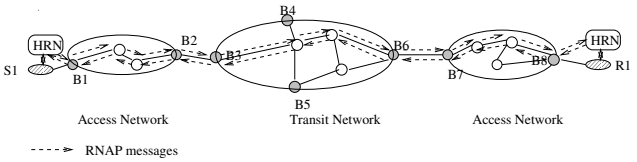


Fig. 2. RNAP-D Architecture

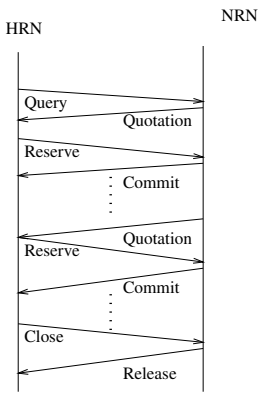


Fig. 3. RNAP messaging sequence between HRN and NRN.

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 [30]. The NRN either has a well-known address, or is located via the service location protocol [36]. 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 after introducing specific RNAP messages 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 renegotiate services. The protocol messages are briefly discussed in Section IV-A, followed by an explanation of how resource reservation works end-to-end, in Section IV-B. The aggregation and de-aggregation of RNAP messages to make the protocol scalable in core networks, and resource negotiation for flow-aggregates in the core network, are discussed in Section IV-C. The use of RNAP for reservation for a multicast transmission is briefly discussed in Section IV-D, and some protocol reliability and security mechanisms are presented in Section IV-E. A more detailed description of RNAP is given in [43].

A. Protocol Messages and Message Fields

We now briefly introduce the names and functionalities of the important RNAP message fields, followed by the protocol messages.

A.1 Message Fields

Id: The *Id* field contains 3 sub-fields: *Flow Id*, *Aggregation Flag*, and *Aggregate Flow Id*. The*Flow Id* defines a flow for which services are negotiated. For individual flows, the *Flow Id* is based on one or a combination of source IP address and port, destination IP address and port, and transport protocol. For aggregates of flows, it is based on the source or destination subnet addresses. The purpose of the *Aggregation Flag* and *Aggregate Flow Id* sub-fields is explained when we discuss message aggregation, in Section IV-C.

Service: The *Service* field contains a number of sub-fields identifying a particular delivery service, for example, the service name, the starting and ending times for the service, negotiation capability, negotiation interval, preemption capability, parameters defining customer traffic such as peak and average rate, and parameters specifying the QoS provided by the service such as loss rate and maximum delay.

Price: The *Price* field contains a number of sub-fields, including the price being quoted by the service provider for a service, and the charges corresponding to a particular customer *Flow Id*.

A.2 Messages

Query: The HRN uses *Query* messages to request a price quotation from the NRN for one or more services, for a flow or group of flows belonging to the customer. The message consists of a set of *Flow Ids*, and one or more service fields accompanying each *Flow Id*. The HRN specifies a set of requirements (such as service time and QoS) with each service, by setting appropriate service sub-fields. When the HRN sends a *Query* for the first time, the NRN also sets up state information for a new *RNAP session*.

Quotation: Upon receiving a *Query* message, the NRN determines the price for each service for which quotations were requested, and returns a list of *Service* and *Price* pairs inside a *Quotation* message. When a *Query* message has null *Service* list, the NRN returns quotations for all available services using default values of unspecified service parameters; it does not return quotations for services which have one or more mandatory parameters to be specified by the customer. In addition to asynchronously sending *Quotation* messages, the NRN also sends out *Quotation* messages periodically. A *Quotation* message sent periodically contains price quotations for all services requested by the customer.

Reserve: The HRN sends a *Reserve* message to apply for services with specified service parameters for a flow or group of

flow. A *Reserve* message is sent at the beginning of a session to request services for the first time, and thereafter, periodically at *Negotiation Intervals*, and also asynchronously at any time. In general, each *Reserve* message carries one or more *Flow Id-Service-Price* triples. The function of the *Price* structure in this context is explained in Section V-A.

Commit: The *Commit* message is generated by the NRN in response to a *Reserve* message, and consist of a set of *Flow Id*, *Service*, *Status* and *Price* 4-tuples. The reservation request indicates the desired and minimum acceptable QoS (in *HRN Data* fields presented in Section V-C). The *Reserve* message keeps track of the currently allocated resources, which fall within that range. For each *Flow Id* , the NRN determines the admissibility of the flow, and returns this information in the *Status* field, as *Admit_Complete*, *Admit_Incomplete* or *Reject*. The admission policy is specific to the service, and need not be administered by the NRN. The *Status* is *Admit_Incomplete* 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 NRN returns the price for providing the service in the *Price* field (normally the same as the price quoted in the preceding *Quotation* message). If the *Commit* is in response to a re-negotiation *Reserve* request in an ongoing session, the NRN also returns the amount charged for each service in the preceding negotiation period, and the accumulated charge since the beginning of the session.

Preempt: If a *Service* is set as preemptable, the NRN may send a *Preempt* message to the HRN to unilaterally change the service parameters or price, or even terminate the service, at any time during a *Negotiation Interval*. This allows the provider to make room for other more important flows urgently, without waiting for the end of a *Negotiation Interval*. A preemptable service would be expected to cost less than an equivalent non-preemptable service.

Close and Release: A *Close* message is sent from the HRN to the NRN to signal the end of an RNAP session. The NRN sends a *Release* message in acknowledgment.

B. RNAP Messaging Scenarios

We have discussed how RNAP messaging works between a HRN and its access network NRN (or equivalently in RNAP-D, the first-hop LRN). We now consider how RNAP works to reserve resources for an user flow **end-to-end**, across multiple network domains.

