A RESOURCE BASED PRICING POLICY FOR REAL-TIME CHANNELS IN A PACKET-SWITCHING NETWORK.

Colin Parris and Domenico Ferrari
The Tenet Group
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California
and International Computer Science Institute
Berkeley, California, U.S.A.

Abstract

In the packet switching networks of the future the need for guaranteed performance on a wide variety of traffic characteristics will be of paramount importance. The generation of revenue, to recover costs and provide profit, and the multiple type of services offered will require that new pricing policies be implemented.

This paper presents a resource based pricing policy for real-time channels (ie., channels with guaranteed performance) in a packet switching network. The policy is based on a set of specific criteria, and the charges for any channel are based on the resources reserved for use by the channel. This reservation charge is based on the type of service requested, the time of day during which the channel exists, and the lifetime of the channel. We argue that the traditional resources are not sufficient to determine a fair reservation charge for a channel offering guaranteed delay bounds, and we introduce the notion of a delay resource in our charging formula. The type of service requested is thus characterized by the amount of the bandwidth, buffer space, CPU, and delay resources reserved. The analysis of this pricing policy is reduced to the analysis of a single node of the network, assuming a homogeneous network. This single-node characteristic increases the scalability and flexibility of the policy. An example of an implementation of this policy is provided.

This research was supported by the National Science Foundation and the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, by AT&T Bell Laboratories, Hitachi, Ltd., Hitachi America, Ltd., Pacific Bell, the University of California under a MICRO grant, and the International Computer Science Institute. The views and conclusions in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of any of the sponsoring organizations.

1. Introduction.

With the ever increasing desire for multimedia applications on computer networks, the need for real-time communication services (i.e. services with guaranteed performance) is readily apparent. In conjunction with the fulfillment of this need, efforts are now being made to determine pricing scheme formats suitable for use in this type of networking environment. While pricing policies exist for non real-time networks (datagram networks) as well as for conventional public utilities networks (telephone and power system networks), they cannot fulfill the requirements that we desire of a pricing policy for an integrated-services network.

This paper presents these requirements, discusses the inadequacies of the conventional pricing policies, and proposes a new pricing policy. The paper begins in Section 2, with the requirements of a useful pricing policy and an overview of the disadvantages of conventional schemes. Next, Section 3 presents and analyzes the proposed pricing scheme. An example of the scheme is discussed in Section 4. Section 5 concludes the paper.

2. Motivation for the Pricing Policy.

This section states the requirements of a useful network pricing policy, provides a brief overview of the conventional pricing schemes, and discusses the disadvantages of each scheme.

The requirements of a network pricing policy are:

- Allow the service provider to collect revenue commensurate with the quality of service provided to the client.
- Allow clients to predetermine the charges for the services they need.
- Discourage client actions that will decrease the efficiency of the network.
- Allow the policy mechanism to be implemented with minimal overhead.
- Allow a simple relationship between the performance characteristics of the network and the revenue derived from the network.

Although most of the requirements stated above are self-explanatory, a brief discussion of some of these requirements may provide some further insight. It is essential that the clients be able to predict and verify the charges for their usage. In some pricing schemes the price of the resource is based on the demand at that time, and clients can bid for the resources that they need. The demand can be viewed as a function of the state of the network and can not be predicted easily. Thus, with such a scheme, clients would not be able to forecast their charges.

The pricing of network services should encourage clients to undertake actions that increase the efficiency of the network and discourage actions that decrease the efficiency of the network. This efficiency should be thought of as a combination of performance and pareto efficiency. A relevant example exists in the use of video conferencing. Video conferencing sessions should generally take place during "working hours", as most of the members involved in the session would generally be available during those hours. Writing large video files to multiple archive video servers could possibly utilize resources equivalent to those of video conferencing, thereby maintaining the same level of efficiency in the network. However, the conferencing session should be given preference over the archive session as by doing so the pareto efficiency is increased. This preference could be given by charging a higher price for service during a specific

time of day, thus forcing "time of day" insensitive traffic out of "working hours". The clients sending the videos to be archived are likely to be discouraged, by the higher charge, from network usage at specific intervals during the day.

The relationship between the performance characteristics and the revenue generated should be such that the effect on the revenue of any change in network resources can be easily determined. This is essential in dealing with the loss of resources due to failures in the network as well as in the quick prediction of the additional revenue that would be generated by the buying or leasing of additional resources or the addition of new guaranteed performance services.

2.1. Disadvantages of Conventional Pricing Policies.

The conventional pricing policies that will be discussed here are: Flat Per-Packet pricing, Peak Load pricing, and Priority Based pricing. A brief overview of each policy will be provided and its inadequacies, with respect to the requirements presented above, stated. A more thorough treatment of each policy can be found in the corresponding reference.

