ARCHITECTURES FOR CONGESTION-SENSITIVE PRICING OF NETWORK SERVICES

By

Murat Yuksel

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Computer Science

Approved by the
Examining Committee:

Dr. S. Kalyanaraman, Thesis Adviser

Dr. T. Ravichandran, Member

Dr. B. K. Szymanski, Member

Dr. B. Sikdar, Member

Rensselaer Polytechnic Institute
Troy, New York

July 2002
(For Graduation August 2002)

ARCHITECTURES FOR CONGESTION-SENSITIVE PRICING OF NETWORK SERVICES

By

Murat Yuksel

An Abstract of a Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Computer Science

The original of the complete thesis is on file
in the Rensselaer Polytechnic Institute Library

Examining Committee:

Dr. S. Kalyanaraman, Thesis Adviser
Dr. T. Ravichandran, Member
Dr. B. K. Szymanski, Member
Dr. B. Sikdar, Member

Rensselaer Polytechnic Institute
Troy, New York

July 2002
(For Graduation August 2002)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

First, I would like to thank my advisor Assoc. Prof. Shivkumar Kalyanaraman, who were very understanding to me throughout my doctoral study. His understanding and patience was unforgettable when I was not productive. He was inspiring and helpful in all aspects of the thesis. This thesis would not happen without his support and motivation.

I cannot forget Assist. Prof. Biplab Sikdar's friendship during my graduate study. When I was a newcomer, he helped me with all my stupid questions in the area of networking. I will not forget his dedication to work and nice talks we had in the Networks Lab.

I would like to thank Prof. Boleslaw Szymanski who was helpful throughout my graduate study. His ideas were always inspiring and his suggestions made the thesis more valuable.

I also would like to thank Assist. Prof. Thiagarajan Ravichandran for serving in the committee and for discussions in the pricing project. Also, thanks due to Assist. Prof. Aparna Gupta for her valuable discussions and feedbacks about our research.

I would like to thank all my colleagues in the pricing project. Ranjeeta, Gurjeet, Ritesh, Anuj. We had a lot of fruitful discussions together, and I received a lot of feedback about the thesis from them, especially Anuj. Also, thanks due to all members of Networks Lab for their nice friendship and feedbacks about my research.

I would like to thank Turkish Ministry of Education for providing initial financial support to my graduate study.

Finally, I would like to thank my wife Bahar for her support and her unbelievable dedication to me. Also, thanks go to my children Abdullah Yusuf and Meryem for all the hard and nice times they gave.. I hope they will grow up and read this one day. Now, this is the time to thank my parents. I have to thank them for listening to my cries and taking care of me when I was a baby.

ABSTRACT

Several adaptive pricing proposals have been made in the last decade for the Internet. Usually, however, those proposals studied optimal strategies and did not focus on implementation issues. We address implementation issues for adaptive pricing over a single differentiated-services (diff-serv) domain. We propose a new adaptive pricing framework Distributed Dynamic Capacity Contracting (Distributed-DCC), which is deployable over diff-serv architecture of the Internet. Essence of Distributed-DCC is to employ edge-to-edge coordination along with techniques for estimation of user willingness-to-pay (e.g. budget estimation). Particularly, congestion can be detected on an edge-to-edge basis, which enables congestion-sensitive pricing at the edges.

Distributed-DCC is able to provide a range of weighted fairness types (i.e. from max-min to proportional) in rate allocation by using pricing as a tool. The provider can tune a parameter, fairness coefficient, to change fairness of the framework.

To illustrate possibility of congestion-sensitive pricing in the framework, we develop a congestion-sensitive pricing scheme, Edge-to-Edge Pricing (EEP), within the framework. We derive optimal prices for EEP and investigate effects of user's elasticity on these optimal prices.

We also investigate congestion control by pricing, especially in terms of time-scale. We illustrate that control of congestion by pricing requires very small time-scale pricing (i.e. frequent updates to prices), which complicates human involvement into negotiations. To solve this time-scale problem, we propose two pricing architectures: Pricing for Congestion Control (PFCC) and Pricing over Congestion Control (POCC). PFCC uses small time-scale pricing directly for controlling congestion and employs end-placed software/hardware agents which take inputs from human user at large time-scale while negotiating with the provider at small time-scale on behalf of the user. POCC uses an underlying edge-to-edge congestion control mechanism by overlaying pricing on top of it. This way, POCC controls congestion at small time-scales while allowing pricing at time-scales large enough for human involvement. We also illustrate how to adapt Distributed-DCC to these architectures.

CHAPTER 1
INTRODUCTION

1.1 Importance of Network Pricing

As the Internet grows in size and heterogeneity, the need increases for new economic models and inter-domain services as well as new tools and techniques for managing congestion. Technical tools such as differentiated services [27], TCP/IP improvements like SACK [56], over-provisioned ISP cores have been developed and deployed in the Internet. However, economic tools, such as responsive pricing, auctioning of bandwidth, proper accounting, have been proposed [19, 33, 42, 50] but not deployed. Such economic tools are necessary building blocks in order to build a superior economic model for the Internet and inter-domain services. Especially, a flexible pricing framework, which can provide environment for a variety of pricing techniques (i.e. static, dynamic, or hybrid), is a crucial building block.

A major impediment in the deployment process has been the minimal "best-effort" service model of the IP, which does not provide a standard mechanism to specify packet-forwarding behavior other than the best-effort service. In other words, several of the proposed schemes have remained in the theoretical domain due to lack of models for implementing them using IP. This scenario has recently changed as the Internet Engineering Task Force (IETF) standardized two approaches to support service differentiation: Integrated Services (int-serv) [12] and Differentiated Services (diff-serv) [27]. Though the two approaches can coexist and inter-operate, it is expected that the latter (diff-serv) or its variants will be the choice of service providers. Given this backdrop, it is important to develop better and flexible (i.e. that can provide environment for both static and dynamic pricing) pricing architectures that can work on diff-serv.

Also, one of the major problems in the Internet is controlling congestion. Currently, the dominant congestion control algorithm in the Internet is TCP[3, 39]'s congestion control algorithm. Several improvements (RED [29], TCP-NewReno [38, 28], Forward ACK [55], TCP-Vegas [13]) have been made to TCP's congestion control algorithm. These congestion control schemes reduce the *sending rate*, but cannot control the increasing *user demand*. For example, during a congestion epoch, a user transferring a large file generally does not stop the transfer, even though underlying TCP reduces sending rate at the background. So, reducing the sending rate is not enough to make sure that the service is good, since capacity of the Internet generally is just not enough to satisfy the active user demand. To provide good service in such cases, user demand must be reduced by causing some users to stop using the network. So, although these congestion control schemes guarantee stability of the network, they have no ability to reduce the user demand. As the user demand and the number of active users increase, the service provided to each user gets worse and worse. At some point the service gets so bad that, the users who want better service stop using the network although they would be willing to pay more for a better service. The fairer and more reasonable way to resolve such a severe congestion epoch is to increase the price of the network service and cause the users with lower willingness-to-pay to reduce their network usage. This way the provider will gain more revenue, and moreover the social utility (welfare) will be more.

The next major problem in the Internet is fairness among the users. One obvious fairness problem is the fact that UDP [73] traffic beats down TCP traffic because it does not have any congestion control algorithm. At the time of congestion, TCP backs off while UDP does not. This causes unfair treatment of user applications using TCP, relative to the ones using UDP. So there is a need to have control over traffic regardless of the underlying transport protocol.

Again as a fairness issue, some applications require better treatment because of their internal requirements. For example, some applications are very sensitive to loss of their packets, while some other application is very sensitive to delay of their packets. All these requirements lead to studies to differentiate services given to the user applications. As we stated earlier, two important standardization happened to accommodate needs of different types of user applications: Integrated Services [12], Differentiated Services [27].

Currently, however, users do not have opportunity to express the value of the application they are running. All the user applications are treated equally (other than strict differentiation provided by diff-serv, RSVP, etc.) regardless of their value to the user. All the users are charged flat-rate (other than strict differences, e.g. cable users and dial-up users) and are provided approximately equal network capacity. This strictly limits the amount of utility the user can get out of the network service, especially that user is willing to pay significantly more for higher capacity.

So, although several improvements have been made to congestion control and service differentiation techniques, the Internet is still missing three important features especially at the time of severe congestion epochs:

- Mechanisms to reduce and control user demand
- Fairness
- Flexible economic models to increase providers' revenue

One proposed method of controlling network conditions is to use pricing. By adaptive pricing (or static pricing if user demand model is known), it is possible to provide efficient techniques to resolve the above listed issues. Since user demand model is unknown and Internet traffic is so bursty, static pricing is not a proper solution.

As we stated earlier in this section, it is possible to increase the service price during congestion epochs and reduce it otherwise. This way pricing can control the user demand adaptively. If the Internet is left as it is without an adaptive pricing architecture, it will always be possible to get into phases where the network service gets terribly bad. However, when adaptive pricing is employed, increased prices will make sure that the users with the higher willingness-to-pay will use the network.

Also, adaptive pricing will make sure that the users will express the value of their applications. Without knowing the value of the applications to their owners (as it is currently for the Internet), it is not possible to maximize or at least increase total user utility. In terms of fairness, it will be possible to give more network capacity to those users who have more willingness-to-pay for the network service.

As another fairness issue, the users who are causing more cost to the network (congestion is also regarded as cost) than the others must be charged more. However, there is no way of doing that by currently used flat-rate pricing. We can give a more solid example as follows: Consider two users A and B in New York. Suppose both of these users are talking to their friends over the Internet. User A's friend is in Los Angeles whereas user B's friend is in Phoenix. So, these two users are basically running a voice-over-IP application from New York to different locations. It is perfectly possible that the route between New York and Los Angeles is passing through more congested areas than the route between New York and Phoenix. Currently, both of these users are being charged the same amount. However, a fairer case would be to charge the user B more than the user A. Moreover, this situation can change dynamically over time. At some time later, the route between New York and Phoenix may become more congested than the route between New York and Los Angeles. So, this requires the charging scheme to be adaptive.

In the next section we lay out the challenges for proper pricing of network services, and then in Section 1.3 we present our approaches to these challenges. Finally in Section 1.6, we outline the thesis.

1.2 Challenges of Network Pricing

Most of the challenges for adaptive pricing of network services are related to implementation and structural obstacles. Several adaptive pricing schemes have already been proposed, but none of them were able to provide an easy implementation. Implementation difficulties for adaptive pricing are because of the high overhead caused by adaptiveness of the pricing scheme.

One important problem is that adaptive pricing needs to update price dynamically over time. Frequency of these price updates are expected to be more when the network gets congested or traffic is so bursty. However, *each price update causes extra overhead in terms of both accounting and control traffic.* When the pricing scheme updates the prices less frequently, it cannot adapt to the changing network conditions and loose control over the user demand and network congestion. So, there is a trade-off between being more adaptive and having less overhead.

Another important implementation difficulty for adaptive pricing is the fact that *users desire to know the price before using the service.* Since adaptive pricing aims to change the price dynamically over time, it becomes necessary to inform the user about the new price. This introduces extra control traffic overhead to the network, which is going to be proportional to the number of active users.

We now focus on challenges to provide three major purposes of an adaptive pricing framework: better congestion control, better fairness, and more revenue.

1.2.1 Congestion Control Challenges

As we stated earlier in Section 1.1 one of the purposes of adaptive pricing is to directly control or help controlling the network congestion by controlling user demand through adaptive prices. In reality, time-scale of congestion control is much smaller than time-scale of pricing. As the time-scale of pricing gets smaller, pricing will directly contribute to congestion control more.

Adaptive pricing can help congestion control, but there are two major problems regarding this issue. One is that *the pricing scheme must be able to detect congestion level and communicate this congestion information to corresponding nodes (the nodes that need current congestion information to update the price) very quickly.* This requires robust and efficient congestion detection techniques that do not cause a lot of implementation overhead.

The second problem is that *the pricing scheme must update the price quickly in order to have more control over congestion.* However, frequent price updates disturb users and also cause more overhead. So, the pricing scheme must balance that trade-off.

1.2.2 Fairness Challenges

The pricing scheme should ensure the fairness of the network. For a network, the most fair case is the case where the total user utility is maximized. Since each user application has a different value to its owner, total user utility cannot be maximized by always giving equal network capacity to the users. In general, however, it has been argued that the network should give equal network capacity to all users to maintain fairness. For example, TCP's congestion control algorithm gives the same share (proved in [17]) of the network capacity to the users. So, the only fairness criteria for congestion control algorithms has been equal allocation of network capacity to the users. This has been the case, because the congestion control schemes do not have any controlled way of getting user's willingness-to-pay, i.e. value of the application to the user. This gap can only be filled by having an adaptive pricing scheme that can estimate user's willingness-to-pay.

We now give a counter example on the definition of fairness. Consider two users A and B, whose traffic are going through the same bottleneck with a capacity of c. Suppose that user B's application is r times more sensitive to bandwidth than user A's application. So, user B is willing to pay r times more than user A. In other words, user B's budget for the network service per unit time is $b_B = rb_A$, where b_A is user A's budget for the network service per unit time. Our question is what is the most fair allocation of c to these two customers?

Since our aim is to maximize the total user utility, we first need to represent user A and B's utility functions. Several studies showed that marginal user utility decreases as the amount of bandwidth given to the user increases, similar to "diminishing rate of returns" phenomenon in economics. This implies that the utility function should be a concave function. Although there are several different concave functions, $u(x) = logx$ has been commonly used in the area where x is the bandwidth given to the user. A more general form would be to include a multiplicative factor in order to represent user's sensitivity to bandwidth. [58, 79] So, we represent the utility functions of users A and B as follows:

$$u_A(x) = b_A logx$$

$$u_B(x) = b_B logx$$

where x is the bandwidth given to the user, b_A and b_B are sensitivity to bandwidth for users A and B respectively. Let x_A and x_B be amount of capacities given to users A and B correspondingly. Since the total capacity given to the two users can at most sum up to c, we can write that $x_B = c - x_A$. The total user utility will be:

$$U = u_A(x_A) + u_B(c - x_A)$$

To find the x_A value that maximizes U, we differentiate U with respect to x_A:

$$\frac{dU}{dx_A} = \frac{b_A}{x_A} + \frac{-rb_A}{c - x_A}$$
$$\frac{dU}{dx_A} = b_A \frac{c - (r+1)x_A}{x_A(c - x_A)}$$

After solving $\frac{dU}{dx_A} = 0$, we get the solution as follows:

$$x_A = \frac{1}{r+1}c$$

$$x_B = \frac{r}{r+1}c$$

So, this means that user B must be given r times more capacity than user A in order to maximize the total user utility. Notice that the ratio of x_A/c is the same as the ratio of $b_A/(b_A + b_B)$. This implies that each user must be given rate in proportion to his/her willingness-to-pay relative to willingness-to-pay of all users. We prove this property for the general case of multiple users in Appendix A.

Notice that this above rate allocation is very much dependent on knowing each user's willingness-to-pay, in other words sensitivity to bandwidth. Congestion control schemes do not have any way of determining this, whereas an adaptive pricing scheme has by changing the price dynamically.

To summarize, an adaptive pricing scheme can provide better fairness than congestion control schemes, since it can determine user's willingness-to-pay. With the knowledge of users's willingness-to-pay, *the pricing scheme must make sure that the users causing the same amount of congestion costs* [1] *are being priced equally.* This is hard to do in a wide area network because of delays, since price commu-

[1]In economics, three main type of costs, [23, 80], are defined: *sunk, variable, marginal.* For a network, sunk costs are the costs of physically building the network in order to make it operational. Variable cost of transferring a packet is the cost of allocating enough capacity for the transmission of that packet. Finally, marginal cost of a packet is amount of disturbance that packet causes on the other packets. In a network, disturbance is basically the congestion. Congestion cost of a packet (or a user) refers to that packet's (or that user's) contribution to the congestion.
In pricing, sunk costs and variable costs are generally recovered by fixed prices, i.e. flat-rate pricing or usage-based pricing. However, since marginal costs vary, dynamic pricing based on *marginal cost-prices* is needed for recovering them. Notice that we mainly focus on congestion pricing.

nication among the nodes of a wide area network will have to include delays. So, finding the optimum price (that maximizes total user utility) and communicating it to several nodes of the provider's network is a challenge for an adaptive pricing scheme.

Also, *the pricing scheme is supposed to make sure that the users causing more congestion costs than the others must be priced proportionally higher.* This is another challenge, since identifying which user's traffic is causing more congestion costs is a non-trivial task for a wide area network. Congested areas in wide area networks change dynamically over time. The level of congestion changes too. So, it is very hard to determine congestion cost information for each traffic flow, since updates are supposed to be done at the network edges or network ends for easy implementation purposes.

1.2.3 Provider Challenges

Besides maximization of total user utility, *the pricing scheme should be able to advertise prices such that provider revenue is maximized.* This includes consideration of several issues, when advertising a price to a particular customer:

- Each customer has different type of utility function.
- Provision of capacity to each customer costs differently. Also, these costs change dynamically according to the current conditions of the network.

So, the problem is not to only maximize total user utility, rather it is to maximize social welfare, *which is the sum of user surplus and provider revenue.* The question becomes what is the best way to allocate a capacity of c to n users such that the social welfare is maximized? We now make a simple analysis of the situation to find out the best capacity allocation and the best prices that lead to this allocation.

We assume that user's utility function is in the form $u_i(x) = b_i log x$. Suppose p_i is the price advertised to a particular user i. The user i will maximize his/her surplus by making sure that he/she contracts for $x_i = b_i/p_i$, i.e. :

$$S = u_i(x_i) - x_i p_i$$

$$\frac{dS}{dx_i} = \frac{b_i}{x_i} - p_i = 0$$

$$x_i = \frac{b_i}{p_i}$$

Assuming that the customers obey this above procedure, the provider can now figure out what price to advertise to each user by maximizing the social welfare $W = S + R$, where R is the provider revenue. Let $k(x)$ be a linear function and be the cost of providing x amount of capacity to a user.

$$W = \sum_{i=1}^{n} [u_i(x_i) - x_i p_i + x_i p_i - k(x_i)]$$

$$W = \sum_{i=1}^{n} u_i(x_i) - k(c)$$

Maximizing W with respect to x_i is the same problem presented in Appendix A. So, any user i should be given a capacity of

$$x_i = \frac{b_i}{\sum_{i=1}^{n} b_i} c$$

Since we assumed that the user will contract for b_i/p_i when advertised a price of p_i, then the provider should advertise:

$$\frac{b_i}{p_i} = \frac{b_i}{\sum_{i=1}^{n} b_i} c$$

$$p_i = \frac{\sum_{i=1}^{n} b_i}{c}$$

This means that the provider should advertise the same price to all users. However, notice that this above study assumed three major things:

- the cost of capacity provisioning to each user is the same
- all users have the same type of utility function, i.e. $u(x) = b\, logx$
- users are honest and do not play unexpected games

1.3 Our Approaches

Implementation difficulties caused by the nature of wide area networks make it impossible to price optimally. We propose an adaptive pricing framework, Dynamic Capacity Contracting (DCC), to achieve most of the challenges mentioned in the previous section. DCC is specifically designed for diff-serv architecture of the Internet. Edge routers of a diff-serv domain are the stations of ISPs, at which customers and the ISP can negotiate for the network service. At the end of negotiation customer and the ISP decide the contract parameters (i.e. price, volume, length). The ISP uses estimation techniques to determine user's willingness-to-pay.

DCC relies on congestion detection techniques that can be implemented only at the network edges, with very little support from interior nodes. Several different congestion detection techniques can be used in order to provide necessary congestion information to DCC. One simple method detecting congestion would be to use ECN bit as the congestion indication bit. Interior routers can mark the packets by updating the ECN bit at the time of congestion. The egress edge router can detect the congestion after receiving the marked packet(s). This is a fast and efficient way of detecting congestion in a wide area network, which is crucial for adaptive pricing. In a recent work, [36], even the updating of ECN bits at interior nodes is not required, which means edge-to-edge congestion detection can be achieved in several ways, and hence DCC is plausible in this sense. Egresses can make estimation of network capacity by using techniques like AIMD (Additive Increase Multiplicative Decrease). We embed congestion pricing into the framework by making the capacity estimation at the egress intelligently. The egress edge router decreases the estimated capacity when it receives a marked packet. Egress edge routers inform the ingress edge routers about the estimated capacity by sending feedback packets. The ingress edge routers can, then, use the estimated capacity and user's estimated willingness-to-pay to figure out optimal price.

We define two main architectures to use pricing for managing congestion control in a diff-serv network:

1. **Pricing for congestion control (PFCC):** In this pricing architecture, provider attempts to solve congestion problem of its network just by con-

gestion pricing. In other words, the provider tries to control congestion of its network by changing service prices. The problem here is that the provider will have to change the price very frequently such that human involvement into the price negotiations will not be possible (we will prove that later in Chapter 5). This problem can be solved by running intermediate software (or hardware) agents between end-users and the provider. The intermediate agent receives inputs from the end-user at large time-scales, and keeps negotiating with the provider at small time-scales. So, intermediate agents in PFCC architecture are very crucial in terms of acceptability by users.

If PFCC architecture is not employed (i.e. providers do not bother to employ congestion pricing), then congestion control will be left to the end-user as it is in the current Internet. Currently in the Internet, congestion control is totally left to end-users, and common way of controlling congestion is TCP and its variants. However, this situation leave open doors to non-cooperative users who do not employ congestion control algorithms or at least employ congestion control algorithms that violates fairness objectives. For example, by simple tricks, it is possible to make TCP connection to capture more of the available capacity than the other TCP connections.

The major problem with PFCC is that development of user-friendly intermediate agents is heavily dependent on user opinion, and hence requires significant amount of research. A study of determining user opinions is available in [11]. In this thesis, we do not focus development of intermediate agents. But, in Chapter 6, we will adapt our propsed framework Distributed-DCC to PFCC architecture.

2. **Pricing over congestion control (POCC):** Another way of approaching the congestion control problem by pricing is to overlay pricing on top of congestion control. This means the provider undertakes the congestion control problem by itself, and employs an underlying congestion control mechanism for its network. This way it is possible to enforce tight control on congestion at small time-scales, while maintaining human involvement into the price

negotiations at large time-scales.

So, assuming that there is an underlying congestion control scheme, the provider can set the parameters of that underlying scheme such that it leads to fairness and better control of congestion. The pricing scheme on top can determine users's willingness-to-pay and set the parameters of the underlying congestion control scheme accordingly. This way, it will be possible to favor some traffic flows with higher willingness-to-pay (i.e. budget) than the others. Furthermore, the pricing scheme will also bring benefits such as an indirect control on user demand by price, which will in turn help the underlying congestion control scheme to operate more smoothly. However the overall system performance (e.g. fairness, utilization, throughput) will be dependent on the flexibility of the underlying congestion control mechanism.

We will investigate POCC architecture in more detail later in Chapter 7, where we adapt the Distributed-DCC framework to POCC architecture.

Given any edge-to-edge congestion detection mechanism, DCC allows both of the above methods to be implemented. In the latter method, the only extra requirement is to mapping DCC's parameters to the parameters of the underlying edge-to-edge congestion control mechanism.

One crucial aspect of DCC is that it imposes *short-term contracts* to accommodate adaptive pricing. The ISP can change the length of these contracts dynamically depending on the current network and traffic conditions. Short-term contracts allow more frequent price updates, and hence more control over user demand can be achieved. To reduce the overhead, DCC makes pricing at a granularity level of *contracts* rather than *packets*. This reduces accounting overhead significantly by sacrificing some optimality, amount of which is to be investigated.

Since network edges are the nodes with high computation power, we can perform several complex operations at these points. To address the fairness issues, DCC makes estimation of each user's willingness-to-pay for the network service and uses that information during the price updates. In order to make sure fairness to the users, the edge routers communicate these estimated willingness-to-pay information to each other at a time-scale that is large enough (generally tens of round-trip-times) to accommodate within a wide area network. In this way, each edge router has overall information about the users's willingness-to-pay, which enables them to figure out optimal prices. This willingness-to-pay communication makes sure that the edge routers are advertising equal prices to those users causing the same amount of congestion costs.

