A Delay Pricing Scheme for Real-Time Delivery in Deadline-Based Networks

A thesis presented

by

Xiao Huan Liu

to

The Department of Computer Science

in partial fulfillment of the requirements

for the degree of

Master of Science

in the subject of

Computer Science

The University of Manitoba

Winnipeg, Manitoba

April 2006

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a MSc thesis entitled:

A Delay Pricing Scheme for Real-Time Delivery in Deadline-Based Networks

submitted by: *Xiao Huan Liu*

in partial fulfillment of the requirements for the degree of: *MSc*

Dr. Yanni Ellen Liu, Advisor
Computer Science

Dr. Ruppa Thulasiram
Computer Science

Dr. Ekram Hossain
Electrical and Computer Engineering

Date of Oral Examination: *February 28, 2006*

The student has satisfactorily completed and passed the MSc Oral Examination.

Dr. Yanni Ellen Liu, Advisor
Computer Science

Dr. A. Neil Arnason
Chair of MSc Oral

Dr. Ruppa Thulasiram
Computer Science

Dr. Ekram Hossain
Electrical and Computer Engineering

(The signature of the Chair does not necessarily signify that the Chair has read the complete thesis.)

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

A Delay Pricing Scheme for Real-Time Delivery in Deadline-Based Networks

by

Xiao Huan Liu

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

of

Master of Science

Thesis advisor

Dr. Yanni Ellen Liu

Author

Xiao Huan Liu

A Delay Pricing Scheme for Real-Time Delivery in Deadline-Based Networks

Abstract

There has been an increasing demand to transport real-time data over packet-switched computer networks. Deadline-based network resource management is a novel approach to supporting real-time data transfer in computer networks. In a deadline-based network, each document to be transmitted is associated with a deadline specified by the document sender. These document deadlines are mapped to deadlines at the network layer, which are carried by packets and used by routers for channel scheduling; packets with more urgent deadlines are serviced first.

In deadline-based networks, the delay performance observed by real-time data largely depends on the traffic deadline and the level of load along the data path. To prevent greedy users from gaining an advantage by specifying arbitrarily urgent deadlines and to aid in network load control, I introduce a novel delay pricing and charging scheme in deadline-based networks to support real-time data delivery. In my pricing scheme, the concept of channel delay price is introduced, and it is decided using a market-based approach based on the traffic urgency level and the network load level at each channel. A user's charge is determined based on the amount of his/her traffic, the channel delay prices, and the delay performance that his/her traffic

receives. My pricing and charging scheme can provide differential charges to users receiving different delay performance, thus preventing greedy users from gaining an advantage by specifying arbitrarily urgent deadlines. In addition, price-sensitive users may adapt his/her traffic in response to price. My pricing and charging scheme easily enable user adaptations, which in turn may significantly improve the performance of real-time delivery in deadline-based networks.

Contents

List of Figures

List of Tables

Acknowledgments

Firstly, I am grateful for the guidance, support and financial assistance from my supervisor: Dr. Yanni Ellen Liu. This thesis would not have been completed without her help in the entire research process. She has given me a lot advice, encouragement, and effort in helping me with this research. I am very thankful to have done this research with her. She is an active researcher in Computer Networks with a very nice personality, and I have learned a lot from her. This research was supported by the University of Manitoba, Canada, under the University Research Grant Program, and by the Natural Sciences and Engineering Research Council of Canada. I would like to thank them for their financial assistance.

In addition, Dr. Thulasiram, one committee member for my thesis, spent a lot of time reading my thesis. He revised my thesis very earnestly, and gave me helpful advice on my thesis and future work. Dr. Hossain, another committee member for my thesis, also gave me helpful comments, and pointed out some problems I had not considered before. Their comments were very helpful on improving the quality of the final version of my thesis.

I would also like to thank Dr. Neil Arnason and the committee members for providing useful comments and suggestions on my initial thesis proposal.

Special thanks also go to my mother, my sister and my husband for their constant support and encouragement along the way.

This thesis is dedicated to my parents, my sister, my husband, and my daughter.

Chapter 1

Introduction

1.1 Pricing in Computer Networks

Current and future computer networks are expected to accommodate a wide variety of network applications. These applications may have different levels of Quality of Service (QoS) requirements on the delivery service of the underlying network. These QoS requirements may be in terms of performance metrics such as delivery latency, transmission bandwidth, packet loss ratio, and packet jitter. An important class of such applications is real-time applications that require timely delivery of real-time data. Examples of real-time data include stock quote updates, bids in an online auction, state update messages in distributed multi-player interactive games, audio and video data in video conference, and voice data in IP telephony.

To adequately support these applications, researchers have been designing computer networks that can provide multiple levels of QoS support to end users, and also the corresponding network pricing schemes. When multiple services are available at

different prices, it is possible for users to select specific services, signal the network to service them according to the requested service quality, and trigger accounting and billing records generation. Comparing with a single-service architecture and a flat-rate pricing model, by adopting a multi-service paradigm and a differential pricing framework, service providers have the potential to offer necessary incentives for users to choose the service that best matches their needs and capacity, thereby creating a fair pricing environment, improving economic and network efficiency, and maximizing revenue and/or social welfare.

An example of multi-service network is the Internet Differentiated Services (DiffServ) framework [4]. DiffServ provides multiple QoS classes over IP networks. The service guarantee provided to the packets of a QoS class depends on the amount of resources allocated to the class, the current load of the class, and, in case of congestion, the drop precedence within the class. The end-result is that multiple levels of services are provided to users[1] that consume multiple amount of resources. More often than not, a fair pricing scheme in such a network is based on the level of service received and the amount of resources consumed – the higher the level of service, i.e., the higher the amount of resources consumed, the higher the price.

On the other hand, prices that users are willing to pay also affect the level of service they receive. In computer networks, multiple levels of service to end users are implemented by way of resource management strategies such as resource allocation and reservation, channel scheduling, and buffer management. The price that a user is willing to pay can be used by resource allocation mechanisms to provide different levels

[1]Here, the users may be network applications or sequences of application data that are transmitted.

of QoS. For example, Fulp et al. [14, 44] used pricing as a method to allocate network bandwidth; Mackie-Mason et al. [29] used pricing to affect the drop precedence and the priority in channel scheduling. Therefore, pricing has good interactions with resource management functions in QoS-enabled networks.

To come up with a proper pricing scheme in computer networks, in literature, microeconomic principles have been applied. In microeconomics, producers own the resources, and consumers purchase the resources from the producers. Prices reflect the relationship between demand and supply so that prices can affect the behavior of consumers and producers. In computer networks, the network service providers are like the producers, and the users (or applications) are like the consumers. The network resources can be priced to affect users' behavior. When the network load is heavy, prices can be raised to discourage users' usage; when the load returns to manageable levels, prices can be lowered to encourage users' usage. Thus when taking into account current network load conditions in determining price, a pricing scheme can also be used to control congestion and to improve the performance of a network.

Aside from the issues of fairness and congestion control, there are other issues in determining prices. For example, proper charging schemes are needed to cover the cost of network construction, maintenance, and upgrade. Another issue arises from a social welfare point of view. If the price is too high, poor users may be starved from network services, whereas if the price is too low, the network may be overloaded; a price may be set appropriately to achieve a social welfare criterion. In my research, however, I am not concerned with these other issues, instead, I focus on the two issues of fair charge and load control. In particular, I address these two issues in

deadline-based networks.

1.2 Motivation

Deadline-based network resource management [23, 24, 25, 45] is a novel approach that was developed to support real-time document delivery applications over packet-switched networks. In this framework, the notion of application data unit (ADU) is used. An ADU may correspond to a file or a frame in audio or video transport. Each real-time ADU is associated with a delivery deadline, which is provided by the sending application. It represents the time at which the ADU should be delivered at the receiver. The ADU deadlines are mapped to packet deadlines at the network layer, which are carried by packets and used by routers for channel scheduling. Deadline-based channel scheduling algorithms are employed inside network routers; packets with more urgent deadlines are transmitted first.

Compared to the best-effort service, which is provided by the Internet today, this novel technique may provide better performance in delivering real-time data, because routers schedule packets according to the urgency of packet deadlines. It has been shown that deadline-based scheduling achieves superior performance to FCFS (First-Come First-Served), which is the scheduling discipline used on the current Internet, with respect to the percentage of ADUs that are delivered on time [23, 24, 45].

In deadline-based scheduling, the delay performance experienced by real-time packets is largely affected by the deadline information that they carry, which depends on the ADU deadlines provided by sending applications. If one is free to specify the deadline for each ADU, a sender may try to gain an advantage by using arbitrarily

tight deadlines. This raises the issue of fairness as seen by network users. A pricing scheme should be provided to fairly charge users according to the delay performance that they receive.

In addition to deadline urgency, the delay performance in deadline-based networks is also affected by the load conditions along the ADU path. When the load is light, the delay performance is good. When the load is heavy, congestion may occur; queues at bottleneck links may grow significantly, the delay performance deteriorates. If a pricing scheme takes into account current network load conditions in determining price, it may also be used to alleviate congestion and to improve the network performance.

So far, the pricing issue in deadline-based networks has not been studied. In this research, I aim to develop a pricing and charging mechanism for deadline-based networks that can be used to achieve the following goals: (1) the network charges users fairly according to the level of service received, in another word, the network offers the incentive for users to submit requests that best match their needs, (2) the delay performance of the network can be improved and the network load can be controlled.

1.3 Research Outline

To prevent greedy users from specifying arbitrarily urgent deadlines, and to control the level of load in order to maintain good delay performance, in this research I develop a novel delay pricing and charging scheme that takes into account both deadline urgency and network load conditions. Different from other researches in which bandwidth or buffer occupancy are priced, the novelty of this work is that the unit

delay on each channel is priced, and the user is charged in term of the delay performance received. At each network channel, a *delay price* is periodically computed based on the traffic deadline urgency and the traffic load so that (1) the higher the level of deadline urgency, the higher the price, and (2) the heavier the network load, the higher the price. Each passing-by packet is charged based on the delay it experiences at this channel and the current channel price: the lower the delay it experiences, the higher the charge; the higher the current channel price, the higher the charge. This charge is carried by the packet and is accumulated along the entire packet path. Depending on the size of network maximum transfer unit (MTU), an ADU may be fragmented into multiple packets for transmission. If an ADU is delivered to the receiver on-time, the ADU is charged based on the packet charges of all its packets. In determining the channel price, a market-based approach from the field of microeconomics is taken in my scheme, in which the price reflects the relationship between the demand and supply. At each channel, the demand is derived from the deadline information carried by real-time packets, and the supply reflects the amount of time that is needed to service these packets. Such a delay pricing scheme encourages users to submit deadline requirements that best match their needs and capacity. Given limited user budget for network transmissions, such a delay pricing scheme may aid in the process of load control so that performance degradation due to congestion can be alleviated. In this thesis, I present my pricing and charging scheme, and evaluate its performance by simulation. A simulation model written in C++ is developed for this purpose.

1.4 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, background knowledge related to this research is reviewed. In Chapter 3, the impact of deadline urgency and network load on the delay performance experienced by real-time data in a deadline-based network is first studied. This motivates the design of my delay pricing and charging scheme. In Chapter 4, the pricing and charging scheme developed in this research is presented and its performance is evaluated. In chapter 5, the concept of user adaptation is introduced. I show that with user adaptation, the network performance can be much improved. Finally, in Chapter 6, the contribution of this research is summarized and a list of topics for future research is provided.

Chapter 2

Background

This research aims at developing a pricing and charging scheme in deadline-based networks. In this section, I give background information on deadline-based networks and pricing in computer networks respectively.

2.1 Deadline-Based Network Resource Management

In classical Internet, routers employ first-come, first-served (FCFS) algorithm when scheduling packets at each router channel. With FCFS, the arrival time establishes the priority of a packet. For certain type of real-time applications, FCFS may not be appropriate. For example, a real-time video transmission to a remote expert during a heart operation is much more important than an e-mail transmission. Therefore, it might be desirable if router channels first serve the more time-critical traffic, provided that traffic urgency information is available.

In deadline-based networks [45], each ADU is associated with a delivery deadline,

which represents the time by which the ADU should be received by the receiver. At the network layer, an ADU may be subjected to packet fragmentation. Each packet of an ADU will carry a deadline that is derived from the ADU deadline, and routers on the path schedule the packet based on its deadline.

It has been found that a deadline-based algorithm, called T/H, performs well in terms of the fraction of ADUs that are delivered on time [23, 45]. T is the time left, which is calculated at packet arrival by subtracting the current time from the deadline of the packet. H is the number of hops remaining. For each arriving packet, the value of T/H is calculated before the packet is entered into the queue, and packets in queue are ordered by ascending values of T/H. When the channel becomes idle, the packet in queue that has the lowest T/H value is serviced next. The T/H value represents the urgency of a packet, thus in T/H, the lower the packet T/H value, the more urgent the packet, the higher the priority it has in channel scheduling.

The performance in deadline-based networks has been previously studied [45], but the pricing issue in deadline-based networks has not been studied before. I address this issue in this thesis.

2.2 Pricing in Computer Networks

In this subsection, I introduce the price theory from microeconomics, raise some questions in applying that theory to computer networks, and present some evaluation criteria for pricing schemes in computer networks. I will use some of these criteria to evaluate my developed scheme.

In microeconomics, each competitive market has consumers, producers, and de-

sirable resources (goods) that are in scarce supply and high demand. The producers own the scarce resources, and sell them to the consumers to maximize revenue. Each consumer possesses finite amount of wealth and acts independently (selfishly) to purchase resources to maximize his/her utility (satisfaction) [44]. Price is the exchange rate of a resource. Price reflects the relationship between supply and demand. Price can be determined in many ways. When demand is greater than the supply, the price increases; when demand is lower than supply, the price drops. Therefore, pricing may constitute an effective way to control the demand for resources.

The concept of competitive market in economics can be applied to the field of computer networks. In computer networks, network service providers are like the producers, owing the network resources such as channel bandwidth. Network users are like the consumers who require the network resources, and users' resource usages are charged. Thus, the network resource can be priced based on the users' demand for the network resource and the resource capacity of the network. In reality, users are price-sensitive, so pricing can affect users' behavior, and users' behavior affect the demand; furthermore, users' behavior can, in turn, affect the network performance. Therefore, pricing in computer networks may constitute an effective way to control the users' behavior.

A number of economical, social, and technical problems arise when designing pricing schemes for computer networks. As stated above, price reflects the relation between supply and demand. If the bandwidth is considered as a resource, what is the demand function for bandwidth consumption? How does the demand change over time? What pricing structures shall be used? Additionally, how can the charge for an

individual user be computed? Should it be based on the number of packets serviced or some other measures? How can a pricing scheme be incorporated into existing networking technologies?

Falkner et al. [12] gave the following evaluation criteria for network pricing schemes:

1. Compliance with existing technologies: pricing schemes that are more compatible with existing technologies are easier to implement.