In the Flat Per-Packet pricing policy users are charged according to the number of packets that are sent by them to any destination. This policy fails to satisfy two of the requirements, ie., (1) and (3), stated above. The revenue collected by the service provider is not commensurate with the service provided in that a client who uses a real-time service and sends *p* packets and a client who sends a non real-time message with *p* packets are charged the same amount. In effect, the non real-time users are carrying some of the real-time users charges. Clients are not discouraged from inefficient actions. As the charge to send *p* packets is the same regardless of time of day or network state, clients are not discouraged from sending low priority data during peak hours.

Peak Load pricing policies assume that there are regular times of the day at which the load on the network is heavy. The charge for usage is raised to shift flexible clients off the peak load intervals. When the peak periods are predictable (Static Peak Load pricing), the clients are made aware of both the period and the charges for the period, at some predetermined time before the peak period. When the peak periods are not predictable (Dynamic Peak load pricing), peak load rates and their corresponding charges are varied dynamically with the network load. The charges are applied to the duration of time the channel existed or the number of packets sent during the peak load and non peak load periods [1]. In the case of Static Peak Load pricing, requirements (1) and (3) are not met. The first three requirements, (1),(2), and (3), are not met in the case of Dynamic Peak Load pricing. As with the previous policy, all the clients who use the service during the peak periods must receive the same quality of service, otherwise the clients with lower quality service will be paying for the clients with higher quality service. The same can be said for the non peak periods. In the case where clients are charged based on the number of packets sent, clients may acquire connections and not send any traffic on them, thereby blocking efficient clients from using them while not allowing the service provider to collect any revenue on these connections. In the case that the periods are not predictable, the clients are not able to predetermine their charges since the charges are changing dynamically. Another problem with the pricing mechanism is that the clients cannot verify their charges, as they must rely on the network to give them the current price per packet, which may vary dynamically. In Static Peak Load pricing, clients know the number of packets sent and can verify their charges without network intervention. While Static Peak Load does discourage some inefficient actions by the clients by forcing flexible clients off peak periods, it does not deal with inefficient actions in

non-peak periods, nor does it deal with inefficiencies among clients who remain in the peak periods. Further granularity is needed in discouraging such actions.

The last policy to be reviewed here is that of Priority Based pricing. In this policy, the network will serve clients in the order of their priority levels, and the per-packet charges will be computed accordingly. This policy is more adaptable than Peak Load pricing, as the priorities present a basis for the network to delay lower priority traffic in favor of higher priority traffic. The clients are allowed to set a certain priority level for their traffic , and, if the experienced performance is too low, they may increase their priority level until their performance criterion is met. The per-packet rates may be set and changed due to the demands of the other clients as they increase their priority levels [2]. The main problem with this scheme is that it fails to meet requirements (1), (2) and (4). As with the Peak Load pricing policy, the service provider may not collect revenue commensurate with the service provided, as clients may acquire connections and not use them. This inhibits the use of the connection by other clients, resulting in loss of revenue to the network, as clients are charged on the basis of the packets sent, not on the duration that their connection is held. Clients cannot do productive long-term budgeting as the charges vary dynamically, and the problem of charge verification exists also in this case. Implementation is difficult in this scheme since each packet must be examined to determine its priority and the accounting mechanism will have to record the packets, their priorities, and the associated costs. The bidding among clients as they vary their priorities will also have to be regulated by the accounting mechanism. The overhead of this policy may be unbearable.

A simpler version of priority-based charging is one in which the priority of packets is based entirely on the quality of service that the clients select and is fixed. There is no bidding to increase priorities, as these priorities can only be increased by acquiring a higher quality of service. The charges for each quality of service are fixed and available to the client. Charges are per-packet, and are applied to each packet dependent on its level of service. This policy satisfies requirements (2), (4), and (5). Requirement (1) and (3) are not satisfied, as clients can acquire connections and not send any packets across them. While these clients are not using any of the resources allocated to them, they are decreasing the efficiency of the network by not allowing other clients access to those resources. The total revenue of the network is also decreased, as no revenue is generated by these connections. This policy will be referred to in the sequel as the Multiple Fixed-Priority Based Policy. An example of this pricing policy is given in [3].

As can be seen from the reviews presented above, the conventional schemes are inadequate for our purposes. A new policy is needed that meets all the requirements stated above. This policy is presented in the next section.

3. A Pricing Policy for Real-Time Channel Establishment Scheme.

In this section, the proposed pricing policy will be presented. It will then be analyzed with regard to the requirements, and applied to our real-time channel establishment scheme. The reader is assumed to have some knowledge of the traffic characterization and the channel establishment scheme found in reference [4]. For the sake of simplicity, we assume that only real-time traffic will be present on the network. It should be noted however that the policy could easily support non real-time traffic services.

3.1. The Proposed Pricing Policy.

In its simplest format the proposed pricing policy may be expressed as:

$$Total\_Charge_i = Reservation\_Charge_i \ ,$$

where $Reservation\_Charge_i$ is the charge applied for the reservation of resources for channel i. All of the resources reserved for the channel are available for immediate utilization by that channel.