A very interesting aspect of DCC is that it provides flexibility to penalize the traffic flows that are causing more congestion costs than the others. DCC can achieve this by intelligently monitoring the edge-to-edge traffic. DCC can achieve to determine the amount of congestion cost that a particular traffic flow is causing. Determination of such information enables DCC to penalize those that cause more congestion than the others by lowering the willingness-to-pay of the corresponding flows.

1.4 Motivation

For our work, the main motivation is that the Internet lacks better economic models and tools for the purpose of traffic engineering. Several technical tools (e.g. TCP improvements, diff-serv) have been deployed in the Internet, but economic tools such as adaptive pricing have not. Although several adaptive pricing proposals have been made for the Internet, all of them stayed in theoretical domain.

Given efficient economic tools, better economic models can possibly be implemented. Current economic model of the Internet is traditional wholesaler-retailer (also known as 1-tier, 2-tier, ...) model. A better model would also include *intermediaries, exchanges*[2] and similar types of components. Such new economic models will provide opportunity for more optimization both in terms of network efficiency and economic efficiency.

In order to implement such better economic models in the Internet, there is a need for more flexible and more efficient economic tools to be deployed. Specifically, the need for an adaptive pricing (which is mainly congestion-sensitive pricing in the case of networks) tool is crucial as an economic tool. In large networks, deployment of congestion-sensitive pricing is tough because it requires very recent information

[2]These type of components are already implemented in bandwidth market.

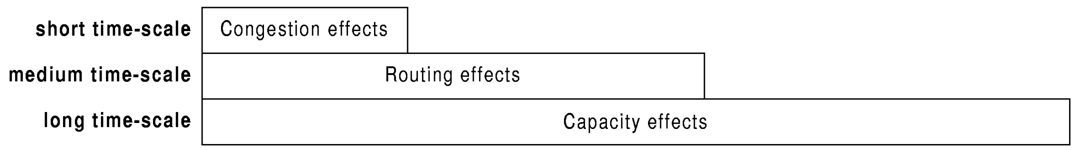


Figure 1.1: Pricing time scales depending on the major factors.

about the actual congestion level while this is very hard to do in large networks. So, deployment of congestion-sensitive pricing schemes has been a major obstacle for better economic models and optimization in the Internet.

Also, although sunk costs of bandwidth are large, the bandwidth has become a commodity recently. Value of the network service lies in packaging and presenting it to users rather than the main cost of bandwidth. So, the value mainly changes at shorter time-scales especially with changes in congestion level. At the time of congestion the value increases which should cause increase in price.

We can classify value changes according to their time-scales. Figure 1.1 represents three major time-scales at which value of network service changes. At long time-scale, provisioned capacity can change and this causes changes in value. At medium time-scale, routing can change and this causes changes in value. Finally at short time-scale, congestion can change and this causes changes in value. Our focus is mainly on pricing issues that can be addressed at short time-scales, i.e. congestion-based. Notice that routes are fixed for short time-scale, so we do not try to address pricing issues related to routing changes.

So, we can summarize our motivation in two main items that are needed in the Internet:

1. **Need for new economic models and tools in the Internet**
   In order to implement new economic models the followings are needed:

   (a) *More flexible pricing architectures:* Current pricing architecture (i.e. framework) does not allow adaptive pricing schemes to be implemented. So, there is a crucial need for a more flexible pricing architecture that provides environment for adaptive pricing. Specifically, it must obtain and provide recent congestion information for possible implementation of congestion-sensitive pricing.

(b) *Building blocks with a range of pricing capabilities:* Given a flexible and knowledgeable pricing architecture, there is also need for very flexible economic tools that can support a range of pricing techniques. Such building blocks will allow more optimization to be deployed. Examples of these building blocks are billing techniques, accounting techniques, pricing schemes.

2. **Need for adaptive pricing**

Adaptive pricing is specifically useful in the following manners:

(a) *A way of controlling user demand and network congestion:* For the purpose better congestion control, adaptive pricing provides opportunity to control user demand and hence help controlling congestion.

(b) *Fairness:* Another benefit of adaptive pricing is that it operates regardless of the underlying transport protocol. For example in the Internet, UDP traffic beats down TCP traffic because it does not back off at the time of congestion. This causes TCP traffic to be treated unfairly compared to UDP traffic. Adaptive pricing can control both UDP and TCP traffic fairly since it can potentially treat them equally.

Also through adaptive pricing, provider can reflect cost differences to users. Note that each traffic flow causes different amount of cost per unit volume. This is mainly because of dynamic nature of the Internet. Routes change dynamically, congestion costs change dynamically, and so on.

1.5 Scope of Thesis

In this thesis, we mainly focus on developing a framework on which congestion-sensitive pricing schemes can be implemented. Figure 1.2 represents scope of our work among the related networking problems[3]. Figure 1.3 represents tree version of

[3]Note that sizes of the shapes does not represent sizes of research problems being represented.

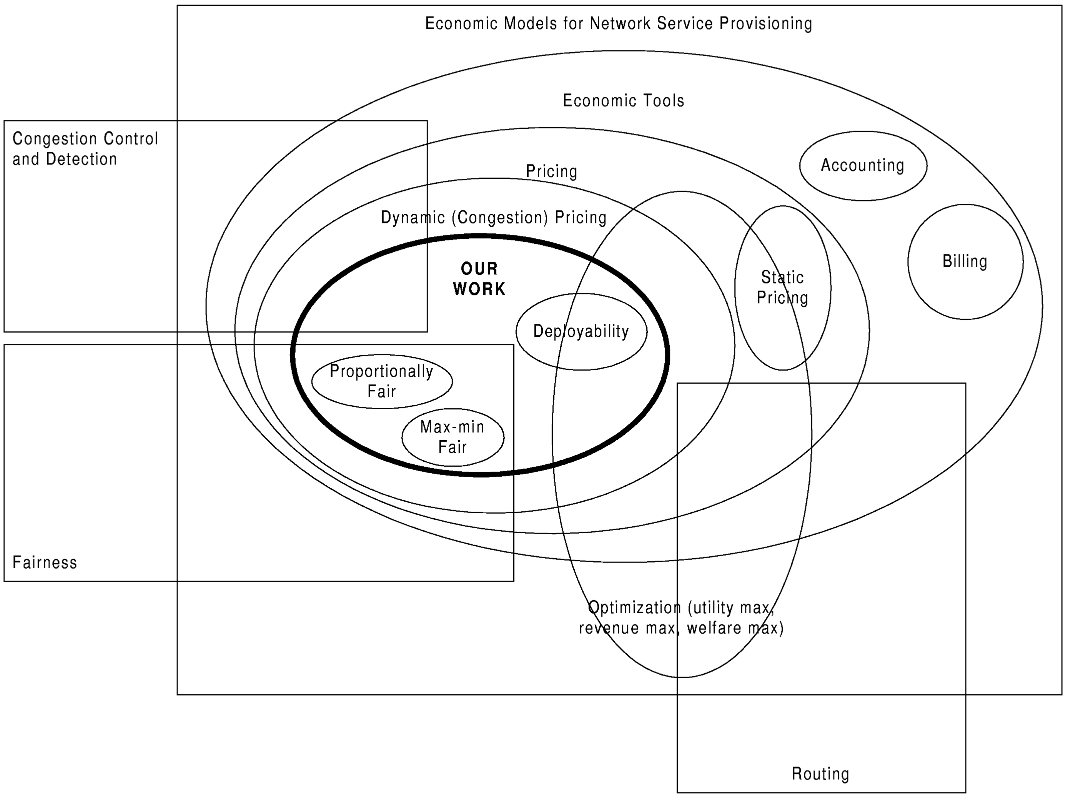


Figure 1.2: Scope of thesis among the networking problems.

scope definition for our work. The gray nodes in Figure 1.3 represents the problems covered by the thesis.

The main problem being solved in the thesis is deployability of congestion-sensitive pricing schemes. The thesis also tries to address congestion control and fairness issues through adaptive pricing. Specifically, it determines ways of using adaptive pricing to control congestion and to provide different types (i.e. proportional, max-min) of fairness among traffic flows. Also, it focuses on providing more optimization (i.e. utility maximization, social welfare maximization) within implementation constraints of adaptive pricing.

1.6 Thesis Outline

This thesis attempts to provide solutions to the problems identified in Section 1.2 within consideration of single diff-serv domain and single provider only. It de-

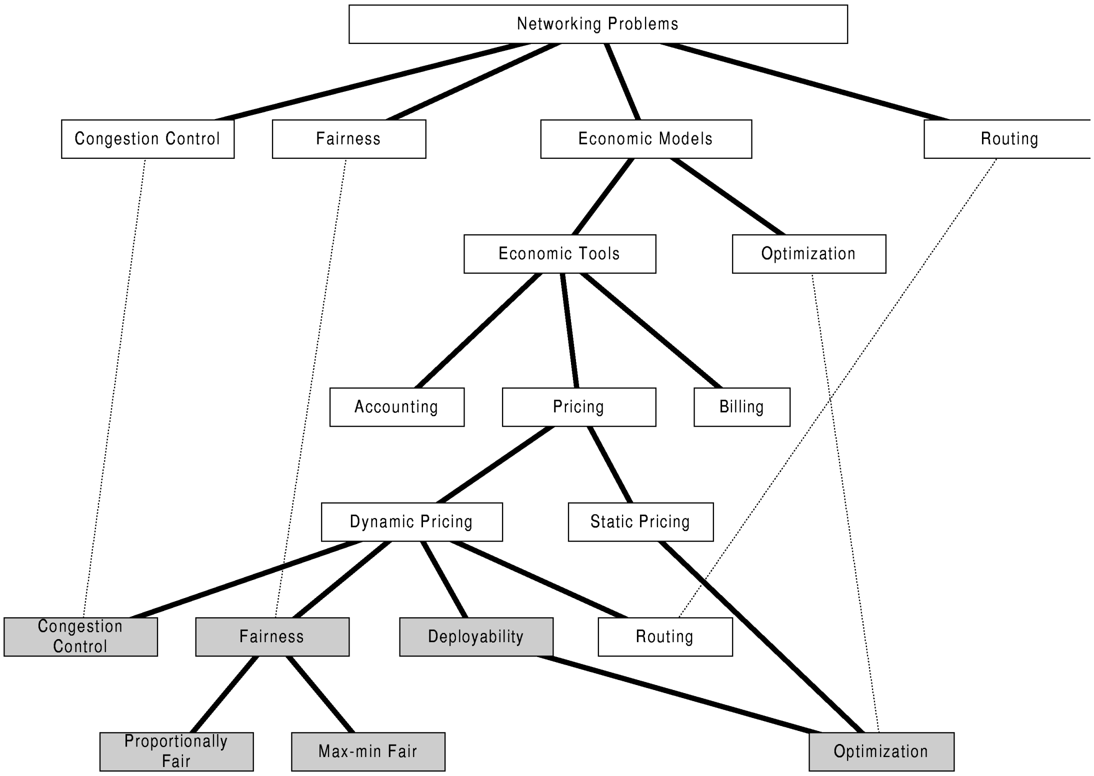


Figure 1.3: Scope of thesis as tree representation.

velops basic building blocks that are necessary components in order to construct a flexible economic model for the Internet and wide area networks.

In Chapter 2, we survey the literature in the area of network pricing and pricing proposals for the Internet. We classify the pricing proposals into two main classes: dynamic pricing schemes, static pricing schemes. We provide descriptions of the pricing proposals in both classes along with evaluations of them according to various criteria: deployability, network efficiency and Pareto efficiency.

For a good understanding of the dynamics of congestion-sensitive pricing, we first study one of the primary pricing proposals in the area. In Chapter 3, we study the well-known pricing proposal Smart Market, for the Internet. The importance of the Smart Market lies behind the facts that it is the first pricing proposal for the Internet and also it defines an optimal congestion pricing method. We lay out implementation problems it has and propose an implementation strategy for the Smart Market on diff-serv networks. We then develop a packet-based simulation of the Smart Market according to the implementation strategy we developed. We finalize our study of the Smart Market by presenting performance results and possible future research on it.

In Chapter 4, we propose Dynamic Capacity Contracting (DCC) framework for adaptive pricing within a single diff-serv domain. Customers perform *short-term* negotiations with the provider at the edges of the network. Since negotiations happen at short-term intervals in DCC, the provider can employ congestion-sensitive pricing by updating congestion condition of the network. We use edge-to-edge congestion detection techniques to facilitate congestion-sensitive pricing in the framework. We develop two pricing schemes within the DCC framework: Edge-to-Edge Pricing (EEP) and Congestion Index. We assume that all the ingress edge routers advertise *the same price* to their customers. By using the EEP pricing scheme, we develop simulation of DCC and present initial performance results for proof-of-concept. We also provide a comparative experimental evaluation of DCC and Smart Market. We finalize the chapter by determining benefits and weaknesses of DCC. One of the major implementation issues that raise about DCC is the question of how long can the short-term negotiation intervals be? Another major deployment issue is to relax the assumption of same prices being advertised at the ingress edge routers.

In Chapter 5, we attempt to answer the question about the length of negotiation intervals (i.e. *pricing intervals*) which was raised in the previous chapter. As the pricing intervals get larger, the control achieved on the network congestion by adaptive prices reduces. We represent the level of control achieved by the prices by the correlation between prices and congestion measurements in the network. We then develop an approximate analytical model of the correlation, which includes length of pricing intervals as a parameter. By using this model we derive three major conclusions, and then we validate the model and the conclusions with experimental results by using simulations of DCC with different pricing intervals. The study of pricing intervals reveals that pricing-based congestion control deteriorates very sharply as the pricing intervals get larger. According to our experimental work, the level of congestion control achieved by congestion-sensitive prices vanishes when pricing intervals exceed 40 round-trip-time (i.e. 2 or 3 seconds for the most cases in the current Internet) even under a traffic load with a low variance.

In Chapter 6, we relax the assumption of same price advertisement in DCC. The ingress edge routers can now advertise locally computed prices, which requires several updates to DCC framework. We call the updated version of DCC framework as *Distributed-DCC*. Advertisement of locally computed prices makes it difficult to maintain coordination among the edge routers. Because of this, we introduce a new component, *Logical Pricing Server (LPS)*, to facilitate the overall coordination of the network. Although having different prices at each ingress makes it difficult to coordinate the network, it provides a new ability of behaving differently to each edge-to-edge flow. Because of this new ability, Distributed-DCC can provide a variety of fairness (e.g. max-min, proportional) in rate allocation to edge-to-edge flows. We describe the Distributed-DCC framework within the PFCC architecture, determine related problems and their solutions. We also revise the EEP pricing scheme within the context of Distributed-DCC framework. We finalize the chapter by presenting performance results and determining problems with Distributed-DCC followed by necessary future work for it.

In Chapter 7, we focus on the case where pricing is overlaid on top of an underlying edge-to-edge congestion control scheme. We determine the issues raised by overlaying pricing over congestion control, provide solutions to them within Distributed-DCC framework. We simulate Distributed-DCC over Riviera [36, 37] and present experimental results.

In Chapter 8, we show differences between Distributed-DCC and other congestion pricing proposals in terms of calculation and optimality of prices. We also provide theoretical proof for optimality of price calculations in the EEP pricing scheme. We solve total user utility maximization problem, and show that the solution leads to EEP. We also investigate the term "elasticity" in the area of networking. We mathematically define utility-bandwidth elasticity, which is the elasticity known in networking area. Then, we determine relationship between the well-known demand-price elasticity and utility-bandwidth elasticity.

In Chapter 9, we investigate theory behind Distributed-DCC's ability to provide multiple fairness types in rate allocation. We derive all possible rate allocations in Distributed-DCC. This way, we show how one can achieve different fairness types within Distributed-DCC framework. Finally, we investigate Distributed-DCC's sensitivity to several parameters, such as contract length, observation interval, and LPS interval.

Finally in Chapter 10, we first summarize and list contributions of the thesis. We then determine the possible future research agenda.

CHAPTER 2
BACKGROUND

2.1 Introduction

Internet pricing has drawn considerable interest of researchers from diverse fields in the last ten years. The scope of the literature is wide-ranging. Pricing can be used for several different purposes for the Internet: Increase revenue [9], uniform users' traffic by time-of-day pricing, reduce congestion epochs by congestion pricing [51, 42], improve fairness [53], network management [34], etc. There are many studies of pricing alternatives based on the existing Internet service model (often referred to as the best-effort service model); since a particular user's Internet experience is often provided by the resources of several service providers, the multiple provider Internet business model is certainly an important factor in analysing both the provider-consumer and provider-provider pricing strategies. The most fervent debates in Internet pricing, however, are driven by proposals tied to alternative service models for the Internet, where different quality of service are provided when network resources are limiting.

What have we learned from this large body of literature? This survey attempts to provide a taxonomy, a classification, some digest and discussion of important works, of course with author's opinions interjected.

Broadly speaking, there are three types of pricing of Internet services: *Flat-rate pricing*, *usage-based pricing*, and *dynamic pricing*. The first two are also regarded as *static pricing* schemes, because price of transmitting unit amount of traffic is fixed.

Flat-rate pricing [26] is basically to charge the user with a fixed amount, p_f, per unit time regardless of his/her usage and actual network conditions. This is the most common mode of pricing to the retail customers currently for the Internet. ISPs usually charge a fixed amount for a contract time period of a month.

Usage-based pricing is to charge the user with a fixed amount, p_u, per unit amount of usage. For the Internet, the amount of usage can be in two forms: traffic amount or connection time. In the first one, p_u is fixed for unit traffic amount (e.g.

p_u dollars per megabyte). In the second one, p_u is fixed for unit amount of connection time (e.g. p_u dollars per one hour of connection). Notice that in the second one the user is free to send/receive any amount of traffic up to a peak whereas in the first one the user pays for every megabyte of traffic he/she caused to the Internet.

The first form of usage-based pricing is being employed in the Internet at ISP-to-ISP level. [61] Usually, ISPs buy their connection from a carrier by defining a fixed rate and a burstable rate. The charge for the fixed rate is flat but the excess rate is charged per traffic amount. The burstable rate defines the maximum value for the sum of fixed rate plus the excess rate that the ISP can use. The carriers generally do not make an exact accounting of the excess rates, since it would be too much of overhead to account. They, rather, sample the excess rates of the ISPs at some reasonably large time intervals, and calculate the total amount of excess usage for each ISP.

Currently in the Internet, the second form of usage-based pricing is employed at customer level by some ISPs. Generally within a month, ISPs charge a flat price (p_f) for the first predefined number of hours, and then charge usage price (p_u) per extra hour of connection.

Dynamic pricing of networks is a more general case of usage-based pricing. It is to charge the user with $p_d(t)$ per unit amount of usage, varying over time t. $p_d(t)$ can be varied depending on the network conditions. Since the price value changes according to the actual network conditions, it is also called as *adaptive* or *responsive pricing*. Most of the research focused on varying p_d according to the congestion level of the network and aimed to relieve the congestion in this way. This special case [4] of dynamic pricing is called *congestion-based pricing* or *congestion pricing*. However, because of high complexity of the proposed dynamic pricing schemes, none of them has been employed in the Internet.

Most of the research on Internet pricing focus on development and implementation of dynamic (responsive) pricing schemes over the Internet. Since 1992, several dynamic pricing schemes [50, 49, 33, 42, 59, 19, 79, 67] have been proposed. Among

[4]Notice that dynamic pricing can be based on several things other than congestion, e.g. number of active users, level of utilization. However, the main part of marginal costs is from congestion. Because of that research has been focused particularly on congestion-based pricing.

those dynamic pricing schemes, most proposed congestion pricing that requires complex pricing techniques, while some of them opposed complex pricing strategies and supported either usage-based pricing or classified flat-rate pricing (e.g. dividing the service into different classes and assigning a flat price to each class).

Evaluation of network pricing schemes can be done according to two major metrics: Network efficiency, economic (Pareto) efficiency [75]. Conditions for network efficiency are stability and high utilization. A network is stable if packet drop rate is close to zero and queue lengths (and hence delay) are bounded. On the other side, Pareto efficiency can be achieved by maximizing social welfare. A situation is Pareto efficient if *nobody can be better off without making somebody worse off* [75]. For a network, this can be achieved by maximizing both the provider's revenue and the users' total utility. Total utility of the users can only be maximized by fair rate allocation to them, i.e. the network capacity should be given to the users according to their willingness to pay. This way the users with more valuable traffic will be given more network capacity since they have more willingness to pay than the others. So, the pricing schemes are supposed to allocate the network capacity fairly to the users while maximizing the provider's revenue. Besides, they are supposed to bound the network queues and lower the drop rate by uniforming users' demand over time. In this chapter, we will make overall evaluation of the pricing proposals according to network and Pareto efficiency.

Although most of the Internet pricing research focus on dynamic and complex pricing techniques, the discussion of whether the market desire such fancy pricing plans or not still continues [60, 61, 25, 4, 65, 64, 45]. There is actually trade-off between better quality-of-service (QoS) and simpler pricing plans. Better QoS can be achieved by employing more complex pricing strategies while, on the other hand, people do not want complex pricing plans. Most researchers, however, believe that the Internet needs a pricing scheme better than the currently employed flat-rate pricing because of its sub-optimality and non-adaptiveness to congestion. The main discussion is on whether to employ complex dynamic pricing schemes or just employ simpler pricing schemes such as usage-based pricing. We will return to this discussion later in Section 2.3.

Currently, flat-rate pricing makes the users not to restrict themselves which increases traffic amount and usage significantly [61]. This unrestricted nature of the flat-rate pricing eases market development, which is pleasing to the service providers. When the usage increases a lot, however, it also causes congestion epochs to be more severe and QoS to be very low. Flat-rate pricing has no contribution to the network efficiency, rather it effects network efficiency negatively especially by putting no constraint on the traffic at peak hours. Also, for the users who have more valuable traffic there is no way to express their willingness to pay. Since there is no way of differentiation in flat-rate pricing, the users with more valuable traffic have to use best-effort traffic as the other users. Because of these reasons, flat-rate pricing cannot achieve Pareto efficient Internet.

In the following survey of the existing work, we first consider the major works on dynamic pricing, which are often combined with various proposals for new Internet service models, new resource allocation algorithms, and protocols. In this case, pricing is used as incentive to achieve better network efficiency, or optimality of bandwidth usage on the basis of a relatively short window of time. These are mostly proposals of dynamic pricing schemes. Then, we survey works that argue against the use of dynamic pricing, for a variety of reasons; and other works that analyze (mostly on economic basis) the pros and cons of flat-rate and usage-based pricing.

The rest of the chapter is organized as follows: We first survey important dynamic pricing schemes in Section 2.2. Then we survey major arguments against dynamic pricing in Section 2.3. In Section 2.4, we survey static pricing schemes, which are basically different compositions of flat-rate and usage-based pricing techniques. Next in Section 2.5, we compare overall characteristics of the presented pricing schemes. Finally, we conclude with conclusions and discussions.

2.2 Dynamic Pricing Schemes

2.2.1 Smart Market

The smart market [50] is one of the first proposed pricing schemes for the Internet. The smart market imposes a per-packet charge, which is dependent on the network congestion. The price-per-packet changes dynamically over time, which makes the smart market a dynamic pricing scheme. Price changes are justified according to the congestion level.

The smart market assumes that each packet will have a *bid* value in its header. The owner (sender) of each packet is supposed to assign the bid values of his/her packets. Basically, the smart market introduces a new packet header to give the users opportunity to express their willingness to pay.

In the network, each router keeps a *threshold* value and forwards only the packets with bid values greater than or equal to the threshold value. The packets that have bid value of less than the threshold are simply dropped. These threshold values at the routers are updated dynamically over time according to the congestion being experienced at the particular router. So, the threshold values are expected to be different at different routers and to be changing by time.

Each packet sent by the user tries to make it to the destination. If it can make to the destination, the price that the owner of the packet pays is called *market-clearing price*, which is the maximum of the threshold values that the packet passed through.