2. Measurement requirement for billing and accounting: this requirement directly contributes to the implementation complexity of a pricing scheme.

3. Support for congestion control or traffic management: it indicates whether a pricing scheme can give price-sensitive users the incentive to send more traffic during light load and less traffic when the load is heavy.

4. Provision of QoS guarantees to individual users: it indicates whether a scheme is capable of providing QoS guarantees to individual users.

5. Degree of network efficiency: it shows the expected level of channel utilization when employing a given pricing scheme. High utilization is commonly preferred.

6. Degree of economic efficiency: it reflects the degree to which a pricing scheme emphasizes the overall user benefit, rather than individual users' benefit.

7. Impact on social fairness: it indicates whether or not a pricing scheme starves the poor users.

8. Pricing time frame: it indicates what time frames a pricing scheme supports. Because congestion often occurs in short time frames, a pricing scheme support-

ing congestion control should support short time frames so that the price can be changed in response to the varying network load.

I will refer to some of above criteria when evaluating my pricing scheme.

2.3 Related Work

In this section, I review the literature of pricing in computer networks. I classify existing network pricing schemes into flat pricing, resource-based pricing, and expected resource consumption pricing.

2.3.1 Flat Pricing

Under flat pricing, users are charged a fixed amount per time unit (e.g., a month) [2, 13]. Currently, flat pricing is widely used for Internet services. Flat pricing is very simple to implement and account. Because it is a static scheme in which prices are independent of network load and usage, flat price is inadequate to control the users' behavior, therefore, it cannot prevent users' greedy behavior which may cause network congestion and degrade network performance. As a result, flat pricing is not suitable for congestion control and traffic management. It cannot facilitate multiple levels of QoS to end users either.

2.3.2 Resource-Based Pricing

In resource-based pricing, different amount of resources are allocated to provide multiple levels of QoS support to network users; charges are determined based on the

price of the resource and the amount of resource that users consume. Resource price may be static or may be dynamic. Static price is determined off-line through long-time measurement and planning. Dynamic price is determined online and reflects the level of load or congestion. In resource-based pricing, price can affect the network resource and traffic management.

There are many resource-based pricing schemes in literature. Based on their major objectives and mechanisms, I classify them into four categories: Paris-Metro pricing, differential service pricing, fair bandwidth allocation, and user adaptation.

2.3.2.1 Paris-Metro Pricing

Paris-Metro pricing (PMP) [36], a slight variation of flat pricing, partitions a physical network into a set of logical subnetworks. Different logical subnetworks can provide different services with different fixed-prices. Prices and capacities of the subnetworks stay constant for extended time periods and prices are set based on customer surveys, user complaints, and the traffic pattern variations. The last one allows for time-of-day price variations, such as the evening discount on phone calls.

PMP encourages users to separate their traffic into different classes based on service level requirement and their available wealth. The different levels of services on different logical subnetworks are implemented through proper resource management strategies such as channel scheduling and buffer management. Compared to the flat-rate pricing scheme, PMP may improve economic efficiency, but the utilization of higher priced subnetworks may be low. Because each subnetwork operates on a best-effort basis, PMP does not support individual QoS guarantees. In addition, the

number of subnetworks in PMP is small. Thus, the number of QoS classes supported is limited. It may not satisfy the diverse QoS demands of users.

2.3.2.2 Differential Service Pricing

In computer networks, channel scheduling and buffer management are two important resource management strategies to provide differential services. A channel scheduling algorithm decides which packet in queue will be serviced first. A buffer management strategy decides which packets will be dropped in time of congestion. In differential service pricing, each packet indicates the service level it belongs to, routers schedule or drop packets according to this indicator. The charge is based on the actual service received. Two prominent differential service pricing schemes are priority pricing and smart market pricing.

Priority Pricing: In *Priority pricing* [6, 7, 16], packets carry the priorities that users specify; routers service the packets according to their priorities. Packets with higher priorities receive better services than the packets with lower priorities. A higher priority incurs a higher charge. The priority reflects a user's utility[2]. Packets with a lower priority level, meaning lower value to users, may be dropped first. Thus, the priority pricing can be used to raise the economic efficiency of networks.

Gupta et al. [16] propose to dynamically set an appropriate price for each priority level based on the arrival rate and the delay cost parameters, etc. Delay cost parameters refer to the following: the jobs in the highest priority class impose delays

[2]Utility is a function to represent the performance of a network application for a certain network service. A network service contains all the relevant measures such as bandwidth, delay and packet drops. For example, utility can represent the different levels of perceived quality of a telephony application when with different amount of allocated bandwidth.

on the jobs in all other priority classes, whereas the jobs in lowest priority classes impose very little delay on the jobs in other priority classes. They are called to have different delay cost parameters. In this way, the price can reflect the relationship between demand and supply, and the congestion may be reduced and multiple levels of QoS can be achieved. Similar to deadline-based networks, priority pricing provides different levels of QoS by way of priority-based channel scheduling. Their method to set the price for different priority levels is valuable to my work.

Smart-Market Pricing: In Smart-Market pricing [28, 29], the differential services are in terms of packet service precedence and packet drop policy. The priority is indirectly specified with a packet header field called "bid". In addition to a fixed connection charge and a charge proportional to the number of packets transmitted, a usage charge in time of congestion is introduced. The usage charge is determined through an "auction". Each packet's "bid" represents the user's willingness to pay for the transmission. The network determines a threshold price based on the current load and channel capacity. The threshold value is the market-clearing price which is the price when demand equals supply. Packets with higher bids are served before those with lower bids, and packets with bid value lower than the threshold will be dropped in time of congestion. Each transmitted packet is charged this threshold value instead of the bid value. This scheme guarantees only relative priority which means a packet with a higher bid has higher priority than the packet with a lower bid, but there are no service guarantees such as guaranteed delivery time. Because of the introduction of an auctioning mechanism at each channel, this scheme would require changes to existing network protocols. The charge of smart-market pricing is

per packet. My proposed scheme also employs per-packet charges.

2.3.2.3 Fair Bandwidth Allocation

Fair bandwidth allocation deals with scenarios when resource available is scarce compared to resource demand. At such times, a fair and efficient allocation for limited network resource is needed. Here, the term fairness can be referred to from traditional network theory [5] and the field of microeconomics [35]. Different fairness criteria have different welfare functions which show the goal of the fairness. Welfare function is a function of the utilities of competitive entities such as users or flows. For example, max-min fairness [10] aims to allocate the resource as equally as possible, and proportional fairness [10] aims at maximizing the overall performance. Kelly et al. [15, 18, 19] first introduced price into a fairness definition for network bandwidth allocation which is called Proportional Fairness per unit charge. In this scheme, a fair bandwidth allocation is in proportion to a user's willingness to pay.

Fulp et al. [14, 44] introduced price into not only proportional fairness but also max-min fairness for bandwidth allocation, and introduced a network resource pricing technique based on the competitive market model. In this scheme, the price for each link's bandwidth is iteratively adjusted to reflect supply and demand by the following modified *tâtonnement process*[3] [39] at discrete time intervals: $p^i_{n+1} = p^i_n \cdot \frac{d^i_n}{\alpha \cdot s_i}$, where n is the index of time intervals. d^i_n is the aggregated user demand for the bandwidth of link i, and s_i is the link capacity. p^i_n is the current price, and p^i_{n+1} is the new price in the next time interval. The price increases after α percent of link bandwidth has been used, where $0 < \alpha \leq 100$. An equilibrium price p^i_* is reached at link

i when α percent of bandwidth equals the demand. Experiments showed that this modified *tâtonnement* process could quickly adjust to market changes. As an example implementation, this scheme is applied to the ATM Available Bit Rate (ABR) service class [8]. The ABR bandwidth is priced to control the sending rate of users in a fair and efficient manner. Resource Management (RM) cells are used to carry the price information.

Even though the fair bandwidth allocation is not my goal, the method to dynamically adjust prices based on network conditions and the mechanism to transfer the price information are valuable to my research.

2.3.2.4 Pricing and User Adaptation

Recently, more studies have focused on exploiting the adaptive nature of users to increase economic and network efficiency. The network dynamically sets prices according to the changing network conditions, and the prices serve as incentive for user to adapt their traffic. This is called *responsive pricing* [26, 27, 30, 31, 32, 33, 34]. At heavy load, the network increases the price of resource because the resource is scarce compared to demand. Price-sensitive users may reduce their service requirements or load level to adapt to the price change or pay more without adaptation. Conversely, when network utilization is low, the network decreases the price for the resource and users may increase their service requirements or load level. Thus, responsive pricing supports congestion control and traffic management.

Users adapt their requirements and traffic according to their utility functions.

[3]The *tâtonnement process* is a price adjustment process to decide the upward and downward movement of market prices. The current price is arrived at by a rise in price when demand exceeds offer and by a fall in price when offer exceeds demand.

Elastic applications such as file transfer and electronic mail are tolerant of delay, so bandwidth can be the only service measure. In contrast, real-time applications are delay sensitive and usually rate-adaptive. For them there may exist a minimum level of service and a maximum level of service; real-time applications cannot operate below the minimum service level and cannot gain additional utility above the maximum service level.

To support users' adaptation in a multi-service network, Wang et al. [40, 41, 42, 43] proposed a dynamic, usage and congestion dependent network pricing system called resource negotiation and pricing protocol (RNAP) which allows dynamic re-negotiation of services during a session. In this environment, the network offers multiple services to users, and dynamically adjusts each service price in response to the change of network load. Each session is charged the sum of three charges on the time period that the session spans. These three charges are: usage charge, holding charge, and congestion charge. The *usage charge* is determined by the actual amount of consumed resource and is set by a retail network to recover the cost of the purchase from the wholesale market and other static costs associated with a service. The *holding charge* reflects the revenue lost due to selling the allotted resource of a given service to the lower level of service. This is under the assumption that if a particular flow or aggregated flows do not use the resource, the resource will be used by the traffic from a lower level of service.

The *congestion charge* is only imposed in congestion periods, and is determined based on amount of bytes transmitted and a *congestion price.* The congestion price p_c for a service class for time interval n is calculated as an iterative *tâtonnement*

process [38, 41]: $p_c(n) = min[\{p_c(n-1) + \sigma(D,S) * (D-S)/S, 0\}^+, p_{max}]$, where D and S represent the current total demand and supply respectively. When the resource is bandwidth, D is the aggregated reserved bandwidth; when the resource is buffer, D is the average occupancy of the buffer during the period. σ is the convergence rate which may be a function of D and S. p_{max} is the maximum congestion price, and all new traffic will be refused when the congestion price reaches p_{max}. If p_{max} is reached frequently, it indicates that more resources are needed. For interval n, the congestion charge C_c is given by $p_c(n) * V(n)$, where $V(n)$ is the total number of transmitted bytes.

RNAP provides means to communicate users' traffic requirements and price information. Users' requests (service parameters) are submitted to the network by a RNAP message, the network replies with the availability of services and the service prices by another RNAP message. Wang et al. [40] introduced two architectures to support RNAP: the centralized architecture (RNAP-C) and the distributed architecture (RNAP-D). In RNAP-C, RNAP is implemented by centralized negotiators in different domains. The centralized negotiator in each domain maintains price and charge information and session states. In RNAP-D, RNAP is implemented at each router; each router maintains session charge states for the flows passing thought it.

After receiving the price information via RNAP, an adaptive application with a budget constraint can adjust its service requests according to its utility function in response to the price variations. Experiments [41] showed that usage-sensitive pricing and user adaptation can effectively reduce the blocking rate at call admission, allowing the bandwidth to be shared fairly among competing users. For instance, the

bandwidth share for elastic applications is reduced. Whereas, the bandwidth share for inelastic, i.e., non-adaptive applications remains fairly constant by charging them more. Wang et al. [43] also compared their congestion-price-adaptive (CPA) scheme with a fixed price (FP) scheme. It was shown that while utilization and the offered load increase, under CPA, the packet delay and loss rate are well controlled within a range, and the user arrival rates can be controlled at a target level, whereas under FP, the delay and the loss rate increase sharply. Additionally, users' benefit (utility) decreases much less under CPA than under FP.

It is important to note that responsive pricing needs to set resource price dynamically and a service price can be set by many methods. The modified *tâtonnement* process in 2.3.2.3 is one such method.

2.3.3 Expectation-Based Pricing

Pricing schemes in previous subsections are usage-based. This means that the charge is based on the amount of resources that a user actually consumes. Two other types of pricing called edge pricing and expected capacity pricing, in contrast, are not usage-based.

2.3.3.1 Edge Pricing

Shenker et al. [37] criticized the usage-based pricing schemes by arguing that they may not produce sufficient revenue to fully recover total cost and thus is perhaps of limited relevance. They also deemed that some factors for setting the price are inherently inaccessible to the network, and thus cannot reliably form the basis for

pricing. Therefore, they advocate shifting the research agenda to pricing schemes that focus more on structural and architectural issues such as: allowing the local control of pricing policies, fostering interconnection, handling multicast appropriately, and allowing receivers to pay for the transmission. They propose the edge pricing scheme in which a charge is locally computed at the edge based on the expected traffic conditions, rather than being computed in a distributed fashion at every node along the entire route. In edge pricing, the actual congestion condition is replaced by the expected congestion condition, and the cost of the actual path is replaced by the cost of the expected path. The edge pricing is flexible in enabling the charge to be billed to either senders or receivers, it may also enhance network efficiency by encouraging users to use multicast.

2.3.3.2 Expected Capacity Pricing

Expected capacity pricing [9] is another pricing scheme that is based on the expected instead of the actual resource usage. In expected capacity pricing, the charge is based on users' expected capacity usage from a long-term contract with the network. Users' expected usage can be specified in many ways. For instance, users can specify a minimum required capacity or a maximum transfer time of data objects, or can use an effective bandwidth based characterization. Combined with edge pricing, one can easily determine a user's charge at the edge. This method saves the effort of measuring the actual resource usage and pushes networks to manage traffic and to allocate resources according to contracts. The ATM protocols and RSVP can support this scheme. Kalyanaraman et al. [17] proposed a dynamic version of this scheme.

2.3.4 Discussion

In my research, I aim at developing a pricing and charging scheme that encourages proper user behavior and improves the delay performance of the network. Therefore, I emphasize on resource-based pricing. In my scheme, I propose that the network provides users with incentives to submit appropriate requests, and I assume users adapt their traffic based on varying prices. Thus, ideas from responsive pricing schemes can be useful. Even if I have different goals with other pricing schemes, their methods to compute the charge, set the price, transfer the pricing information are valuable to my research. Different from other research, I am going to price a service in terms of delay performance, rather than bandwidth or buffer occupancy. In my environment, I propose to set prices for a unit delay, and then charge users based on the delay performance that they receive. In this work, the delay performance of an application is reflected by the delay that is experienced by its traffic along the traffic path.