B.1 End-to-end Resource Negotiation for a Single Customer in RNAP-D

We consider how a customer reserves resources for a flow or group of flows to a particular destination address.

1. The HRN sends a *Query* message to the first hop LRN (FHL). 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 and sends it upstream. 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. Periodic *Quotation* messages are also sent by the LHL hop-by-hop upstream, as above.
2. The HRN periodically sends a *Reserve* message to the FHL, and receives a *Commit* message in an identical manner to the *Query-Quotation* pair. As the *Commit* message is forwarded upstream, in addition to the committed price being incremented at each router, the incremental charge for each service at that router is added on as well. *Query* and *Reserve* messages may also be sent asynchronously at any time in a similar manner.

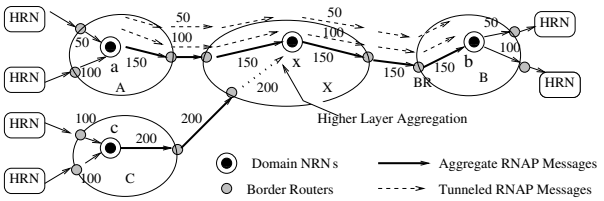


Fig. 4. Example RNAP-C message aggregation.

B.2 End-to-end Resource Negotiation for a Single Customer in RNAP-C

We now consider resource reservation for a customer flow traversing multiple network domains; each domain implements RNAP-C, with a controlling NRN. As mentioned earlier, 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 (mechanisms for doing this are discussed in Section IV-B.1).

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

C. End-to-end Resource Negotiation with Aggregation of Customer Flows

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 first 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. We then address the related issue of how RNAP would be used between two adjoining network domains, to negotiate for resources in bulk for a flow-aggregate.

C.1 Aggregation and De-aggregation

We consider aggregation of RNAP messages belonging to a number of different senders with the same destination network address. The aggregation is carried out by a NRN at node 'a' (Fig. 4). The destination network for the aggregate of flows is network B. A corresponding de-aggregation is carried out by a NRN at node 'b', in destination network B. A single level of aggregation and de-aggregation - per-flow RNAP message aggregation is carried out upstream of node 'a'. We discuss a refinement of this method to permit multiple levels of aggregation (but a single step de-aggregation) in [44] - this is a tree-based aggregation algorithm similar to that in [40][41]. Due to limited space, we do not attempt to describe multi-level aggregation in this paper. Also, for simplicity, we assume the RNAP-D architecture - aggregation and de-aggregation can also be done by RNAP LRNs by a virtually identical mechanism.

The NRN at node 'a' aggregates RNAP messages for user flows which have destination network address B (determined by Internet Routing Registry (IRR) databases, such as RADB [29], look up), and also use the same service (or similar services) and have similar negotiation intervals. Consider the aggregation of *Reserve* messages (this also applies for *Query* messages). NRN 'a' creates an aggregate *Reserve* message, with the source address in the *Flow Id* set to 'a', and the destination address set to the network address B. It also sets the *Aggregation Flag* in the *Id* structure, identifying the message as an aggregate message. NRN 'a' also "tunnels" the incoming per-flow *Reserve* messages up to node 'b', so that per-flow reservation can be resumed in destination network. In each per-flow *Reserve* message, the *Aggregate Flow Id* is set to the *Flow Id* of the aggregate *Reserve* message, 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.

At border routers of access networks, filters are set up so as to intercept aggregate RNAP messages as well as tunneled per-flow RNAP messages. For instance, the border router 'BR' (Fig. 4) is set up to intercept UDP packets with destination address set to the network address B and port number set to the RNAP port. Aggregate *Reserve* messages and tunneled per-flow messages are sent up to the transport layer. The border router looks up the address of the NRN for domain B (node 'b') through DNS SRV, and forwards all such messages to node 'b'. It is possible that a NRN is implemented at the border router itself, in which case the forwarding is not necessary. The per-flow *Reserve* messages arriving at NRN 'b' are processed normally, allowing per-flow resource reservation within domain B. The aggregate *Reserve* message (identified as such by its *Aggregation Flag*) is not processed, but is used to maintain state information mapping the aggregate to its constituent flows, for charging purposes.

There is a similar message flow for RNAP response messages (*Quotation*, *Commit* and *Preempt*) in the reverse direction. NRN 'b' acts as the aggregating entity for the response messages and the aggregate response message is sent to the aggregation point 'a'. The per-flow messages are also tunneled to node 'a'.

The aggregation entity on the source network side is also responsible for de-aggregation of RNAP response messages. That is, it checks the mapping between an aggregate session and per-flow RNAP response messages, and, 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.

C.2 Resource Negotiation for Flow-Aggregates and Advance Reservation

At the aggregation point, the NRN acts as the client negotiator (or HRN) for an aggregate session in negotiations with the downstream NRN (for example, referring to Fig. 4 again, the aggregating NRN 'a' negotiates with NRN 'x'). In general, the client negotiator will negotiate resources for an aggregate session corresponding to the per-flow reservation requests. To avoid frequent re-negotiation, however, it is likely that the client negotiator will increment or decrement the requested resources with some minimum granularity. When the sum of per-flow requests approaches the resources reserved (or reach some threshold) for the aggregate, the client negotiator will reserve an additional block of resources. Similarly, the requested reservation is decremented in blocks as required. The larger the block, the less frequently the aggregate session needs to be re-negotiated, but a higher holding cost is incurred for resources which may be under-utilized. In general, negotiation of resources in blocks results in a fairly static service, and periodical re-negotiations, if any, would occur with a much longer negotiation interval. Hence, price *Quotation* messages for the aggregate session will probably only be sent asynchronously in response to *Query* messages, when an additional block of resources needs to be reserved or removed.