The smart market also defines the order of service according to the bids. It proposes that the routers first serve the packets which have higher bids. The idea behind this is to make *auction* among the users. However, this is very hard to implement since it requires sorting of the packets according to their bids and can potentially lead to starvation of packets with lower bid values. Also, this effects TCP's RTT estimation techniques and congestion control algorithm negatively since packet drops are crucial for TCP's congestion control techniques.

Smart Market proposes to solve total user utility maximization problem. It assumes that user i has a utility function $u(x_i) - D(Y = X/K)$, where x_i is the number of packets sent by the user, X is the accumulation of all users demand (i.e. $X = \sum_i x_i$), K is the total network capacity, $Y = X/K$ is the utilization of network, and D is the total delay experienced by the user as a function of utilization Y.

Based on this utility assumption, Smart Market solves the problem:

$$\max_{x_i} \sum_{i}^{n} x_i - nD(X/K)$$

where n is the total number of users. Solution to this above maximization problem is achieved with the following price p:

$$p = u'(x_i) = \frac{n}{K} D'(X/K)$$

which is the price value that is supposed to be charged at each local router. In other words, the above price formula is the way of calculating threshold values at routers.

One can apply the smart market pricing scheme over a differentiated-services ([27]) network as follows: The sender sets the bid value, b, in the packet and sends it to the network through the ingress node. The packet passes through a series of interior routers, each of which has a threshold value, T. The interior routers simply drop the packet, if it does not satisfy the condition of $b \geq T$. If the packet satisfy the condition, then it is placed into the sorted queue according to its bid value. If the packet could make through the network, the accounting is done for the owner of the packet. To calculate the market-clearing price of each packet an extra packet header is needed. This new header field can be updated at each interior router depending on what is available in it and the T value of that particular router. Each customer sends a probe-packet to find out the current market-clearing price. The egress node that receives the probe packet sends feedback to the customer including the probed market-clearing price. Then, the customer adjusts his/her network usage according to that market-clearing price and the value of the application. A more detailed study of implementing the smart market over differentiated-services architecture of the Internet is available in [82] and in Chapter 3.

The smart market has several problems for implementation over the Internet. It requires the interior routers to be able to sort packets according to their bids, which requires new routers to be deployed. Also, there is need for extra packet header fields, e.g. bid value, market-clearing price. Additionally, it has bad interaction with TCP, which constitutes 90% of the current Internet traffic.

Although it has several implementation issues, the smart market was proven to be ideal in terms of Pareto efficiency in [50]. Also in [51], MacKie-Mason and Varian prove that the smart market can also be useful to figure out the optimal amount of extra provisioning needed to improve performance of a running network, i.e. optimal capacity expansion of a network.

2.2.2 Priority Pricing

Gupta et al. proposed a priority pricing scheme with congestion-based prices [33, 32]. For each service type (e.g. ftp, email, WWW) within a priority class, the service provider maintains a congestion-based price, e.g. r_{mi} for priority class of i and service type of m. The service provider keeps updating these prices at some time interval, T, according to the load and the congestion level of the node that provides the service m. In order to determine the congestion level, priority pricing calculates experienced delay by sampling the delay over some time intervals. Since the scheme uses delay as congestion measure of the network, it is not only a priority-based pricing scheme but also a congestion-based pricing scheme.

In priority pricing, the network is viewed as customers, network core, and the servers that serve requests from the customers. The provider basically prices the services that these servers are providing. For a specific type of service, customer sends request to the service provider which in turn causes the service provider to calculate and advertise its current price for the particular service type in every priority class. Then, the customer decides about which service to buy and informs the service provider about his/her decision.

Priority Pricing solves benefit maximization problem for the provider, which is formulated as:

$$B(p, w) = X_0 \int \int \sum_i \sum_j \sum_k \lambda_{ij} [V_{ij} - \delta_{ij} \tau(j, k, m)] p_{ijkm} dF(V_{ij}, \delta_{ij})$$

subjected to

$$w_{mk} \equiv \omega_k(X_m; v_m)$$

where X_0 is demand scaling parameter, λ_{ij} is fraction of X_0 associated with customer

i for service j, V_{ij} is the revenue obtained from customer i for service j, δ_{ij} is delay cost of customer i for service j, $\tau(j,k,m)$ is expected throughput time for service j at server m in priority class k, p_{ijkm} is probability that customer i will request service j at server m in priority class k, X_m is the matrix of service request arrival rate by job size and priority at server m, v_m is the capacity at server m, w_{mk} is expected waiting time at server $m$for priority class k, and ω_k is the waiting time for priority class k as a function of service request arrival rate and capacity. Based on this optimization problem, Priority Pricing calculates price of a job sized q at server m for priority class k as:

$$r_{mk}(q) = \sum_l \sum_q [d\omega_l / dX_{mkq}] \sum_i \sum_j \delta_{ij} x_{ijlm}$$

where X_{mkq} is the arrival rate of jobs sized q at server m in priority class k, and x_{ijlm} is the flow rate of service j for consumer i with priority k at server m.

Among the problems with priority pricing, there is no defined implementation strategy. Also, one another problem is that there are several parameters to be tuned, e.g. price update intervals, price smoothing parameters. Moreover, there is a big question of what should the number of priority classes be in order to get good classification of the customers. The answer to this question is critical to evaluate the scheme in terms of Pareto efficiency.

In the network core, priority pricing does not propose any specific way about how to treat the traffic belonging to different priority classes. It just focuses on treating the traffic differently at the servers where the requests being implied by these traffic are served. In terms of reaching to the server, traffic of all the priority classes are equal. This becomes an issue since RTTs are very large for some cases. So, the customers with larger RTTs to the servers will get less of the resources than the customers with smaller RTTs within the same priority class, which violates fairness objectives.

Also, most of the user requests need multiple servers to participate in serving the request. For instance, an email passes through multiple email servers, an HTTP request is served by multiple HTTP servers. Priority pricing does not investigate problems regarding the cases when requests are served by multiple servers. Notice that priority pricing views the servers as bottlenecks. Different topology of servers (i.e. serial, parallel) must be investigated in terms of performance and fairness of the scheme.

Following the trend on priority-based pricing, Marbach [52, 53] studied priority-based pricing in diff-serv environment and fairness of priorit-based pricing. In [52], he showed that priority-based pricing always leads to equilibrium by modeling the system as a non-cooperative game. In [53], as an extension to [52], he also showed that equilibrium rate allocation of such non-cooperative game is max-min fair.

2.2.3 Packet Marking (Proportional Fair Pricing)

In [42], Kelly et al. proposed a pricing-based congestion control scheme which is dependent on marking the packets when there happens congestion at the interior routers. Then either the interior router itself or the destination end-node of the packet feeds a congestion price (i.e. *shadow price*) back to the owner of the packet. The owner, then, adjusts his/her sending rate according to the shadow price and the value of his/her traffic.

In [42], the authors prove that such a packet marking scheme will lead to stability and proportional fair rate sharing of the network. The intuition behind the scheme is that the shadow price will increase at the time of congestion and hence the users will lower their sending rates, which will keep the congestion under control. Similarly, the shadow price will reduce at the non-congested time periods and hence the users will increase their sending rates, which will keep the network being utilized. Also, for those whose packets are passing through more number of congested areas of the network, more feedback (each of which is a congestion charge) will be sent. Thus, the number of bottlenecks that user's traffic traverses is also taken into account when charging the user. So, the users whose traffic is contributing more to the congestion are charged more and will get less share of the network capacity.

In [42], the authors show stability and fairness properties of PFP based on total user utility maximization problem. First, they split the system problem (i.e. total user utility maximization) into two smaller and separable sub-problems as user's problem and network's problem. The system problem is sub-divided by introducing "price" as Lagrange multipliers. Assuming that user utilities are concave and feasible region is convex, they show that optimal rate allocation vector is proportionally fair, i.e. the following condition is satisfied:

$$\sum_{r \in R} \frac{{x_r}^* - x_r}{x_r} \leq 0$$

where x_r is the rate of user flow on route r and R is the set of all possible routes in the network. The authors also provide solution to dual of the system problem, and algorithms for price calculation in both primal and dual cases.

In terms of congestion control, proportional fair pricing is different than TCP's congestion control mechanism in two ways. In TCP, the packets are simply dropped in the case of congestion whereas, in proportional fair pricing, the routers mark the packets sends a measure of congestion (i.e. shadow price) to the packet's owner as a feedback. Secondly, in TCP, when a user gets a congestion indication (either as a time-out or a re-transmission request), it employs multiplicative decrease followed by an additive increase to its sending rate. In proportional fair pricing, however, users are free to adjust their own sending rates according to their wishes. In [31], the authors prove that such a pricing scheme leads to network and Pareto efficiency of the networks.

Proportional fair pricing can be implemented over a differentiated-service network as follows: The customers send their packets through ingress nodes and the packets passing through congested areas are marked. This marking can be done by using the ECN bit in TCP header. The egress nodes make accounting of the marked packets and charges the packet owner's according to the number of marked packets received from them. This charge can be conveyed to the packet owner's at small enough time intervals. Then, the customers (packet owners) can adjust their sending rates accordingly. However, this way of accounting will not reflect the exact idea behind the proportional fair pricing. The reason is that some of the packets will pass through multiple congested areas but the egress node will not be able to realize that since it will count that packet as only one marked packet. If the interior routers do send the feedback for each marked packet as the proportional fair pricing proposes, then the customer will receive feedback for each marked packet. This would let the customer to know the exact number of congested areas (i.e. bottlenecks) that his/her packets are passing through, and he/she would react differently. However, the latter way of accounting requires all routers to be updated, which is not practical. So, proportional fair pricing can only be implemented in part if it is restricted to update only ECN bit at interior routers and to generate feedback packets at the edges.

As a follow-up to PFP, Low et al. [47] proposed a congestion pricing framework where the concepts of PFP can be generalized to non-logarithmic user utility functions. Additionally, Low et al. introduced ways of using congestion pricing for flow control. Also generalizing the concept, Low used PFP's optimization techniques to model behavior of congestion control algorithms in the Internet [48, 46].

2.2.4 Resource Negotiation and Pricing (RNAP)

Wang and Schulzrinne, [76, 77], have recently proposed a resource negotiation and pricing framework, RNAP, through which dynamic pricing of network services could be done. The authors propose both centralized and distributed ways of implementing RNAP. RNAP allows customers to negotiate and contract with the service provider about several QoS parameters, e.g. peak rate, loss rate, maximum delay.

RNAP maintains two crucial time intervals to manage the contracting and pricing. First one is *negotiation interval* which is the time between subsequent contracts between the customer and the service provider. Second one is *price update interval* which is the time interval to update the prices of the system: Usage price, holding price, and congestion price. The total charge to a customer is composed of three parts:

- *Usage Charge:* $V(t) * p_u$ where $V(t)$ is the amount (volume) of the traffic that the customer sends during the time period t and p_u is the price per unit traffic volume (i.e. usage price) for that time period.

- *Holding Charge:* The charge made per traffic volume not used by the customer, although the customer contracted for it. This happens when the customer does not use all of the rate he/she contracted for. RNAP defines a holding price for each service class which is supposed to be higher for the service classes with better QoS parameters.

- *Congestion Charge:* This charge is made to the customer when he/she uses congested area(s) of the network. The customer pays an extra congestion price for each of his/her packet traversing a congested area. The congestion price varies dynamically over time according to the congestion level of the particular area of the network.

Since RNAP employs congestion-based pricing, it can utilize the network resources and hence achieves network efficiency. It can also achieve Pareto efficiency provided that the negotiation and price update intervals are small enough, which has not been analyzed so far. One drawback of RNAP in terms of pricing is that it has too much overhead, since it integrates several network management techniques: Resource management, admission control, and pricing. Also, another major problem with RNAP is that it requires all network nodes (including interior nodes) to participate in price calculation. At every negotiation interval, RNAP control agents probe the network by sending a special control packet, *Path* packet, for each traffic flow. *Path* packets are updated at each node they are passing through. Each node, basically, sums its current local price to a header field of *Path* packets. So, these *Path* packets simply find out about the total price for each traffic flow in the network. Notice that this is a procedure with very high overhead. This requires all interior routers to be updated, which is not practical in the Internet.

Additional to the RNAP, Wang and Schulzrinne introduces a very elegant utility function in [76] that captures behavior of most adaptive user applications:

$$U(x) = U_0 + w\, log \frac{x}{x_m}$$

where x is the bandwidth available to the user application, U_0 is the utility for the lowest QoS level (i.e. perceived utility for best effort service), w is the sensitivity of user application to bandwidth (i.e. elasticity), and x_m is the minimum amount of bandwidth required by the application for proper execution. Given a price p per unit traffic volume, the user can maximize surplus (i.e. $U(x) - xp$) by requesting a bandwidth of $x = w/p$. So, w basically represents the value of that application to the user, which is equivalent to user's budget for that particular application. Finally, assuming that user's budget for the application is b, the user will maximize his/her perceived surplus out of the application by contracting a bandwidth of $x = b/p$ for the application.

2.3 Arguments against Dynamic Pricing

While people do not want complex pricing plans, it is possible to provide better QoS by employing more complex pricing strategies. This trade-off has attracted significant attention in the area of Internet pricing [60, 61, 4, 25]. In [61], Odlyzko provides historical evidence that people want simplicity in pricing of their Internet services. He proposes that any possible pricing scheme for the Internet must be simple to the users, otherwise it will not be accepted by people. On the other hand, several studies [25, 78] showed that flat-rate pricing performs worse than more complex pricing schemes in terms of both network and Pareto efficiency. The studies in [25] and [78] do not focus on recovery of sunk and variable costs, but they prove that the provider makes more revenue by employing complex pricing plans.

Generally, people are risk averse and do not want to face with varying prices for Internet service. Employing usage-based pricing (not even congestion pricing) irritates Internet users [5] significantly [61], and ISPs do not want their customers to be irritated since their main concern is market share. The reason why the providers strive for market share is that most of their costs are sunk costs (i.e. deployment of physical fibers) and marginal costs (i.e. maintenance costs, congestion costs) are very small relative to deployment costs. [2, 57] On the other side, in the project of Internet Demand Experiment (INDEX), Edell and Varaiya showed that it is really possible to differentiate among the users by employing usage-based pricing [25]. In the experiments Edell and Varaiya made, although the differentiation of the users

[5] Only home users who generally have dial-up access. For other users (i.e. enterprise users, other ISPs), there is no practical evidence that they are irritated by usage-based pricing.

is achieved by usage-based pricing, the total Internet usage decreased as Odlyzko states later in [61]. So, the question still remains to what extent complexity is desired in pricing of the Internet services.

Paschalidis and Tsitsiklis made a very interesting study on modeling and analyzing static and dynamic pricing strategies in [65]. Assuming that the customer's demand model is known, they have shown that static pricing can perform almost as good as dynamic pricing for many small users (i.e. users that make connections of small length) case. By using continuous-time Markov Decision Processes, they model the case of multiple traffic classes where the customers can change their class over time (which is in fact a complex and dynamic decision required from the customers). Within this context, they formulate revenue and welfare (utility) maximization problems, and prove that static pricing techniques can be as good as dynamic pricing techniques *if the customer demand model is known.* Later, in [64], they showed that demand substitution effects do not change the result obtained in [65]. The caveat in Paschalidis and Tsitsiklis's work is that they used "connection time" in their analytical models as a basis for how long a user will stay in the system. However, the current Internet is moving from connection-based dial-ups to "always-on" services such as cable, DSL. So, it is questionable whether their models will hold for the future's always-on Internet or not.

As a follow-up to the work of Paschalidis and Tsitsiklis, Patek and Campos-Nanez [66] have made a study of the case where users with long connection times are also in the scenario. Again assuming that customer's demand model is known, Patek and Campos-Nanez also showed that static pricing can be as good as dynamic pricing even if there are users making long connections.

Again, following the same trend, Lin and Shroff [45] showed that control of very large (i.e. many users many nodes) networks can be achieved by static pricing. Particularly, they showed that static pricing performs asymptotically as good as dynamic pricing in terms of control of very large networks.

Shenker et al., [70], also argue that dynamic pricing does not seem to be attractive because of its high implementation costs and structural requirements. To support their argument, Shenker et al. posed three major questions:

- *Are marginal congestion costs relevant?* For several cases marginal congestion costs are so small that they can be ignored and are irrelevant to the overall efficiency of the network. Also, usage-based prices and flat prices can constrain the usage and they can be enough to improve efficiency if they are set properly. Moreover, many users (generally the ones that use small capacity connections, e.g. dial-up) will reduce their connection times and usage significantly if either usage-based or flat prices are increased.

- *Are marginal congestion costs accessible?* For large networks, like the Internet, it is usually very hard to measure the exact contribution of a particular flow to the congestion. In order to measure the exact contribution, support from the network core is required, which is very hard to implement on wide area networks. Wide area networks require these type of activities to be done at the edges. However at the edges, current mechanisms can only approximate the contribution to congestion for a particular flow. In our work, DCC, we use edge-to-edge congestion detection techniques to approximate marginal congestion cost of the flows.

- *Is optimality the only goal?* A plausible pricing architecture must also address structural and implementation issues for practical purposes. It must provide a flexible and efficient accounting infrastructure. Optimal pricing will cause non-practical and extensive amount of overhead. A pricing architecture must balance the trade-off between optimality and practicality. In our work, DCC, we balance this trade-off.

We believe that pricing based on marginal congestion costs is viable as long as implementation issues (which are posed in the last two of the above questions) are solved. In our work, we address implementation and structural issues by sacrificing some optimality in pricing. Although congestion pricing is complex and difficult to implement, it is still a viable method of providing better fairness and service especially during congestion epochs. Regarding the first question above, we believe that flat-rate charges can be employed along with congestion-based charges, i.e. both the flat-rate price and the congestion-based price are charged to users. By doing this way, the flat-rate prices will guarantee recovery of sunk costs, while the congestion-based prices will improve QoS and fairness of the service.

2.4 Static or Usage-Based Pricing Schemes

2.4.1 Time-of-Day Pricing

Actually, time-of-day pricing is an anonymous pricing scheme just like flat-rate pricing. Intuitively, increasing the network service price at peak hours of day or week should lead to better results in terms of supply-demand balance. In the contract, the service provider advertises different prices for different time periods of a day and a week. This assumes that the peak hours (i.e. user demand model) are known beforehand. Since prices do not change within the contract term, Time-of-Day pricing is not a dynamic pricing scheme. It is already deployed in several network services, e.g. long-distance phone service, wireless phone service.

In [65], the authors show that time-of-day pricing performs almost as good as dynamic pricing when the user demand model is known beforehand. However, they also agree that dynamic pricing performs much better than time-of-day pricing when the demand model is imprecise, since dynamic pricing can adapt the prices to actual traffic pattern whereas time-of-day pricing cannot. Odlyzko, [61], claims that the current Internet demand model is generally precise, and because of time-of-day will perform good enough for the most time. However this may be true only at large time-scales. For example, generally, Internet traffic is heavy at noon hours and light at morning hours. This does not guarantee that no large burst will take place during the morning hours. So, time-of-day will not be able preserve supply-demand balance during unexpected demand bursts, even though they might be temporary.

2.4.2 Paris Metro Pricing (PMP)

Odlyzko [59] proposed an Internet pricing scheme that is inspired by the old city metro system of Paris, France. In the old metro system in Paris, there were two classes of cars with exactly the same quality in terms of both seats and arrival time. So, the cars were just logically divided into classes and the only difference between the classes was the price. First class passengers were paying more than second class passengers. Although the quality is the same, some people were interested in using first class cars at high price since they knew that the first class cars would be less crowded. PMP claims the analogy between this old city metro system of Paris and the Internet. In the Internet, all the packets receive the same quality of service in a best effort manner.

PMP proposes to divide the Internet into fixed capacity logical sub-networks each of which has a long-term flat price. Each sub-networks has a different price, which will cause users to classify themselves. The only difference between PMP and old metro system of Paris is that packets of different sub-networks are treated differently at the routers whereas in old metro system of Paris passengers of different classes were being treated equally. The traffic belonging to higher priced sub-networks are served first at the routers, which is essentially a priority-based service system. Notice that PMP differs from many other pricing proposals by not having a closed-loop system. In other words, PMP does not introduce any feedback mechanisms to make the pricing. Also realize that it is priority-based and imposes no bandwidth controls.

Odlyzko proposes that PMP will make users to adjust themselves and QoS will almost be guaranteed to the users using the highly priced sub-networks. In [40], authors developed an analytical model for PMP for a single provider case and showed that PMP can be profitable to the provider under appropriate price settings based on demand behavior.

In order to implement PMP, the routers need to maintain a priority queue. Each priority corresponds to a sub-network. The packets need to include a priority value which was set by the service provider to the customer. Basically, the customer signs up a contract and the service provider sets the priority values of the customer's packets according to that contract. The priority value can be inserted into the available TCP/IP headers.

Although it is reasonably easy to implement PMP, it has low network efficiency since the sub-networks with higher prices will be under-utilized. Also, the number of sub-networks will be an issue since it represents the number of service classes and choices available to customers. That number will be restricted by the current header fields.

Similar to PMP, Dube et al. [24] proposed Tirupati pricing scheme where users pay different prices for joining to different queues that are being served by the same server in a round-robin fashion. Dube et al. develops stochastic model of such a queueing system, and shows that service quality perceived by users of each queue will be proportional to prices of the queues. Actually, the essence of Tirupati scheme is the same as PMP. PMP's "classes" correspond to Tirupati's "queues". The only difference is that Tirupati scheme allows possiblity of allocating more resources to one queue thant the others.

2.4.3 Expected Capacity Scheme

Clark, [19, 18], proposed a scheme that combines both capacity allocation and pricing for the Internet, named as *expected capacity*. Clark argues that the Internet users are generally not willing to have a guaranteed permanent capacity, since most of their traffic is very bursty. The reason for this is that users generally want to finish their transactions in a small amount of time, e.g. browse the Web quickly, transfer a large file quickly. For example, a user browsing the Web generally wants to open a web page in seconds, but reads it for minutes. Sometimes, the user wants to transfer a large file in seconds, but he/she does not do that permanently, probably once in 10 minutes. This user behavior causes very bursty traffic to the Internet. So, the users generally do not require a fixed guaranteed rate, rather they want to have their expectations satisfied. Clark uses these facts to support his argument for expected capacity scheme.

Expected capacity scheme proposes to have long-term contracts between users and service provider. The service related (other than price and contract term) parameters of the contracts are supposed to be (but not necessarily) maximum burst size and maximum burst frequency (e.g. 10MB of burst at every 10 minutes). Notice that these parameters also define the delay requirements of the user. The provider has to make sure that it can serve the expected rates (which can be derived from the service related parameters of the contract) of the users. Clark does not propose a way doing that for the provider, which is a lacking point in the scheme.

The prices of the long-term contracts can be defined either as flat-rate or usage-based or both. Accounting overhead needed for expected capacity scheme is very less. Although it might be possible to pose dynamic prices, it would not be useful since the contracts are supposed to be long, which does not allow frequent price changes. So, the scheme does not work for controlling instantaneous congestion epochs. Since Clark's does not define a way of implementing dynamic prices on expected capacity scheme, we classify it as a static pricing scheme.

2.4.4 Edge Pricing

Shenker et al., [70], proposed a general pricing architecture, named as *edge pricing*. Edge pricing defines a very general framework by proposing that the providers calculate their prices locally and advertise those locally computed prices. It is possible that provider A is buying service from another provider B. Provider A's customers can send packets that traverses through provider B, however those customers do not know about this, since provider A advertises the local price. So, in edge pricing a packet might be valued differently at different parts of the network. Assume that provider A is charging p_A per packet to its customers. Also assume that provider A bought service from provider B at a charge of p_B per packet. Initially, when the customer sends out the packet, its value is p_A. However, when that packet gets into provider B's domain, its value becomes p_B. This property of edge pricing makes it non-optimal since the packet owner does not pay the exact cost of the packet to the overall network. Shenker et al. did not provide any study on the amount of lost optimality, which is crucial in terms of economic equilibrium and stability of the scheme.