Chapter 3

Deadline-Based Data Delivery

In this chapter, I study the impact of ADU deadline urgency and traffic load on the delay performance experienced. Discrete-event simulation [3] is used. A C++ program is constructed for this purpose. The pseudo code of my simulator is give in Appendix B. Differ from previous studies where only aggregated performance was studied, in this work, I study the performance observed by individual ADUs and individual users. The different delay performance received by different ADUs and different users that I observe in this chapter motivates the design of my delay pricing and charging scheme.

3.1 Performance Model

I first describe my performance model. At a sender, each generated ADU is characterized by: source and destination addresses, size, arrival time, and deadline. For simplicity, only real-time ADUs are considered. The support to best-effort traffic

will be discussed in Section 4.3. Segmentation of an ADU into packets is performed at the sender before the packets are admitted to the network. The maximum packet size (MTU) at the network layer is assumed to be 1500 bytes. Packets are routed through the network until they reach their destination node. They are then delivered to the receiver where packet re-assembly is performed. I assume that fixed shortest-path routing is used and there are no transmission error. For simplicity, the processing time for segmentation at the sender and for re-assembly at the receiver are not included in my model, and each packet carries the deadline of the ADU to which it belongs.

The deadline-based channel scheduling algorithm implemented is the T/H algorithm. T stands for the time left, which is calculated by packet deadline minus current time. H is the number of remaining hops to destination. The value of T/H is calculated when a packet arrives at a router, it can be viewed as the urgency of a packet; specifically, a packet with a smaller T/H means that it is more urgent. There is one packet queue at each channel, packets in it are ordered by their T/H values. When the channel becomes idle, the most urgent packet in queue is serviced next. At each channel, I also implemented a "late packet drop" mechanism. If an arriving real-time packet is already late at the current hop with respect to its deadline, i.e., if its time-left is less than the sum of transmission time and propagation delay on the outgoing channel, it is certain that the packet is going to be late at its destination. In this case, the packet is simply dropped at the current hop. This is based on the assumption that packets arriving late at its destination is of little value in real-time applications.

3.1.1 Network Model

A 13-node network model is used in my simulation. Its topology is depicted in Figure 3.1. The routing tables used in this network model are given in Appendix A.

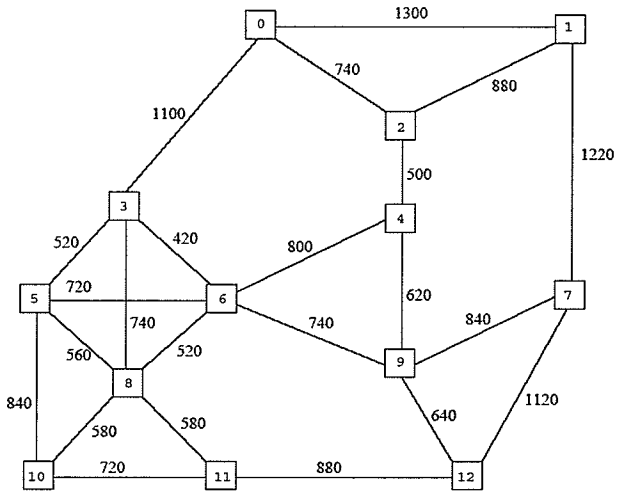

Figure 3.1: Network model

The capacity of each channel is assumed to be 155 Mbit/second. The number shown on each link is the distance in miles. It is used to determine the propagation delay. The propagation speed on all channels is assumed to be 120 miles/ms. At each scheduler, I assume that a finite buffer is in place. Modern routers usually have large buffer sizes that can accommodate between 100 and 200 ms' worth of data with respect to channel capacity [11]. I will consider buffer sizes within this range. In my simulation, the buffer size was assumed to be 3.1 MByte for each outgoing channel. This corresponds to 160 ms' worth of data with respect to the channel capacity. For simplicity, I assume that packets dropped due to buffer overflow are not

re-transmitted.

3.1.2 Traffic Model

I introduce the notion of traffic class in this study. A traffic class is identified by a source/destination pair. Ignoring identical source/destination cases, in the 13-node network model used, there are 156 traffic classes in total. The ADU interarrival time of each traffic class is assumed to be exponentially distributed, and the mean ADU interarrival time is $1/\lambda$ second. The size of each ADU is assumed to belong to one of two ranges: [500, 1500], and [1500, 500000], in bytes. The first range reflects the sizes of small ADUs, i.e., one packet per ADU. The proportion of small ADUs is kept at 50%. ADU size is assumed to be uniformly distributed within each of these two ranges.

For a real-time ADU, the delivery deadline is modeled as follows. Let x be the end-to-end latency when there is no queueing and no segmentation. Neglecting the processing delay, x includes end-to-end transmission delay and end-to-end propagation delay. Let x_p be the end-to-end propagation delay, y the size of the ADU, and c_j the capacity of the j-th channel along the path based on shortest-path routing. Then x can be calculated by $x = x_p + \sum_j y/c_j$. The allowable delay of an ADU is assumed to be proportional to x. Hence, the delivery deadline for the ADU is given by $deadline = arrival\ time + kx$, where k is referred to as a *deadline parameter* ($k > 1$). In general, a smaller k means that the ADU has a more urgent deadline.

3.2 Delay Performance in Deadline-Based Networks - Initial Observations

Two experiments were carried out. The first experiment is to compare the performance of two benchmark ADUs when they have different deadline urgency while all other attributes of these two ADUs are the same; the second experiment is to compare the performance of a benchmark session when its level of deadline urgency is varied. Here the session denotes a sequence of ADUs that are sent between a given sender and a given receiver. It is introduced to denote a subset of packets within a traffic class.

3.2.1 Delay Performance of Two Benchmark ADUs

The first experiment is to compare the delay performance of two benchmark ADUs when they carry different deadlines. To see the effect of network load on performance, I also vary the load level of background traffic. I use end-to-end ADU response time as the delay performance metric. The end-to-end ADU response time is defined as the time between when an ADU is submitted to the network for transmission and when the ADU is delivered to the destination.

3.2.1.1 Experiment Description and Result

There are two benchmark ADUs numbered 1 and 2. For both ADUs, the source is node 8, the destination is node 1, they both go through the bottleneck channel $3 \rightarrow 0$. Both ADUs are sent to the network at the same time. They have the same

size of 5000 bytes. Given the MTU size of 1500 bytes, each ADU requires four packets to transmit. This may be more realistic than a "one packet per ADU" model. The end-to-end latency x in this case is about 27 ms. Let end-to-end deadline (D) denote the time period between when an ADU is submitted by a sending application for transmission and the ADU deadline. The end-to-end deadlines for ADUs 1 and 2 are chosen to be 100, 600 ms respectively. The value of 100 accounts for the case when the allowable queueing delay is in the same order as the end-to-end latency. The value of 600 allows for much larger queuing delay. Thus ADU 1 carries a more urgent deadline than the other. Let $expo(d)$ denote the exponential distribution with mean d. The following model parameters are used for background traffic. The deadline parameter k for all background ADUs generated is assumed to be given by $1+expo(d)$, where d is 0.4. This way a variety of deadline urgency can be modeled, and the average urgency level of background traffic is neither too urgent nor too loose. d is called the deadline urgency parameter, and is used extensively in the following experiments. A smaller d means more urgent deadlines. Background traffic is sent along all traffic classes. For each traffic class, the λ is varied from 10 to 11.1 ADUs/second. They correspond respectively to 74.5% and 87.7% utilization on the bottleneck. I am interested in moderate to heavy load levels, because at these levels, the delay difference between the two ADUs with different deadline urgency is more obvious than at very light load. The results are shown in Table 3.1.

Table 3.1: Delay performance of two benchmark ADUs

Background traffic λ (ADUs/second)	ADU 1 (D=100 ms) Response time (ms)	ADU 2 (D=600 ms) Response time (ms)
10	42.22	64.44
11.1	71.21	535.67

3.2.1.2 Result Analysis

Firstly, I compare the performance of the two benchmark ADUs with different deadlines at the same network load level (see results on the same row). From the results, it can be observed that the ADU with a more urgent deadline has lower response time than the one with a less urgent deadline. I thus conclude that ADU deadline urgency largely affects the delay performance experienced; the ADU that carries a more urgent deadline receives better performance, when all other attributes are equal.

Secondly, I compare the end-to-end response times of the two benchmark ADUs when background traffic load level is varied (see the results on the same column). As we know, a larger λ value indicates a heavier load level. From the results, it can be observed that an ADU experiences longer end-to-end response time when background traffic is heavier. This is because when network load is heavier, queues at each channel may grow longer, packets thus experience longer queuing delay, thus longer response time. Therefore, the delay performance deteriorates when network load is heavier. I also observe that the difference between the response times of the two benchmark ADUs is larger when network load is heavier. I thus conclude that the deadline urgency as well as the traffic load have large impact on ADU delay performance.

These results are rather intuitive. Experiment results reinforced the intuition.

3.2.2 Delay Performance of a Benchmark Session

In the second experiment, I study the performance of a benchmark session when its level of deadline urgency is varied. In addition, I vary the load level of background traffic to investigate the effect of network load on the session delay performance. I again use the mean response time of on-time ADUs as the performance metric.

3.2.2.1 Experiment Parameters and Result

All background traffic is the same as in the first experiment, except that one of the 156 traffic classes is designated as the benchmark class, in addition, I assume that all packets on this class belong to one session. I call this session "Session 0". Session 0's arrival rate is assumed to be 11.1 ADUs/second. The deadline parameter k for session 0 is assumed to be given by $1 + expo(d)$, where d is varied from 0.1 to 0.4. The former is when the benchmark session has more urgent deadlines than background traffic; the latter is when the benchmark session has the same deadline urgency as all other traffic in the network. The load level of the background traffic is varied from 10 to 11.1, and then to 12.5, the corresponding measured utilization at the bottleneck are 75%, 88%, and 94% respectively. The measured utilization is the total time that a channel is busy divided by the simulation duration. The results are shown in Table 3.2. For completeness, I also included the ADU on-time rate results for session 0. The ADU on-time rate is calculated using the number of ADUs on-time divided by the total number of ADUs sent.

Table 3.2: Delay performance of a benchmark session

Background traffic λ (ADUs/second)	$d_{session0}$=0.4		$d_{session0}$=0.1	
	Mean on-time ADU response time (ms)	ADU on-time rate	Mean on-time ADU response time (ms)	ADU on-time rate
10	39.36	94.46%	36.48	94.08%
11.1	38.67	91.22%	34.68	89.31%
12.5	36.99	86.64%	34.30	85.11%

3.2.2.2 Result Analysis

I first compare the delay performance of session 0 when its ADUs have different deadline urgency levels (see the results on the same row). I can observe that when the deadlines are more urgent, i.e., when d=0.1, the mean response time of on-time ADUs is lower than when deadlines are less urgent. I further notice that this holds for all three network load levels. I thus conclude that a session may gain better service by assigning more urgent deadlines on its ADUs. It can also be observed that when deadlines in a session are more urgent, the session's ADU on-time rate is lower than when its traffic has less urgent deadlines. This is as expected, as deadlines become more urgent, less ADUs are delivered on-time. Yet for those that are indeed on-time, the mean response time is low compared to the case when deadlines are less urgent.

Secondly, I compare the delay performance of session 0 at different network load levels when its ADU deadlines are at the same urgency level (see the results on the same column). I can observe that as the background traffic load increases, the mean response time of on-time ADUs for session 0 decreases. Further experiments with multiple replications demonstrate that this trend is not always maintained; the confidence intervals overlap each other at 90%, 95%, 99% levels. On the other hand,

for ADU on-time rate, the three load levels did generate non-overlapping confidence intervals among the three loads, thus using on-time rate as performance metric, the heavier the load, the higher the performance.

3.3 Discussion

From the experiments in this chapter, I conclude that in deadline-based networks, the delay performance largely depends on the deadline urgency specified. When competing with the same background traffic, an ADU or a session of ADUs can raise their service priority, thus obtaining better response time performance, by using more urgent deadlines that exceed the actual needs. An important objective of my pricing and charging scheme is to prevent such greedy behaviours.

In addition, I observe that the traffic load in the network also affects the delay performance, the delay performance here means the ADU response time for the case of discrete ADUs, and it means the ADU on-time rate for the case of a real-time session. The delay performance is better when the load is lighter. Another objective of my pricing and charging scheme is to help with the network load control so that good performance can be maintained.

Chapter 4

Delay Pricing in Deadline-Based Networks

In this chapter, I present the channel delay pricing scheme, and the packet and ADU charging scheme that I have developed, discuss some related implementation issues. I then use simulation to show that my scheme can assign differential charges to traffic with different deadline urgency. In addition, the charges are higher when network load is heavier. At the end, I observe the dynamics of channel delay price over time on a bottleneck link to illustrate some important characteristics of my proposed delay pricing scheme.

4.1 A Market-Based Approach to Delay Pricing

Defer from other researches, I take a market-based approach from the field of microeconomics to price unit delay for packets on a channel in my scheme. In this

research, I call the price of unit delay *channel delay price*, and the channel delay price is updated for every *price update interval*. As we know, the price usually reflects the relationship between supply and demand in microeconomics. Therefore, to determine the channel delay price, the delay demand and delay supply need to be decided first.

The delay demand of a channel should reflect the expected delays (response times) on the channel. In deadline-based networks, each packet carries a deadline, which specifies the requirement on its delay performance. At each hop, the T/H value calculated indicates the delay requirement of this packet on this channel; namely, if the response time on this hop is less than or equal to T/H, and if every hop along the packet path manages to achieve so, then the packet will arrive at the receiver on-time. Thus, I use T/H to represent the expected delay of a packet on a channel. Suppose that M packets are serviced within the price update interval n at the channel i. Then I define delay demand at channel i (D_i) within the price update interval n as

$$D_i(n) = \frac{1}{D_i^T(n)}, \quad where \quad D_i^T(n) = \sum_{m=1}^{M} (T/H)_m. \tag{4.1}$$

From an economic point of view, the finite capacity and the transmission service at each channel is the scarce resource sought by real-time packets. The delay supply of a channel should reflect the actual delays on the channel which is related to the channel capacity and traffic load. Then I define delay supply at channel i (S_i) within price update interval n as

$$S_i(n) = \frac{1}{S_i^T(n)}, \quad where \quad S_i^T(n) = \sum_{m=1}^{M} (response - time)_m. \tag{4.2}$$

Therefore, to calculate the delay demand and delay supply on each price update interval, the following information is recorded and accumulated for each departing

packet: (i) the packet T/H value, and (ii) the packet response time. The packet response time is defined as the sum of the queuing delay, the packet transmission time, and the channel propagation delay.