The equation given above can also be written as

$$Total\_Charge_i = TOS_i \cdot t_i \cdot tod_i \ ,$$

where

- TOS_i is the type of service required by the client for channel i.
- t_i is the lifetime of channel i.
- tod_i is the period factor (time of day) associated with channel i.

TOS_i is chosen from a menu specifying the performance parameters of the service provided. There are n different *TOS* and TOS_i is the service type from 1..n requested by channel i. Each of the n *TOS* has a fixed charge per unit time associated with it, and these prices are made available to all clients. tod_i is the factor associated with peak periods. This factor is used to encourage clients with flexible needs to shift their workloads to periods where the *tod* factor is less. The tod_i values for each period are available to the client. The value of *tod* associated with channel i changes if the channel is present when a period ends and another begins, and the effects of such changes in the charge are calculated during accounting. t_i is the duration of the channel. $TOS_i \cdot t_i$ gives the charge due to reserving the resources for the service type given by TOS_i for time t_i . The factor tod_i influences this charge based on the periods during which the resources were reserved.

To be adequate, this pricing policy needs to meet all of the requirements given above. The policy satisfies requirement (1), since clients with higher performance service requests get charged a higher rate. The service provider can scale the value of TOS_i to recover costs and realize a reasonable profit. The reservation of resources guarantees that these resources are available to the clients when they demand them, but the resources may be utilized by other clients (usually the non real-time clients) when not in use by their "owners". If the client's request is accepted by the network, the reserved service is immediately provided.

Requirement (2) is met in that the TOS_i, and tod_i charges are fixed and available to the clients. Clients also know the pricing policy, and usually the lifetime of each channel (in some cases exactly, in some others only approximately), and can therefore determine the charges that their channels will incur, as well as verify the charges applied to those channels by the network.

Clients are encouraged to do efficient actions by the TOS_i and tod_i values. Clients needing less stringent levels of performance will choose a *TOS* with a correspondingly lower charge (i.e., a lower price is paid for a lower quality of service) and will have a reduced amount of resources available to them (as compared to clients with higher valued *TOS*). The use of tod_i will force flexible clients with the same *TOS* to shift loads off of the peak periods. If clients use the network inefficiently by reserving resources that they do not intend to use, they are penalized by having to pay the higher charges associated with that quality of service. This encourages

efficient use by the clients and thus satisfies requirement (3).

Requirement (4) is met, in that the policy can be implemented with minimal overhead. The TOS_i value is determined at the time of establishment. Timestamps are kept upon the establishment and termination of all channels. The t_i and tod_i values are determined using these timestamps. The charges can then be computed offline and submitted at a later date to the client.

The last requirement to be met is that of scalability and flexibility. The revenue (i.e., the total charge) should be related to the network utilization so that changes to the topology or the service menus can be easily incorporated into the current pricing scheme. The scalability aspect refers to the addition of new nodes, while the flexibility refers to the addition of new services. Our policy meets this requirement, as pricing is handled on a per-node basis, and the total charge is the sum of the charges incurred in the nodes along the route. To examine the effects of additional real-time services in the network, only a single node need be analyzed. Therefore, network analysis actually collapses to the analysis of a single node of the network. Obviously, a homogeneous network is assumed in the analysis. This collapse of the analysis to a single node is accomplished by deriving the *TOS* value directly from a network utilization function. In the network utilization function selected, the bandwidth, cpu, buffer space, and delay resources are parameters. The function operates on these parameters and generates suitable *TOS* values. This utilization function and its relationship to the policy are discussed in the next section.

3.2. The Utilization Function.

In this section a description of the network and traffic models that were used in our simulations will be presented. A discussion will follow of the functions used to calculate bandwidth, buffer space, cpu time, and delay resource reservations.

In our network model, the network consists of an interconnection of nodes. Nodes are connected to each other using links. Nodes can be hosts or switches. In each case the classification as a node is based on the fact that the entity in question provides bandwidth, buffer space, and CPU cycles as resources for channels.

The traffic model and the performance specifications we used consist of the parameters as given below. A complete description of this model is available in [6]. The traffic and performance parameters are the following:

- $X_{\min}$ is the minimum packet interarrival time on the channel.
- X_{ave} is the minimum value of the average packet interarrival time over an interval of duration I.
- $S_{\max}$ is the maximum packet size in bytes.
- $T_{\max}$ is the maximum service time in the node for the packets.
- D is the source-to-destination delay bound for the packets.
- Z is the minimum probability that the delay of a packet is smaller than the given bound D.
- $W_{\min}$ is the minimum packet loss probability due to the buffer overflow.