The providers can adapt dynamic prices into edge pricing by approximating congestion costs by monitoring the traffic at the edges. However, edge pricing scheme lacks a good way of approximating the dynamic (congestion-based) prices only by performing operations at the edges. Shenker et al. do not focus on dynamic pricing since they believe it is not plausible way of pricing the Internet service. This is why we classify edge pricing as a static pricing scheme.

Edge pricing provides scalability and allows each provider to implement its

Pricing Scheme	Dynamic	Granularity	Charging Time
Smart Market	yes	packets	aposteriori
Priority Pricing	yes	packets	apriori
Packet Marking	yes	packets	aposteriori
RNAP	yes	contracts	apriori
Flat-Rate	no	N/A	apriori
Time-of-Day	no	N/A	apriori
PMP	no	N/A	apriori
Expected Capacity	not defined	long-term contracts	apriori
Edge Pricing	not defined	packets	apriori

Table 2.1: Evaluation of pricing schemes according to various characteristics.

own pricing strategy. Also, accounting overhead is minimal too. As far as structural and implementation issues are concerned, edge pricing is a very reasonable pricing architecture, although it is sub-optimal from Pareto efficiency point of view. So, edge pricing proposes to sacrifice some (amount of which needs to be investigated) optimality in order to address implementation issues.

2.5 Taxonomy of Pricing Schemes

In the previous sections we surveyed major dynamic or static pricing proposals. There have been several other pricing proposals in the area. Some of them are as follows: Market Pricing [68, 67], CHiPS [30], Ex-Post Pricing [8], MARKETNET [81], Network Path Pricing [71].

In this section we provide a taxonomy of the pricing schemes evaluated in this paper. Table 2.1 shows the comparison of the surveyed pricing schemes according to three criteria.

- First criterion is if the pricing scheme is *dynamic or static*. Only the dynamic pricing schemes can achieve optimality in pricing.

- Second criterion is the *pricing granularity* (i.e. unit that the scheme makes charging) of the scheme. For implementation purposes finer pricing granularity levels are not desired.

- The last criterion is related to *charging time relative to service*. If the charging time is prior to service (i.e. *apriori*), the user knows how much he/she is supposed to pay for the service. This is generally the case for contracting frameworks. If the charging time is posterior to service (i.e. *aposteriori*), the user finds out about the total charge at the end of the service. Naturally, users want prior pricing, which is one of the reasons for the trend to flat-rate pricing.

2.6 Bandwidth Market

In this section we investigate economic models being used in bandwidth market. In bandwidth market, companies sell different types of capacity (e.g. point-to-point fiber or coaxial lines, satellite frequency band) with various parameters: assured rate, burstable rate, contract term. Several wholesaler companies have been in this billion-dollar market: Qwest, Nortel, AT&T, Verizon, Sprint, etc. These companies are mainly wholesalers, which lay down the physical links and sell their bandwidth capacity.

Just like the Internet, economic model of the bandwidth market has been wholesale-retail model. However, some companies have started to employ new economic models recently, e.g. Enron, RateXChange, TelcoExhange. These companies implement the concept of *bandwidth intermediary*, where several different functionality can be provided to both buyers and sellers. A bandwidth intermediary can function in different ways: *exchange, broker*. In a bandwidth exchange, buyers and sellers meet with each other and perform trading. The bandwidth exchange itself earns revenue by charging fees to subscribers (i.e. the buyers and sellers) or by charging commissions for each trade. So, a bandwidth exchange is basically a marketplace similar to stock market. A bandwidth broker actually buys bandwidth from wholesalers and sells it to retailers or customers. So, bandwidth broker actually provides the bandwidth to its customers, whereas bandwidth exchange just provides mediation between bandwidth buyers and bandwidth sellers. Enron functions both as a broker and an exchange along with being a wholesaler itself. RateXChange functions just as an exchange, since it does not sell or own bandwidth by itself. At the other side, TelcoExchange functions just as a broker.

Enron, for example, owns physical links and also plays a role of bandwidth intermediary. So, along with selling its own bandwidth, Enron also buys bandwidth from other wholesalers and sells it. Bandwidth intermediary provides flexibility to perform several optimizations automatically. For example, given several wholesalers (bandwidth sellers), Enron can find the cheapest way of constructing a leased-line between two given points in the World. This is basically simplification of buyer's job. Enron also provides risk management tools for buyers and sellers to handle price uncertainty. Similar to stock market, buyers can buy a capacity at a predetermined price before a preset deadline in exchange for a premium, i.e. call option. Similarly, sellers can sell a capacity at a predetermined price before a preset deadline again in exchange for a premium, i.e. put option.

Our work in this thesis focuses on developing building blocks to implement these type of above-mentioned economic models in the Internet or other wide area networks.

2.7 Deployability: An example

Deployability of congestion-sensitive techniques and complex pricing plans on wide area networks requires flexibility in accounting and dynamism in provisioning. Recently, GigaBit Ethernet technology has attracted significant attention of industry. Given the advances in fiber technology which reduce propagation delay significantly, GigaBit Ethernet has been considered for metropolitan area networks (MANs) additional to traditional usage of it in local areas networks (LANs). Major companies, such as AT&T, started to invest in this trend, [63]. GigaBit Ethernet vendors are able to provide Ethernet-based MAN service, however they use IP-based transmission to implement inter-MAN transport.

GigaBit Ethernet provides nice flexibility of dynamism in provisioning. Within seconds, the vendor can change bandwidth provided to user. GigaBit Ethernet vendors (e.g. Yipes Communications, Cogent Communications, Telseon) offer their customers this flexibility along with real-time price quoting. Within 5 seconds, [7, 54], the customer can quote the current price and increase/decrease his/her contracted bandwidth in units of 1Mb/s.

So, GigaBit Ethernet provides environment for implementation of adaptive pricing on public networks.

2.8 Summary

We surveyed major pricing proposals for the Internet. The debate on whether to implement dynamic pricing schemes or static pricing schemes is still going on. It is evident that Internet users are seeking simplicity in pricing, as a result they desire flat prices. This shows that any plausible dynamic pricing scheme must address this issue by making sure that the scheme will be able to provide simple pricing plans to the users.

We believe that the only way of implementing dynamic pricing schemes is to provide an intermediary agent that can handle fluctuations of dynamic prices and advertise less fluctuated prices to the users. Otherwise, no dynamic pricing scheme will be desired by both users and providers. The users naturally want flat prices. The providers also choose to advertise flat (or close to flat) prices since they do not want to irritate the users and loose market share.

Another important problem for dynamic pricing schemes is to address implementation and structural issues. Most of the dynamic pricing proposals in the area are not practical and will have to stay in theoretical domain in future. For the ones that are practical, it is still most likely that they will not be deployed on individual users (e.g. dial-up or DSL users) because of their extensive overhead relative to current Internet speed. Deployment for the dynamic pricing schemes may be possible if the relative overhead is small enough. This can happen in future when most Internet users have high speed access and total demand is large.

By using underlying edge-to-edge congestion detection and control mechanisms, our work specifically addresses implementation and structural issues to implement dynamic pricing on large networks. Within consideration of implementation issues, we also determine ways of using adaptive pricing to help congestion control and to improve fairness in large networks.

CHAPTER 3
ADAPTATION OF SMART MARKET TO DIFF-SERV ARCHITECTURE

3.1 Introduction

One of the earliest pricing proposals is MacKie-Mason and Varian's [50] Smart Market, which is specifically designed to address both resource allocation and congestion management issues and therefore is referred to as a congestion-sensitive pricing scheme. This scheme charges the users on a packet-by-packet basis depending upon the current level of congestion in the network, and the users in turn lower their demands according to their utility. In other words, it theoretically achieves both economic efficiency (optimal distribution of capacity among users [75]) and network efficiency (e.g. high utilization, low queue length) goals. However, the way it is defined, Smart Market is not possible to implement because it requires several upgrades to the current wide area networks.

In this chapter, we study the Smart Market and focus on developing a baseline implementation strategy and simulation of it on the diff-serv [27] architecture. We specifically investigate necessity of packet sorting (to be explained in Section 3.2) at routers. We also investigate Smart Market's abilities in terms of service differentiation and fairness.

The chapter is organized as follows: First, we describe details of the Smart Market briefly in Section 3.2. Next in Section 3.3, we outline our implementation strategy for the Smart Market and determine deployment limitations for it. Section 3.4 develops a set of simulation experiments and presents results along with discussions. Finally, Section 3.5 states implications and conclusions of the work and proposes possible future work.

3.2 The Smart Market Scheme

This section presents important and overall characteristics of the Smart Market scheme. The Smart Market imposes a per-packet-charge, which reflects incremental congestion costs (which could be zero). The price-per-packet varies dynamically depending on the level of congestion in the network. Users assign a "bid" value for each packet sent into the network. The network routers maintain a current threshold (cutoff) value and only pass those packets with bids greater than the cutoff value. This threshold value depends on the level of congestion at the particular router, and is adjusted by that router. Finally, user pays the highest threshold value among all routers that it passed through, called the "market-clearing price". Moreover, the successfully sent packets are sorted according to their bid values at each router. This behavior simulates an auction where the capacity is divided among the bidding packets on a packet-by-packet basis.

Though the Smart Market scheme is theoretically attractive, we can observe some implementation and deployment difficulties. For example, the smart market scheme does not provide a guaranteed service to users and can lead to packet re-ordering. Moreover, the "bidding" procedure requires support at end-systems (or proxy agents) and the "clearing" procedure is required at all potential bottlenecks in the network [50]. Within these limitations, we now design an implementation strategy for the Smart Market scheme.

3.3 Implementation Strategy

There are two key implementation issues of the Smart Market scheme:

- How to communicate the necessary information (customer's bids, network's charges) between customers and the network?
- How to calculate threshold value at a local router?

We now address these issues within diff-serv framework.

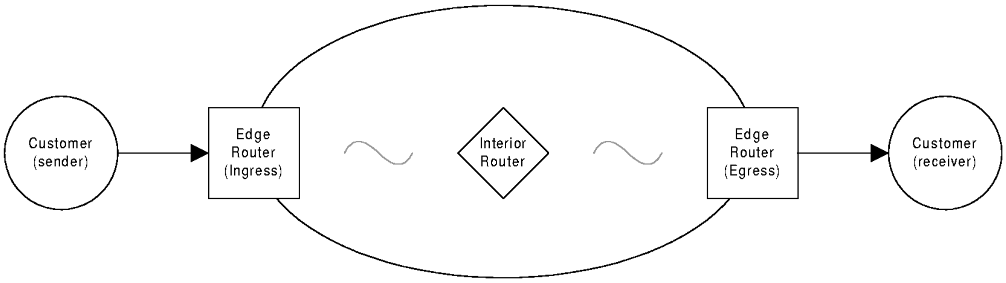


Figure 3.1: Representation of Differentiated-Services Architecture.

3.3.1 Communication Between Customers and The Network

The diff-serv architecture classifies network routers (or nodes) into two types: edge router (ER) and interior (core) router (IR). It constrains complex data and control plane functionality to be implemented at network edges to simplify the core. For a traffic flow passing through a diff-serv domain, the ER at the entry point is called as *ingress*-ER and the one at the exit point is called as *egress*-ER.

To implement Smart Market on diff-serv, we propose that the sender (or proxy) sets the bid value, b, in the packet and sends it to the network. The packet passes through an ingress-ER and series of IRs, each of which has a threshold value T. IRs simply drop the packet if it does not satisfy the condition $b \geq T$. If the packet satisfies the condition, it is placed into a priority queue and sorted according to its bid value. The priority queue may potentially reorder packets, which leads to negative effects on TCP congestion control [3, 10]. We will investigate importance of the sorting of packets later by experiments in Section 3.4.

For the communication of the bid and clearing price, there is a need for two fields in packet headers. The *bid field* is written only by the customer and is read by IRs. The *clearing price field* (initialized to 0 by the customer) is updated at each IR to the maximum of the prior value of the field and the current threshold value, T, at that particular IR bottleneck. In other words, if the value of the threshold, T, is greater than the value of the clearing price field of the packet, the value of T is copied into that field. Else, the field is left unchanged. When the packet reaches the egress edge router, it contains the maximum of the threshold at all the IRs it passed through.

The egress-ER acts a measurement proxy and accounts for the clearing price of the packet. In other words, the source pays the clearing price determined by the egress-ER. Egress-ERs accumulate each packet's clearing price and send periodic indications to each corresponding source. So far, everything in the Smart Market can be adapted to the diff-serv architecture, albeit with two new fields in the packet header. However, the information required by the customers in order to adjust their sending rates and bids require new feedback mechanisms. Theoretically, the Smart Market assumes that the customers are fed back such information immediately without any delay, which is not possible to implement on a wide-area network. So, an approximation is needed.

We propose to use a simple probing procedure which happens at fixed-time *probing intervals* set to be larger than round-trip time (RTT) as a way to handle this feedback problem. The customer (or the ingress-ER on behalf of the customer) sends a *probe packet* (in addition to data packets) to investigate the current status of the network at these fixed time intervals. This probe packet goes through IRs and finds the current clearing price. The egress-ER receives this probe packet and sends the customer a *feedback packet* containing the current clearing price of the network. The feedback packet also includes the total of clearing prices (bill) for that customer in the latest interval. The customer uses the feedback information to adjust her packet's bid values, capacity demand (number of packets to send) and available budget.

Note that if we want the IRs to treat the probe packets and the feedback packets just like data packets, they must have bid values as high as possible to ensure that they will not be lost and will encounter minimum delay. That means there has to be a maximum value for the bids of the data packets, which is a deviation from the Smart Market because it does not impose any limiting value for the bids. The fixed length bid field also implicitly constrains the size of bids. Alternatively, the IRs of the network have to behave differently for these probe and feedback packets. However, this will increase the processing time of a packet at the IRs, i.e. each packet will be checked whether it is a data, probe or feedback packet. We choose to normalize the bid values into a range (e.g. 0 to 1) and hence define a maximum value for the bids, i.e. 1. Once normalization (mapping to [0,1]) is done, there must also be a way of reversing this mapping back. What is going to be the actual money in dollars to charge for a clearing price of, for example 0.75? We currently leave this question, which is important for the service providers, unanswered. We simulated the Smart Market in ns [1] according to the ideas presented above. In the next section, we present the details of the simulation and our assumptions.

3.3.2 The Threshold Value T

MacKie Mason and Varian [50] determine the congestion price of a packet as

$$T = p = \frac{n}{K} D'(X/K) \tag{3.1}$$

where n is the total number of customers in the network, K is the capacity of the network, X is customer's capacity demand, and $D(X/K)$ is the delay experienced by customer. So, we propose that the IRs maintain fixed time *update intervals* at the end of which they calculate the rate of change in the delay, $D'(X/K)$, and update the threshold value, T. Specifically, threshold value for update interval i is calculated as follows:

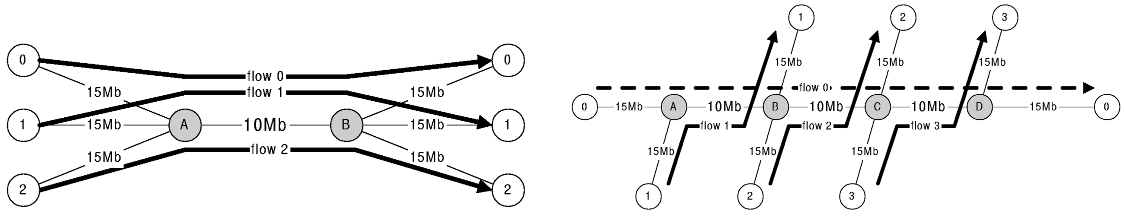
$$T[i] = \begin{cases} T[i-1] & , X[i-1] - X[i-2] = 0 \\ \frac{n}{K}\frac{D[i-1]-D[i-2]}{X[i-1]-X[i-2]} & , otherwise \end{cases} \tag{3.2}$$

where $D[i]$ is the delay experienced by the customer in interval i. Notice that the term $\frac{D[i-1]-D[i-2]}{X[i-1]-X[i-2]}$ corresponds to the term $D'(X/K)$ in (3.1).

3.4 Simulation Experiments

3.4.1 User Model

We model the user's utility function with the well-known logarithmic utility function $u(x) = wlog(x)$ [42, 47, 44], where x is the capacity given to the user, and w is user's willingness-to-pay. User will maximize his/her surplus by making sure that her capacity demand is $x = w/p$ where p is the current clearing price. Notice



(a) Single-bottleneck network (b) Multi-bottleneck network

Figure 3.2: Topologies for Smart Market experiments.

that the parameter w is equivalent to user's budget for expenditures to network service. So, in our simulations, user i is initially assigned a budget value W_i, and at the end of every probing interval the user i adjusts her bids such that her capacity demand is w_i/p for the next interval.

3.4.2 Experimental Configuration

We perform our experiments on single-bottleneck and multi-bottleneck topology. The single-bottleneck topology has a bottleneck link, which is connected to n edge nodes at each side where n is the number of users. The multi-bottleneck topology has $n-1$ bottleneck links, that are connected to each other serially. There are again n ingress and n egress edge nodes. Each ingress edge node is mutually connected to the beginning of a bottleneck link, and each egress node is mutually connected to the end of a bottleneck link. All bottleneck links have a capacity of 10Mb/s and all other links have 15Mb/s. Propagation delay is 0.1ms on bottleneck link(s) and 10ms on all other links. Figure 3.2-a shows a single-bottleneck topology with $n = 3$. Figure 3.2-b shows multi-bottleneck topology with $n = 4$. The white nodes are edge nodes and the gray nodes are interior nodes. These figures also show the traffic flow of users on the topology. To ease understanding the experiments, each user sends its traffic to a separate egress. For the multi-bottleneck topology, one user sends through all the bottlenecks (i.e. long flow) while the others cross that user's long flow.

In order to investigate importance of packet sorting (please refer to Section 3.3.1) on performance, we simulate two versions of Smart Market: Smart Market with Sorted Queues (SM-SORTED), Smart Market with FIFO Queues (SM-FIFO).

We run experiments both on UDP and TCP traffic. The UDP traffic will have an average packet size of 1000B.

The initial value of T at all IRs is set to 0.1. The probing interval at ERs and the update interval at IRs are set to 1sec, unless otherwise said so.

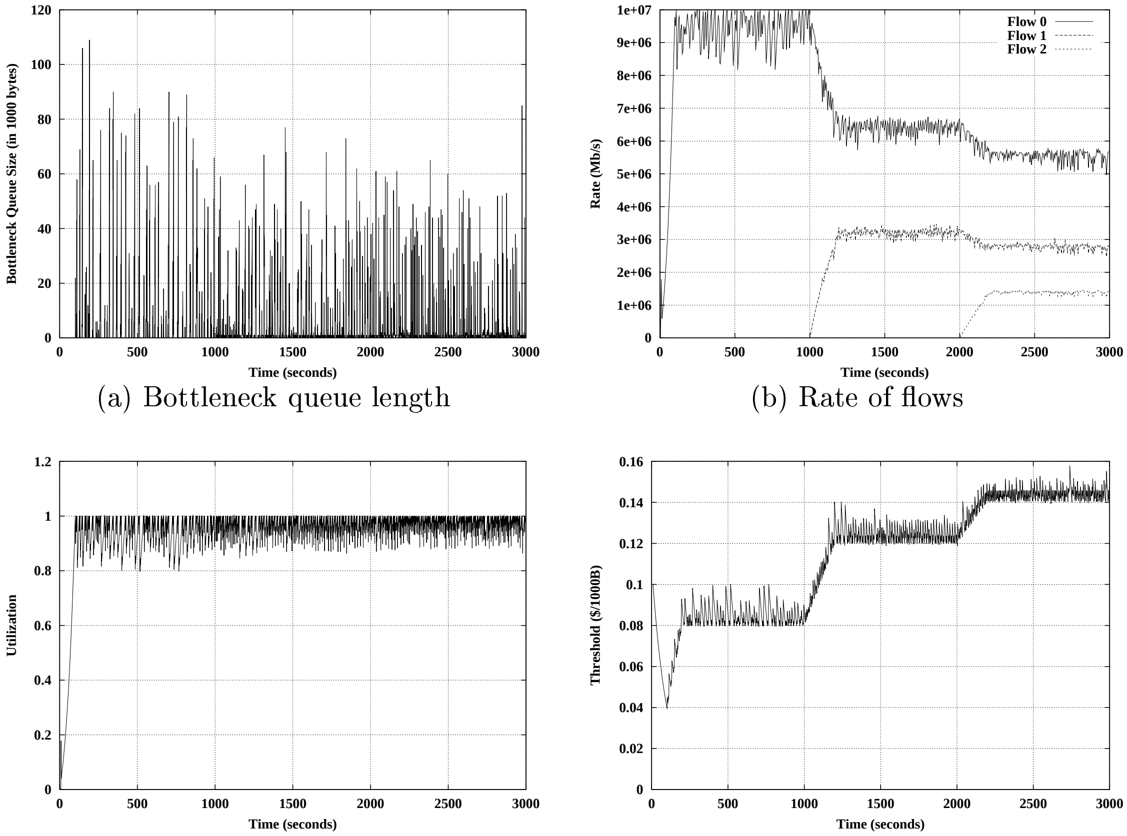
3.4.3 Experiments

3.4.3.1 System Dynamics and Stability

To see Smart Market's dynamics, we first run an experiment of SM-SORTED on the single-bottleneck topology with three flows as represented in Figure 3.2-a. The flows generate UDP traffic. Budgets of flows 0, 1, 2 are $w_0 = 100$, $w_1 = 75$, $w_2 = 25$ respectively. Total simulation time is 3000 seconds. Initially, only flow 0 is active, and the other flows get active with 1000 seconds intervals. Figure 3.3-a shows the instantaneous queue at the bottleneck, and Figure 3.3-c shows bottleneck utilization. Controlled queue and high utilization show that Smart Market provides stable operation. Figure 3.3-b shows instantaneous rates of the flows. We can observe that flows share the bottleneck capacity in proportion to their budgets. Figure 3.3-d shows the instantaneous threshold value at the bottleneck. As new flows join in, Smart Market adapts its threshold value accordingly.

In order to compare SM-SORTED and SM-FIFO in terms of system dynamics, we run a series of experiments on the single-bottleneck topology for various values of the update and probing intervals from 1 to 30 seconds. Moreover, to see traffic effects we run the experiments for both UDP and TCP traffic. Figures 3.3-e, 3.3-f, and 3.3-g show average bottleneck queue length, maximum bottleneck queue length, and average bottleneck utilization respectively for various values of update and probing intervals. We observe that SM-SORTED performs significantly worse (approximately 30%) in utilizing the bottleneck, which causes almost zero average and maximum queue length. This is because packet sorting causes extra timeouts and hence causes TCP to back off unnecessarily. Also, it causes Duplicate-ACKs to be generated, which then causes the TCP source to trigger fast re-transmit phase by halving its window size.