After deciding the delay demand and delay supply, the channel delay price is decided next. The goal at each channel is to utilize a pricing mechanism to urge the adjustment of delay demand so that the difference between the delay supply and the delay demand can be kept minimal. An iterative tâtonnement process from [43] is used to calculate the channel delay price. The channel delay price is updated for every price update interval. At the end of price update interval n, the channel delay price for the update interval $n+1$ on channel i ($p_i(n+1)$) is defined as:

$$p_i(n+1) = \{p_i(n) + \sigma * \frac{D_i(n) - S_i(n)}{S_i(n)}, 0\}^+ \tag{4.3}$$

where σ is an *adjustment factor*, which can be used to trigger faster or slower responses of the channel delay price to the amount of difference between the delay demand and delay supply. At system initialization, p_0 is set to zero. In addition, only positive channel delay prices are defined.

It should be noted from Eq. 4.3 that, the channel delay price on channel i increases when the delay demand is greater than the delay supply. Furthermore the price is higher (i) when the deadline urgency is higher, this is because a higher deadline urgency level results in smaller values of T/H, thus a smaller value of D_i^T and a larger value of D_i; and (ii) when the load is heavier. This is because a heavier load results in longer response times, thus a larger value of S_i^T and a smaller value of S_i. Conversely, the channel delay price decreases when delay demand is lower than the delay supply.

4.2 Calculation of Packet and ADU Charges

Using the channel delay pricing scheme presented above, I describe a method to calculate packet and ADU charges in deadline-based networks. Note that in this work, I focus on devising a *delay charging* scheme that aims at two objectives: (1) to provide an incentive for network users to submit requests with the QoS requirement that best matches their need, and (2) to control network load so that good delay performance can be maintained. In general, network charging schemes usually contain certain charges in order to assure the return on investment; these charges may cover the cost for constructing, maintaining, and upgrading the network. In this research, however, I do not consider these charges and focus on the delay charge only.

A per-packet per-channel charging scheme is employed in my framework. At each channel, upon each packet departure, the packet response time d_a is calculated: the queuing delay can be obtained by subtracting the packet arrival time from the current time at this channel, the transmission delay can be computed with the packet size and the channel capacity, the propagation delay is fixed and given. Let p be the current channel delay price. The packet charge g at this channel is defined as: $g = p/d_a$. Define a new packet header field called "accumulated charge". It keeps track of the total delay charge incurred by this packet at all channels along its path. If a packet arrives at the receiver on-time, the value of this field is retrieved and is taken as the packet charge. If an ADU is delivered on-time, its ADU charge is defined as the sum of all its packets' charges. Late packets and ADUs are not charged.

4.3 Network Layer Issues

In my pricing scheme, the channel delay prices are updated periodically at constant time intervals. In general, the length of the update interval should not be too short, this way a good number of T/H value and response time samples can be collected to estimate the current resource demand and supply. I suggest to use a length that is much longer than the average packet transmission time. The update interval should not be too long either, in this way the short-term traffic conditions can be accounted for.

My pricing and charging scheme introduces some processing overhead inside routers. This includes packet response time calculation, and accumulation of the T/H values and the packet response times for all departing packets. However, because none of these operations depends on queue size, I consider this overhead to be inexpensive in terms of implementation. The computation of delay prices only occurs once every update interval, which is much longer than the mean packet transmission time, therefore is not considered costly either. The "accumulated charge" header field can also be easily added using packet header options or similar mechanisms.

The social fairness aspect of a pricing scheme is concerned with whether some users will be prevented from accessing the network only because of their inability to pay [12]. In my discussion so far, I have assumed that there is only real-time traffic in the network. In fact, best-effort traffic can easily be accommodated in my framework. Two queues on each outgoing channel can be set. One is the high-priority queue for real-time packets, the other is low-priority queue for best-effort packets. All best-effort traffic can carry a deadline of infinity. At each outgoing channel inside the network,

a certain amount of bandwidth can be allocated to best-effort traffic only. This can be implemented using a fair-queuing algorithm with two classes. The deadline-based scheduling is used only within the real-time class. The best-effort class can use a low flat-rate pricing scheme. Those users who can not afford the delay charges of real-time class can use the amount of resource for best-effort class.

4.4 Performance Evaluation

In this section, I evaluate the performance of my pricing and charging scheme by simulation. In Section 3.2, I have shown that it is possible for a user to gain higher service priority in deadline-based networks by specifying more urgent deadlines and this holds for both individual ADUs and for a series of ADUs of a session. To prevent greedy users from gaining an advantage by using arbitrarily tight deadlines that exceed their actual needs, I introduce a packet delay charge that is tied to the packet response time. In this section, I show that after introducing my delay pricing and charging scheme, more urgent traffic would incur a higher charge than less urgent traffic and this holds for various scenarios. In addition, I show that the traffic charges are higher (I) when the network load is heavier, (II) when the deadline urgency is higher, and (III) when an ADU size is larger.

In my experiments, I used the performance model that is described in Section 3.1 with the addition of my pricing and charging scheme implementation. The following values are chosen for algorithm parameters. The adjustment factor σ in Eq. 4.1 is set to 0.06, referred to the value chosen in [43]. The price update interval is chosen to be 1 second. The simulation is run for 50 seconds.

4.4.1 Differential Charges of Two Benchmark ADUs

Three experiments are conducted in the individual ADUs case. The first experiment is identical to the one reported in Section 3.2.1, except that the ADU charges obtained are added. The second experiment examines the cases when the deadline urgency of background traffic is varied. The third experiment examines the cases when the benchmark ADU size is varied. The ADU charges obtained are in *charge unit* (cu). In this work, I do not associate the charge unit with any concrete monetary value, and leave this choice to network operators.

4.4.1.1 Experiment I: Differential Charges When Load Varies

The goal of this experiment is to compare the charges of the two benchmark ADUs that have different deadlines when load varies. All experiment parameters are the same as in Section 3.2.1. The results are shown in Table 4.1. This table is identical to Table 3.1 except the addition of ADU charges.

Table 4.1: Charges of two benchmark ADUs

Background traffic λ (ADUs/second)	ADU 1 (D=100 ms)		ADU 2 (D=600 ms)	
	Response time (ms)	Charge (cu)	Response time (ms)	Charge (cu)
10	42.22	0	64.44	0
11.1	71.21	6.9664	535.67	0.7584

It can observed that when $\lambda = 11.1$ ADUs/second, which corresponds to a bottleneck utilization of 88%, the ADU that has the more urgent deadline is charged more than the one with less urgent deadline. When $\lambda = 10$ ADUs/second, which corre-

sponds to a bottleneck utilization of 74%, both ADUs received a zero charge. This is because when load is light, the delay demand is less than the delay supply, thus the delay price at various channels is zero during the period when the benchmark ADUs are serviced. So both ADUs received a zero charge. Note that the exact load level at which the charge becomes non-zero depends on many factors, and may be much difficult to determine before-hand. I thus conclude that when delay demand is higher than delay supply (which is the case when $\lambda = 11.1$ ADUs/second), my delay pricing and charging scheme can provide charge differentiation to ADUs that have different deadline urgency; the more urgent ADUs are charged higher.

In my scheme, I initialize the channel delay price to zero. If the delay demand keeps being lower than the delay supply, the price will keep to be zero, in this case, my scheme cannot provide charge differentiation. If a service provider would like to provide charge differentiation in term of ADU end-to-end response time under this condition, they can enforce a lowest channel delay price, and initialize a positive channel delay price and a positive lowest channel delay price at system start.

4.4.1.2 Experiment II: Varying Background Traffic Deadline Urgency

The second experiment aims to compare the charges of the two benchmark ADUs when the deadline urgency of background traffic is varied. This is to demonstrate that the delay differential charges of my scheme sustains in various scenarios. All experiment parameters are the same as in the last experiment, except that the λ of background traffic is 11.1 ADUs/second. In the last experiment, the parameter d for the deadlines of background traffic is 0.4, in this experiment, I add the case for when

d equals 0.2. The results are shown in Table 4.2.

Table 4.2: ADU charges at different background deadline urgency levels

Background traffic d	ADU 1 (D=100 ms)		ADU 2 (D=600 ms)	
	Response time (ms)	Charge (cu)	Response time (ms)	Charge (cu)
0.4	71.21	6.9664	535.67	0.7584
0.2	86.08	11.31	535.67	1.5725

It can be observed that (a) for both $d = 0.2$ and $d = 0.4$, ADU 1 has more urgent deadlines, thus achieves better response time performance than ADU 2; using my delay pricing and charging scheme, ADU 1 is always charged higher than ADU 2. (b) ADU 1 experiences higher response time when the background traffic has more urgent deadlines, and ADU 2 experiences the same response time. This is because that the priority of ADU 1 decreases when the background traffic has more urgent deadlines. However, ADU 2's deadline is always much looser than the other ADUs, thus its priority stays the same in both scenarios. (c) Both ADU 1 and ADU 2 are charged more when background traffic has more urgent deadline, even though ADU1 experiences longer response time and ADU2 experiences the same response time when d of background traffic is 0.2. This is because the deadline urgency level of the traffic reflects the delay demand on a channel; when the traffic's deadline urgency level is higher, the price is higher. Therefore the channel delay price is higher when d of background traffic is 0.2, and a higher price results in a higher charge. I conclude from this experiment that the charge differentiation among the two benchmark ADUs that have different deadline urgency sustains when background traffic deadline urgency is

varied.

4.4.1.3 Experiment III: Differential Charges When Varying ADU Size

The third experiment aims to compare the charges of the two benchmark ADUs when their sizes vary. In previous experiments, the benchmark ADUs' sizes have been 5000 bytes. In this experiment, I add the case when ADU sizes equal to 1500 byte. Given the MTU size of 1500 bytes in my performance model, the old case corresponds to 4 packets per ADU, while the new case corresponds to one packet per ADU. The d value of background traffic is 0.4. Other parameters are the same as in the last experiment. The results are shown in Table 4.3.

Table 4.3: Charges of benchmark ADUs with different sizes

ADU size (byte)	ADU 1 (D=100 ms)		ADU 2 (D=600 ms)	
	Response time (ms)	Charge (cu)	Response time (ms)	Charge (cu)
1500	70.85	1.7428	535.31	0.1896
5000	71.21	6.9664	535.67	0.7584

It can be observed that ADUs with a larger size are charged more. This is because I calculate the charge per packet. An ADU that requires more packets to transmit could incur a higher charge.

4.4.2 Differential Charges of A Benchmark Session

Two experiments are conducted in the individual session case. The first experiment is identical to the one reported in Section 3.2.2, except that the mean ADU

charge is added. The second experiment examines the cases when the deadline urgency of background traffic is varied.

4.4.2.1 Experiment I: Differential Charges When Load Varies

There are two goals of this experiment. The first goal is to observe the mean ADU charges of a benchmark session when its level of deadline urgency is varied. The second goal is to investigate the effect of network load on the mean ADU charges obtained by the benchmark session. The result are shown in Table 4.4. This table is identical to Table 3.2 except the addition of mean ADU charges.

Table 4.4: Mean ADU charges of a benchmark session

Background traffic λ (ADUs/second)	$d_{session0}$=0.4			$d_{session0}$=0.1		
	Mean response time (ms)	ADU on-time rate	Mean ADU charge (cu)	Mean response time (ms)	ADU on-time rate	Mean ADU charge (cu)
10	39.36	94.46%	38.47	36.48	94.08%	50.89
11.1	38.67	91.22%	1474.09	34.68	89.31%	1781.48
12.5	36.99	86.64%	5301.97	34.30	85.11%	6037.62

Let's first look at the mean ADU charges of Session 0 under the same network load level (see the results on the same row). It can be observed that the mean ADU charge of Session 0 is higher when its ADUs' deadlines are more urgent. Secondly, let's look at the mean ADU charges of Session 0 under different network load levels (see the results on the same column). It can be observed that the mean ADU charge of Session 0 is higher when the network load is heavier. Thus, it can be concluded that my delay pricing and charging scheme can provide differential charges to individual

ADU sessions; the more urgent the session's traffic, the higher the charge. In addition, an ADU session is charged more at heavier network load.

4.4.2.2 Experiment II: Varying Background Traffic Deadline Urgency

The second experiment studies the mean ADU charges of a benchmark session when the deadline urgency of background traffic is varied. All experiment parameters are the same as in the last experiment, except that (a) the λ of background traffic is fixed at 11.1 ADUs/second; (b) in addition to 0.4, I added one more level, 0.2, for the parameter d of background traffic. The results are shown in Table 4.5.

Table 4.5: Session ADU charge at different background deadline urgency levels

Background traffic d	$d_{session0}$=0.4			$d_{session0}$=0.1		
	Mean response time (ms)	ADU on-time rate	Mean ADU charge (cu)	Mean response time (ms)	ADU on-time rate	Mean ADU charge (cu)
0.4	38.67	91.22%	1474.09	34.68	89.31%	1781.48
0.2	37.48	85.69%	2570.14	35.10	85.88%	3100.69

It can be observed that, (a) for both levels of background traffic urgency, the mean ADU charge of Session 0 is higher when its ADU deadline urgency is higher. This is a desired effect of my scheme, when traffic deadlines are more urgent, the traffic is charged higher. (b) For both levels of Session 0's traffic urgency, when the background traffic has more urgent deadlines, Session 0's mean ADU charge increases. This is because when network traffic deadline urgency level is higher, delay demand is higher, thus the price is higher. A higher price would result in a higher charge.

4.4.3 The Factors That Affect Channel Delay Prices

To better understand the characteristics of my delay pricing and charging scheme, in this section, I observe the dynamics of the channel delay price at a bottleneck channel over time. I have seen from previous results that two factors, namely the traffic deadline urgency and the level of network load affect the traffic charges. I focus on these two factors.

4.4.3.1 Experiment I: Effect of Deadline Urgency on Delay Price

I first emphasize on the effect of traffic urgency on delay prices. The network load level λ of 12.5 ADUs/second is used. This corresponds to a bottleneck link utilization of 94%. Five levels of d value for all traffic are experimented, they are 0.2, 0.3, 0.4, 0.6, and 1.2. All other parameters are the same as in the previous experiments. The price dynamics are shown in Figure 4.1(a). Because the initial value of channel prices is zero in my model, all price curves start with price 0.