The last three of these parameters define the quality of service offered by the network. The *TOS* value assigned to each channel type is based on the resources reserved by the channel and on the channel's effect on the admission of other channels into the network. The resources reserved are directly related to the client requirements, as can be seen in the equations below. If

all nodes are identical, it is sufficient that these resources be determined for a single node for a complete analysis of the network.

$$\text{Bandwidth reserved} = \sum(1/X_{min}) \cdot S_{\max}$$

$$\text{Buffer space reserved} = \sum Buffers() \cdot S_{\max}$$

$$\text{CPU time reserved} = \sum(1/X_{min}) \cdot T_{\max}$$

$$\text{Delay resource reserved} = \sum(1/D) \cdot Z \cdot K(D,Z)$$

where:

- N_p is the number of nodes to be traversed by the channel.
- Buffers() is the buffer space function that calculates the number of buffers required by the channel at each node based on the values of client parameters.
- *K(D,Z)* is a heuristically determined delay function dependent on the parameters *D* and *Z*.

The bandwidth reserved in a node is determined by multiplying the maximum packet sending rate, which is the reciprocal of the minimum packet interarrival time, by the maximum packet size. The total bandwidth reserved is the summation across all the nodes in the path. If the nodes are identical, this summation is achieved by simply multiplying tge bandwidth reserved in a node by the total number of nodes in the path.

The total amount buffer space is determined by summing the result of the Buffers () function across all nodes multiplied by $S_{\max}$. A thorough description of the Buffers () function will be provided in Section 4.2 As explained above, this can be easily achieved by multiplying by the number of nodes in the path.

The *TOS* value is computed as follows:

$$\text{TOS} = \alpha \cdot \text{ Bandwidth reserved} + \beta \cdot \text{ Buffers reserved} + \gamma \cdot \text{ CPU time reserved} + \delta \cdot \text{ Delay}$$

The values of the coefficients α, β, γ, and δ are chosen by the network designers to reflect the relative scarcity of each of the resources. For instance, if bandwidth is the bottleneck resource, α must be made larger than the other constants in order to encourage clients to use the resource effectively.

The bandwidth, buffer space, and CPU time resources are easily defined and are directly related to the quality of service requirements. However, the delay resource cannot be directly related to that quality. The delay resource can be thought of as a quantification of the loss of potential for the network to admit other channels due to the admission of the channel being created. The admission of a channel into the network may reduce the network's ability to admit other channels due to the increased difficulty the node scheduler encounters in satisifying all the local deadlines. We therefore say that the new channel consumes some of the delay resource of the network. The delay resource is related to the delay (*D*) and the probability of no lateness (*Z*) requirements of the channel. We make the delay resource inversely proportional to the delay parameter, since the shorter the delay required by the channel, the greater the restrictions placed

on the scheduler, and directly proportional to the probability parameter since the greater this probability, the greater the restrictions placed on the scheduler. The proportionality constant *K(D,Z)* is explained in the next section.

3.2.1. The delay function *K(D,Z)*.

The purpose of the delay function , *K(D,Z)*, is to bias the values of the *D* and *Z* parameters of all the different channel types so that the amount of the delay resource consumed by each channel of a specific type correctly reflects the loss of potential to admit other channels into the network.

Before an explanation of *K(D,Z)* can be provided, a few terms need to be defined. The *request space* is an *n*-dimensional region containing all possible combinations of the *n* types of channels which can be requested by clients. In the example to be provided in section 4, there are three channel types, thus the *request space* is a 3-dimensional space. The values of each of the 3 elements of a vector in the *request space* are non-negative, and indicate the number of channels requested of that type. The *admissions space* is a subset of the *request space*, and contains those vectors that can be admitted into the network by the admission control scheme. The admission control scheme is based on the resources consumed by each channel and the total resources available in the network at the time of the request. These vectors admitted into the network are called *admissible vectors*. For these vectors to be in the *admissions space*, all possible permutations of the channels, representing all possible orderings in which the request arrive, must be admissible into the network. An example of this situation is that of the *admission vector* (2, 1, 1), where this vector indicates that 2 Type I, 1 Type II, and 1 Type III channels can be admitted. If (2, 1, 1) is admissible, all possible permutations are admissible (eg. (I,I,II,III),(I,II,I,III) etc).

The *safe region* is a subspace of the *admissions space*. It is a region bounded by the axes and extending a finite distance from them. This region is determined by the network designer. The extent of this region is such that as many *admissible vectors* as possible will lie within it. In situations where the *request space* is large (i.e., a large number of channel types and a large number of channels of each type can supported) , it is computationally inefficient to determine the *safe region* which encompasses all of the admissible channels. The inefficiency is due to the large number of simulations that would be needed in order to determine all the possible *admissible vectors* . In this situation, a computationally efficient method must be employed to determine the *safe region* that encompasses as many *admissible vectors* as possible. The equation of the boundary of this safe region will provide us with the delay function *K(D,Z)*, as this equation depicts the relationships between the channel types in terms of the effect of admitting a channel of one type versus another. In a network offering three types of services, the equation of the boundary of the *safe region* is of the form $a\,N_1 + b\,N_2 + c\,N_3 = d$ where a , b, c , d are the coefficients of the equation and the N_i's variables corresponding to the number of channels of type *i* supported by the network. In this equation the values of a , b, and c provide the ratios which indicate the loss of ability of the network to accept another channel upon acceptance of this channel. As an example, assume a , b , c are 1 , 2 , and 4 respectively. Therefore the acceptance of one channel of type III (i.e., the value of N_3 is increased by 1) means that the network has lost the ability to accept 4 channels of type I or 2 channels of type II or 2 channels of type I and 1 channel of type II. This situation occurs as the boundary equation must be satisfied. These coefficients allow us to bias different channel types so that a fair reflection of the preference for one channel over the other is given in the price. An example of this region is given in Figure 1