The NRN at an aggregation point may also forecast a certain demand to a particular destination network, and could negotiate a large block of resources in advance, using the advance reservation mechanism. The HRN or client NRN indicates an advance reservation using the *Starting Time* and *Ending Time* fields in the Service description. The server NRN initializes session state at the conclusion of the advance negotiations, and maintains the state until the actual transmission has been completed.

If the client negotiator chooses to cancel part or all of a reservation made in advance, it can re-negotiate with the server negotiator to try to 'sell back' previously reserved resources at an agreed price. The price eventually agreed upon would probably reflect any cancellation or holding cost fee the server negotiator wishes to charge. The server negotiator may also offer to buy back resources reserved in advance, for more important usage.

D. Negotiation for a Multicast Session

RNAP request messages for a multicast session will negotiate services for flows to multiple destinations. In this case, the *Flow Id* field carries the IP multicast address. On receiving such a message, the NRN (or LRN) communicates with a multicast routing protocol to determine if the multicast tree diverges into two or more branches within the NRN-administered domain (or at the LRN-router). If so, multiple copies of the message are formed and forwarded to different directions.

The NRN or LRN maintains session state allowing it to aggregate RNAP response messages traveling upstream at divergence points on the multicast tree. As response messages are aggregated, the pricing and charging information from individual response messages are summed to obtain the corresponding information for the aggregate message.

E. Protocol Reliability and Security

RNAP messages are sent using the UDP protocol. In both RNAP-C and RNAP-D models, periodical RNAP messages provide a natural way of protecting against loss. RNAP also allows the HRN to asynchronously solicit any service related information at any time during the negotiation session. To protect against asynchronous message loss and as an additional protection against synchronous message loss, the HRN continues to retransmit a request with exponential back-off (for congestion control) until a response is received. If one or more *Reserve* messages are lost in transmission, the network continues to provide service based on the latest *Reserve* for a specified time-out period.

Failure of a negotiation server or of a device storing RNAP state information, and network partitions need also to be considered. A back-up NRN may be needed for the RNAP-C model. If there is any possibility of resource unavailability due to element failure or route change, a re-negotiation process is triggered from the influenced domain to the corresponding neighboring domains to allow new resource agreements to be reached. When a device that stores customer charging information is down for a period, the charge for the period is asynchronously retrieved after the device recovers.

For security, RNAP messages may be authenticated and encrypted either similar to RSVP [38] or using IPSEC [39].

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. In Section V-C, we discuss a mechanism for sharing of charges by the sender and receiver, which is followed by multicast charging.

Finally, in section V-D, 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* structure carried in 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. The *HRN Data* contains various sub-fields for sender-receiver negotiation (Section V-C).

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-D, 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.

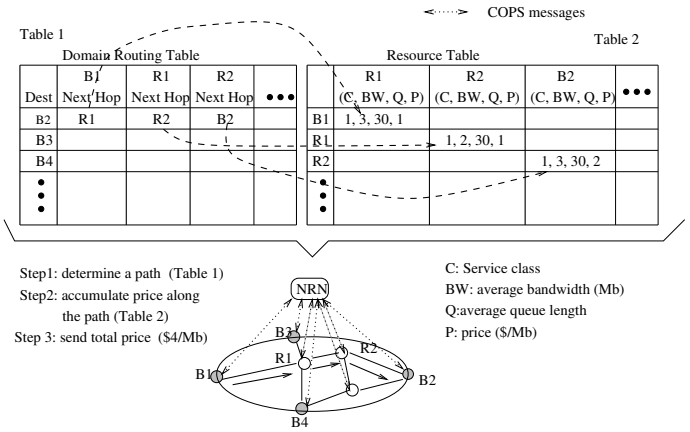


Fig. 5. Price Formulation in RNAP-C

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 in the domain using Dijkstra's algorithm. A similar idea has been explored in [40] 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 transmit domain (Fig. 5). 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-D). 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 [30] 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.
2. Prices are computed at the network boundary, and communicated to the NRN. For price calculation, there are two alterna-

tives.

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.

3. 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 [42] 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.

In the above schemes, we assume that a domain has one NRN. A domain could also have multiple NRNs, each NRN residing at an ingress router. In this case, the ingress router does not need to send periodic per-session reports to a centralized NRN, and pricing, charging, and RNAP messaging are done directly from the ingress router. Reliability concerns make a more distributed architecture (multiple NRNs, or RNAP-D) preferable. But some management goals (for instance, all NRNs in one domain need to have coherent view of the resource at internal routers to allow them to make correct admission decisions) may make a centralized policy more attractive.

B.3 Charge Formulation for Multiple Domains and Flow Aggregates

When a customer flow spans more than one administrative domain, each domain computes incremental prices and charges for the flow using its own pricing strategy and architecture, and the total end-to-end price and charge are obtained in a hop-by-hop manner (with each domain representing a single hop) as in Section V-B.1.

When a set of flows enter a domain as a flow-aggregate, the NRN (or network domain as a whole in RNAP-D) carries out messaging and charging as if the aggregate belonged to a single customer. The NRN in the aggregating domain (or LRN at the aggregation point) is responsible for mapping the total charge into charges for individual customer flows or flow-aggregates.