It can be observed that (a) the channel delay prices at a higher deadline urgency level are always higher than the prices at a lower deadline urgency level. Among the five levels of d, d=0.2 results in the highest prices. (b) The prices for d=0.2, 0.3, 0.4, and 0.6 monotonically increase over time, while the prices for d=1.2 are close to zero. This is because in the former cases, the delay demands are greater than the delay supplies, thus the delay prices keep increasing. In the latter case, the demand approaches the supply, thus the price stays flat at values close to zero. (c) Among the cases when demands are greater than supplies, the lower the value of d, i.e., the higher the deadline urgency level, the faster the delay prices increase. This is caused

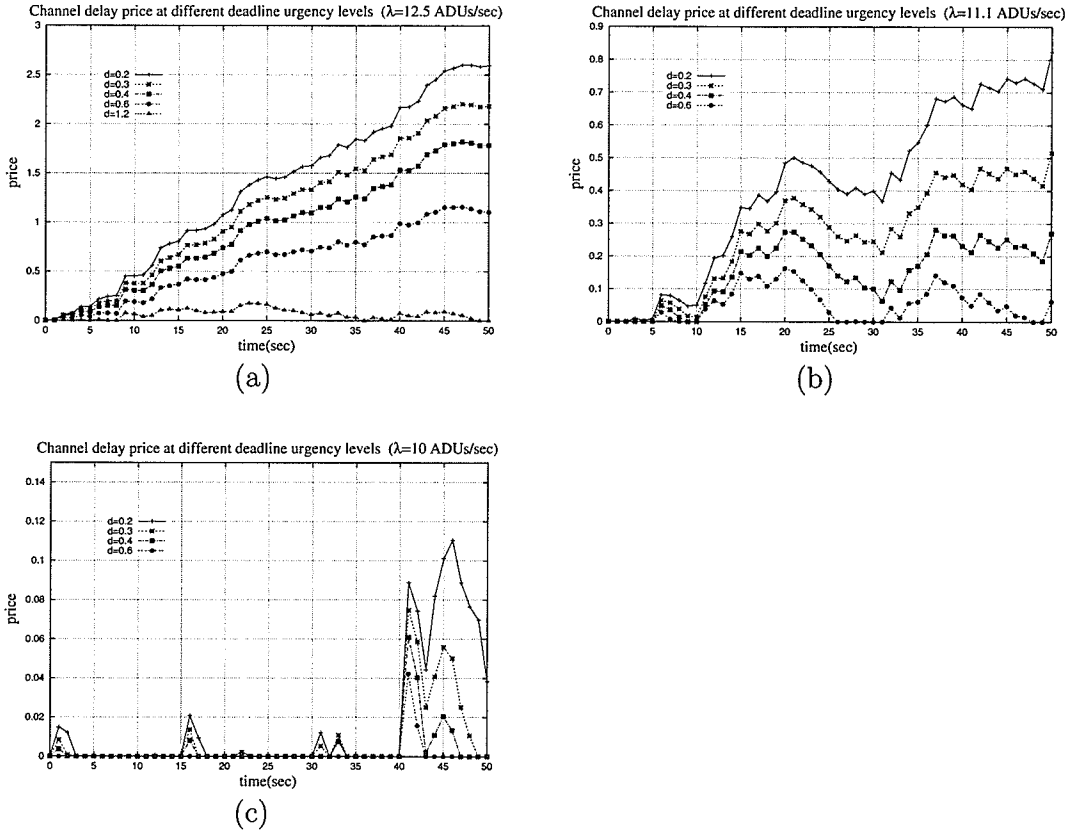

Figure 4.1: Channel delay price at different traffic deadline urgency levels

by the larger mismatch between demand and supply when deadline urgency level is higher. It can be concluded that the channel delay price is largely affected by the traffic deadline urgency; when the delay demand is greater than the delay supply, the more urgent the deadlines, the higher the channel price.

Similar observations can be made for other load levels. In Figures 4.1(b) and (c), I plot the price dynamics for when λ=11.1 ADUs/second and when λ=10 ADUs/second. Note that the scales on the Y axis are very different in these graphs, but the trend stays the same.

4.4.3.2 Experiment II: Effect of Network Load on Delay Price

The second experiment emphasizes on the effect of network load on the channel delay prices. The value of d is assumed to be 0.2. I observe the price dynamics when λ is 10, 10.5, 11.1, 11.8, and 12.5 ADUs/second respectively. The corresponding values of bottleneck link utilization are given in Table 4.6. The results are shown in

Table 4.6: Network Load Parameters

λ(ADUs/second)	Utilization
10	76%
10.5	81%
11.1	87%
11.8	90%
12.5	94%

Figure 4.2(a).

It can be observed that (a) the channel delay prices at a higher load level are always higher than the prices at a lower load level. Among the five levels of λ, λ=1.5 results in the highest prices. (b) The prices for λ=12.5, 11.8, 11.1, and 10.5 monotonically increase over time, while the prices for λ=10 are close to zero. This is because in the former cases, the delay demands are greater than the delay supplies, thus the delay prices keep increasing. In the latter case, the demand approaches the supply, thus the price stays flat at values close to zero. (c) Among the cases when demands are greater than supplies, the higher the load level, the faster the delay prices increase. This is caused by the larger mismatch between demand and supply when network load level is higher. It can be concluded that the channel delay price is largely affected by the network load level; when the delay demand is greater than

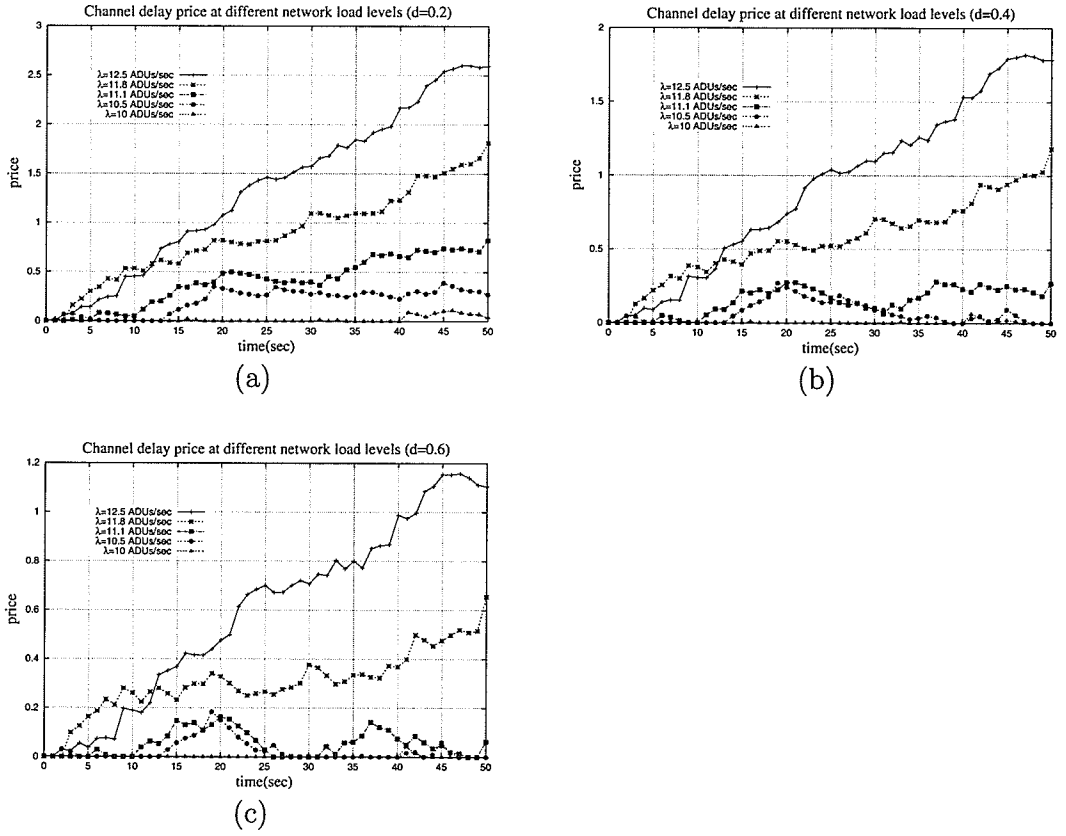

Figure 4.2: Channel delay price at different network load levels

the delay supply, the higher the network load, the higher the channel price. When demand and supply are approximately equal, the price is close to zero.

Similar observations can be made for other traffic deadline urgency levels. In Figures 4.2 (b) and (c), I plot the price dynamics for when d is 0.4 and 0.6 respectively. Note that the scales on the Y axis are very different in these graphs, but the trend stays the same.

4.5 Discussion

From the experiments in this chapter, I have shown that when at the same network load level, my pricing and charging scheme can yield differential charges on individual ADUs with different deadline urgency levels or with different ADU sizes; an ADU with a more urgent deadline or a larger size is charged more. Thus a user who assigns more urgent deadlines on his/her ADUs and generates more traffic will be charged more. In other words, my scheme can be used to prevent user's greedy behavior of assigning arbitrary urgent deadline to his/her traffic. In addition, my scheme provides a higher charge at a higher network load level. Therefore can also aid in network load control.

The experiments in this chapter also show that both the traffic deadline urgency and the network load level affect the channel delay prices. When delay demand is greater than delay supply, the channel delay price is higher when the network load level is higher or when the traffic deadline urgency level is higher. Therefore, the channel delay prices reflects the current network load conditions with respect to the current traffic deadline urgency conditions. In the next chapter, I introduce the concept of user adaptation, and show that my pricing and charging scheme easily enables user adaptation, which in turn can significantly improve the network performance.

Chapter 5

User Adaptation

In a network with a service pricing and charging scheme in place, users are normally assumed to be price-sensitive; users may adapt their traffic transmissions in response to price changes. If users cannot afford their traffic transmission activities given the current price, they either lower their transmission requirements, lower their traffic, or opt not to transmit. Conversely, if users have abundant budget, they may raise their transmission quality requirements to gain a better service or increase their traffic to accomplish more tasks. In short, adaptive users or applications can adapt their traffic requirements. In this chapter, I focus on adaptive users with limited budget for their service requests. My goal is to demonstrate that my developed pricing and charging scheme can easily enable user adaptations, which in turn can lead to much improved application and network performance.

In this chapter, I first present an example user adaptation model for real-time traffic. I then use simulation to show the on-time performance improvement after introducing user adaptation. Two traffic scenarios are experimented: discrete real-

time ADU traffic, and a mixture of real-time on-line game traffic and continuous-media traffic.

5.1 Price-Sensitive User Adaptation

I first describe an example user adaptation model. Note that my goal is not to come up with an accurate or even a good user adaptation model, but rather to use a "somewhat" reasonable one to demonstrate the potential of my pricing and charging scheme in network load control.

I assume that user adaptation is performed periodically at regular time intervals. These time intervals are called *adaptation intervals.* I also assume that each user (or application) has a fixed amount of budget for every price update interval called *interval budget.* Each ADU, upon arrival at the destination, is charged using my charging scheme. The ADU charges are made available to the senders via application-layer acknowledgments (ACKs), and each user records the total charge that is received in every price update interval.

Based on the charge received in the previous adaptation interval, a sender can adjust one or more of the following three traffic attributes for the subsequent traffic: the ADU deadline urgency level, the ADU sending rate, and the ADU size. When the charge received in the previous adaptation interval is higher than the interval budget, a user may lower its traffic requirements in terms of less urgent deadlines, a lower ADU sending rate, and smaller ADUs. When deadlines are less urgent, its priority in deadline-based scheduling would drop, the ADU response time would increase, which results in a lower charge; when the ADUs are sent at a lower rate, fewer ADUs are sent

every fixed time interval, thus a lower charge; when ADUs are smaller, the network load level can be reduced, the mismatch between the delay demand and the delay supply would decrease, which results in lower prices and lower charges. Conversely, when the charge received in the previous adaptation interval is lower than the interval budget, the user can raise the requirements of his/her traffic. The exact algorithm for increasing or decreasing service requirements are application-traffic specific.

For simplicity, in my simulation model, I assume that the user adaptation intervals coincide with the price update intervals. In my evaluation, two types of application traffic are experimented: discrete real-time ADUs and a mixture of real-time on-line game traffic and continuous-media traffic. The former represents generic real-time traffic while the latter represents a more realistic traffic mix. I first describe the adaptation scheme used by each type of traffic, and then show the performance gain achieved by the combination of pricing and user adaptation. The case when there is no user adaptation serves as my benchmark scenario to be compared with.

5.2 Case I: Discrete ADU Traffic Only

In this section, I use simulation to study the effect of user adaptation on performance using the traffic model that I have used in previous chapters, namely, discrete ADU traffic. I still use the same 13-node network model.

5.2.1 Adaptation Model

I still use the notion of a session, ADUs belong to the same session traverse the same path between the given sender and the given receiver. I assume that the user

adaptation is based on session, and within each adaptation interval, each session is allocated a fixed amount of user budget called *session interval budget* (IB_s). A session's charge in the previous interval is represented by (IC_s). The pseudo code for session adaptation is given in Table 5.1.

Table 5.1: Pseudo code for session adaptation in Case I

```
/* The function for session adaptation in Case I*/

FUNCTION (Session-Adaptation)
    BEGIN
        IF (IC_s > IB_s)
            Decrease the session's traffic requirement;
        IF (IC_s < IB_s)
            Increase the session's traffic requirement;
    END
```

For simplicity, I assume that there is only one session on each traffic class. Each session continuously generates ADUs during the entire simulation duration. Each session is associated with three attributes: (1) deadline urgency parameter d, (2) mean ADU interarrival time I, in seconds, and (3) ADU size parameter θ. The deadlines and ADU interarrival time follow the same distribution as before where the ADU deadlines are given by $deadline = arrival\ time + (1 + expo(d))x$ and the ADU interarrival time of each session is assumed to be exponentially distributed with mean I. The ADU size is assumed to be the product of θ and a random variate z, in bytes. θ is a parameter whose value increases (or decreases) when service requirement of ADU size is increased (or decreased) in application adaptation. z is generated from two ranges with equal probability: Uniform(500, 1500), and Uniform(1500, 500000), in

bytes. The increase or decrease of service requirements is modeled as follows. Define an *adaptation parameter* α. In my experiments, I assume that a session adapts one of three attributes. Let the value of an attribute in the update interval n be v_n. The value of the attribute in the update interval $n+1$, v_{n+1}, is calculated by either

$$v_{n+1} = v_n * (1 + \alpha) \tag{5.1}$$

or

$$v_{n+1} = v_n * (1 - \alpha). \tag{5.2}$$

I use the aggregated on-time ADU throughput (in the number of ADUs per second) as my performance metric. It is defined as the total number of on-time ADUs across all traffic classes in the network divided by the simulation time.

5.2.2 Aggregated Performance With User Adaptation

To evaluate the ADU on-time performance when with application adaptation, four cases are studied: one without adaptation, in the other three, the three attributes: d, I, and θ are adapted respectively. All sessions' initial value of d is set to 0.4. The minimum value of d is 0.2. There is not an enforced upper bound on the value of d. All initial values of I are 0.08 seconds, which corresponds to an offered load level of 1.01[4] on the bottleneck when without user adaptation. There is not an enforced upper bound and lower bound on the value of I. The initial value and the maximum value of θ are set to 1. There is not an enforced lower bound on the value of θ.

The price update interval and the adaptation interval are assumed to be 1 second

[4]The offered load is calculated using the measured total bit arrival rate at the bottleneck divided by the bottleneck channel capacity.

long. The initial price at each channel is assumed to be zero. All sessions' interval budget is assumed to be 0.1. I select a small value of interval budget so that user adaptations can be triggered quickly after the start of simulation and can be triggered more frequently. In my experiments, an α value of 0.4 is used. Each simulation is run for 150 seconds. I assume that each sender starts to perform adaptation at time 10 seconds. This is to make sure that users begin their adaptation after the router buffers are filled to some degree.