below. In Figure 1, a network containing two channel types is considered. The maximum number of channels of type I is 9 and of type II is 14. As 2 channel types are considered, the *request space* is 2-dimensional with the maximum points on the axes being the maximum number of Type I and Type II channels, ie., (0,9) and (14,0), respectively. As the number of simulations need to determine the *admissions space* is small, the *safe region* is the same as the *admissions space*. The boundary of the *safe region* is identified as a dotted line.

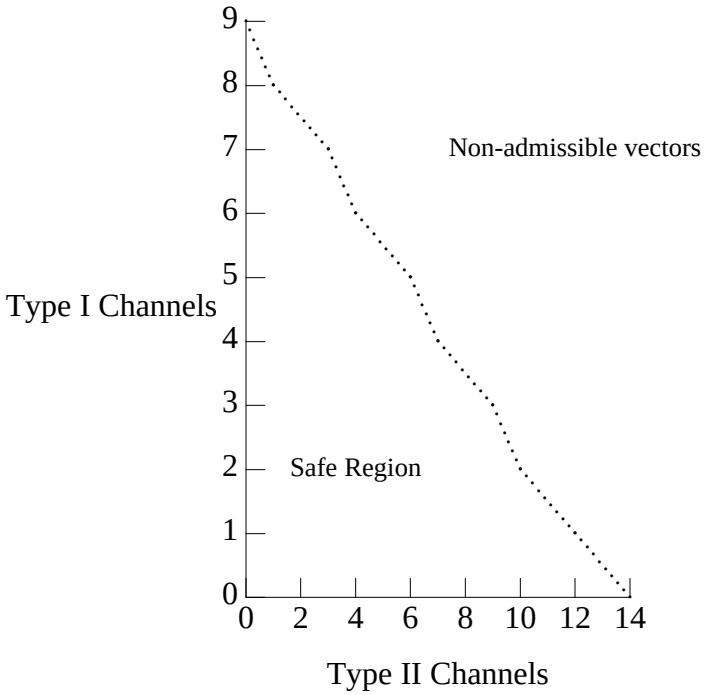

Figure 1

3.2.2. Estimating the safe region.

For the analysis of networks with a large number of channel types and the ability to support a significant number of channels of each type, it is not feasible to determine the exact boundary of the *safe region* because of the amount of simulations (i.e., computational resources) needed. Also, the equation of the boundary of the *safe region* is used to determine the relative weight of each channel type and an accurate estimate of this boundary would provide all the needed information. The accuracy of the estimate is dependent on the number of simulations performed.

The algorithm used to obtain an estimate of the boundary utilizes an important property of the *request space*. If a vector, $(n_1, n_2, \cdots, n_t)$, in the *request space* is admissible, then all vectors of the form $(\hat{n}_1, \hat{n}_1, \cdots, \hat{n}_t)$ where $n_k \geq \hat{n}_k$ for all $k = 1,...,t$, are *admissible vectors*. Vector $(n_1, n_2, .., n_t)$ is said to *dominate* vector $(\hat{n}_1, \hat{n}_2, .., \hat{n}_t)$. This property is obvious in that vector $(\hat{n}_1, \hat{n}_2, .., \hat{n}_t)$ is the *admissible vector* $(n_1, n_2, .., n_t)$ with the appropriate channels removed. Hence vector $(\hat{n}_1, \hat{n}_2, .., \hat{n}_t)$ will always be admissible. A divide and conquer strategy is used to exploit this *dominance* property. The *request space* is bounded along each dimension by the maximum number of channels of that type that are admissible. This bounded portion of the *request space*

is then divided into 2^t sub-regions where t is the number of channel types available. These initial sub-regions, referred to as *level*1 sub-regions, may or may not contain the boundary of the *safe region*. To determine if a sub-region contains the boundary of the *safe region* , two vectors need to be tested for admissibility. These vectors are the vectors in the sub-region which are nearest to the origin and furthest from the origin. A sub-region contains the boundary if the vector nearest to the origin is admissible and the vector furthest from the origin is not admissible. Any other combination of results indicates that the boundary is not in that sub-region and that sub-region is then discarded. Therefore each sub-region inspected needs two simulations to verify the two vectors. Each vector need be simulated only once, as the admission control scheme uses the worst possible permutation of those channels represented by the vector to determine admissibility. Upon verifying that a level 1 sub-region contains the boundary of the *safe region*, this sub-region can be further subdivided into 2^t level 2 sub-regions and the process repeated until the threshold set by the parameter K is reached.