C. Shared Charging and Multicast Charging

RNAP also allows sender and receivers to share the charge, which is facilitated by the *HRN Data* fields in the negotiation messages. The *HRN Data* contains subfields *HRN Account* and *HRN Charge Fraction*, as well as optional subfields *Minimum Data Rate*, *Maximum Data Rate*. The *HRN Account* field identifies the account from which charges are to be debited. The corresponding *Charging Fraction* field indicates the fraction of the total charge to be borne by the HRN. The minimum and maximum data rate fields are included to allow the sender and receiver HRNs to reach a basic agreement on the desired transmission rate.

The sender HRN and receiver HRN negotiate through the exchange of *Query/Reserve* and *Quotation/Commit* messages. Any *Query* or *Reserve* message with *HRN Charge Fraction* less than one is forwarded automatically by the last hop NRN to the receiver. The receiver HRN sends its modified *Query* or *Reserve* message to the last hop NRN, which generates a *Quotation* or *Commit* message and send it upstream. A *Commit* message either accepts or denys the service requested by the *Reserve* message depending on if the respective *HRN Charge Fraction* fields in the *Query* (or *Reserve*) messages received from the sender and receiver HRNs add up to greater than one.

In a multicast session, either sender or the receivers could negotiate separately, or they could both participate in negotiation. If the sender or receiver is solely responsible for negotiation and payment, messaging sequence is similar to the simple scenario considered for unicast. In receiver negotiation with partial or full sender payment, the sender may specify a maximum expenditure through the *Price:New Price*. If the sender negotiates, with partial or full receiver payment, the sender negotiates resources based on the overall demand and willingness to pay, and may request partial reservation on some paths on which receivers have a lower willingness to pay.

More detailed descriptions regarding the negotiation process can be found in [43][44].

D. Pricing Strategy

In the previous sections, we discussed the mechanisms for pricing in RNAP, including the pricing structure used by RNAP messages, and the formulation of end-to-end prices and charges in the RNAP architecture. 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 competitive market model defines two kinds of agents: consumers and producers. The price computation is performed periodically, with a price update interval τ. In general, the price update interval at a router is 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.

The router supports multiple services. We also assume that the router is partitioned to provide a separate link bandwidth and buffer space for each service, on each port. In the discussion that follows, we consider one such logical partition. 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 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. It can be represented as:

$$p_u = f(Service, destination, time_of_day, ...) \quad (1)$$

The usage_charge $c_u(n)$ for a period n in which $V(n)$ bytes were transmitted is given by:

$$c_u(n) = p_u * V(n) \quad (2)$$

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:

$$p_h^i = \alpha^i * (p_u^i - p_u^{i-1}) \quad (3)$$

Where α^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:

$$c_h(n) = p_h * R(n) * \tau \quad (4)$$

where τ 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.

Defining a usage charge and a holding charge separately allows the customer to reserve resources conservatively, without penalizing him excessively for unused resources. As an example, an audio stream can have periods of silence, when the reserved resources are not used by the customer. Also, not charging the customer purely on the basis of reserved resources makes it easier for the customer to keep his reservation level constant even during idle periods.

Congestion Charge:

Network congestion arises from a scarcity of network resources, generally bandwidth and buffer space. The congestion charge is imposed only if 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]:

$$\begin{aligned} p_c(n) &= \min[\{p_c(n-1) \\ &\quad +\sigma(D,S) * (D-S)/S, 0\}^+, p_{max}] \end{aligned} \quad (5)$$

Where D and S represent the current total demand and supply respectively, and σ is a factor used to adjust the convergence rate. σ 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

$$c_c(n) = p_c(n) * V(n). \quad (6)$$

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

$$session_charge = \sum_{n=1}^{N} (c_h(n) + c_u(n) + c_c(n)) \quad (7)$$

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

E. Stability of Negotiation Process

Since the re-negotiation of network services will probably be driven mainly by price changes, it is appropriate to briefly consider the stability of the negotiation process. The stability of the tâtonnement process, through which the supply and demand reach equilibrium, has been demonstrated in the literature [25][26]. Transmission delays should be small compared to the negotiation interval, which we envision to be on the order of minutes, and are hence unlikely to significantly affect this stability. A more important factor is frequent bandwidth adaptation by a number of users sharing the same link bandwidth. In Section VII-B, we will discuss a situation in which this may lead to oscillatory behavior, but this can 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.

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. The user must define quantitatively, through a *utility function*, the benefit or value associated with a sending rate

and QoS parameter-set obtained from a particular network service. Note a user utility function is private to the user itself. The rest of this section is divided into two parts. In Section VI-A, we discuss the definition and general characteristics of an utility function. In Section VI-B, we discuss how sending rate and transmission QoS parameters are selected based on the utility function.

A. The Utility Function

We consider a set of user applications, required to perform a task or *mission*, for example, audio, video, and whiteboard 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 *utility* of a reservation request denotes how beneficial the corresponding network resource allocation would be 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. The object of the adaptation is to select a set of transmission parameters that gives the maximum possible utility relative to the cost of obtaining this service, subject to the user budget constraint.

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] under diff-serv, and several classes of service under Assured Forwarding (AF) [8], also 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.

The measurement of subjective quality of multimedia transmissions has been reported by a number of researchers [48][51][52]. 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. But in our framework, perceived value very strongly reflects individual user preferences, and the application task being performed. We consider it likely, therefore, 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 the utility function

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. For example, 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. For example, 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. For example, the requirement of a weekly meeting between native speakers will probably have a lower quality requirement than a conferencing system teaching a foreign language and hence the users will see a higher value for using high quality transmission for the latter.