The results from my simulations are shown in Table 5.2. It can be observed that

Table 5.2: Delay performance without and with adaptation

Adaptation attribute	ADU On-time throughput (ADUs/second)
Without adaptation	1680.61
Adapting deadline d	1724.48
Adapting inter_arr_t I	1736.78
Adapting ADU size θ	1693.19

the ADU on-time throughput is improved with application adaptation. Using the simple adaptation model described above and a fixed value of α, it appears that adapting the ADU sending rate results in the best performance gain. I also plot the price dynamics over time on the bottleneck when without and with adaptation (see Figure 5.1). It can be observed that the channel delay price on the bottleneck can be well controlled when with user adaptation, in contrast the price keeps increasing when without user adaptation. As the offered load is larger than 1 in this case, the bottleneck is overloaded, thus demand is much greater than supply, without user adaptation, as a sustained high level of load, price keeps increasing. In contrast, the price can be much better controlled with user adaptation; the price stabilizes at simulation time 50 seconds, and stay close to zero after then. Combined with the

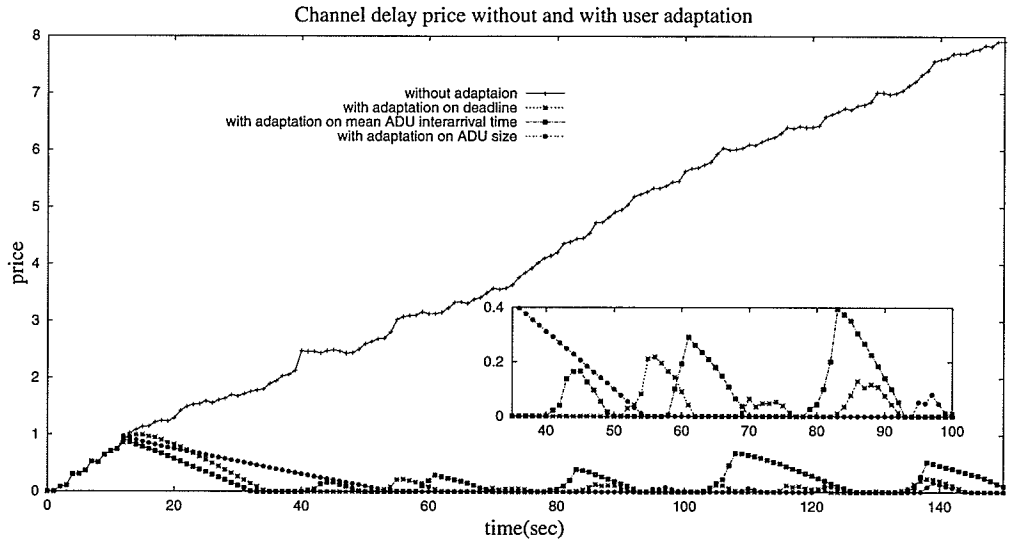

Figure 5.1: Channel delay price without and with adaptation, Case I

throughput results, it can be concluded that with a simple user adaptation model as ours, pricing together with user adaptation can improve network performance.

5.2.3 Effect of User Budget in User Adaptation

In this experiment, I study the effect of user budget in user adaptation. Intuitively, the higher the user budget, the higher the user service requirements that s/he can afford. I observe the adaptation of a benchmark session for three levels of interval budget. The session from node 11 to node 0 (session 1) is chosen. This session is one that pass by the bottleneck. The three levels of interval budget experimented are 0.1, 600, and 6000. For proof of concept, I choose deadline urgency parameter d as the adaptation attribute. Other parameters are the same as in the last experiment. All sessions start performing adaptations at time 10 seconds.

I plot the d values of session 1 over time as the result of user adaptation in Figure 5.2. Only the time range 50 to 150 seconds is shown. This is after the initial

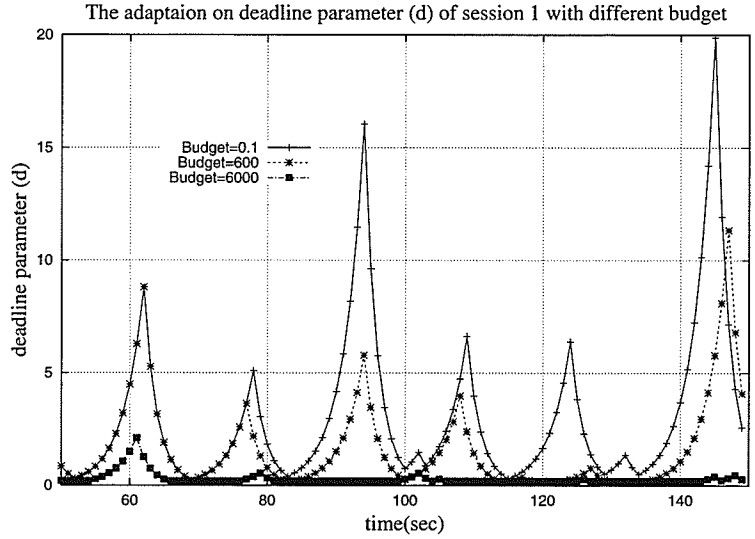

Figure 5.2: User deadline urgency adaptation when with different budgets, Case I

transient period (as shown in Figure 5.1). After time 50 seconds, the channel delay price is stabilized. It can be observed that the value of d fluctuates over time as the result of periodic adaptation. Further, the fluctuation is more severe for lower levels of budget. When interval budget is 0.1, the maximum value of d is almost 20. This corresponds to a very loose deadline. When user budget is higher, however, the maximum value of d is around 2. This corresponds to a much more urgent deadline. In other words, the session can send traffic with more urgent deadlines when s/he has a higher interval budget.

Similar results are observed when adapting the ADU sending rate and the ADU size. A session with a higher budget can afford higher ADU sending rate and larger ADUs. Thus, it can be concluded that a session with a higher budget can submit

higher service requirements for their transmissions.

5.3 Case Study II: A Mixture of Game and Multimedia Traffic

The traffic scenario in Case I represents a generic type of traffic with ample space for traffic attribute adaptation. In more realistic traffic scenarios, each user may open multiple real-time applications together, and each real-time application may have its own traffic characteristics. In particular, some parameters may not be adapted or may only be adapted within a small range. For example, a video application sends video frames at a rate of 25 frames/second to the network, in this case each video frame can be considered an ADU, and arbitrarily increasing or decreasing the ADU sending rate may not be possible. In addition, if an application can no longer adapt its traffic parameters, it may be wise to temporarily pause the session, later when network load drops, the stopped session may be resumed. In this section, I experiment with a more realistic traffic scenario in which both distributed multiplayer interactive game traffic and distributed multimedia application traffic are involved.

There are many types of distributed multiplayer interactive games [21], in this study, I focus on first-person-shooter (FPS) games. There are a large variety of distributed multimedia applications. In this study, I consider two classes: streaming multimedia, which include streaming stored audio/video and streaming live audio/video, and real-time interactive audio/video [20]. In my performance model, there are three types of traffic: game, interactive multimedia, and streaming multimedia.

In this scenario, the session represents one real-time application which is either a game, an interactive multimedia application, or a streaming multimedia application. In addition, one user opens multiple sessions.

5.3.1 Adaptation Model

Session and User Adaptations In this case, each session represents the traffic of a real-time application. When an application performs service requirement adaptation, it may not only consider the charge received in the past, but also consider the delay performance received in the past. For example, even though the charge in the previous interval is lower than the interval budget, which means the user has abundant budget, the user may also received unsatisfactory delay performance in the previous interval, in this case, the user may still decrease the service requirements. So, in this more realistic scenario, I also consider the delay performance in the previous interval when doing adaptation. Besides recording the total charge that is received in every price update interval, each session also records the number of on-time ADUs (N_t) based on ACKs and the number of ADUs sent (N) per adaptation interval. I assume that each session specifies an expected level of ADU on-time rate, *EptRate*. When the value of N_t/N is greater than *EptRate*, a session is regarded as receiving satisfactory delay performance in the previous interval. Otherwise, a session is deemed unsatisfied. To set up a more realistic adaptation model, I also define a *threshold budget parameter* β to quantify the ratio of interval charge and interval budget. My adaptation model for a session operates as follows. If a session's charge (IC_s) in the previous interval is lower than β percent of the session's interval budget (IB_s) and the value of N_t/N of the

session in the previous interval is greater than the session's *EptRate*, which indicate that the session has abundant budget and received satisfactory delay performance, the session performs adaptation by increasing its service requirement. Conversely, if a session's charge in the previous interval is greater than the session's interval budget or the value of N_t/N of the session in the previous interval is lower than the session's *EptRate*, which indicate that the session does not possess enough budget or the session received unsatisfactory delay performance, the session performs adaptation by decreasing its service requirements. If the charge is in between β percent and 100% of the interval budget and the value of N_t/N of the session in the previous interval is at least the session's *EptRate*, no adaptation is carried out. The pseudo code for session adaptation is given in Tabel 5.3.

Table 5.3: Pseudo code for session adaptation in Case II

```
/* The function for session adaptation in Case II*/

FUNCTION (Session-Adaptation)
    BEGIN
        IF (IC_s > IB_s)  OR  (N_t/N < EptRate)
            Decrease the session's traffic requirement;
        IF (IC_s < β * IB_s)  AND  (N_t/N > EptRate )
            Increase the session's traffic requirement;
    END
```

Each session initially selects an end-to-end deadline D that is used to determine the ADU deadlines of this session. The initial D is modelled as follows. Let x_p be the end-to-end propagation delay. The initial D=$(1 + expo(d))^*$ x_p. Differ with the method before, I do not consider the transmission delay. Because the ADU sizes are

relatively small (the mean ADU size for game session is 72.3 bytes, and the maximum ADU size for multimedia session is 19700 bytes), the transmission delay is much smaller than propagation delay. Thus I ignore the transmission delay when deciding the initial D. Each session has a minimum value of D ($minD$) and a maximum value of D ($maxD$). The end-to-end deadline of a session's ADUs can only be adapted within the range from $minD$ to $maxD$.

I have used the notions of traffic class and session before. In this section, I introduce the notion of a user. A user can send traffic across multiple traffic classes from the same source. Along each traffic class, a user may initiate multiple sessions. Each session carries the traffic from one of the three types of aforementioned applications. To simplify the simulation, there is only one user on each traffic class, and a user generates a number of sessions belonging to the same traffic class at the beginning of simulation. All sessions continuously transmit ADUs during the entire simulation run except being stopped by their user, this way sessions perform the adaptation during the entire simulation. Each user has an interval budget. A user's interval budget (IB_u) is the sum of all his/her sessions' interval budgets. A user's charge (IC_u) in an adaptation interval is the sum of all his/her sessions' charges in the adaptation interval. If IC_u is less than β*IB_u and the user has stopped sessions, the earliest stopped session is resumed. In addition, each ongoing session of this user performs adaptation. If IC_u is greater than β*IB_u (this includes the case when IC_u is greater than IB_u), normal session adaptation described above is performed. The pseudo code for user adaptation is given in Table 5.4.

There are three types of session in the traffic mix: real-time interactive game,

Table 5.4: Pseudo code for user adaptation in Case II

```
/* The function for user adaptation in Case II*/

FUNCTION (User-Adaptation)
    BEGIN
        IB_u=∑user.IB_s;
        IC_u=∑user.IC_s;
        IF (IC_u < β * IB_u) AND (has stopped sessions)
            Resume the earliest stopped session
        FOR all user's sessions
            Function(Session-Adapatation);//Perform session adaptation
    END
```

interactive multimedia, and streaming multimedia. Each type of session has its own characteristics, thus its own specific adaptation model. Next, I introduce each type of traffic and its adaptation model.

Game Traffic The traffic model of FPS games has been studied previously and is available in open literature, in this study, I use the one in [22]. In multiplayer interactive games, when players make moves, state update messages are sent from game clients to the game server, and then being forwarded to other affected game clients by the server. Each state update message can be considered an ADU. The traffic model is slightly different between the "game clients to game server" traffic and the "game server to game clients" traffic. For simplicity, in this study, I only include the "game clients to game server" traffic in my traffic model. The ADU interarrival times are bimodal, half of them are 33 ms fixed, the other half are 50 ms fixed. The two values randomly alternate. The ADU sizes follow a normal distribution with mean of 72.3, and standard deviation of 7.0, in bytes.

The adaptation of game traffic is modelled as follows. Let D denote the end-to-end deadline of an ADU. The maximum D ($maxD$) for a game ADU is assumed to be 0.2 second. For game traffic, I assume that the ADU sizes and ADU interarrival times are not adjustable. Thus only ADU deadline can be adapted. The adaptation model in Eq. 5.1 and Eq. 5.2 is used on the parameter D.

The pseudo code for the adaptation of a game session is given in Table 5.5.

Table 5.5: Pseudo code for game session adaptation in Case II

```
/* The function increasing the service requirements for a game session*/

FUNCTION (GAME-Increase)
    BEGIN
        IF (D > minD)
        {
            D = D * (1 − α); //Decreasing end-to-end deadline
            IF (D < minD);
                D = minD;
        }
    END

/* The function decreasing the service requirements for a game session*/

FUNCTION (Game-Decrease)
    BEGIN
        IF (D < maxD)
        {
            D = D * (1 + α); //Increasing end-to-end deadline
            IF (D > maxD);
                D = maxD;
        }
        ELSE      //D cannot be adapted
            STOPPED      //The session is stopped
    END
```

Multimedia Traffic The traffic model for the interactive and streaming multimedia traffic is also obtained from open literature. A total of 10 video traces from [1] are used in my study. A variety of videos are included, there are drama movies, action movies, cartoon movies, TV shows, sports, and office camera video. Each trace includes three trace files, corresponding to the low, the medium, and the high quality levels of the same video. I use Q to denote the quality level of a video; three values of Q: 1, 2, and 3, are used to represent low, medium, and high quality respectively. Each trace file consists of ADU sizes of video frames, in bytes. These frames are captured at a rate of 25 frames/second. Therefore the ADU interarrival time for multimedia traffic is 0.04 second.

The adaptation of multimedia traffic is modelled as follows. The maximum D for streaming multimedia is assumed to be 10 seconds. The maximum D for interactive multimedia is assumed to be the same as that of the game traffic, which is 0.2 second. I still adapt D using Eq. 5.1 and Eq. 5.2. Because human eyes need to have multimedia frames played back at the rate of 25 frames/second, I assume that the ADU interarrival times cannot be adapted. For ADU sizes, I assume that an application can adapt by using a frame from a higher or a lower quality traces of the same video. In my experiments, when a multimedia session does adaptation, I assume that it adapts D first. If D has reached the maximum or minimum value, which indicates D cannot be adapted further, the session adapts Q.

The pseudo code for the adaptation of a game session is given in Table 5.6.