This parameter is set by the network designer and reflects the accuracy of the estimate needed. K is used to determine the minimum size of the sub-regions. A level n sub-region, which contains the boundary, cannot be further subdivided if its area is less than $\prod_{i=1}^{i=t} K \cdot N_i$ where N_i is the maximum possible number of channels of type i that can be supported by the network. If the level n sub-region, containing the boundary, cannot be subdivided further, the vector nearest to the origin (ie., the admissible vector) is kept and testing of that level n sub-region ceases. The value of K is the fraction of error that can be tolerated in estimating the boundary. For a channel type where the maximum possible number of channels in 30 a K of 0.1 indicates that the maximum tolerance is 10 % of the 30 channels (ie., 3 channels), hence the vector estimated as being on the boundary will be at most 3 channels away along the dimension corresponding to that channel type.

This divide and conquer process is repeated for each level 1 sub-region until all level 1 sub-regions have been inspected completely. The vectors resulting from these inspections are then analyzed, and all dominant vectors are retained. The equation of the boundary is then determined by doing a multiple least squares regression on the retained data. This algorithm is summarized below:

- Determine N_i (maximum number of channels of type i) for each channel type i , $i = 1,...,t$.
- Set the level of tolerance K needed. K is a fraction such that $\lceil K \cdot N_i \rceil$ channels is the maximum possible distance of the estimated safe region from the actual safe region along that dimension. The larger the value of K the less the number of calculations needed.
- With N_i (maximum number of channels of type i) for each channel type i , $i = 1,...,t$. defined, the level 0 sub-regions is defined by the vectors $(N_1, N_2, \cdots, N_t)$ and (0, 0, ... , 0).

The following recursive algorithm is then employed.

- Divide each level n sub-region into 2^t level $n+1$ sub-regions under the constraint of the threshold area as defined by K. These level $n+1$ sub-regions are defined by two vectors. The vectors are always those nearest to the origin and furthest from the origin, respectively. Obviously, we must have $K < 0.5$ or we cannot initially subdivide the sub-regions.
- Do simulations on the vectors nearest to and furthest from the origin. Use table below to determine the vectors to select.

Nearest	Furthest	> or <= A_K	Action
S	S	>	Discard sub-region.
S	S	< =	Discard sub-region.
S	F	>	Repeat 4 .
S	F	Approx=	Keep vector nearest to origin.
F	S	>	Not possible.
F	S	<=	Not possible.
F	F	>	Discard sub-region.
F	F	< =	Discard sub-region.

- Collect all appropriate vectors at threshold points and select dominant vectors. The threshold points are points that are nearest to the origin whose sub-region's area is equal to the threshold area.

The maximum overhead incurred by this algorithm is the product of the maximum number of existing lowest level sub-regions by the number of simulations done on each sub-region. The maximum number of lowest level sub-regions possible is the total area of the *request space* divided by the area of a lowest level sub-region (i.e., $\prod_{i=1}^{i=t} N_i / \prod_{i=1}^{i=t} K \cdot N_i$ where N_i is the maximum number of channels of type i that can be supported by the network). The maximum number of simulations to be run is the maximum number of lowest level sub-regions multiplied by 2. Therefore, the maximum overhead is proportional to $2 \cdot (1/K^t)$, which is independent of the maximum number of channels of each type supported. In actual use the overhead is dependent on the shape of the boundary; simulations using this algorithm show that at most the overhead incurred was only 9 % of the maximum overhead. The maximum possible error incurred is always less than the length of the diagonal of the lowest level sub-region. This value is dependent on the values of K, and N_i.

4. An Example and its Analysis.

In this section the proposed scheme will be applied to a simple homogeneous network and the values of the *TOS* constants and permutations will be determined for each of the channels. The constants that need to be determined can be classified into two categories: constants that need to be determined computationally, and parameters that must be provided by the network designers. The computationally determined constants are generally those associated with the resources reserved by a channel of a specific type, and are shown in Table I. These are determined by arithmetic and simulation operations. The parameters provided by the network designers are those based on the revenue needed by the network, and are shown in Table II. These parameters are usually obtained by considering the overall system resources, as well as the economic and political climate. The current supply and demand situation for the services being offered and the FCC restriction on permissible charges for these services are examples of the economic and political considerations. Each of the computationally determined constants are discussed in their respective subsections below. The parameters will be discussed in the final subsection. The network topology and specifications of the channel types will first be provided.