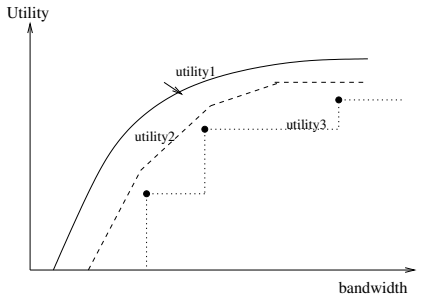


Fig. 6. Some example utility functions

A.2 Utility as a Function of Bandwidth

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 [51][52] 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. 6 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

In this section, we consider how changes in utility function may influence the resource distribution. The operations we consider are an offset applied uniformly to the utility over all bandwidths, and multiplicative scaling of the utility function. We discuss the operation qualitatively here, and present some experimental results in section VII-B.4.

The utility function represents the relative preference of the user for different bandwidths. A constant (bandwidth-independent) offset to the utility function will not influence the resource distribution as long as the valuation of a bandwidth is higher than its cost.

On the other hand, a constant offset of the utility function changes the minimum perceived value. The minimum perceived value represents how much the user is willing to pay to just keep the application alive. Lowering this value allows the application to be terminated more readily during congestion (high cost). If a user values an uninterrupted service very highly, he increases the perceived value of the "keep alive" bandwidth.

A multiplicative scaling of the utility function by a factor greater than one tends to increase its bandwidth share, since it results in a bigger additive increase in perceived surplus at higher bandwidth than at lower bandwidths. Effectively, the demand elasticity of the application is reduced. The opposite effect is observed when the

scaling factor is less than 1.

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, α and β, 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 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:

$$\begin{aligned} \max\ & [U(x) - C(x)] \\ \text{s. t.}\ & C(x) \leq b \\ & x_{min} \leq x \leq x_{max} \end{aligned} \tag{8}$$

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.

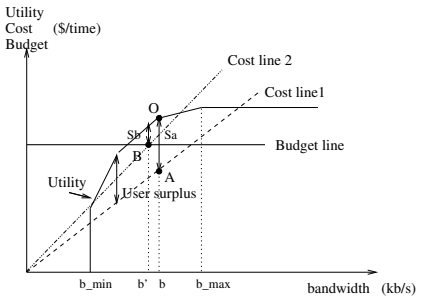


Fig. 7. A perceived value based rate adaptation model

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. 7. 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. 7 case A)
or else the bandwidth corresponding to the intersection point of the cost line with the budget line (b' in Fig. 7 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 8, 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 [49][50][52][53]. 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 [49]. To take into account the interdependencies among applications, the utility of the i_{th} application should,

in general, be written as $U^i(x^1, ..x^i, ..x^n)$, 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 $\hat{U}(U^1(\cdot), ..., U^n(\cdot))$, 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 :

$$\hat{U} = \sum_i [U^i(x^1, ..., x^i, ..., x^n)] \tag{9}$$

and the optimization of surplus can be written as

$$max \sum_i [U^i(x^1, ..., x^i, ..., x^n) - C^i(x^i)]$$
$$\text{s. t.} \sum_i C^i(x^i) \leq b$$
$$x^i_{min} \leq x^i \leq x^i_{max} \tag{10}$$

Where x^i_{min} and x^i_{max} 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.

A simpler method is to use a similar approach as to that for the single application case, along with some heuristics. In this case, we make the simplifying assumption that the individual application utility functions 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^i_l $(l = 1, 2, ..L^i)$, and the utility function is piece-wise linear. Segment l (the straight line between levels l and $l + 1$) has a slope k^i_l. 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 $l_{opt}(l = 1, 2, ..L^i)$, with bandwidth B^i_{opt}, at which the surplus (utility minus cost) is maximized for that application. Let the cost of the targeted bandwidth be $C^i_{opt}(B^i_{opt})$.
2. If the total expenditure needed for the system, $\sum_i C^i_{opt}(B^i_{opt})$, 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 $(u^i_l - C^i_l)$ 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.

A more complicated search algorithm and/or more sophisticated heuristics can be envisioned for a more general case in which the utility function of each application depends on some subset of transmission parameters of the application in question as well as those for other application streams. The focus of the current work is to develop and demonstrate the basic adaptation framework, and develop some insights based on the results. Also, the development of more sophisticated heuristics may be easier once particular system-wide utility functions have been developed and tested.

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 Controlled Load service proposed within the intserv model.

Although our implementation was highly 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 *Com-*

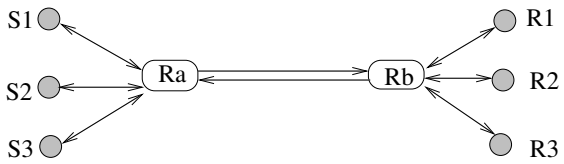


Fig. 8. Testbed setup

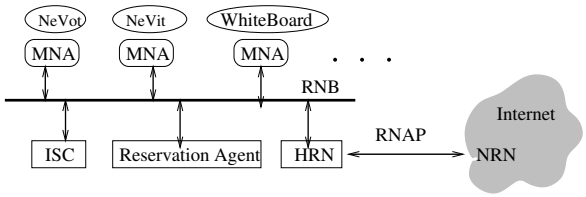


Fig. 9. The architecture of the extended MINT system

mit information. This information includes "bandwidth available" information in the *Price:HRN Data → Maximum Rate* field.

rsvpd version 4 from ISI [33] was extended to support RNAP. Resource reservation on a link was performed using Class-Based Queueing (CBQ) [34], as part of the ALTQ package [35].

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 [32][31] 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 [30][31]. 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 RNAP session at the node.

Since per flow queuing was used, congestion charging was not applied when a per-flow overflows. 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. 8.

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 a session to adapt to any data rate it intended to send.