Table 5.6: Pseudo code for multimedia session adaptation in Case II

```
/* The function increasing the service requirements for a multimedia session*/

FUNCTION (MULTIMEDIA-Increase)
    BEGIN
        IF (D > minD)
            D = D * (1 − α); //Decreasing end-to-end deadline
            IF (D < minD);
                D = minD;
        ELSE      //D cannot be adapted
            IF (Q < 3)
                Q = Q + 1; //Increasing the quality (ADU sizes)
    END

/* The function decreasing the service requirements for a multimedia session*/

FUNCTION (MULTIMEDIA-Decrease)
    BEGIN
        IF (D < maxD)
            D = D * (1 + α); //Increasing end-to-end deadline
            IF (D > maxD);
                D = maxD;
        ELSE      //D cannot be adapted
            IF (Q > 1)
                Q = Q − 1; //Decreasing the quality (ADU sizes)
            ELSE      //Both D and Qcannot be adapted
                STOPPED      //The session is stopped
    END
```

5.3.2 Experiment Parameter and Result

To obtain the on-time performance in the last adaptation interval, in the subsequent experiments, I added the modeling of ACK in my simulation. It is assumed that no ACK is lost, and the end-to-end delay of ACK packet is uniformly distributed on the range of [35, 85] ms. This range is selected based on the longest end-to-end

propagation delay, which is about 32.8 ms in my network.

Two experiments are carried out to evaluate the performance gain when with application adaptation. In the first experiment, the number of sessions of a user is uniformly distributed on [10, 30], and the offered load on the bottleneck is 0.98 without user adaptation. In the second experiment, the number of sessions of a user is uniformly distributed on [10, 40], and the offered load on the bottleneck in this case is 1.27 without user adaptation. With these two levels, I can observe the adaptation behavior when the network load is heavy and when the network is overloaded respectively. In both experiments, among all sessions in the network, 10% of them are game sessions, 70% of them are interactive multimedia sessions, and 20% of them are streaming multimedia sessions. At the beginning of simulation, a multimedia session randomly chooses a video from the available 10 videos, 50% of multimedia sessions use trace files in the high quality, 30% of multimedia sessions use trace files in the medium quality, and 20% of multimedia sessions use trace files in the low quality. The parameter β is chosen as 80. The parameter d for initial D is chosen as 0.3 for all sessions, and the initial D is defined as the minimum D of the session. All sessions' interval budgets are 1. The adaptation and price update interval is 1 second long. The simulation duration is 150 seconds. Two values of *EptRate*: 0.7, 0.9, and two values of α: 0.4 and 0.8 are experimented.

The results are shown in Tables 5.7 and 5.8. I report both ADU on-time throughput and ADU on-time rate. The ADU on-time rate is defined as the fraction of total ADUs sent that are delivered on-time. I collect the aggregated on-time throughput and ADU on-time rate, which are across all traffic classes in the network. I also collect

the ADU on-time rate per user. In addition, I include the results for the user that had the lowest ADU on-time rate when without and with adaptation. I also include the offered load which is obtained from measurement on the bottleneck to show the average level of network load.

Table 5.7: Delay performance without and with adaptation, no. of sessions of a user: Uniform[10,30]

Adapting attributes	Aggregated ADU on-time rate	Lowest ADU on-time rate of a user	On-time ADU throughput	Offered load on bottleneck
Without adaptation	66.10%	14.09%	52088.28	0.9829
(α=0.4) ($EptRate$=0.9)	86.48%	72.51%	59253.75	0.6802
(α=0.8) ($EptRate$=0.9)	82.58%	63.50%	57260.54	0.7462
(α=0.4) ($EptRate$=0.7)	78.39%	48.87%	55520.99	0.7170
(α=0.8) ($EptRate$=0.7)	75.65%	48.84%	53721.35	0.7250

Table 5.8: Delay performance without and with adaptation, no. of sessions of a user: Uniform[10,40]

Adapting attributes	Aggregated ADU on-time rate	Lowest ADU on-time rate of a user	On-time ADU throughput	Offered load on bottleneck
Without adaptation	54.87%	4.68%	54395.65	1.2711
(α=0.4) ($EptRate$=0.9)	82.92%	62.45%	63376.93	0.4188
(α=0.8) ($EptRate$=0.9)	79.13%	61.44%	62000.24	0.5540
(α=0.4) ($EptRate$=0.7)	74.40%	52.71%	58779.41	0.4258
(α=0.8) ($EptRate$=0.7)	71.97%	49.69%	58117.97	0.5103

From the results, it can be observed that by way of application adaptation, both

the aggregated on-time performance and individual users' on-time performance can be significantly improved. In addition, it can be observed that, smaller value of adaptation parameter α results in lower offered load and slightly better ADU on-time performance, and a higher value of *EptRate* results in a slightly lower level of offered load but much improved on-time performance.

I also plot the price dynamics in this mixed traffic case, comparing the channel delay price on the bottleneck when without and with adaptation. The results are shown in Figures 5.3 and 5.4. It can be observed that the channel delay price can

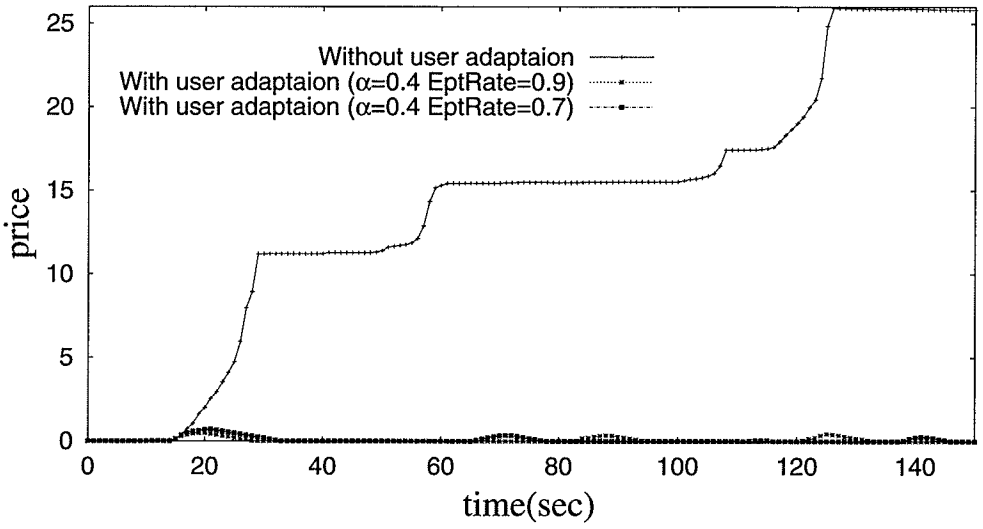

Figure 5.3: Channel delay price without and with adapatation, Case II, no. of sessions of a user: Uniform[10,30].

be well controlled when with user adaptation; in contrast, the price keeps increasing when without user adaptation.

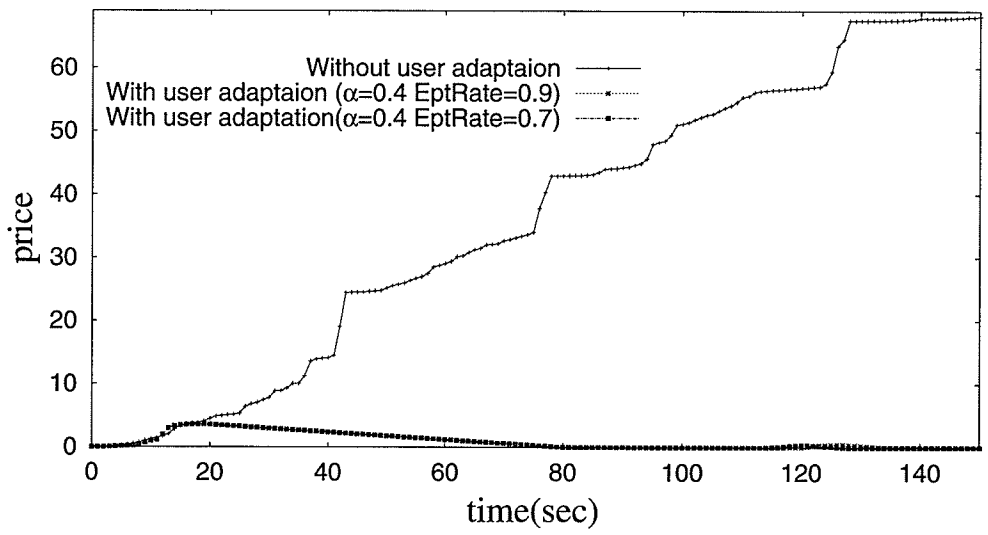

Figure 5.4: Channel delay price without and with adapatation, Case II, no. of sessions of a user: Uniform[10,40].

5.4 Discussion

In this chapter, I studied the effect of application adaptation for two types of real-time traffic in deadline-based networks. Such an application adaptation is enabled by my developed pricing and charging scheme. I assume that price-sensitive users with limited budget constraints adapt their traffic requirements based on the charges received earlier. I have shown through simulation that the delay performance of the entire network and of individual users can be improved as the result of such adaptation.

Chapter 6

Conclusion

6.1 Summary and Contributions

This thesis addresses the pricing issue in deadline-based networks. Differing from previous studies on deadline-based networks, in which only the aggregated performance of all traffic that is transmitted over the network was studied, in this research, I study the delay performance of individual users in deadline-based network. I found that the delay performance of a user largely depends on the deadline urgency level of his/her traffic, as well as the deadline urgency of other traffic on the network, and also on the network load level. With deadline-based scheduling at each channel, a packet with a more urgent deadline is serviced first, an ADU with a more urgent deadline experiences less queuing delay thus lower response time. In addition, the packets on a channel that is lightly loaded experience less queuing delay thus better response time performance. Because packets with more urgent deadline will be served with a higher priority in deadline-based networks, a greedy user can obtain a better service

in terms of experiencing lower queuing delay, by assigning more urgent deadlines. A greedy user may also generate a lot of traffic to downgrade the aggregated network performance.

To prevent users' greedy behavior from assigning arbitrarily urgent deadlines and to aid in network load control, in this thesis, I developed a novel delay pricing and charging scheme in deadline-based networks to support real-time data delivery. In my scheme, I make use of the concept of competitive market and determine a channel delay price based on a delay demand and a delay supply at each channel. The delay demand is derived from packets' deadlines. The delay supply is derived from the response times of the packets that are serviced, which in turn depend on the link capacity and the amount of traffic. Therefore, the channel delay price reflects the traffic's deadline urgency level and the network load on the channel. In my charging scheme, a user's charge depends on the amount of his/her traffic and the delay performance of its traffic. A user with more traffic is charged more, and a user whose traffic experiences less queuing delay will be charged more. Simulation results show that my scheme can provide differential charges in terms of different delay performance. Therefore, my scheme can be used to urge users to submit appropriate service requirements for their traffic.

In reality, users are price-sensitive, they may adapt their traffic requirements in response to the price. In my study, I introduced a user adaptation method. In this method, time is slotted, a traffic attribute, which is related to the deadline or the amount of traffic, can be adapted by the traffic sender according to the charge that is received by the same user in a previous time interval and according to his/her

budget. Simulation results show that my scheme can easily enable user adaptation, hence improve the delay performance and aid in network load control.

Compared with other studies on network service differentiation and pricing, in which pricing of bandwidth is concerned, in this thesis, I have introduced the novel delay pricing concept. This is made available by the deadline-based framework in which each packet carries its delay requirement. The demand can easily be derived from this deadline information.

6.2 Future Work

There are a number of interesting future work of this study. This section gives a brief outline of them.

- Further improvement of my pricing scheme.

  I have not considered dropped packets due to buffer overflow in my scheme. Therefore, my scheme cannot control congestion absolutely. For example, when the traffic deadlines are very loose, the delay demand may still be lower than the delay supply for those packets that are not dropped, but indeed many packets are dropped. One solution to this would be to increase the channel delay price when the number of dropped packets increases.

- Charge estimation scheme.

  After introducing the delay pricing scheme, network prices vary over time. According to the channel delay prices, a user may ask the network to provide an

estimation of the charges for his/her traffic, based on his/her requirements before traffic submission. A charge estimation scheme can be introduced in the future work.

- User adaptation based on utility functions.

  I introduce a linear adaptation method for all types of traffic in my research. In reality, different applications may have different utility functions. Therefore, they may adapt their service requirements in different ways. The investigation of strategies to maximize the user utility can be researched in the future.

Appendix A

Routing Tables of the 13-Node Network Used

The routing table of Node i includes the next hop information and the number of hops to all other destination nodes in the network from Node i; one row per destination node.

Node 0		
Dest.	Next	H
1	1	1
2	2	1
3	3	1
4	2	2
5	3	2
6	3	2
7	1	2
8	3	2
9	2	3
10	3	3
11	3	3
12	1	3

Node 1		
Dest.	Next	H
0	0	1
2	2	1
3	0	2
4	2	2
5	0	3
6	2	3
7	7	1
8	0	3
9	7	2
10	7	4
11	7	3
12	7	2

Node 2		
Dest.	Next	H
0	0	1
1	1	1
3	0	2
4	4	1
5	4	3
6	4	2
7	1	2
8	0	3
9	4	2
10	0	4
11	1	4
12	1	3

Node 3		
Dest.	Next	H
0	0	0
2	2	2
1	1	1
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	11	11
12	12	12

Node 4		
Dest.	Next	H
0	0	0
1	1	1
2	2	2
3	3	3
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	11	11
12	12	12

Node 5		
Dest.	Next	H
0	0	1
1	1	1
2	0	2
3	4	1
4	4	3
6	4	2
7	1	2
8	0	3
9	4	2
10	0	4
11	1	4
12	1	3

Node 6		
Dest.	Next	H
0	3	2
1	3	3
2	4	2
3	3	1
4	4	1
5	5	1
7	9	2
8	8	1
9	9	1
10	5	2
11	8	2
12	9	2

Node 7		
Dest.	Next	H
0	1	2
1	1	1
2	1	2
3	1	3
4	9	2
5	9	3
6	9	2
8	12	3
9	9	1
10	12	3
11	12	2
12	12	1

Node 8		
Dest.	Next	H
0	3	2
1	3	3
2	6	3
3	3	1
4	6	2
5	5	1
6	6	1
7	11	3
9	6	2
10	10	1
11	11	1
12	11	2

Node 9		
Dest.	Next	H
0	4	3
1	7	2
2	4	2
3	6	2
4	4	1
5	6	2
6	6	1
7	7	1
8	6	2
10	12	3
11	12	2
12	12	1

Node 10		
Dest.	Next	H
0	5	3
1	11	4
2	8	4
3	5	2
4	5	3
5	5	1
6	8	2
7	11	3
8	8	1
9	8	3
11	11	1
12	11	2

Node 11		
Dest.	Next	H
0	8	3
1	12	3
2	8	4
3	8	2
4	12	3
5	10	2
6	8	2
7	12	2
8	8	1
9	12	2
10	10	1
12	12	1

Node 12		
Dest.	Next	H
0	7	3
1	7	2
2	7	3
3	9	3
4	9	2
5	11	3
6	9	2
7	7	1
8	11	2
9	9	1
10	11	2
11	11	1

Appendix B

Simulator Pseudo code

I give the pseudo code of my simulator.