In general, we observe that packet sorting affects system performance nega-



(a) Bottleneck queue length

(b) Rate of flows

(c) Bottleneck utilization

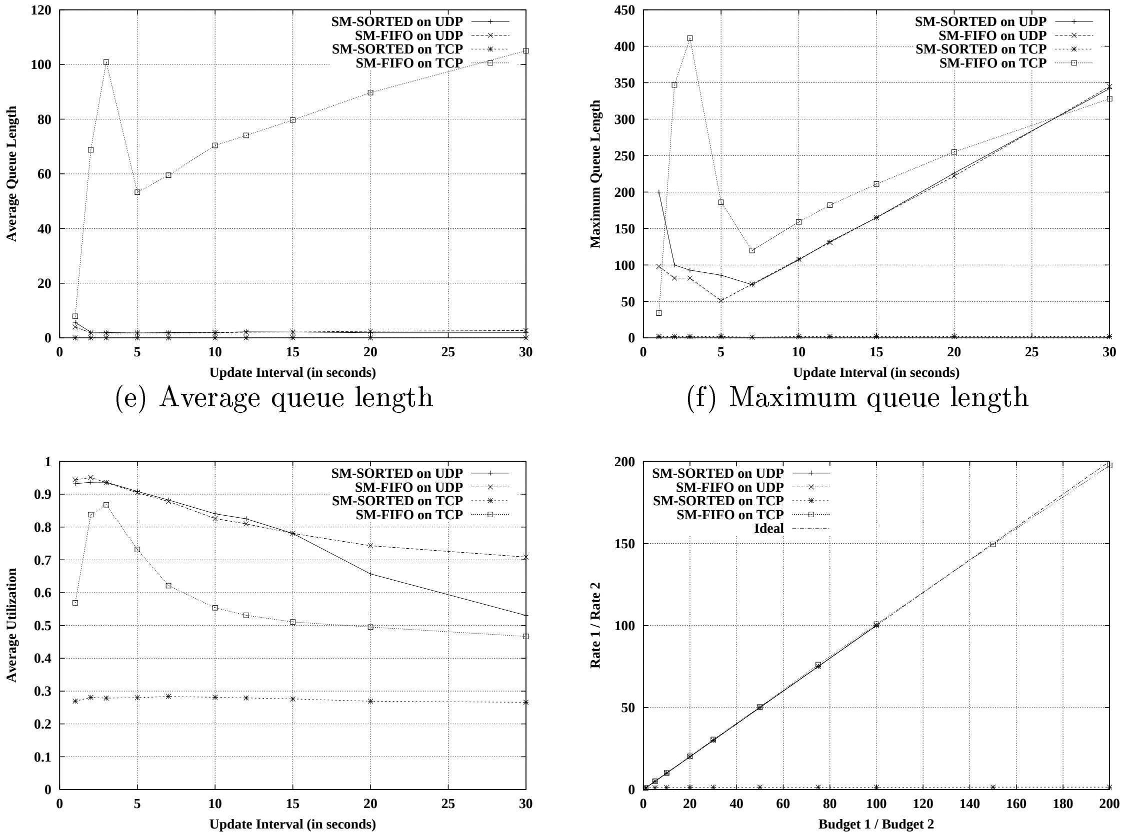
(d) Threshold at the bottleneck

Figure 3.3: Simulation results of SM-SORTED with UDP traffic on single-bottleneck topology.

tively. We can see this by comparing results for SM-SORTED and SM-FIFO on TCP traffic, and also by comparing results for SM-SORTED and SM-FIFO on UDP traffic. For example, SM-SORTED on TCP traffic utilizes bottleneck a lot less than SM-FIFO on TCP traffic. Also, from Figure 3.3-g, SM-SORTED on UDP traffic utilizes less than SM-FIFO on UDP traffic when update interval exceeds 15 seconds.

Overall, both versions of Smart Market (i.e. SM-SORTED and SM-FIFO) perform better on UDP traffic than TCP traffic. This is mainly due to burstiness of TCP traffic.

Also, we can observe that as the update and probing intervals get larger performance metrics get worse for all the cases. This is because Smart Market's fidelity of control lowers as update/probing intervals get larger.



(e) Average queue length (f) Maximum queue length

(g) Average utilization (h) Service differentiation

Figure 3.4: Comparison of SM-SORTED and SM-FIFO on UDP and TCP traffic.

3.4.3.2 Service Differentiation

We run a series of experiments for SM-SORTED and SM-FIFO on the single-bottleneck topology with UDP and TCP traffic. There are two flows in the experiments and for each experiment we vary the ratio of their budgets from 1 to 200, i.e. $w_0/w_1 = 1..200$. Given the budget ratio w_0/w_1, we then observe ratio of the two flows's rates during the simulation. Figure 3.3-h shows results of these experiments. The horizontal axis shows the ratio of the flows's budgets, which is set at the beginning of simulation, and the vertical axis shows the ratio of the two flows's average rates observed during the simulation. Observe that only SM-SORTED on TCP traffic cannot differentiate the two flows, while Smart Market is able to differentiate between them pretty well in all other cases. The reason why SM-SORTED

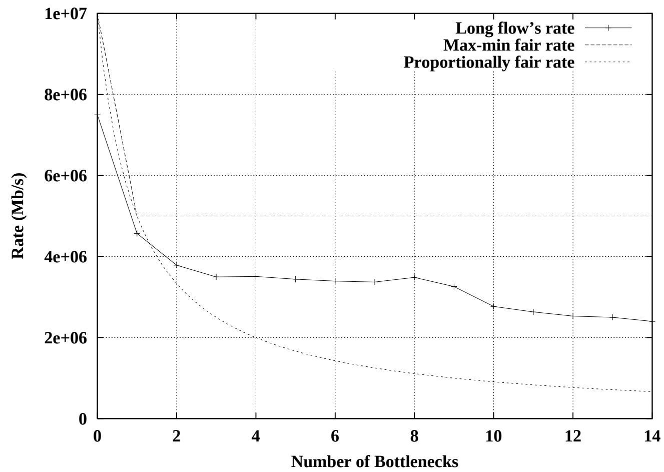


Figure 3.5: Simulation of SM-SORTED with UDP traffic on multi-bottleneck topology. Long flow's rate vs. number of bottlenecks.

on TCP does not perform well in service differentiation is again due to the negative effects of packet sorting on TCP performance.

3.4.3.3 Fairness

Our last series of experiments are on the multi-bottleneck topology with $n = 14$ bottlenecks (the case of $n = 3$ is shown in Figure 3.2-b). All the flows have equal budget of 10 \$/Mb, and they generate UDP traffic. We simulate SM-SORTED. Our aim is to observe behavior of the long flow (i.e. flow 0)'s rate as number of bottlenecks on its way increase.

At time 0, only the long flow is active. The other flows (i.e. cross flows) join in one after another with an interval of 1000 seconds. Total simulation time is 15,000 seconds. So, as new flows join in the number of bottlenecks in the system increases. Figure 3.5 shows the long flow's rate as the number of bottlenecks increases on its way. The figure also plots theoretical rates of the long flow for max-min and proportional fair cases. In the max-min fair case, the long flow and the cross flows share the bottleneck capacity equally, i.e. 10/2 Mbps in our experimental topology. In the proportional fair case, the long flow gets less than the cross flows in proportion to the number of bottlenecks on its way, i.e. the long flow gets $10/(r+1)$ Mbps and the cross flows get $10r/(r+1)$ Mbps in our experimental topology. We observe from Figure 3.5 that Smart Market allocates the bottleneck capacity in such a way that it is between max-min and proportional fair rate allocations.

In the definition of Smart Market, each flow pays the clearing-price for its route, which is the maximum of the bottleneck thresholds in the route. For our experimental topology, the long flow should be paying approximately the same price as the cross flows since the bottleneck capacities are equivalent. So, one may expect that the capacity allocation in our experiments should be max-min fair. The result shows that it is not. The reason behind this is that the long flow is experiencing more delay (both propagation delay and queuing delay) than the cross flows. This makes effective capacity for the long flow less than it is supposed to be, which in turn causes the long flow to get less of the capacity than the cross flows.

3.5 Summary

We investigated the difficulties in implementing Smart Market, a well-known congestion-sensitive pricing scheme for the Internet, on a network with diff-serv architecture. We found that Smart Market cannot be implemented to a real network without important changes (e.g. modeling, packet format, architectural issues). Also, it has limitations on deployment (e.g. requires upgrades in both hosts and routers), and is very sensitive to offered workload (e.g. effect on TCP flows). We proposed the following major changes to implement the Smart Market on diff-serv architecture:

- delay in feeding back the congestion information of the network to the customers
- mapping the threshold value of the interior routers to an interval such as [0,1]
- concentrating more functionality at the ERs versus less functionality at the IRs to suit a diff-serv implementation

By applying the above changes, we developed a packet-based simulation for Smart Market and presented simulation results along with their analysis. We observed that Smart Market is able to control congestion with low bottleneck queue length and high bottleneck utilization. Also, we observed that packet sorting at IRs affects system performance negatively when traffic type is TCP. To see the real importance of packet sorting in Smart Market's performance we simulated the Smart Market with and without packet sorting, i.e. SM-SORTED and SM-FIFO. Simulation results showed that packet sorting does not really improve system performance. In fact, it degrades system performance especially on TCP traffic, which is currently the dominant traffic type in the Internet. Also, SM-FIFO is a lot easier to implement than SM-SORTED. Because of these reasons, it makes more sense to implement SM-FIFO.

Also, we have shown by simulation that Smart Market provides fairness in between max-min and proportional. Future work should consider multiple diff-serv domains case, and Smart Market's behavior on bursty traffic patterns.

In general, open question is whether one desires congestion pricing or not on the long run. This chapter shows that congestion pricing is implementable using flexibilities offered by diff-serv architecture.

CHAPTER 4
DYNAMIC CAPACITY CONTRACTING (DCC)

4.1 Introduction

Most of the proposed pricing schemes have remained in theoretical domain due to lack of models for implementing them using IP. In this chapter, we focus on developing a pricing framework, Dynamic Capacity Contracting (DCC), that utilizes the advanced traffic management features offered by the differentiated-services architecture. The main focus of this chapter is to address pragmatic new ideas in Internet pricing and related technical and deployment issues. However, we do not focus on complete refinement of the schemes or thorough performance analysis. This is the topic of later chapters.

The rest of the chapter is organized as follows. We first position our new scheme, DCC, within the literature in Section 4.2. Then in Section 4.3, we describe DCC framework: a flexible and dynamic framework to implement a range of pricing schemes within the diff-serv architecture. In Section 4.3.1, we describe two congestion based pricing schemes developed using this framework. Further in Section 4.4, we present performance results to demonstrate the potential of this framework. Finally in Section 4.6, we summarize the work and give ideas for possible future work.

4.2 Related Pricing Proposals

Among the proposed pricing proposals, flat-rate pricing [26], is the most common mode of payment today for bandwidth services, and is popular for several reasons. It has minimal accounting overhead, and encourages usage. However, flat rate pricing has problems. During congestion, the marginal cost of forwarding a packet is not zero, and flat pricing does not offer any (dis)-incentive for users to adjust their demand, leading to potential "tragedy of commons" [33]. Our view is that there is no congestion issue if capacity and provisioning speed exceeds demand. But, there exist several niches (e.g. international links, tail circuits to remote markets, peering points or complex meshed networks) where bandwidth, even though technically available, cannot be added fast enough. This is because the company probably does not own the links (or jointly-owns it with a partner, or no company owns them), or the part of the network is so large that carrier-class upgrades take time (upgrade-cycles of one year are quite common).

Two prominent pricing proposals are:

- to regulate usage by imposing a fee based upon the amount of usage (data or connection time) actually sent (called usage-based pricing)
- use a fee based upon the current state of congestion in the network (called congestion-sensitive pricing).

On the commercial side, usage-based pricing has been deployed. ISPs are starting to sell OC-3 (155 Mbps) access rapidly, but with usage-based pricing on port-usage (not on an end-to-end basis). Usage is typically measured over 5 min intervals and a monthly charge is assessed based upon the average of these measurements after removing the extra-ordinary values. The rates vary around $600-800/Mbyte/month. The problem with usage-based pricing is that usage costs are imposed regardless of whether the network is congested or not. Further, it does not address the congestion problem directly, though it does indirectly make users more responsible for their demands. Also, some users may not like a posterior pricing unless it is a very small part of their overall expenditures.

Researchers, on the other side, have been studying congestion-sensitive pricing recently. Some of the schemes that have been proposed are: MacKie-Mason and Varian's Smart Market scheme [51, 50], Gupta's Priority Pricing scheme [33] and Kelly's Proportional Fair pricing and rate-allocation scheme [42].

Clark, [19], proposed an Expected Capacity scheme where users pay a price for high expectation of delivery of a given volume of traffic. Note that this scheme is not congestion-sensitive. This "contracting framework" requires simple negotiation between a customer and a provider and does not require end-system support. Though it does require some network upgrades, it was appealing to both ISPs and users, and as a result inspired the development of the differentiated-services architecture itself.

As a purely congestion-sensitive scheme, Smart Market is ideal in terms of performance. Theoretically, it achieves economic and network performance goals. However, as we studied in Chapter 3, it is very hard to implement.

In summary, while the congestion-sensitive proposals do not have a clear deployment or service-assurance models, Clark's model is not congestion-sensitive and is not able to address all performance goals. Our work is an effort to provide a middle ground between these approaches.

4.3 DCC Framework

In this section, we describe the proposed "Dynamic Capacity Contracting" (DCC) framework. This framework extends Clark's model to incorporate short-term contracting and adds mechanisms to make it congestion-sensitive. DCC framework provides opportunity for making congestion-sensitive pricing of short-term contracts. The most important features of DCC framework are its granularity level and its easy implementation on diff-serv architecture.

Similar to Expected Capacity scheme, DCC framework introduces granularity level of "contracts" rather than "packets". The reason we need short-term contracts for congestion-sensitive pricing is because long-term contracts do not give the flexibility to change the current price of a contract based upon congestion. Short-term contracts naturally expire and force re-negotiation, at which point provider can revise the price based upon congestion measures of the network. One key issue is the maximum length of these short-term contracts. We will look into details of this issue in later chapters.

We can model a short-term contract (service) for a given traffic class as a function of price per unit traffic volume, maximum volume (maximum number of bytes that can be sent during the contract) and the term of the contract (length of the contract):

$$Contract = f(P_v, V_{max}, T) \tag{4.1}$$

Throughout the thesis, we will assume that the user can send up to the maximum volume negotiated within the term of the contract. We do not investigate

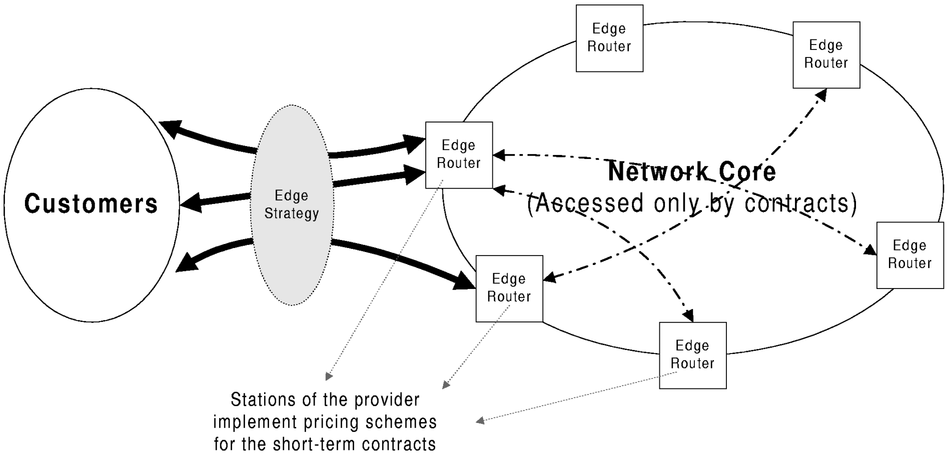


Figure 4.1: DCC framework on diff-serv architecture.

possible strategies when the user sends more than contracted volume, since this is an issue for admission control and hence out-of-scope for this thesis. As in Clark's Expected Capacity model the provider will assure that the contracted traffic will be carried with a high expectation of delivery. In general, the user may send this traffic to any destination of its choice (i.e. a point-to-anywhere service); however again throughout the thesis, we focus on the case of point-to-point service since the measurement of congestion information in the former case is non-trivial, and is a very broad research area. We make one simplification to Equation 4.1 by assuming that the term parameter, T is fixed i.e. different users cannot choose different term values. In short, the provider offers the user service with the following characteristics:

- the flexibility to contract a desired volume, V, up to a predefined maximum volume V_{max}
- a predefined price per unit volume, P_v, for the contract.
- a fixed contract length, T

The other important property of DCC framework is its possible implementation on diff-serv architecture. Figure 4.1 shows big picture of implementing DCC framework on diff-serv architecture. Customers can only access network core by making contracts with the provider stations placed at the edge routers. Accessing to contracts can be done in different ways, what we call *edge strategy*. Two basic edge strategies are "bidding" or "contracting", which will be described in detail later in Section 4.3.2. We leave performance analysis of different edge strategies for further research and use contracting.

The provider may advertise different or the same prices at the stations. The provider stations apply a pricing strategy to identify prices of the short-term contracts. Currently we assume that the same price is being advertised at all stations. We call the framework in which different prices are advertised at stations as *Distributed-DCC*, which will be developed in later chapters.

Stations can perfectly advertise congestion-based prices if they have updated information about the congestion level in the network core. This congestion information can come from the interior routers or from the egress edge routers depending on the congestion-monitoring mechanism being used. In our studies, we use an edge-to-edge congestion detection scheme developed by another group in our team [35]. Our framework assumes that the congestion detection mechanism is able to give congestion information in small time-scale (i.e. what we call *observation interval*) by saying whether the network was congested during the last observation interval or not.

In summary, DCC framework has been designed to use pricing and dynamic capacity contracting as a new dimension in managing congestion, as well as to achieve simple economic goals. The key benefits of our framework are:

- a congestion-sensitive pricing framework employable on diff-serv architecture
- does not require per-packet accounting (works at granularity of contracts)
- does not require upgrades or software support anywhere in the network except at the edges

In this sense, DCC is well positioned as a pragmatic intermediate approach between Clark's Expected Capacity approach and a purely congestion-sensitive scheme, e.g. Smart Market.

4.3.1 Pricing Schemes

The basic idea behind our pricing schemes is that the contract is sub-divided into smaller observation intervals and each observation interval is identified as "congested" or "not-congested" according to the feedback from the congestion detection mechanism. We now develop two pricing schemes for the provider stations, i.e. ingress edge routers. Development of better pricing schemes is an open issue.

4.3.1.1 Edge-to-Edge Pricing (EEP)

In this pricing scheme we assume that egress edge routers feed back the egress output rate, μ, at congested observation intervals and nothing is fed back during not-congested observation intervals. The EEP scheme basically uses that fed back μ to estimate the available network capacity and uses that estimation to adjust the price.

The provider stations advertise price per unit volume, P_v, based upon the following formula[6]:

$$Pv = \frac{\sum_i B_i}{\texttt{average\_rate\_limit} * \texttt{contract\_length}} \tag{4.2}$$

where $\sum_i B_i$ is the estimated total budget of customers for network service and `average_rate_limit` is the parameter represents the current estimated network capacity. When the network is not congested the `average_rate_limit` should increase such that the price becomes less, and vice versa. So, the `average_rate_limit` is calculated during each contract, and is what captures the "congestion-sensitivity" of the pricing scheme. Also notice that the denominator in Equation 4.2 calculates the total volume to be contracted and numerator calculates the total budget available for that volume. Hence, P_v becomes price per unit volume.

The parameter `average_rate_limit` is calculated in the following manner: The station keeps updating a parameter, `rate_limit`, at the end of every observation interval. During not-congested observation intervals, the station increases `rate_limit` using an additive increase policy. Specifically, the rate limit is incremented by `delta_rate_limit` = 1 packet/round-trip-time. In every congested ob-

[6]Optimality of this pricing formula will be discussed later in Chapter 8.

servation interval, the station equates the `rate_limit` to $\beta * \mu$, where β is a fixed fraction in (0.5,1), and μ is the output rate fed back by the egress node. The basic reason behind this setting is to prevent overpricing by guaranteeing that the `rate_limit` is not set to a value much lower than the network capacity. Since β is close to 1 and μ will be very close to the network capacity during congestion, this will make sure that `rate_limit` is not set to a very low value which would disturb capacity estimation. The `average_rate_limit` is simply the mean of `rate_limit` values of the observation intervals that passed during the contract. The `rate_limit` for the first observation interval in a contract is initialized to the `average_rate_limit` that was calculated at the end of the previous contract. Unlike traditional rate-based congestion control, we assume in this scheme that the `rate_limit` is not directly enforced, but is used indirectly to calculate prices and let the user reduce its demand based upon price.

The level of control is a function of how accurately the price can be set such that it neither overloads nor under-utilizes the network. The accuracy of the price value depends on two parameters in Equation 4.2:

- $\sum_i B_i$, i.e. budget estimation capability
- `contract_length`, i.e. how large the contract can be in order to be truly "congestion-sensitive" while balancing the transaction costs of the user

4.3.1.2 Congestion Index

In this pricing scheme, we assume that the provider station receives an indication during the congested observation intervals and receives nothing during the not-congested ones. Since the observations happen at a smaller time-scale, the ratio of the number of congested observation intervals to the total number of observation intervals is a measure of congestion. We call this measurement as *congestion index*. Let n be the number of observations during a contract, and r be the number of observations at which congestion was observed, then the congestion index for that contract is $z = r/n$. This congestion index and the congestion indexes of the previous contracts can be used to determine the price for the next contract, i.e.

$p_i = f(z_{i-1}, z_{i-2}, ...)$ where subscripts represent contracts in time. Several possible mappings (the f function) of congestion indexes to price can be done.

Notice that the accuracy of congestion index, z, depends directly on the number of observations per contract, n. On the other hand, having more frequent observation intervals causes more overhead of congestion indication traffic. So, one key performance trade-off of this scheme is the number of observation intervals.

4.3.2 Edge Strategies

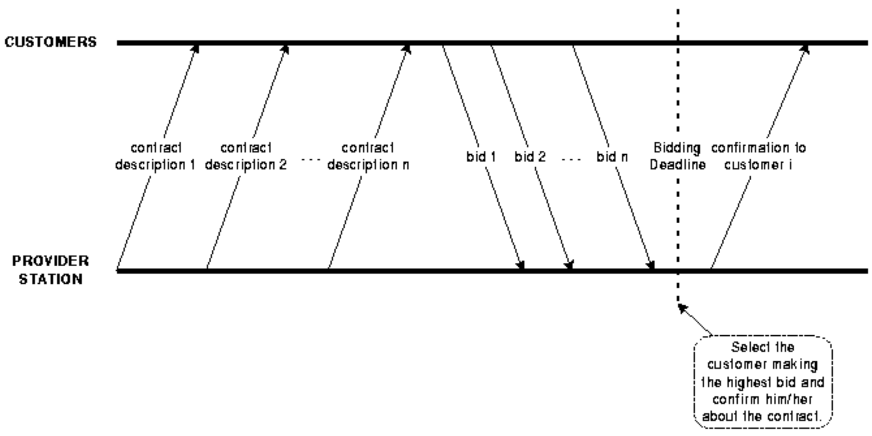
We now introduce two possible edge strategies that can be implemented as decision-making mechanism to identify which customer gets an available contract at the provider station. Edge strategy is important, because it:

- effects the rate allocation and fairness especially in cases where there is high competition for the contracts.
- has to have less transaction overhead because it will be used frequently at the beginning of short-term contracts.
- must handle the trade-off between having less complexity (i.e. less transaction costs) and fairer rate allocation.

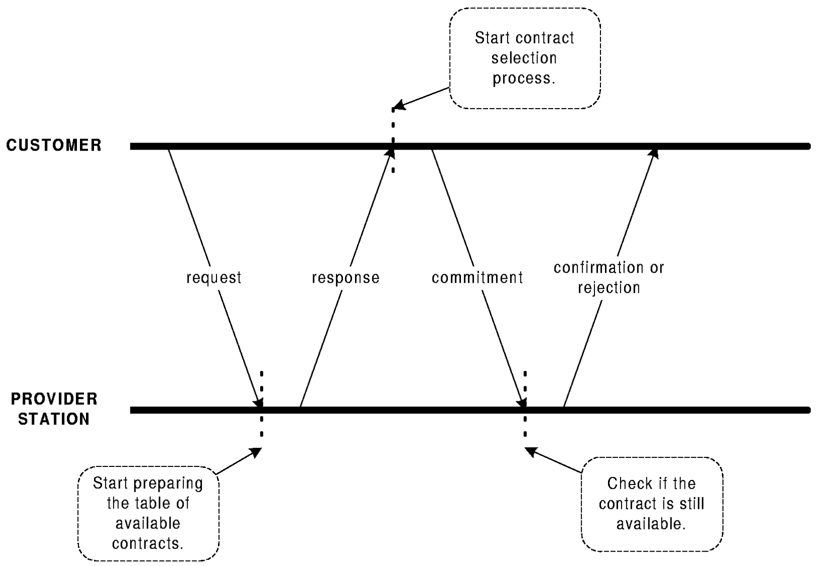
Defining better edge strategies and their performance analysis are open issues for further research.