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). In addition to the CL traffic, 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) [45] system. The audio and video application components of MINT, NeVoT and NeViT, support rate adaptation.

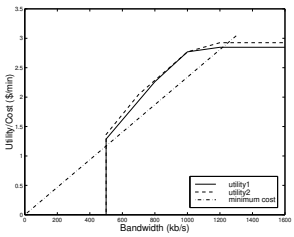


Fig. 10. Utility functions used in the experiments of section VII-B.1 and VII-B.3

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) 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, Utility1 and Utility2, shown in Fig. 10.

Fig. 11-a1, 11-a2, and 11-a3 show different aspects of adaptive behavior when Utility1 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. 11-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. 11-a1 also shows the variation with time of the total bandwidth reservation, and Fig. 11-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. 11 a3 shows a gradual decrease in the surplus obtained by each user until the price sta-

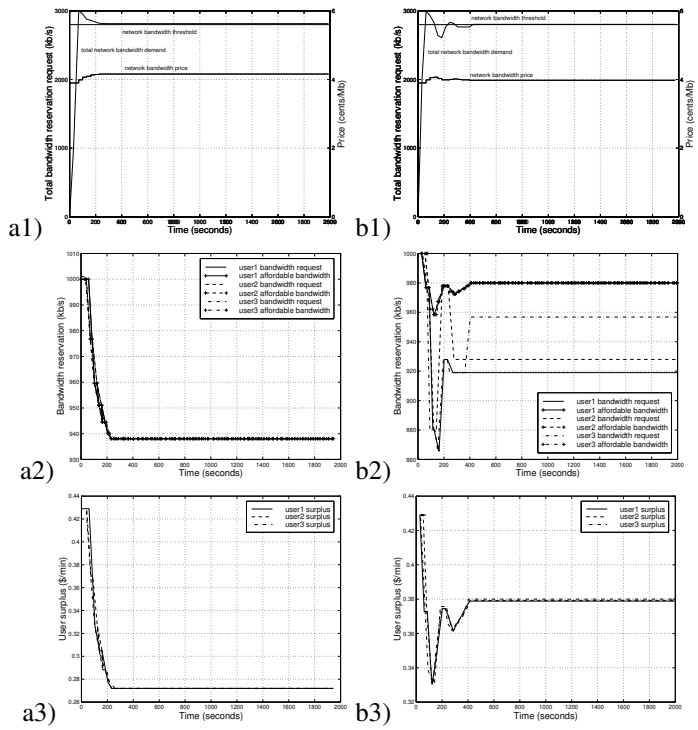


Fig. 11. Allocation of bandwidth and surplus for three competing users sharing a link. a1, a2, and a3 show the results when the users all have the Utility 1 function from Fig. 10, and b1, b2, and b2 show corresponding results when the users have the Utility 2 function from the same figure

bilizes. All users are observed to have nearly identical adaptation traces.

The second experiment uses Utility2 in Fig. 10. Utility2 differs from Utility1 in that the optimal bandwidth (at the initial uncongested 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. 11 b, the adaptation traces are observed to be different from that shown in Fig. 11 a. 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 bandwidth 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 [55] 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:

$$r_{i+1} = \begin{cases} r_i - \alpha_0(r_i - r^*) - \alpha_1(r_i - r_{i-1}), & \text{if } \frac{|SP(r^*) - SP(r_i)|}{SP(r_i)} > \theta \\ r_i, & \text{otherwise} \end{cases} \quad (11)$$

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

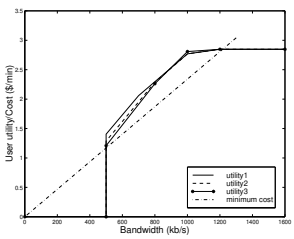


Fig. 12. Utility functions with different bandwidth sensitivity

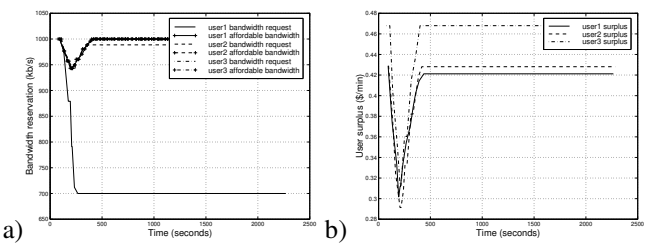


Fig. 13. Bandwidth reservation a) and perceived surplus b) when all users have different demand elasticity

obtaining bandwidth x. Quicker convergence is attained by making α_0 large, while the overshoot is minimized by making α_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 θ %. In the experiment, α_0, α_1 and θ are set separately as 0.4, 0.6, and 2%, which led to the quick convergence without large overshoot.

Fig. 11 b-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 θ % of optimal) choice of some users. All three users end up with final 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. 12) for different users. An utility function with a smaller slope reflects a higher elasticity in the bandwidth requirement of the user. Fig. 13-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. 14-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