Table I - Constants to be computationally determined:

B - Bandwidth reserved.
BU - Buffer space reserved .
C - CPU time reserved
K(D,Z) - Delay function.
DE - Delay resource reserved

Table II - Parameters to be provided by network designers:

α - Weighing constant for bandwidth resources reserved.
β - Weighing constant for buffer space resources reserved.
γ - Weighing constant for CPU time resources reserved.
δ - Weighing constant for delay resources reserved.
Tod- Time of day value.

4.1. The Network.

The network topology considered here is that of a homogeneous network where all the links and switches have identical resources. Each switch and its associated link form a logical node, and the bandwidth, buffer space, CPU, and delay resources associated with the switch and the link are ascribed to this node. Thus, the analysis of this network collapses to the analysis to a single logical node, as each channel consumes the same amount of resources in each logical node. Any arbitrary homogeneous topology can be converted into this logical node form; an analysis of a single node will provide all the pricing information needed for this policy. The total resources reserved by the channel are given by the sum of the resources reserved in each node. This single node will be simulated, to determine the value of *K(D,Z)*, using a *CSIM* based simulator. The 3 channel types that have been selected attempt to capture the relative resource ratios and the flavors of the variety of multimedia traffic that will be carried on future real-time networks. Channel I is a high bandwidth/low delay deterministic channel with respect to channel III, which can be considered a low bandwidth/high delay deterministic channel. Channel II is a medium bandwidth/medium delay statistical channel. Table III provides the specifications for the channel types.

Table III - Channel types.

Channel Type	t (time units)	$x_{\min}$ (time units)	x_{ave} (time units)	D (time units)	Z (prob)
I	1	10	20	100	1.0
II	4	20	40	150	0.5
III	4	40	80	200	1.0

4.2. Resources Reserved.

This section presents the resources reserved by the channel in each logical node. The total resources reserved are given by the summation of the resources in each node.

The bandwidth reserved for Channels I, II, and II at the node are determined by the bandwidth reservation formulae presented in Section 3.2:

$$\text{Bandwidth reserved} = (1/X_{\min})\cdot S_{\max}\cdot N_p$$

The reciprocal of the minimum packet interarrival time, $X_{\min}$, is the maximum rate at which packets are given to the network. $S_{\max}$ is the maximum packet size. The product of the two give the maximum bandwidth per node needed by a channel of this type. As a paradigm of our scheme is to always consider the worst-case scenario the bandwidth reserved is the maximum needed by the channel. For the sake of simplicity we let $S_{\max}$ be 1.

The buffer space reserved at each processing node for each channel type is determined by the buffer space reservation formulae presented in Section 3.2 (as the nodes are homogeneous the summation becomes a multiplication by the number of nodes N_p) :

$$\text{Buffer space reserved} = \mathit{Buffers}()\cdot S_{\max}\cdot N_p$$

The Buffers() functions are given below for both deterministic and statistical channels, respectively:

$$\text{Buffers}() = S_{\max}\lceil [d_i/X_{\min}]W_{\min}\rceil$$

$$\text{Buffers}() = S_{\max}\lceil [d_i + I\cdot P_{do}\cdot(\sum_{j=1}^{j=n}(t_{j,n}/X_{min,j}) - 1]/X_{\min})W_{\min}\rceil$$

where

- d_i is the local delay bound in the current node.
- P_{do} is the overflow probability of a node.

A complete discussion of these buffer space equations can be found in [5]. The Buffers () function provides the number of buffers needed per node using as parameters the channel characteristics of each channel type. This value, multiplied by the number of nodes in the path, gives the total number of buffers needed. The buffer space in bits is determined by multiplying each result by $S_{\max}$, which in this example is 1.

The CPU time reserved at processing nodes for each channel type is determined by the following formula:

$$\text{CPU time reserved} = (1/X_{\min})\cdot T_{\max}\cdot N_p$$

The CPU time reserved is the product of $T_{\max}$ and the number of nodes that the channel traverses. $T_{\max}$ is the amount of CPU time needed to process a packet corresponding to a channel of a specific type. CPU time is customarily measured by CPU cycles, where the time per cycle is a known constant; hence the time is obtained by multiplying the cycles by the time per cycle.

The delay function *K(D,Z)* provides a constant, dependent on all of the channel specifications, that is used to quantify the effect of the delay requirements of this channel on the admission of any additional channels into the network. This constant biases the delay requirement of the channel, *D*, and the probability that the packets arrive before the required delay, *Z*, so that the delay resource reserved represents the loss of potential of the network to accept additional channels after this channel has been accepted. The boundary of the safe region was

determined as described in Section 3.2.2. The estimated equation of the boundary was $1.06 \cdot N_1 + 0.97 \cdot N_2 + 0.98 \cdot N_3 = 5.1$ The coefficients in this equation were used to provide the appropriate delay constant. The equation of the actual boundary, determined by simulating all vectors in the *request space* is $1.1 \cdot N_1 + 1.0 \cdot N_2 + 1.0 \cdot N_3 = 5.6$, and 656 simulations out of a maximum of 16000 ($K=.05$) were done. The values of the coefficients indicate that the admission of 1 channel of type 1 would reduce the ability to accept 1 channel of type 2 or 3. Therefore, the value of $K(D,Z)$ would be that of D/Z for each channel respectively.