4.3.2.1 Bidding

In bidding, the decision is made by the provider station. First, the customers show their interests in network service by joining to the station. The provider station sends *contract description* to the interested customers, and then the customers send their bids to the provider station until a time, *bidding deadline*, previously specified and advertised to the customers. The provider station, then, decides which customer will get the contract by selecting the highest bid. Finally, the provider station sends a *confirmation* to the selected customer. Figure 4.2-a visualizes the bidding edge strategy.



(a) Bidding



(b) Contracting

Figure 4.2: Two sample edge strategies.

4.3.2.2 Contracting

In contracting, customers make selection of the contract among a variety of contracts offered by the provider station. Basically, customers that are interested in making a contract send a *request* to the corresponding provider station. Then, the provider station sends a *response* that includes the table of available contracts to the interested customers. Next, the customer selects one of the contracts and informs the provider station about his/her selection by sending a *commitment*. Finally, if

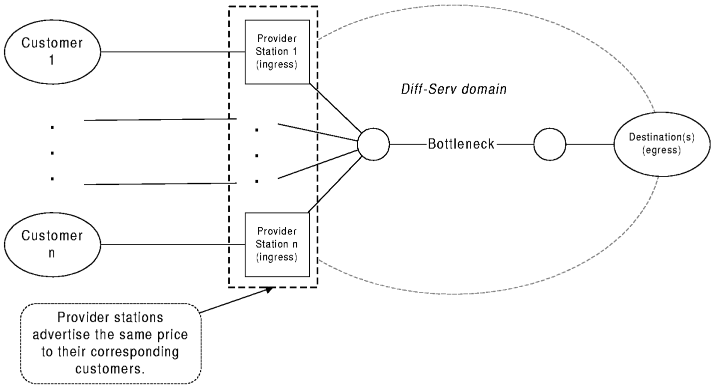


Figure 4.3: Configuration of the experimental network for DCC simulations.

the contract was not given to another customer, then provider station assigns the contract to the customer and sends a *confirmation* to the customer. Figure 4.2-b visualizes the contracting edge strategy.

4.4 Performance

In this section we first present configuration of the simulation experiments of DCC, and then we present our assumptions for customer and provider model. Finally we present simulation results along with their discussions.

4.4.1 Configuration of Experiments

In our experiments, we use a simple network configuration represented in Figure 4.3 and the network simulator, ns [1]. The configuration has multiple customers trying to send traffic to the same destination through a bottleneck. Each customer agent has a corresponding provider agent with which they negotiate for short-term contracts throughout the simulation. Although there are multiple instances of the provider they advertise the same price value to their customers. We use contracting as the edge strategy and EEP as the pricing scheme of the provider stations.

Length of the short-term contracts is assumed to be fixed and is set to 400ms. Length of the observation intervals is set to 80ms, which leads to 5 observations per contract. The round-trip-time (RTT) for a customer is 40ms. Bottleneck rate is

Experiment ID	Mean Budgets of Customers					Total Mean Budgets
	#1	#2	#3	#4	#5	
1	20	20	20	20	20	100
2	40	40	40	40	40	200
3	60	60	60	60	60	300
4	10	20	30	40	50	150

Table 4.1: Parameters of the experiments for DCC simulations.

1Mbps, and customers send UDP traffic shaped by a leaky bucket with fixed packet sizes (1000 bytes). All the queues are strictly FIFO. We constructed the queues such that they do not drop any packet throughout the simulation, since we want to see DCC's performance on providing assured service.

For all the experiments, total simulation time is 40sec and the number of customers is 5. We run four experiments with mean customer budgets defined in Table 4.1. The first three experiments have customers with equal budgets and the last experiment has customers with increasing budgets from 10 to 50 units. We do the first three experiments to see if the pricing scheme can find the optimum price per unit volume when the total budget of the customers is increased. The last experiment aims to find out whether the framework can allocate the bottleneck rate proportionally to the customers' budgets or not.

4.4.2 Customer and Provider Models

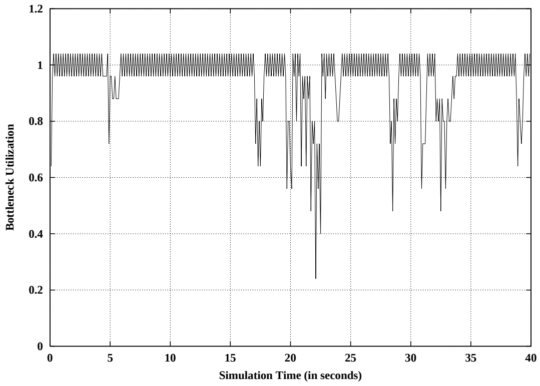
We assume simple models for our initial proof-of-concept. Provider calculates the expenditure that the customer made for the contract and uses that information in estimating the customer's budget for the next contract. The estimated budget for the next contract is the average of the customer's expenditures in previous contracts. For the simulations in this chapter we assume that providers advertise the maximum volume, V_{max}, as the bottleneck capacity. In other words, the providers offer a variety of contracts with a possible volume in the interval $[0, V_{max}]$. Better budget estimation methods and estimation of maximum volume in a multiple-route network are in the scope of future research.

Customers choose a desired volume of premium data traffic to be sent during the contract based upon the price per unit volume, P_v, a demand curve, and their available budget. The demand curve is assumed to be a simple hyperbolic curve between price and corresponding demand (volume), i.e. the volume that is contracted by a customer is calculated as $V = B_i/P_v$ where B_i is the budget of the customer. After this volume selection process, customers have to bound the volume by V_{max}, which was advertised by the provider. In such a case, any leftover budget is carried over to the next term. This choice of a volume is then conveyed to the provider. Observe that this contracting now defaults to the Expected Capacity framework proposed by Clark. Specifically, this scheme provides "service assurances" and is not just a "best-effort" service or a service whose quality is more probabilistic and dynamic like the Smart Market model. For the budget model, we assumed that customers have some budget for each contract. This budget value is randomized by truncated-Normal [74] distribution with a given mean as defined in Table 4.1.

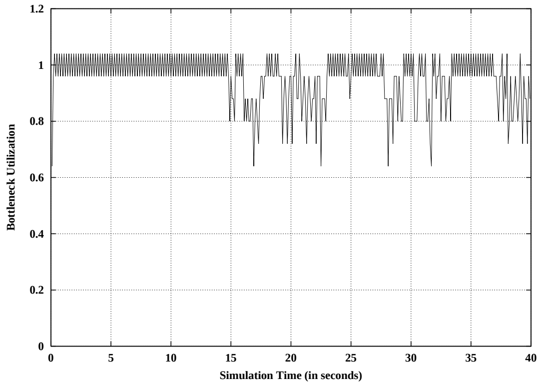
4.4.3 Results

We measure performance of the framework from two perspectives: network efficiency and economic efficiency. Network efficiency is measured by bottleneck utilization and bottleneck queue length. A network efficient framework must stabilize the system by utilizing the resources highly while keeping the queuing delays low. Economic efficiency is measured by looking at fairness of rate allocation and accuracy of the pricing. Specifically, rate allocation must be done proportional to the customers' budgets. Also, value of the price per unit volume must be such that it sells the bandwidth neither cheap nor expensive. In other words, pricing scheme must be such that it balances the customers' demands with the available bandwidth capacity by finding the optimum price value. We make Experiment 4 to see DCC' s performance on fair rate allocation and Experiments from 1 to 3 to see DCC' s performance on finding the accurate price.

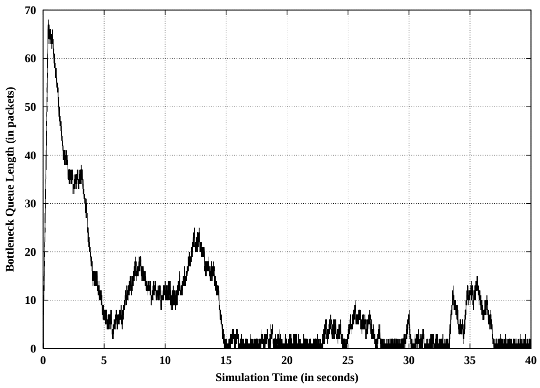
Figures 4.4-a, 4.4-b, and 4.4-d, 4.4-c show that DCC is able to utilize the bottleneck capacity while keeping the bottleneck queue within a reasonable bound. Figures 4.4-e and 4.4-f show that rate allocation is being done proportional to customers' budgets. In Figure 4.4-e, observe that five customers with equal budgets are



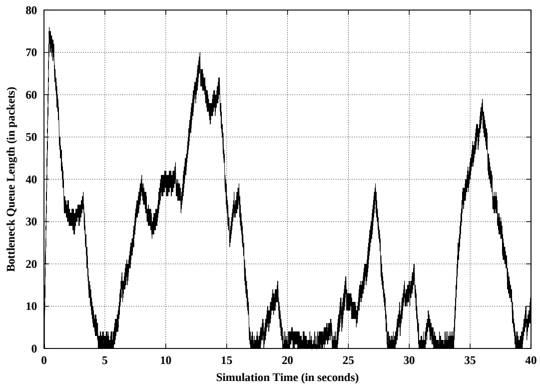
(a) Bottleneck utilization in Experiment 1



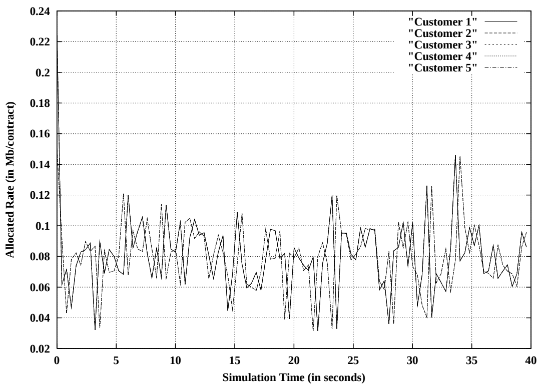
(b) Bottleneck utilization in Experiment 4



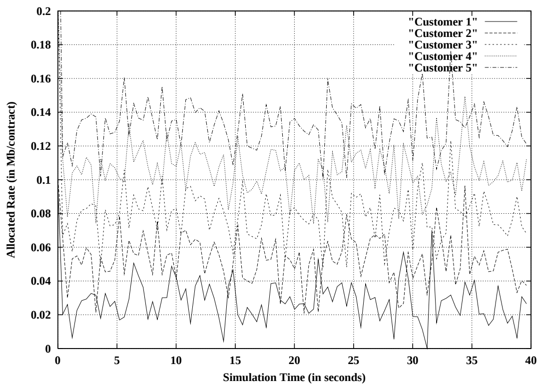
(c) Bottleneck queue length in Experiment 4



(d) Bottleneck queue in Experiment 1



(e) Rate allocation in Experiment 1



(f) Rate allocation in Experiment 4

Figure 4.4: Results of DCC experiments.

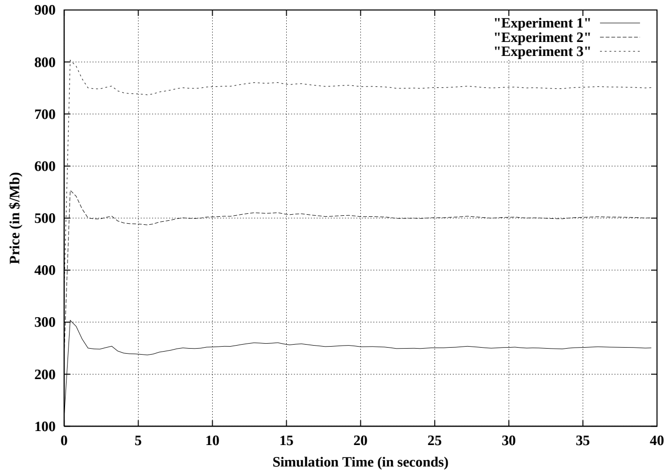


Figure 4.5: Instantaneous prices in Experiments 1, 2, and 3.

being allocated the same rate in Experiment 1, and also in Figure 4.4-f, customers with increasing budgets are being allocated increasing rates proportional to their budgets in Experiment 4.

An interesting result from Figure 4.5 is that DCC framework makes the provider stations to advertise the optimum price value according to total budgets of customers. Notice that in Experiments 1, 2, and 3, customers have total budgets of 100, 200, and 300 respectively. The figure show that the steady-state price value is adjusted proportional to the total budgets of customers, which prevents the bottleneck to be neither overloaded nor under-utilized.

4.5 Comparison of DCC and Smart Market

We conduct simulation experiments to compare the performance of the two pricing proposals: DCC and Smart Market. Our objective is to evaluate their performance in terms of both network efficiency and economic efficiency. The performance measures we use for both proposals are utilization, queue length, relative volume allocations.

We perform our experiments on a simple single-bottleneck network configuration. The bottleneck rate is 1Mb/s, and can be accessed by the customers through an edge router (corresponds to the provider). The customers send constant bit rate UDP traffic with fixed packet sizes (1000 bytes). The contract term in DCC and

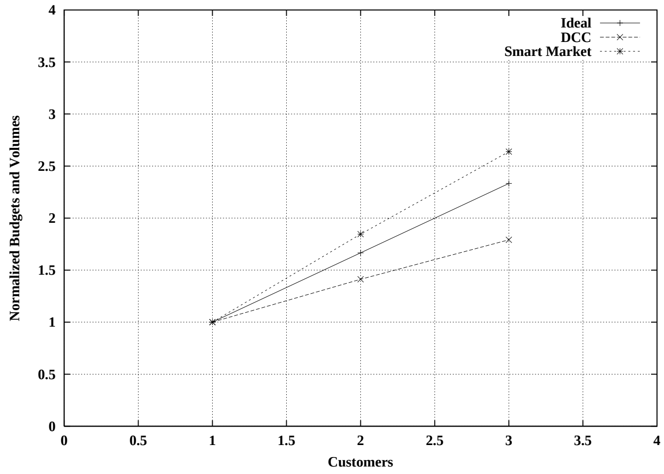


Figure 4.6: Normalized volumes given to the flows in DCC and the Smart Market.

the length of the update interval at interior routers in Smart Market are set to be 0.4sec. Also, the length of the observation interval in DCC is set to 80ms. In the experiment, there are three customers with unequal budgets (15, 25 and 35 units respectively for customers 1, 2 and 3).

Smart Market gives average volumes of 0.1Mb/s, 0.175Mb/s and 0.275Mb/s to customers 1, 2 and 3 respectively, while DCC gives 0.225Mb/s, 0.30Mb/s and 0.375Mb/s. Figure 4.6 plots the normalized values of volumes allocated to the customers. The solid line plots the normalized values of the customer budgets, i.e. the ideal volume allocation. The figure indicates that Smart Market allocates the volume to the customer almost proportionally to their budgets, whereas DCC allocation is a little less proportional to the budgets. This implies that in comparison to Smart Market, DCC has a lower economic efficiency. However, total volume allocated to customers is significantly higher in the case of DCC, i.e. 0.55Mb/s in the Smart Market versus 0.95Mb/s in DCC. This indicates that DCC better utilizes the bottleneck, which implies better network efficiency.

In summary, the experiments suggest that DCC is better from a congestion management perspective because it achieves a higher utilization. Interestingly, this is achieved without seriously distorting volume allocations, which are in fact, close to those attained by Smart Market. Nevertheless, from a pure economic efficiency perspective, Smart Market appears to fare better.

4.6 Summary

We have proposed a "Dynamic Capacity Contracting" (DCC) framework primarily inspired by the work of Clark [19] and MacKie-Mason and Varian [50], and the diff-serv architecture which provides a platform for implementation. The distinguishing features of our work include the idea of "short-term" contracts, mechanisms to support congestion-sensitive pricing of such contracts, use of pricing as a tool for congestion management, and a pragmatic focus on implementation issues. We have also proposed two sample schemes in this framework and showed experimental results of one of them to illustrate the potential of the framework.

DCC, however, has one important implementation problem on wide area networks: DCC assumes that the same price is being advertised at provider stations, which are indeed very far away from each other in a real wide area network. This assumption is unrealistic and needs to be relaxed by letting the stations to be able to calculate prices locally while maintaining stability and fairness of the overall network. We will explore this particular issue later in Chapter 6.

In later chapters of this thesis, we will investigate the following points in DCC:

- Studying optimality of EEP pricing scheme
- Exploring ways to implement DCC on a wide area network, especially by relaxing the assumption of advertisement of same price at provider stations
- Finding a maximum bound on the length of the short-term contracts such that congestion-sensitivity of prices is still maintained and control over congestion is not totally lost
- Addressing congestion control issues, especially regarding co-existence of DCC with underlying edge-to-edge congestion control schemes
- Addressing fairness issues, especially regarding price discrimination among different traffic flows

Further research may include the following open points:

- Using soft admission control techniques to set V_{max} parameter of the contracts
- Expansion of the concept of contracting to point-to-anywhere contracts
- Designing and analyzing better edge strategies
- Exploring inter-domain pricing issues in diff-serv environment:
  - Exploring the concept of bandwidth intermediary to facilitate the mediation between customer and multiple providers by leveraging the DCC framework. The design of such agents for scalability and integration of policy and budget constraints is also an open topic.
  - Finding good pricing strategies for the provider in different market environments, e.g. monopoly, competitive

CHAPTER 5
ANALYSIS OF PRICING INTERVALS

5.1 Introduction

One proposed method for controlling congestion in wide area networks is to apply *congestion-sensitive pricing* [20, 51], which is a form of dynamic pricing. Many proposals have been made to implement dynamic pricing over wide area networks and the Internet [19, 33, 42, 50, 49, 79, 76, 68, 67, 62, 82]. Most of these schemes aimed to employ congestion pricing. The main idea of congestion-sensitive pricing is to update price of the network service dynamically over time such that it increases during congestion epochs and causes users to reduce their demand. So, implementation of congestion-sensitive pricing protocols (or any other dynamic pricing protocol) makes it necessary to change the price after some time interval, what we call *pricing interval*. Pricing time-scale has not been investigated significantly in the area other than [5]. In [5], authors propose a pricing architecture based on pricing time-scale issues. However, in this chapter, we look at a more specific problem, which is effect of pricing time-scale on congestion control by pricing.

Clark's Expected Capacity [19] scheme proposes long-term contracts as the pricing intervals. Kelly's packet marking scheme [42] proposes shadow prices to be fed back from network routers which has to happen over some time interval. MacKie-Mason and Varian's Smart Market scheme [50] proposes price updates at interior routers which cannot happen continuously and have to happen over some time interval. Wang and Schulzrinne's RNAP [79] framework proposes to update the price at each service level agreement which has to happen over some time interval. Hence, congestion-sensitive pricing can only be implemented by updating prices over some time interval, i.e. pricing interval.

It has been realized that there are numerous implementation problems for dynamic or congestion-sensitive pricing schemes, which can be traced into pricing intervals. We can list some of the important ones as follows:

- *Users do not like price fluctuations:* Currently, most ISPs employ flat-rate pricing which makes individual users happy. Naturally, most users do not want to have a network service with a price changing dynamically. In [25], Edell and Varaiya proved that there is a certain level of desire for quality-of-service. However, in [60] and [61], Odlyzko provides evidence that most users want simple pricing plans and they easily get irritated by complex pricing plans with frequent price changes. So, it is important that price updates should happen as less as possible. In other words, users like a service with *larger* pricing intervals.

- *Control of congestion degrades with larger pricing intervals:* Congestion level of the network changes dynamically over time. So, the more frequent the price is updated, the better the congestion control. From the provider's side, it is easier to achieve better congestion control with *smaller* pricing intervals.

- *Users want prior pricing:* It is also desired by the users that price of the service must be communicated to them before it is charged. This makes it necessary to inform the users of the network service before applying any price update. So, the provider has to handle the overhead of that price communication. The important thing is to keep this overhead as less as possible, which can be done with *larger* pricing intervals.

Hence, length of pricing intervals is a key issue for the implementation of congestion-sensitive and adaptive pricing protocols. In this chapter, we focus on modeling and analysis of pricing intervals to come up with a maximum value for it such that the level of congestion control remains in an acceptable range. Beyond this range, pricing could be used to regulate demand, but it becomes less useful as a tool for congestion management. The rest of the chapter is organized as follows: In Section 5.2, we first explore steady-state dynamics of congestion-sensitive pricing with a detailed look at the behavior of prices and congestion relative to each other. We then develop and discuss an approximate analytical model for the correlation of prices and congestion measures in Section 5.3. In Section 5.4, we validate the model by simulation experiments and present the results. Finally, in Section 5.5 we discuss the implications of the work and possible future work.

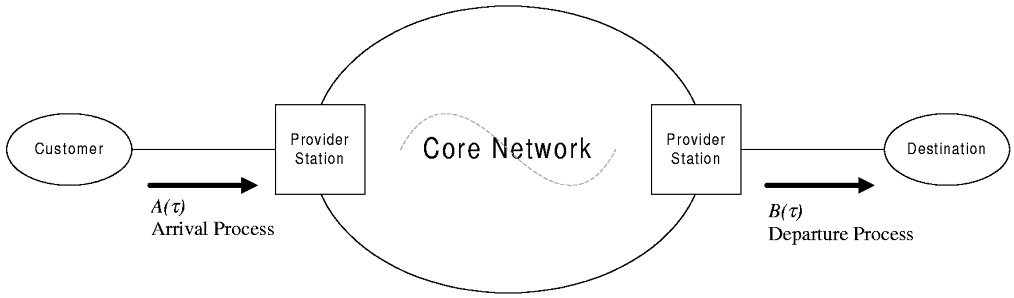


Figure 5.1: A sample customer-provider network.

5.2 Dynamics of Congestion-Sensitive Pricing

This section explains the behavior of congestion-sensitive prices and congestion measures relative to each other in a steady-state system. A sample scenario is described in Figure 5.1. The provider employs a pricing interval of T to implement congestion-sensitive pricing for its service. The customer uses that service to send traffic to the destination through the provider's network. The provider observes the congestion level, c, in the network core and adjusts its advertised price, p, according to it. Note that c and p are in fact functions of time (i.e. $c(t)$ and $p(t)$ where t is time), but we use c and p throughout the chapter for simplicity of notation. It is a realistic assumption to say that the provider can observe the network core over small time intervals, i.e. a few round-trip-times (RTTs). To understand effect of pricing interval to the dynamics of congestion-sensitive pricing, we look at the relationship between c and p over time.

Assuming that we have continuous knowledge of congestion level, c, we can represent the dynamics of congestion-sensitive pricing as in Figure 5.2. Figure 5.2 represents the relationship between c and p for two different pricing interval lengths, $T_1 > T_2$. For both lengths, the steady-state behavior of congestion-sensitive pricing is represented. The advertised price, p, varies around an optimum price, p^*.

When the provider sees that the congestion level has been decreasing, it decreases the advertised price such that the network resources are not under-utilized. Then the customer starts sending more traffic in response to the decrease in price,

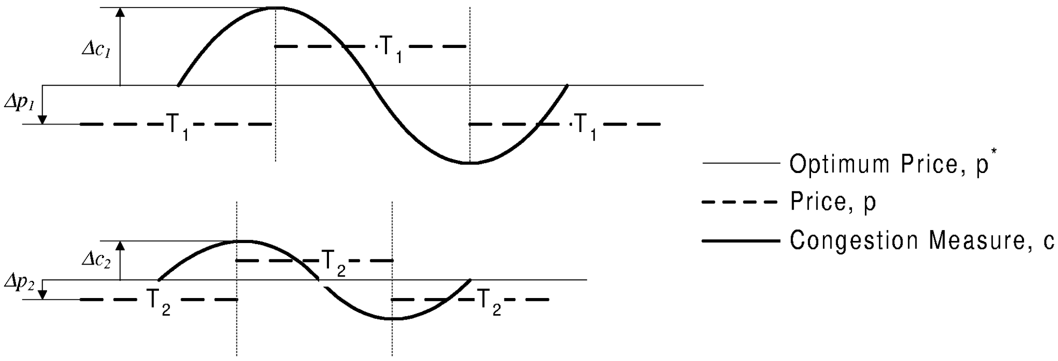


Figure 5.2: Congestion measure relative to congestion-sensitive prices in a steady-state network being priced.

and congestion level in the core starts increasing accordingly. The congestion level continues to increase until the price is increased by the provider at the beginning of the next pricing interval. When the provider increases price because of the increased congestion in the last pricing interval, the customer starts sending less traffic than before. Then congestion level starts decreasing. This behavior continues on in steady-state. This explains how congestion-sensitive prices can control the congestion in a network. The important difference is that with a larger pricing interval the congestion level oscillates larger as represented in Figure 5.2.