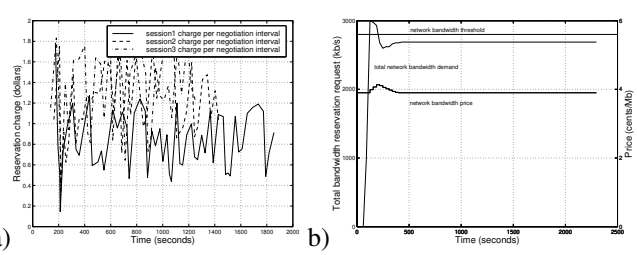
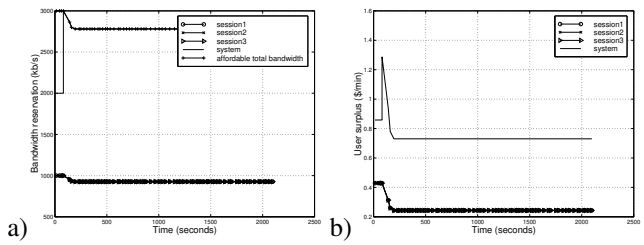
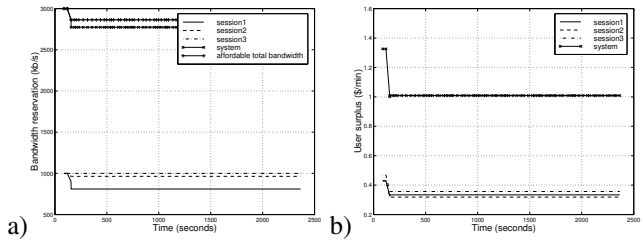


Fig. 14. Network charges for different users a) and the total network bandwidth demand and price b) when the users have different demand elasticity



a) b)

Fig. 15. Bandwidth reservation a) and perceived surplus value b) for adaptation across media sessions in a system, all sessions having the same utility



a) b)

Fig. 16. Resource reservation a) and perceived surplus value b) among sessions of a system with different bandwidth sensitivity

(Fig. 13-b).

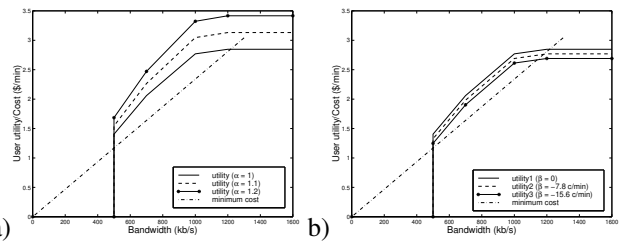
Thus, users with less stringent bandwidth requirements express this flexibility through a less bandwidth-sensitive utility function, and bear a greater share of reductions in bandwidth for congestion-control. Users with more bandwidth-sensitive requirements have to pay a higher charge during congestion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

B.3 Adaptation Across Media

The experiments so far show how a utility function serves to guide bandwidth adaptation by an application in response to congestion-sensitive pricing in the network, and how bandwidth is shared by competing applications based on individual demand elasticities. 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 two experiments are presented.

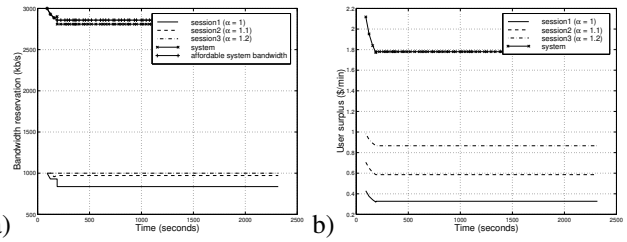
In the first experiment, the system consists of three media sessions, all of which have the same utility function, Utility1 shown in Fig. 10. When the system budget is exceeded due to congestion, the HRN adjusts the application bandwidths downwards according to the adaptation algorithm described in Section 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. 15 a).

The second experiment is similar except that the three media sessions have different utility functions shown in Fig. 12. Fig. 16 a) 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.



a) b)

Fig. 17. Equivalent utilities under multiplicative scaling a) and additive shifting b)



a) b)

Fig. 18. Bandwidth reservation and perceived surplus for utilities scaled multiplicatively by different amounts

B.4 Linear Operations on Utility Functions

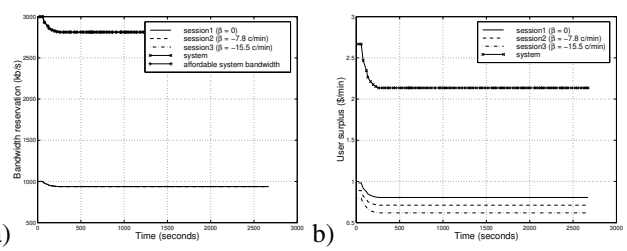
In this section, we show how linear operations on the utility function, namely multiplicative scaling by a weight α, and additive or subtractive shifting by an amount β, influence bandwidth selection and distribution. The experiment studies bandwidth distribution between multiple sessions in a system 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. 12). 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. 17 a).

Fig. 18 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 α 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, so that the utility function shifts upwards or downwards. In Fig. 19 b), the utility1 function (which is the same as utility1 in Fig. 12 a) is shifted downwards and form utility2 and utility3. Three different sessions are assigned different utility functions.

The results shown on Fig. 19 a) shows that all three sessions are allocated the same bandwidth though Fig. 19 b) shows that the allocation results in different values of perceived surplus. This



a) b)

Fig. 19. Bandwidth reservation and perceived surplus for utilities shifted additively by different amounts

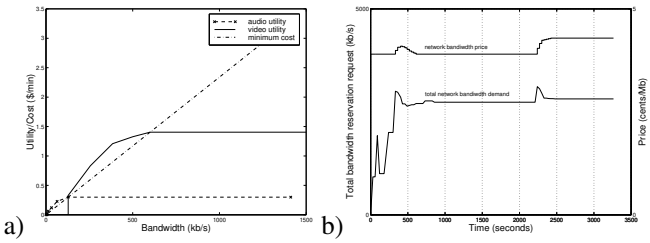


Fig. 20. a) Audio and video utility functions used for adaptation by MINT b) Price and total bandwidth variation in the same experiment

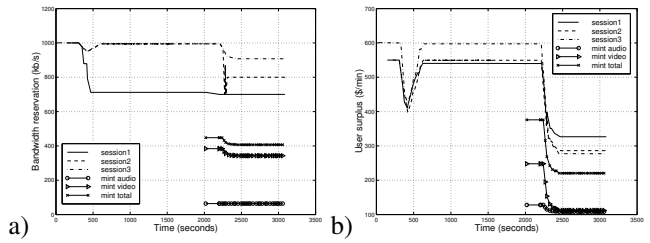


Fig. 21. Individual bandwidth reservations and perceived surplus in the adaptation of Mint applications

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. 20 a).