MAIN:
/*Initialization */
Aggregated-SentADU-No=0;
Aggregated-on-time-ADU-No=0;
Schedule the first Price-Change-Event;
FOR (all Traffic Classes) **DO**
FOR (all Users of the Traffic Class) **DO**
FOR (all Sessions of the User) **DO**
Schedule and ADU-Arrival-Event;
$Session.SentADU\text{-}No = 0$;
$Session.on\text{-}time\text{-}ADU\text{-}No = 0$;
$Session.charge = 0$;
$Session.N = 0$; //for user adaptation in Case II in Chapter 5
$Session.N_t = 0$; //for user adaptation in Case II in Chapter 5
$Session.IC_s = 0$;//for user adaptation in Chapter 5
$User.SentADU\text{-}No = 0$;
$User.on\text{-}time\text{-}ADU\text{-}No = 0$;
$User.charge = 0$;
$User.IC_u = 0$; //for user adaptation in Case II in Chapter 5
$TrafficClass.SentADU\text{-}No = 0$;
$TrafficClass.on\text{-}time\text{-}ADU\text{-}No = 0$;
$TrafficClass.charge = 0$;
FOR (;$clock < Simulation - TIME$;) **DO**
Exact an event from the Event-List
Deal with the event; //refer to "Event Part"
/*Performance Statistics */
FOR (all Traffic Classes) **DO**
FOR (all Users of the Traffic Class) **DO**
FOR (all Sessions of the User) **DO**
Print out Session's Charge=$Session.charge$;
Print out Session's ADU on-time rate=$\frac{Session.on-time-ADU-No}{Session.SentADU-No}$;
$User.SentADU\text{-}No$+=$Session.SentADU\text{-}No$;
$User.on\text{-}time\text{-}ADU\text{-}No$+=$Session.on\text{-}time\text{-}ADU\text{-}No$;
$User.charge$+ = $Session.charge$;
Print out User's Charge =$User.charge$;
Print out User's ADU on-time rate=$\frac{User.on-time-ADU-No}{User.SentADU-No}$;
$TrafficClass.SentADU\text{-}No$+=$User.SentADU\text{-}No$;
$TrafficClass.on\text{-}time\text{-}ADU\text{-}No$+=$User.on\text{-}time\text{-}ADU\text{-}No$;
$TrafficClass.charge = User.charge$;
Print out TrafficClass's Charge =$TrafficClass.charge$;
Print out TrafficClass's ADU on-time rate=$\frac{TrafficClass.on-time-ADU-No}{TrafficClass.SentADU-No}$;
$Aggregated\text{-}on\text{-}time\text{-}ADU\text{-}No$+=$TrafficClass.on\text{-}time\text{-}ADU\text{-}No$;
$Aggregated\text{-}SentADU\text{-}No$+=$TrafficClass.SentADU\text{-}No$;
Print out the Aggregated ADU on-time rate =$\frac{Aggregated-on-time-ADU-No}{Aggregated-SentADU-No}$;
Print out the Aggregated ADU on-time throughput=$\frac{Aggregated-on-time-ADU-No}{Simulation-TIME}$;

```
/*————————————————Event Part ————————————————*/
ADU-Arrival-Event:
  Segment ADU to PCK(packet);
  Schedule a PCK-Arrival-Event;
  Session.SentADU-No++;
  Session.N++;// No. of ADUs sent per adaptation interval
  Schedule the next ADU-Arrival-Event in the same session;

PCK-Arrival-Event:
  CASE Arrival destination:
      ADU.charge+ = PCK.charge;
      IF (All PCKs of the ADU are arrived the destination)
          AND (ADU is on time)
          Session.on-time-ADU-No++;
          IF(Case I of Chapter 5)
              Session.N_t++; //No. of on-time ADUs per adaptation interval
              Session.IC_s+ = ADU'scharge;
          IF(Case II of Chapter 5)
              Schedule an ACK-Event at sender;
  CASE Not arrival destination:
      IF (Buffer is enough to accept the new PCK) AND (PCK is not late)
          Calculate T/H;
          Put PCK into the queue;

PCK-Departure-Event:
  channel.busy = 0; //channel is set to idle
  According to propagation delay, schedule a
      PCK-Arrival-Event at the next channel;
  Response-Time=Queu-Delay+Trans-Delay+Propa-Delay;
  newCharge = channel.price/Response-Time;
  PCK.charge+ = newCharge;
  channel.S^T+ = Response-Time; // For calculating the delay supply

PCK-Service-Event:
  channel.busy = 1; //channel is set to busy
  According to transmission delay, schedule a PCK-Departure-Event;
  channel.D^T+ = PCK.T/H; // FOR calculating the delay demand

ACK-Event:
  Session.N_t++;
  Session.IC_s+ = ADU'scharge;
```

/*——————————Event Part ——————————*/

Price-Change-Event:
FOR (all channels) **DO**
$S = 1/S^T$; //calculate the delay supply
$D = 1/D^T$; //calculate the delay demand
$Delay\text{-}Price = Min(Delay\text{-}Price * \sigma * \frac{D-S}{S}, Lowest\text{-}Price)$
$S^T = 0$;
$D^T = 0$;
Schedule Next Price-Change-Event;

User-Adapatation-Event:
FOR (all users) **DO**
FUNCTION(User-Adaptation); //do user adaptation
FOR (all session of the user) **DO**
$Session.N = 0$;
$Session.N_t = 0$;
$Session.IC_s = 0$;
$User.IC_u = 0$;

Bibliography

[1] http://trace.eas.asu.edu/TRACE/ltvt.html.

[2] Loretta Anania and Richard J. Solomon. Flat-the minimalist price. In *Internet Economics*, pages 91–118, Cambridge, MA, USA, 1997. MIT Press.

[3] Jerry Banks, Barry Nelson, and John Carson. *Discrete-Event System Simulation.* Prentice-Hall, Upper Saddle River, N.J., Third edition, 2000.

[4] Florian Baumgartner, Torsten Braun, and Pascal Habegger. Differentiated services: a new approach for quality of service in the Internet. In *HPN*, pages 255–273, 1998.

[5] Dimitri P. Bertsekas and Robert Gallager. *Data Networks.* Prentice Hall, second edition, 1992.

[6] Ron Cocchi, Deborah Estrin, Scott Shenker, and Lixia Zhang. A study of priority pricing in multiple service class networks. In *SIGCOMM '91: Proceedings of the conference on Communications architecture & protocols*, pages 123–130, New York, NY, USA, 1991. ACM Press. http://doi.acm.org/10.1145/115992.116005.

[7] Ron Cocchi, Scott Shenker, Deborah Estrin, and Lixia Zhang. Pricing in computer networks: motivation, formulation, and example. *IEEE/ACM Trans. Netw.*, 1(6):614–627, 1993. http://dx.doi.org/10.1109/90.266050.

[8] ATM Forum Technical Committee. Traffic Management Specification. 1996.

[9] David D.Clark. Internet cost allocation and pricing. In *Internet Economics*, pages 215–252, Cambridge, MA, USA, 1997. MIT Press.

[10] Robert Denda, Albert Banchs, and Wolfgang Effelsberg. The fairness challenge in computer networks. In *Proceedings of the First COST 263 International Workshop on Quality of Future Internet Services*, pages 208–220, London, UK, 2000. Springer-Verlag.

[11] Internet end-to-end interest mailing list. Queue size of routers. http://www.postel.org/pipermail/end2end-interest/2003-January/.

[12] Matthias Falkner, Michael Devetsikiotis, and Ioannis Lambadaris. An overview of pricing concepts for broadband IP networks. *IEEE Communications Surveys and Tutorials*, 3(2), 2000. http://www.comsoc.org/livepubs/surveys/public/2q00issue/falkner.html.

[13] Peter C. Fishburn and Andrew M. Odlyzko. Dynamic behavior of differential pricing and quality of service options for the Internet. In *Proc. First Intern. Conf. on Information and Computation Economies (ICE-98)*, pages 128–139. ACM Press, 1998. http://www.dtc.umn.edu/ odlyzko/doc/networks.html.

[14] Errin W. Fulp and Douglas S. Reeves. Distributed network flow control based

on dynamic competitive markets. In *Proceedings of the IEEE International Conference on Network Protocols*, pages 119–128, 1998.

[15] Richard J. Gibbens and Frank P. Kelly. Resource pricing and the evolution of congestion control. 1998. http://www.statslab.cam.ac.uk/ frank/evol.html.

[16] Alok Gupta, Dale O.Stahl, and Andrew B. Whinston. Priority pricing of integrated services networks. In *Internet Economics*, pages 323–352, Cambridge, MA, USA, 1997. MIT Press.

[17] Shivkumar Kalyanaraman and T. Ravichandran. Dynamic capacity contracting: a framework for picing the differentiated services Internet. In *1st Int'I. Conf. Information and Computation Economics*, May 1998. Submitted.

[18] Frank Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8(1):33–37, Jan. 1997.

[19] Frank P. Kelly, Aman K. Maulloo, and David K. H. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998. http://citeseer.csail.mit.edu/kelly98rate.html.

[20] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet.* Second edition, 2003.

[21] Jani Lakkakorpi, Andreas Heiner, and Jussi Ruutu. Measurement and characterization of Internet gaming traffic. Research Seminar on Networking, Helsinki University of Technology, Networking Laboratory, Espoo, Finland, February 2002.

[22] Tanja Lang, Philip Branch, and Grenville Armitage. A synthetic traffic model for Quake3. In *ACE '04: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 233–238, New York, NY, USA, 2004. ACM Press.

[23] Yanni E. Liu and Johnny W. Wong. Deadline based channel scheduling. In *Proceedings of the IEEE Global Telecommunications Conference (Globecom'01)*, pages 2358–2362, San Antonio, Texas, November 2001. IEEE Press.

[24] Yanni Ellen Liu. *Deadline-based network resource management.* PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, 2003.

[25] Yanni Ellen Liu and Johnny W. Wong. Admission control in deadline-based network resource management. In *Proceedings of the 23rd IEEE International Performance, Computing, and Communications Conference (IPCCC'2004)*, pages 95–102, Phoenix, Arizona, April 2004. IEEE Press.

[26] Jeffrey K. MacKie-Mason, Liam Murphy, and John Murphy. Responsive pricing in the Internet. In *Internet Economics*, pages 279–303. MIT Press, Cambridge, MA, USA, 1997.

[27] Jeffrey K. MacKie-Mason, Liam Murphy, and John Murphy. The role of responsive pricing in the Internet. In J. Bailey and L. McKnight, editors, *Internet Economics*, pages 279–303. MIT Press, 1997. available from URL http://citeseer.ist.psu.edu/mackie-mason95role.html.

[28] Jeffrey K. MacKie-Mason and Hal R. Varian. Pricing congestible network re-

sources. *IEEE Journal on Selected Areas in Communications*, 13:1141–1149, Sept. 1995.

[29] Jeffrey K. Mackie-Mason and Hal R. Varian. Pricing the Internet. In *Public Access to the Internet*, pages 269–314, Cambridge, MA, USA, 1995. http://www-personal.umich.edu/ jmm/papers.html.

[30] John Murphy and Liam Murphy. Bandwidth allocation by pricing in ATM networks. In *Broadband Communications*, pages 333–351, 1994. http://citeseer.ist.psu.edu/murphy94bandwidth.html.

[31] John Murphy, Liam Murphy, and Edward C. Posner. Distributed pricing for embedded ATM networks. In *Proc. International Teletraffic Congress ITC-14*, pages 1053–1063, North Holland, June 1994. http://www.eeng.dcu.ie/ murphyj/pub-2003/Publications.html.

[32] Liam Murphy and John Murphy. Feedback and pricing in ATM networks. In *Modelling and Evaluation of ATM Networks*, pages 197–212, 1995. http://citeseer.ist.psu.edu/murphy95feedback.html.

[33] Liam Murphy and John Murphy. Pricing for ATM network efficiency. In *Proc. 3rd International Conference on Telecommunication Systems, Modelling and Analysis*, pages 349–356, Nashville, USA, March 1995. http://www.eeng.dcu.ie/ murphyj/publ/publ.html.

[34] Liam Murphy, John Murphy, and Jeffrey K. MacKie-Mason. Feedback and efficiency in ATM networks. In *Proc. of the 1996 Intl Conf.*

on Communications (ICC'96), pages 1045–1049, Dallas, TX, June 1996. http://citeseer.ist.psu.edu/murphy96feedback.html.

[35] Walter Nicholson. *Microeconomic Theory, Basic Principles and Extensions.* The Dryden Press, 1989.

[36] Andrew Odlyzko. A modest proposal for preventing Internet congestion. Technical Report 97-68, 17, 1997. http://citeseer.ist.psu.edu/odlyzko97modest.html.

[37] Scott Shenker, David D. Clark, Deborah Estrin, and S. Herzog. Pricing in computer networks: reshaping the research agenda. In *SIGCOMM Comput. Commun. Rev.*, volume 26, pages 19–43, New York, NY, USA, 1996. ACM Press.

[38] Hal R. Varian. *Microeconomic Analysis.* W.W.Norton and company, third edition, 1993.

[39] Leon Walras. *Elements of Pure Economics.* Richard D.Irwin, 1954. trans. W.Jaffé.

[40] Xin Wang and Henning Schulzrinne. RNAP: a resource negotiation and pricing protocol. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'99)*, pages 77–93, Basking Ridge, New Jersey, Jun. 1999.

[41] Xin Wang and Henning Schulzrinne. An integrated resource negotiation, pricing, and QoS adaptation framework for multimedia applications. *IEEE Journal on Selected Areas in Communications (JSAC)*, 2000. Special Issue on Internet QoS.

[42] Xin Wang and Henning Schulzrinne. Performance study of congestion price based adaptive service. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'00)*, pages 1–10, Chapel Hill, NC., Jun. 2000.

[43] Xin Wang and Henning Schulzrinne. Pricing network resources for adaptive applications in a differentiated services network. In *Proceeding of INFOCOM'2001*, Anchorage, Alaska, Apr. 2001.

[44] Errin W.Fulp and Douglas S. Reeves. The fairness and utility of pricing network resources using competitive markets. *Technical report, North Carolina State University*, 2002. http://citeseer.ist.psu.edu/fulp00fairness.html.

[45] Johnny W. Wong and Yanni E. Liu. Deadline based network resource management. In *Proceedings of the Ninth International Conference on Computer Communications and Networks (ICCCN'00)*, pages 264–268, Las Vegas, Nevada, October 2000. IEEE Press.