Another important characteristic of congestion-sensitive pricing is that the price must be oscillating around an optimum price, p^*, to guarantee both congestion control and high utilization of network resources. In other words, the average of advertised prices must be equal to the optimum price value. Notice that the customer will send less traffic which will under-utilize network resources when $p > p^*$, and the customer will send excessive traffic than the network can handle which will cause uncontrolled congestion when $p < p^*$. So the provider needs to satisfy the condition that the average of advertised prices equals to the optimum price.

The important issue to realize is that congestion control becomes better if the similarity between the advertised price and congestion level is higher. Because of the above explained implementation constraints, the advertised price cannot be updated continuously. This results in dissimilarity between the price and congestion level. Intuitively, if the correlation between the advertised prices and the congestion

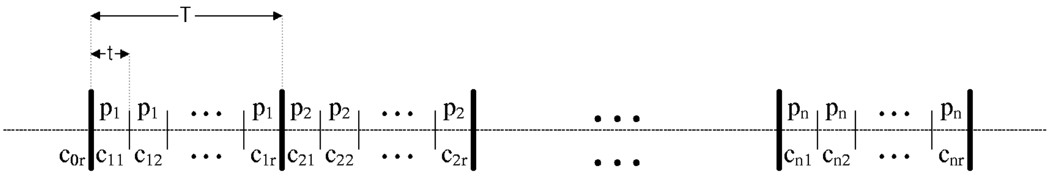


Figure 5.3: Prices and congestion measures for subsequent observation intervals.

measures is higher, fidelity of control over congestion becomes higher. Again by intuition, the correlation becomes smaller if the pricing interval is larger.

Another important issue is the *price oscillation* caused by the discontinuous price updates. As the pricing intervals get larger, the oscillation in price also gets larger. This in effect leads to oscillation in user demand (i.e. traffic) correspondingly. So, larger oscillations in price are expected to cause larger oscillation and *higher variance* in incoming traffic. Then, more oscillated traffic causes more oscillated congestion level. This behavior is represented in Figure 5.2 with the case that $\triangle c_1 > \triangle c_2$ and $\triangle p_1 > \triangle p_2$.

In the next section, we will develop an approximate model of correlation between the advertised prices and congestion measures analytically and find the largest value for the pricing interval such that the system functions in a desired range of service.

5.3 Analytical Model for Correlation of Prices and Congestion Measures

5.3.1 Assumptions and Model Development

Assume the length of pricing interval stays fixed at T over n intervals. Also assume the provider can observe the congestion level at a smaller time scale with fixed observation intervals, t. Assume that $T = rt$ holds, where r is the number of observations the provider makes in a single pricing interval. Assume that the queue backlog in the network core is an exact measure of congestion. [47]

We assume that the customer has a fixed budget for network service and he/she sends traffic according to a counting process, which is a continuous time stationary stochastic process $A(\tau)$, $\tau \geq 0$ with first and second moments of λ_1 and λ_2 respectively. In reality, λ_1 is not fixed, because the customer responds to price changes by changing its λ_1. However, since we assume steady-state and fixed budget for the customer, it is reasonable to say that the customer will send at a constant rate over a large number of pricing intervals. Let m_{ij} be the number of packet arrivals from the customer during the j th observation interval of ith pricing interval, where $i = 1..n$ and $j = 1..r$. So the total number of packet arrivals during the ith pricing interval is

$$m_i = \sum_{s=1}^{r} m_{is} \tag{5.1}$$

Also assume that the packets leave after the network service according to a counting process, which is a continuous time stationary stochastic process $B(\tau)$, $\tau \geq 0$ with first and second moments of μ_1 and μ_2 respectively. Let k_{ij} be the number of packet departures during the jth observation interval of ith pricing interval, where $i = 1..n$ and $j = 1..r$. So the total number of packet departures during the ith pricing interval is

$$k_i = \sum_{s=1}^{r} k_{is} \tag{5.2}$$

Assuming that no drop happens in the network core, the first moments of the two processes are equal in steady-state, i.e. $\lambda_1 = \mu_1$, but the second moments are not.

As represented in Figure 5.3, let p_i be the advertised price and c_{ij} is the congestion measure (queue backlog) at the end of the jth observation in the ith pricing interval. In our model we need a generic way of representing the relationship between prices and congestion. We assumed that the congestion-sensitive pricing algorithm calculates the price for the ith pricing interval according to the following formula[7]

$$p_i = a(t, r)\ c_{(i-1)r} \tag{5.3}$$

where $a(t, r)$, *pricing factor*, is a function of pricing interval and observation interval defined by the congestion pricing algorithm. We assume that $a(t, r)$ is only effected

[7]Note that this is a simplifying formula for tractability, and cannot capture all aspects of congestion pricing.

by the interval lengths, not by the congestion measures. Notice that this assumption does not rule out the effect of congestion measures on price, but it splits the effect of congestion measures and interval lengths to price. We will use a instead of $a(t,r)$ for notation simplicity.

Within this context, the following equations hold:

$$c_{ij} = c_{0r} + \sum_{u=1}^{i-1}(m_u - k_u) + \sum_{s=1}^{j}(m_{is} - k_{is}) \tag{5.4}$$

$$c_{ir} = c_{0r} + \sum_{j=1}^{i}(m_j - k_j) \tag{5.5}$$

where $i \geq 1$. Reasoning behind Equations 5.4 and 5.5 is that the queue backlog (which is the congestion measure) at the end of an interval is equal to the number of packet arrivals minus the number of packet departures during that interval.

Let the average price be $\overline{p}$ and the average queue backlog be $\overline{c}$. By assuming that the system is in steady-state we can conclude that the following equation is satisfied

$$\overline{p} = a\overline{c} \tag{5.6}$$

Since the system is assumed to be in steady-state, we can assume the initial (right before the first pricing interval) congestion measure equals to the average queue backlog, i.e.

$$c_{0r} = \overline{c} \tag{5.7}$$

We want to approximate the model of correlation between p and c according to the above assumptions. We can write the formula for correlation between p and c over n pricing intervals as

$$Corr_n = \frac{E_n[(c-\overline{c})(p-\overline{p})|m,k]}{E_n[(c-\overline{c})^2|m,k]E_n[(p-\overline{p})^2|m,k]} \tag{5.8}$$

assuming that total of m packet arrivals and k packet departures happen during the n rounds.

We can calculate the numerator term in Equation 5.8 as follows:

$$E_n[(c-\overline{c})(p-\overline{p})|m,k] = \frac{1}{rn}\sum_{i=1}^{n}\sum_{j=1}^{r}(p_i-\overline{p})(c_{ij}-\overline{c}) \tag{5.9}$$

By applying Equations 5.3, 5.6 and 5.7 into Equation 5.9 we can get

$$E_n[(c-\overline{c})(p-\overline{p})|m,k] = \frac{1}{rn}\sum_{i=1}^{n}\sum_{j=1}^{r}(ac_{(i-1)r}-ac_{0r})(c_{ij}-c_{0r}) \tag{5.10}$$

Then by applying Equations 5.4 and 5.5 into Equation 5.10, we get the following

$$\begin{aligned}&E_n[(c-\overline{c})(p-\overline{p})|m,k] = \\ &\quad\frac{a}{rn}\sum_{i=1}^{n}\sum_{j=1}^{r}\left(c_{0r}+\sum_{\theta=1}^{i-1}(m_\theta-k_\theta)-c_{0r}\right)\left(\sum_{u=1}^{i-1}(m_u-k_u)+\sum_{s=1}^{j}(m_{is}-k_{is})\right)\end{aligned} \tag{5.11}$$

After going through the derivation, we can put Equation 5.11 into the following form

$$E_n[(c-\overline{c})(p-\overline{p})|m,k] = \frac{a}{rn}\sum_{i=1}^{n}\sum_{j=1}^{r}\left(H_1+\sum_{\theta=1}^{i-1}(m_\theta-k_\theta)\sum_{s=1}^{j}(m_{is}-k_{is})\right) \tag{5.12}$$

where $H_1 = \sum_u(m_u-k_u)^2+\sum_u\sum_{v\neq u}2(m_u-k_u)(m_v-k_v)$, $u = 1..i-1$ and $v = 1..i-1$.

We can calculate the variance of congestion measures similarly as follows:

$$E_n[(c-\overline{c})^2|m,k] = \frac{1}{rn}\sum_{i=1}^{n}\sum_{j=1}^{r}(c_{ij}-\overline{c})^2 \tag{5.13}$$

By applying Equations 5.4 and 5.7 into Equation 5.13 we can get

$$E_n[(c-\overline{c})^2|m,k] = \frac{1}{rn}\sum_{i=1}^{n}\sum_{j=1}^{r}\left(\sum_{u=1}^{i-1}(m_u-k_u)+\sum_{s=1}^{j}(m_{is}-k_{is})\right)^2 \tag{5.14}$$

After going through the derivation, we can put Equation 5.14 into the following form

$$E_n[(c-\overline{c})^2|m,k] = \frac{1}{rn}\sum_{i=1}^{n}\sum_{j=1}^{r}\left(H_1+H_2+2\sum_{u=1}^{i-1}(m_u-k_u)\sum_{s=1}^{j}(m_{is}-k_{is})\right) \tag{5.15}$$

where $H_2 = \sum_s (m_{is} - k_{is})^2 + \sum_s \sum_{z \neq s} 2(m_{is} - k_{is})(m_{iz} - k_{iz})$, $s = 1..j$, $z = 1..j$.

We finally can calculate the variance of price as follows:

$$E_n[(p - \overline{p})^2 | m, k] = \frac{1}{rn} \sum_{i=1}^{n} \sum_{j=1}^{r} (p_i - \overline{p})^2 \tag{5.16}$$

By using Equations 5.3, 5.5 and 5.6 into Equation 5.16 we can get the following

$$E_n[(p - \overline{p})^2 | m, k] = \frac{a^2}{n} \sum_{i=2}^{n} \left(\sum_{j=1}^{i-1} (m_j - k_j) \right)^2 \tag{5.17}$$

Similarly after going through derivation, we can put Equation 5.17 into the following form

$$E_n[(p - \overline{p})^2 | m, k] = \frac{a^2}{n} \sum_{i=2}^{n} H_1 \tag{5.18}$$

Now we can relax the condition on m and k by summing out conditional probabilities on Equations 5.12, 5.15, and 5.18. Specifically, we need to apply the operation

$$E_n[x] = \sum_{m_{ij}=0}^{\infty} \sum_{k_{ij}=0}^{\infty} E_n[x | m, k] P_{m_{ij};k_{ij}} \tag{5.19}$$

for all $i = 1..n$ and $j = 1..r$, where $P_{m_{ij};k_{ij}}$ is $P\{A(t) = m_{ij}; B(t) = k_{ij}\}$. This operation is non-trivial because of the dependency between the processes $A(\tau)$ and $B(\tau)$, and it is not possible to reach a closed-form solution without simplifying assumptions. After this point, we develop two *approximate* models by making simplifying assumptions.

5.3.1.1 Model-I

Although the arrival and departure processes are correlated, there might also be cases where the correlation is negligible. For example, if the distance between arrival and departure points is more, then the lag between the arrival and departure processes also becomes more which lowers the correlation between them. So, for simplicity, we assume *independence* between the arrival and departure processes and derive an *approximate* model. The independence assumption makes it very easy to relax the condition on m and k, since the joint probability of having $A(t) = m_{ij}$ and

$B(t) = k_{ij}$ becomes product of probability of the two events. After the relaxation, we then substitute $\mu_1 = \lambda_1$ because of the steady-state condition, and get the followings:

$$E_n[(c-\overline{c})(p-\overline{p})] = \frac{atr}{2}(n-1)(\lambda_2+\mu_2-2tr\lambda_1^2) \tag{5.20}$$

$$E_n[(c-\overline{c})^2] = \frac{t}{2}(\lambda_2+\mu_2)(rn+1) - t^2\lambda_1^2(1+r-r^2+r^2n) \tag{5.21}$$

$$E_n[(p-\overline{p})^2] = \frac{a^2tr}{2}(n-1)(\lambda_2+\mu_2-2tr\lambda_1^2) \tag{5.22}$$

Let σ_A^2 be the variance of the arrival process and σ_B^2 be the variance of the departure process. By substituting Equations 5.20, 5.22, and 5.21 into 5.8 we get the correlation model for the first n rounds as follows:

$$Corr_n = \frac{1}{at(\frac{\sigma_A^2+\sigma_B^2}{2}+\lambda_1^2)(rn+1) - a(t\lambda_1)^2(1+r-r^2+r^2n)} \tag{5.23}$$

5.3.1.2 Model-II

To make a more realistic model, we try to develop a model where the arrival and departure processes are not considered independent. We consider the system as an $M/M/1$ queueing system with a service rate of μ. Notice that μ is different from the parameters μ_1 and μ_2 which are first and second moments of $B(\tau)$. We now try to derive the joint probability as follows:

$$P_{m_{ij};k_{ij}} = P_{m_{ij}} * P_{k_{ij}|m_{ij}} \tag{5.24}$$

where $P_{m_{ij}} = P\{A(t) = m_{ij}\}$ and $P_{k_{ij}|m_{ij}} = P\{B(t) = k_{ij}|A(t) = m_{ij}\}$. Notice that $P_{m_{ij}}$ is probability of having m_{ij} events for the Poisson distribution with mean $\lambda_1 t$. However, it is not that easy to calculate $P_{k_{ij}|m_{ij}}$, since probability of having k_{ij} departures depends not only on the number of arrivals m_{ij} but also the number already available in the system which is $c_{i(j-1)}$. Let N be the random variable that represents the number available in the system, then we can rewrite $P_{k_{ij}|m_{ij}}$ as follows:

$$P_{k_{ij}|m_{ij}} = \sum_{c_{i(j-1)}=k_{ij}-mij}^{\infty} P_{k_{ij}|m_{ij};c_{i(j-1)}} * P_{c_{i(j-1)}} \tag{5.25}$$

where $P_{c_{i(j-1)}} = P\{N = c_{i(j-1)}\}$. Observe that the minimum value of $c_{i(j-1)}$ can be $k_{ij} - mij$, because the condition $k_{ij} \leq m_{ij} + c_{i(j-1)}$ must be satisfied for all time intervals. In Equation 5.25, $P_{c_{i(j-1)}}$ is known for a steady-state $M/M/1$ system. Let $\rho = \lambda_1/\mu$, then $P_{c_{i(j-1)}} = (1-\rho)\rho^{c_{i(j-1)}}$. [43] However, calculation of $P_{k_{ij}|m_{ij};c_{i(j-1)}}$ is not simple, because the m_{ij} arrivals may arrive such that there is none waiting for the service. Fortunately, this is a very rare case for a loaded system. So, we can formulate $P_{k_{ij}|m_{ij};c_{i(j-1)}}$ for the usual case as if all the m_{ij} arrivals happened at the beginning of the interval t. Within this context, we now derive $P_{k_{ij}|m_{ij};c_{i(j-1)}}$.

Let $E(\mu)$ be an Exponential random variable with mean $1/\mu$, and $E_r(k,\mu)$ be an Erlangian random variable with mean k/μ. Then, we can formulate the probability of having $k > 0$ departures in time t as follows:

$$P_{k>0 \ in \ t} = \int_0^t P\{E_r(k,\mu) < x\} \left[1 - P\{E(\mu) < t - x\}\right] dx \tag{5.26}$$

Now, we can formulate the CDF of $P_{k_{ij}|m_{ij};c_{i(j-1)}}$ as follows:

$$P\{B(t) \leq k_{ij}|m_{ij}; c_{i(j-1)}\} = P_{0 \ in \ t} + \sum_{k=1}^{k_{ij}} P_{k>0 \ in \ t} \tag{5.27}$$

Notice that $P_{0 \ in \ t} = 1 - P[E(\mu) < t]$. We used Maple to derive the CDF formula in (5.27), and got the following result:

$$P\{B(t) \leq k_{ij}|m_{ij}; c_{i(j-1)}\} = e^{-\mu t} + \frac{1}{\mu}\left(k_{ij} - e^{-\mu t}\sum_{i=1}^{k_{ij}}\sum_{j=0}^{i}\frac{(\mu t)^j}{j!}\right) \tag{5.28}$$

By using the CDF formula in Equation 5.28 in Maple, we then find pmf as:

$$\begin{aligned} P_{k_{ij}|m_{ij};c_{i(j-1)}} &= P\{B(t) \leq k_{ij}|m_{ij}; c_{i(j-1)}\} - P\{B(t) \leq k_{ij} - 1|m_{ij}; c_{i(j-1)}\} \\ &= \frac{1}{\mu}\left(1 - e^{-\mu t}\sum_{i=0}^{k_{ij}}\frac{(\mu t)^i}{i!}\right) \end{aligned} \tag{5.29}$$

Afterwards, we apply the operation in Equation 5.25, i.e.:

$$P_{k_{ij}|m_{ij}} = \sum_{c_{i(j-1)}=k_{ij}-m_{ij}}^{\infty} \frac{1}{\mu}\left(1 - e^{-\mu t}\sum_{i=0}^{k_{ij}}\frac{(\mu t)^i}{i!}\right) * \left(1-\frac{\lambda_1}{\mu}\right)\left(\frac{\lambda_1}{\mu}\right)^{c_{ij}} \tag{5.30}$$

Again by using Maple, we finally derive $P_{k_{ij}|m_{ij}}$ as:

$$P_{k_{ij}|m_{ij}} = \frac{1}{\mu}\left(\frac{\lambda_1}{\mu}\right)^{(k_{ij}-m_{ij})}\left[1 - e^{-\mu t}\sum_{i=0}^{k_{ij}}\frac{(\mu t)^i}{i!}\right] \tag{5.31}$$

Even though we have found a nice solution to $P_{k_{ij}|m_{ij}}$ in Equation 5.31, it does not allow us to get a closed-form model for the correlation after the relaxation operation in Equation 5.19. In order to get a closed-form correlation model, we approximated the term with summation in Equation 5.31. Notice that the term with summation is equivalent to ratio of two Gamma [15] functions, i.e.:

$$e^{-\mu t}\sum_{i=0}^{k_{ij}}\frac{(\mu t)^i}{i!} = \frac{\Gamma(k_{ij}+1, \mu t)}{\Gamma(k_{ij}+1)}$$

In Appendix B, we approximated the ratio $\Gamma(x,y)/\Gamma(x)$ and used that method to approximate the term with summation in (5.31). After the approximation, we did get a closed-form correlation model. But, it is not possible to provide it in hardcopy format[8] because it is a very large expression. However, we will provide numerical results of the model later in Section 5.4.

5.3.2 Model Discussion

Since Model-II is a very large expression, we only discuss Model-I. Assuming that the other factors stay fixed, the correlation model in Equation 5.23 implies three important results:

1. *The correlation degrades at most inversely proportional to an increase in pricing intervals (T):* For the smallest n value (i.e. 1), denominator of Equation 5.23 will have $r+1$ as a factor which implies linear decrease in the correlation

[8] It is available upon request.

value while the pricing interval, $T = rt$, increases linearly. Notice that its effect will be less when n is larger.

2. *Increase in traffic variances (σ_A^2 and σ_B^2) degrades the correlation:* From Equation 5.23, we can observe that the correlation decreases when the variance of the incoming or outgoing traffic increases.

3. *Increase in traffic mean (λ_1) degrades the correlation:* Again from Equation 5.23, we can see that the correlation decreases while the mean of the incoming traffic increases.

These above results imply that lower pricing intervals must be employed when variance and/or mean of the traffic starts increasing. We validate these three results in Section 5.4 by experiments. Note that the model reveals non-intuitive effect of traffic mean on the correlation. Also, observe that the model incorporates not only the effect of pricing intervals on the correlation, but also the effects of statistical parameters (e.g. traffic mean and variance).

As previously mentioned, the correlation between prices and congestion measures is a representation of the achieved control over congestion. Congestion-sensitive pricing protocols can use such a model to maintain the control at a predefined level by solving the inequality $Corr_n \geq Corr_{min}$ for r, which defines the length of the pricing interval. If feedback from the other end (i.e. egress node in DiffServ [27] terminology) is provided, then such a model can be implemented in real-time. σ_B^2 can be calculated by using the feedbacks from the other end, and σ_A^2 and λ_1 can be calculated by observing the incoming traffic.

5.4 Experimental Results and Model Validation

5.4.1 Experimental Configuration

We use Dynamic Capacity Contracting (DCC) [72] as the congestion pricing protocol in our simulations. The short-term contracts corresponds to the pricing intervals in our modeling.

Figure 5.4 represents the topology of network in our experiments. There are 5 customers trying to send traffic to the same destination over the same bottleneck

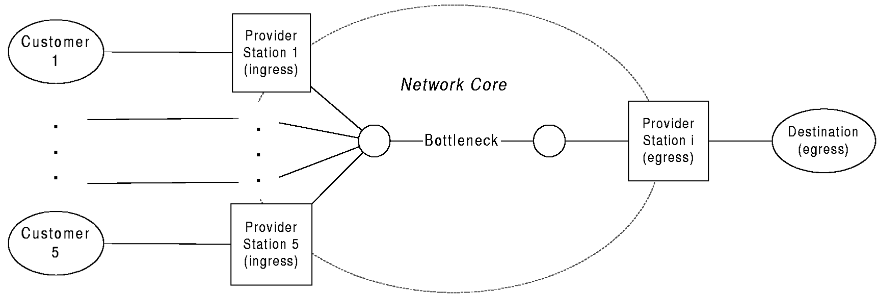


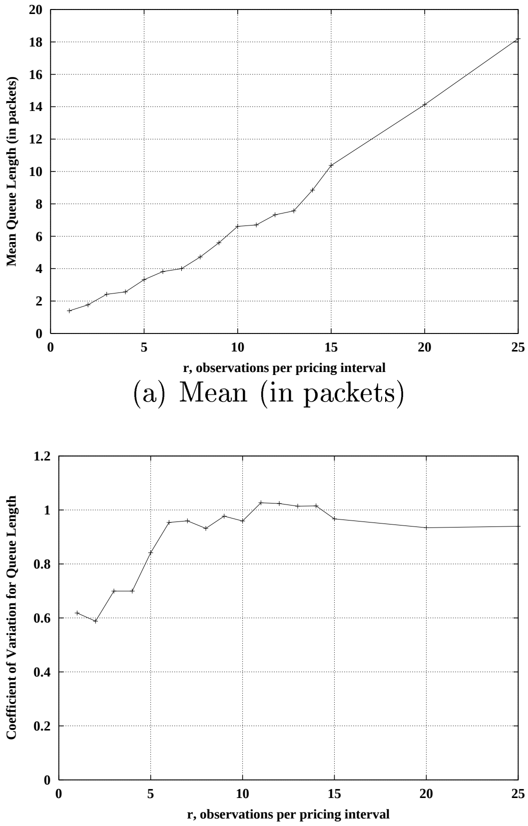
Figure 5.4: Topology of the experimental network.

with a capacity of 1Mbps. Customers have equal budgets and their total budget is 150 units. We observe the bottleneck queue length and use it as congestion measure. The observation interval is fixed at $t = 80ms$ and RTT for a customer is $20ms$. We increase the pricing interval by incrementing the number of observations (i.e. r) per contract. We run several simulations and calculate correlation between the advertised prices and the observed bottleneck queue lengths during the simulations.