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. 12. 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 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 develop a protocol and architecture which enables network service negotiation for multiple delivery services and environments. The RNAP protocol enables service negotiation between user applications and the access network, as well as between adjoining network domains. The protocol permits negotiation and communication of QoS specifications, user traffic profiles, admission of service requests, and pricing and charging information for requested services. The periodic as well as asynchronous re-negotiation framework of the protocol enables dynamic, usage sensitive adaptation of service parameters and pricing by the network, if required, and also enables the user application to respond to changes in application requirements. At the same time, the framework provides sufficient flexibility to support users with limited negotiation capability, or with a requirement for very static and predictable service specifications.

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. In either case, the architecture is scalable because it does not assume service reservation with a particular granularity, and incorporates mechanisms for flow aggregation. The two architectures use the same set of RNAP messages, and can co-exist and inter-operate across multiple administrative domains.

The protocol and architectures provide mechanisms for local or 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.

In addition, we have presented a framework 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.

A prototype implementation of the important RNAP functionalities has been described, along with some preliminary measurement results. 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 weighting (or multiplicatively scaling)

and additive shifts on utility functions is examined.

Future development of this work includes deployment of the full functionality of RNAP and more extensive tests for the performance of RNAP signalling in a large scale network.

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," Request for Comments (Informational) 1633, Internet Engineering Task Force, June 1994.

[5] S. Shenker, C. Partridge, and R. Guerin, "Specification of guaranteed quality of service," RFC 2212, 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," Internet Draft, Internet Engineering Task Force, Feb. 1999. Work in progress.

[8] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," Internet Draft, Internet Engineering Task Force, Feb. 1999. Work in progress.

[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] S. Shenker, and L. Breslau, "Two issues in reservation establishment", *Proc. ACM SIGCOMM'95*, Cambridge, MA, August 1995.

[12] K. Nichols, V. Jacobson, and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet," Internet Draft, Internet Engineering Task Force, Nov., 1997

[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 competive 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 & pany.

[26] Hahn F (1982). Stability. In: Arrow KJ and Intriligator MD (eds), "handbook of Mathematical Economics", Volumn II. Noth-Holland, Amesterdam, pp 745-793.

[27] J. Sairamesh, "Economic paradigms for information systems and networks", PhD thesis, Columbia University, New York 1997.

[28] 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.

[29] RADB, A Distributed Database for Internet Routing Registry. http://www.radb.net/docs/.

[30] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, A. Sastry, "The COPS (Common Open Policy Service) Protocol," Internet Draft, Internet Engineering Task Force, Feb., 1999. Work in progress.

[31] R. Yavatkar, D. Pendarakis, and R. Guerin, "A framework for policy-based admission control," Internet Draft, Internet Engineering Task Force, Nov. 1998. Work in progress.

[32] S. Herzog, "RSVP extensions for policy control," Internet Draft, Internet Engineering Task Force, Mar. 1999. Work in progress.

[33] RSVP software release, ftp://ftp.isi.edu/rsvp/release.

[34] 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.

[35] K. Cho, ALTQ: Alternate Queueing for FreeBSD.

[36] J. Veizades, E. Guttman, C. Perkins, and S. Kaplan, "Service location protocol," Request for Comments (Proposed Standard) 2165, Internet Engineering Task Force, June 1997.

[37] A. Gulbrandsen, P. Vixie, "A DNS RR for specifying the location of services (DNS SRV)," RFC 2052 , Oct. 1996.

[38] F. Baker, B. Lindell, and M. Talwar, "RSVP cryptographic authentication," Internet Draft, Internet Engineering Task Force, Mar. 1999. Work in progress.

[39] R. Atkinson and S. Kent, "Security architecture for the Internet protocol," Request for Comments (Proposed Standard) 2401, Internet Engineering Task Force, Nov. 1998.

[40] 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.

[41] P. Pan, E. Hahne, H. Schulzrinne, "BGRP: A Tree-Based Aggregation Protocol for Inter-Domain Reservations". submitted.

[42] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," RFC 1889, Jan. 1996.

[43] 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)*.

[44] X. Wang and H. Schulzrinne, "A Framework for Resource Negotiation and Pricing in the Internet," Technical Report, Columbia University, Sept. 1999.

[45] D. Sisalem and H. Schulzrinne, "The multimedia Internet terminal (MINT),"*Journal of Telecommunications*, vol. 9, pp. 423-444, 1998.

[46] Henning Schulzrinne, "Dynamic configuration of conferencing applications using pattern-matching multicast," *International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'95)*, 1995.

[47] 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.

[48] ITU-T P.800, "Methods for subjective determination of transmission quality".

[49] 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.

[50] 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.

[51] C. Lambrecht and O. Verscheure, "Perceptual quality measure using a spatio-temporal model of human visual system", *Proc. of IS&T/SPIE*, San Jose, Feb, 1996.

[52] 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.

[53] 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.

[54] 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.

[55] R. Vaccaro, "Digital control, a state space approach", McGraw Hill, New York, 1995