The delay resource reserved for each channel type is determined by the following formula:

$$\text{Delay resource reserved} = (1/D) \cdot K(D,Z) \cdot Z \cdot N_p$$

The results of this pricing analysis are given in Table IV below.

Table IV - Resources Reserved.

Channel Type	Bandwidth reserved	Buffer space reserved	CPU Time reserved	$K(D,Z)$	Delay Resource reserved
I	0.1	3	1	0.01	1
II	0.05	3	4	0.003	1
III	0.025	2	4	0.005	1

4.3. Network Parameters.

The network parameters that the designers are required to determine are, as noted in Table II:

α - Weighing constant for bandwidth reserved.
β - Weighing constant for buffer space reserved.
γ - Weighing constant for CPU time reserved.
δ - Weighing constant for delay reserved.
Tod - Time of day value.

As mentioned previously, these parameters depend on a variety of conditions having to do with facility resource availability, economic climate and political concerns. The complex scenarios encountered when dealing with political and economic concerns will be ignored in this paper, thereby simplifying the determination of the network parameters. It should be kept in mind that the values associated with each of these parameters are determined from very complex functions, which take into consideration a wide variety of inputs. The simplest of solutions were considered for this example. The resources, bandwidth, buffer space, CPU time, and delay, were considered equally scarce and hence only relative weighting considerations were taken into account. Relative weighting considerations are those which seek to balance the effect of each resource equally. For example, bandwidth can be measured in bits per second, and CPU time in milliseconds; the addition of these two quantities (assuming resource-wise that they are weighted equally) would result in the CPU time having a negligible effect in the equation. Therefore the constants, in this case α and γ, have to be chosen so that they will accomplish this balance. This can be accomplished by using percentages. The values of α and γ should be the reciprocal of the maximum possible bandwidth and CPU time available to channels. The same can be said for the

value of β, which should be the reciprocal of the maximum buffer space available to a single channel. δ is different in that it is already normalized by virtue of its definition. This relative weighting consideration is a minimum consideration. The other economic and political considerations can be factored in with this balance consideration. In this example, the maximum bandwidth, buffer space, and CPU time available at each node is 100,000 bandwidth units, 100 buffers, and 1000 CPU time units. For the purpose of simulation time units are used, and these units can be scaled to reflect a 'real-world' situation. For example, one bandwidth unit can be 100 bits per second, making the maximum bandwidth of the link available to that node be 10 Mbits/sec. The same can be done for buffer and CPU time units. In the example α is 0.000001, β is 0.01, gamma is 0.001 and δ is 1.

The values associated with tod_i are determined purely by economic considerations, for example the costs to be recovered and the profits to be generated. These considerations are subjects for further research and will not be addressed in this paper.

5. Conclusion.

This report presented a pricing policy for real-time channels. The requirements of a useful pricing scheme were stated, and a review of the deficiencies of conventional schemes was provided. The conventional schemes reviewed were Flat Rate, Peak Load and Priority Based pricing scheme. The proposed pricing policy was then presented. This policy is based on the resources reserved by a channel in a single logical node of a homogeneous network. This policy is unique in that delay is characterized as a resource, in addition to bandwidth, buffer space, and CPU time. Also, the analysis of a complex homogeneous network collapses to the analysis of a single logical node. The validity of the scheme was then critiqued using the requirements presented previously. A thorough description of the resource reservation formulas was then provided, followed by that of a simple example. The overhead associated with this policy is minimal in that the TOS_i value can be predetermined and the values of tod_i, and t_i can be easily determined during run-time. The actual measurement of overhead and the economic and logistic aspect associated with the selection of network parameters were noted as areas of future work and are currently under investigation.

REFERENCES

[1] P.O.Steiner, "Peak Loads and Efficient Pricing", Quarterly Journal Of Economics (1957), pp.585-610.

[2] H. Chao, R.Wilson, "Priority Service: Pricing, Investment, and Market Organization", The American Economic Review(December 1987), pp 899-916.

[3] R. Cocchi, D. Estrin, S. Shenker, L. Zhang, "A Study of Priority Pricing in Multiple Service Class Networks", Proceedings of ACM SIGCOMM 1991 Conference, pp 123-130.

[4] D. Ferrari, D.Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", IEEE Journal on Selected Areas in Communication (April 1990), pp 368-379.

[5] D.Ferrari, D.Verma, "Buffer Space Allocation for Real-Time Channels in a Packet-Switching Network", Technical Report TR-09-022, International Computer Science Institute, Berkeley, June 1990.

[6] D.Ferrari, "Client Requirements for Real-Time Communication", Technical Report TR-90-007, International Computer Science Institute, Berkeley, March 1990.