Customers send their traffic with a fixed variance but changing mean according to the advertised prices for the contracts. We assume that the customers have fixed budgets per contract with additional leftover from the previous contract. The customers adjust their sending rate according to the ratio B/p where B is the customer's budget and p is the advertised price for the contract. So, customers increase or decrease their sending rate right before the contract starts accordingly. Notice that since the customers' budget is fixed, the sending rate of the customers is actually fixed on long run, which fits to the fixed average incoming traffic rate (λ_1) assumption in the model.

Customers send their traffic with mean changing according to the advertised prices for the contracts. We assume that the customers have fixed budgets per contract with additional leftover from the previous contract. The customers adjust their sending rate according to the ratio B/p where B is the customer's budget and p is the advertised price for the contract [9]. Notice that since the customers' budget is fixed, the *average* sending rate of the customers is actually *fixed on long run*,

[9]Note that $x = B/p$ maximizes surplus for a customer with utility $u(x) = B\ log(x)$.



(a) Mean (in packets)

(b) Variance

(c) Coefficient of variation, $\frac{\sigma}{\mu}$

Figure 5.5: Statistics of bottleneck queue length.

which fits to the fixed average incoming traffic rate assumption in the model.

5.4.2 Results

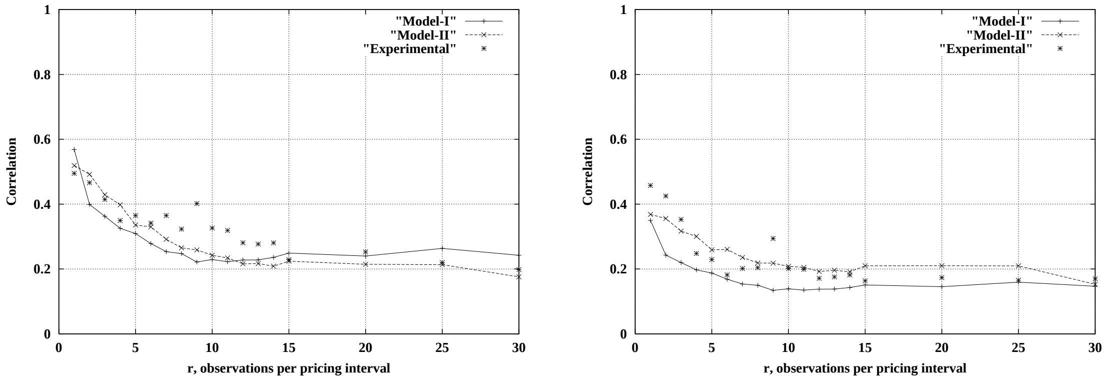
In this section, we present several simulation results for validation of the model and the three results it implies.

Figures 5.5-a and 5.5-b show mean and variance of the bottleneck queue length respectively. We observe steady increase in mean and variance of bottleneck queue as the pricing interval increases. Furthermore, Figure 5.5-c shows the change in the coefficient of variation for the bottleneck queue length as the pricing interval increases. Note that an increase in the coefficient of variation means a decrease in the level of control. We observe that coefficient of variation increases as the pricing interval increases until $10r$, and stays fixed there after. This is because the congestion pricing protocol looses control over congestion after a certain length of pricing interval, which is $10r$ in this particular experiment. These results in Figures 5.5-a to 5.5-c validate our claim about the degradation of control when pricing interval increases. Furthermore, they also show that dynamic pricing does not help congestion control when the pricing interval is longer than a certain length.

To validate the model, we present the fit between our correlation models and experimental results obtained from simulations. Figures 5.6-a and 5.6-b represent the correlations obtained by inserting appropriate parameter values to the model and corresponding experimental correlations, respectively for $n = 15$ and $n = 25$. We observe that Model-II fits better than Model-I, since Model-II considers the dependency between arrival and departure processes. Notice that the model is dependent on the experimental results because of the parameters for incoming and outgoing traffic variances (i.e. σ_A^2 and σ_B^2), pricing factor (i.e. a), and mean of the incoming traffic (i.e. λ_1). We first calculate the parameters σ_A^2, σ_B^2, a (ratio of average price by average bottleneck queue length) and λ_1 from the experimental results, and then use them in the model.

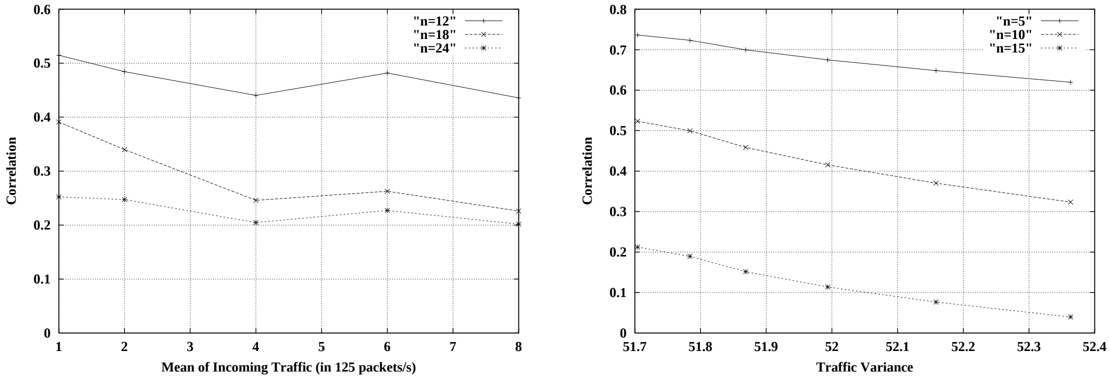
We now validate the three results implied in Section 5.3.2. Figures 5.6-a and 5.6-b show that the correlation decreases slower than $1/r$ when r increases linearly. This validates the first result. Figure 5.7-b represents the effect of change in the variance of incoming and outgoing traffic (i.e. σ_A^2 and σ_B^2) on the correlation. The horizontal axis shows the increase in variances of both the incoming and outgoing traffic. The results in Figure 5.7-b obviously show that an increase in traffic variances causes decrease in the correlation. This validates the second result. Finally for validation of the third result, Figure 5.7-a represents the effect of change in the mean of the incoming traffic (i.e. λ_1) on the correlation. We can see that increase in λ_1 causes decrease in the correlation. Another important realization is that the correlation is more sensitive to variance changes than mean changes as it can be seen by comparing Figures 5.7-a and 5.7-b.

Before concluding this section, we would like to stress on the relationship between the correlation and the level of congestion control. As we previously stated, Figures 5.6-a and 5.6-b show the effect of increasing pricing intervals on the correla-



(a) For 15 pricing intervals (b) For 25 pricing intervals

Figure 5.6: Fitting analytical model to experimental results.



(a) Effect of traffic mean (b) Effect of traffic variance

Figure 5.7: Effect of traffic patterns to the correlation (for $T = 800ms$ and $r = 10$).

tion for different values of n. We can see that the correlation value stays almost fixed after the pricing interval reaches to $10r$. Also, Figure 5.5-c shows the coefficient of variation for the bottleneck queue length. Remember that coefficient of variation for the queue length represents the level of congestion control being achieved. We observe in Figure 5.5-c that it reaches to its maximum value (approximately 1) when the pricing interval reaches to $10r$, which is the same point where the correlation starts staying fixed in Figures 5.6-a and 5.6-b. So, by comparing Figure 5.5-c with Figures 5.6-a and 5.6-b, we can observe that the correlation decreases when the level of congestion control decreases, and also it stays fixed when the level of congestion control stays fixed. This shows that the correlation can be used as a metric to represent the level of congestion control.

5.5 Summary

We investigated steady-state dynamics of congestion-sensitive pricing in a customer-provider network. With the idea that correlation between prices and congestion measures is a measurement for level of congestion control, we modeled the correlation. We found that the correlation decreases at most inversely proportional to an increase in pricing interval. We also found that the correlation is inversely effected by the mean and variance of the incoming traffic. This implies that congestion-sensitive pricing schemes need to employ very small pricing intervals to maintain high level of congestion control for current Internet traffic with high variance [21].

From the model and also from the simulation experiments we observed that the correlation between prices and congestion measures drops to very small values when pricing interval reaches to 40 RTTs even for a low variance incoming traffic. Currently, we usually have very small RTTs (measured by milliseconds) in the Internet. This shows that pricing intervals should be 2-3 seconds for most cases in the Internet, which is not possible to deploy over low speed modems. This result itself means that deployment of congestion-sensitive pricing over the Internet is highly challenging. As the link speeds are getting higher and RTTs are getting smaller, it becomes harder to deploy congestion-sensitive prices.

The results obviously show that there will be need for intermediate middle-ware components (i.e. intermediaries) between individual users and ISPs, when ISPs deploy congestion-sensitive pricing for their service. These middle-ware components will be expected to lower price fluctuations such that price changes will be possible to implement over low speed modems. This scenario suggests that congestion-sensitive prices can be implemented among ISPs to control congestion, but there has to be middle-ware components which can handle the transition of the congestion-sensitive prices to the individual customers in a smooth way. Alternatively, instead of using congestion-sensitive pricing directly for the purpose of congestion control, it can be used to improve fairness of an underlying congestion control mechanism. This way it will be possible to control congestion at small time-scale, while maintaining human involvement to pricing at large time-scale. We believe that the second approach is more realistic way of implementing congestion-sensitive pricing over the Internet. The analysis of pricing intervals that we made in this chapter lead us to investigate the second approach, which will be presented later in Chapter 7.

Another key implementation problem for congestion pricing is that current Internet access is point-to-anywhere. It is not possible to obtain information about the exit points of the traffic. However, it is not possible to determine congestion information and prices without coordinating entry and exit points of the traffic. So, this particular aspect implies that it is highly challenging to implement congestion pricing at individual user to ISP level. But, if an ISP has enough control over the entry and exit points, then it is possible. Alternatively, if ISPs of the current Internet collaborate on providing information about the entry and exit points to each other, then again it will be possible.

Future work should include complex modeling of the dynamics of congestion-sensitive pricing by relaxing some of the assumptions. For example, a model without fixed arrival rate assumption would represent the behavior of the system more appropriately. Also, better budget models are needed in the model.

Another important issue to explore is how much congestion control can be achieved with exactly what level of correlation between prices and congestion measures. In this particular modeling work we assumed that the correlation value is a direct representation of the level of congestion control that was achieved. Although we supported this idea by providing the match between the correlation and the coefficient of variation in Section 5.4.2, this issue needs more investigation.

CHAPTER 6
DISTRIBUTED-DCC:
Pricing for Congestion Control (PFCC)

6.1 Introduction

In chapter 4, we presented a framework, *dynamic capacity contracting (DCC)*, for congestion-sensitive pricing in a single diff-serv domain. This version of DCC assumed the provider stations (that are placed at the edge routers) to be advertising *the same price* for the contracts, which is not possible to implement over a wide area network. This is simply because the price value cannot be communicated to all stations at the beginning of each contract. In this chapter, we relax this assumption by letting the stations to calculate prices locally and be able to advertise different prices than the other stations. We call this version of DCC as *Distributed-DCC*.

In Section 1.3, we described two pricing architectures in terms of managing congestion control through pricing: *pricing for congestion control (PFCC), pricing over congestion control (POCC)*. In PFCC, pricing framework employs an underlying edge-to-edge *congestion detection* mechanism. However, in POCC, it uses an underlying edge-to-edge *congestion control* mechanism. Note that, in the case of POCC, system performance is dependent not only on the pricing framework, but also on properties of the underlying congestion control mechanism. In this chapter, we present the Distributed-DCC framework based on the PFCC architecture, and in the next chapter we will investigate issues regarding Distributed-DCC framework based on the POCC architecture.

Two main purposes of the Distributed-DCC framework are to provide, within implementation constrains, fairer *bandwidth sharing* and better *congestion control* by controlling user demand. In this chapter, we mainly focus on fairness issues related to bandwidth sharing and show that Distributed-DCC can achieve max-min fairness and proportional fairness under proper conditions. In the next chapter, we will show that POCC architecture performs better than PFCC in terms of congestion control.

The chapter is organized as follows: In the next section, we first describe the overall properties of Distributed-DCC framework. We look at several issues (such as stability, scalability, fairness, capacity estimation) regarding Distributed-DCC. In Section 6.3, we re-describe the EEP pricing scheme within the context of Distributed-DCC. In Section 6.5, we evaluate performance of Distributed-DCC by simulating EEP pricing scheme. We finally summarize the chapter in Section 6.6.

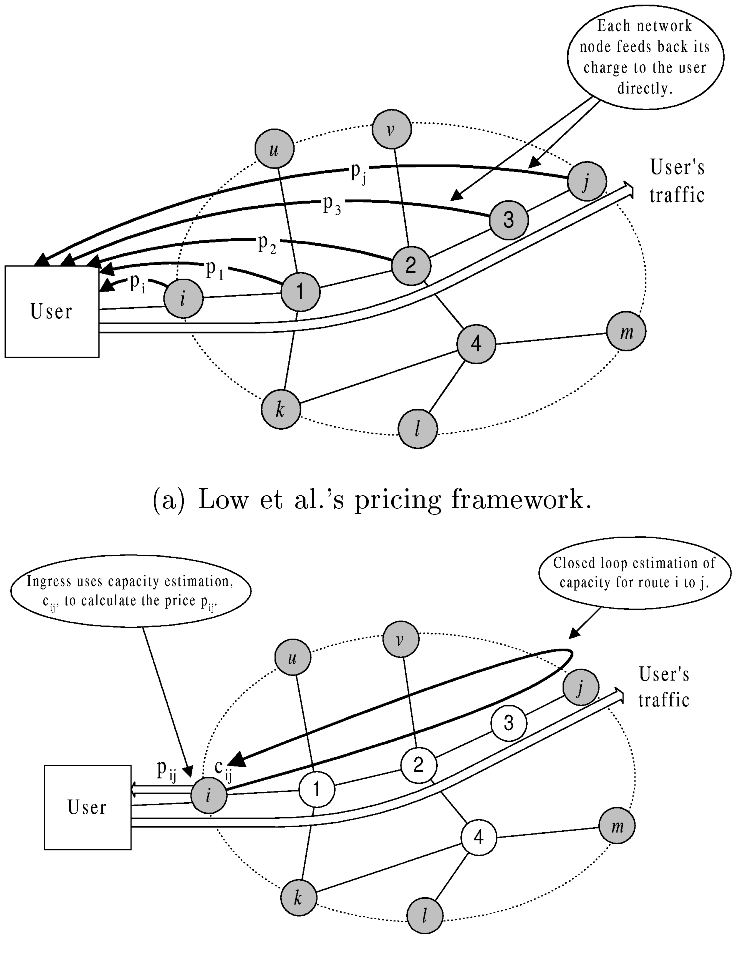
6.2 Distributed-DCC

Distributed-DCC is specifically designed for diff-serv architecture, because the edge routers can perform complex operations which is essential to several requirements for implementation of congestion pricing. Each edge router is treated as a station of the provider. Each station advertises locally computed prices with information received from other stations. The main framework basically describes how to preserve coordination among the stations such that stability and fairness of the overall network is preserved. Essence of Distributed-DCC is two-fold:

- Since upgrade to all routers is not possible to implement, pricing should happen on an *edge-to-edge* basis which only requires upgrades to edge routers.
- Provider should employ *short-term* contracts in order to have ability to change prices frequently enough such that congestion-pricing can be enabled.

As a fundamental difference between Distributed-DCC and the well-known dynamic pricing proposals (e.g. Kelly et al.'s proposal[42], Low et al.'s proposal [47]) in the area lies in the manner of price calculation.

In Distributed-DCC, the prices are calculated on an edge-to-edge basis, while traditionally it has been proposed that prices are calculated at each local link and fed back to users. To make it more concrete, Figures 6.1-a and 6.1-b show the case of Distributed-DCC and the case of Low et al.'s framework. Gray nodes are the ones that participates in price calculation for a user. In Distributed-DCC, basically, the links on a flow's route are abstracted out by edge-to-edge capacity estimation (which is supposed to be congestion-based) and the ingress node communicates with the corresponding egress node to observe congestion on the route of user's traffic. Then, the ingress node uses the estimated capacity and the observed congestion



(a) Low et al.'s pricing framework.

(b) Distributed-DCC framework.

Figure 6.1: Comparison of Distributed-DCC with Low et al.'s pricing framework in terms of price calculation.

information in order to calculate price. However, in Low et al.'s framework, each link calculates its own price and sends it to the user, and the user pays the aggregate price. So, Distributed-DCC is better in terms of implementation requirements, while Low et al.'s framework is better in terms of optimality. Distributed-DCC trades off some optimality in order to enable implementation of dynamic pricing. Amount of lost optimality depends on the closed-loop edge-to-edge capacity estimation.

Distributed-DCC framework has three major components as shown in Figure 6.2: *Logical Pricing Server (LPS), Ingress Stations*, and *Egress Stations*. Solid lined arrows in the figure represent control information being transmitted among

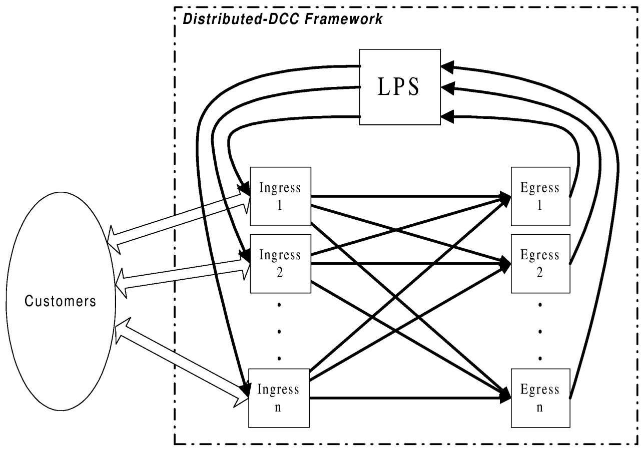


Figure 6.2: Components of Distributed-DCC framework: Solid lined arrows represent flow of control information necessary for price calculation. In PFCC architecture, communication with LPS must be at very short time-scales (i.e. each short-term contract). However, in POCC, LPS is accessed at longer time-scales (i.e. parameter remapping instants).

the components. Basically, Ingress stations negotiate with customers, observe customer's traffic, and make estimations about customer's demand. Ingress stations inform corresponding Egress stations about the observations and estimations about each edge-to-edge flow.

Egress stations detect congestion by monitoring edge-to-edge traffic flows. Based on congestion detections, Egress stations estimate available capacity for each edge-to-edge flow, and inform LPS about these estimations.

LPS receives capacity estimations from Egress stations, and allocates the available network capacity to edge-to-edge flows according to different criteria (such as fairness, price optimality).

Below, we describe functions and sub-components of these three components in detail. To ease understanding of the framework, we show important parameters, their symbols and their descriptions in Table 6.1. Also, we provide pseudo-code for major components of Distributed-DCC in Appendix F.

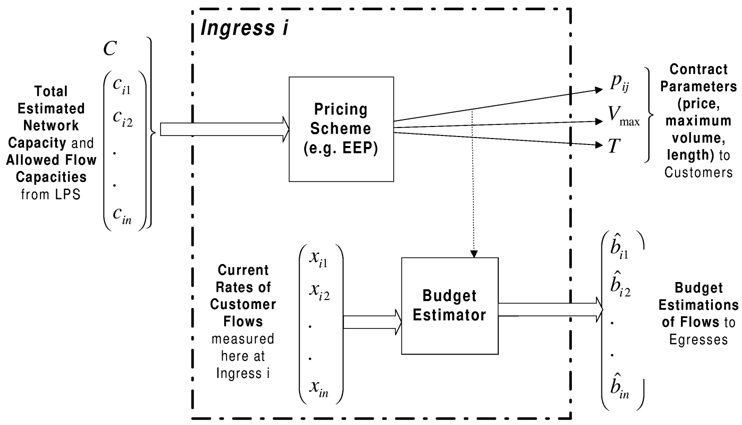


Figure 6.3: Major functions of Ingress i.

6.2.1 Ingress Station i

Figure 6.3 illustrates sub-components of Ingress station i in the framework. Ingress i includes two sub-components: *Pricing Scheme* and *Budget Estimator*.

Ingress station i keeps a "current" price vector p_i, where p_{ij} is the price for the flow from ingress i to egress j. So, the traffic using flow i to j is charged the price p_{ij}. Pricing Scheme is the sub-component that calculates price p_{ij} for each edge-to-edge flow starting at Ingress i. It uses allowed flow capacities c_{ij} and other local information (such as $\hat{b}_{ij}$), in order to calculate price p_{ij}. The station, then, uses p_{ij} in negotiations with customers. We will describe a pricing scheme Edge-to-Edge Pricing (EEP) later in Section 6.3. However, it is possible to implement several other pricing schemes by using the information available at Ingress i. Other than EEP, we implemented another pricing scheme, Price Discovery, which is available in [6].

Also, the ingress i uses the total estimated network capacity C in calculating the V_{max} contract parameter defined in (4.1). Admission control techniques can be used to identify the best value for V_{max}. We use a simple method which does not put any restriction on V_{max}, i.e. $V_{max} = C * T$ where T is the contract length.

Budget Estimator is the sub-component that observes demand for each edge-to-edge flow. We implicitly assume that user's "budget" represents user's demand (i.e. willingness-to-pay). So, Budget Estimator estimates budget $\hat{b}_{ij}$ of each edge-

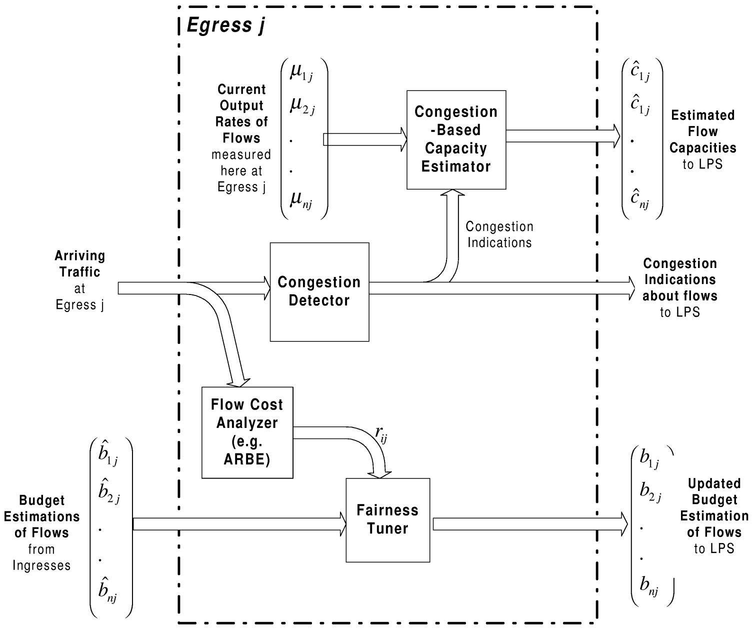


Figure 6.4: Major functions of Egress j.

to-edge traffic flow[10]. We will describe a simple algorithm that calculates $\hat{b}_{ij}$ later in Section 6.2.4.1.

6.2.2 Egress Station j

Figure 6.4 illustrates sub-components of Egress Station j in the framework: *Congestion Detector, Congestion-Based Capacity Estimator, Flow Cost Analyzer*, and *Fairness Tuner*.

Congestion Detector implements an algorithm to detect congestion in network core by observing traffic arriving at Egress j. Congestion detection can be done in several ways. We assume that interior routers mark (i.e. sets the ECN bit) the data packets if their local queue exceeds a threshold. Congestion Detector generates a "congestion indication" if it observes a marked packet in the arriving traffic.

Congestion-Based Capacity Estimator estimates available capacity $\hat{c}_{ij}$ for each edge-to-edge flow exiting at Egress j. In order to calculate $\hat{c}_{ij}$, it uses congestion indications from Congestion Detector and actual output rates μ_{ij} of the flows. The

[10]Note that edge-to-edge flow does not mean an individual user's flow. Rather it is the traffic flow that is composed of aggregation of all traffic going from one edge node to another edge node.

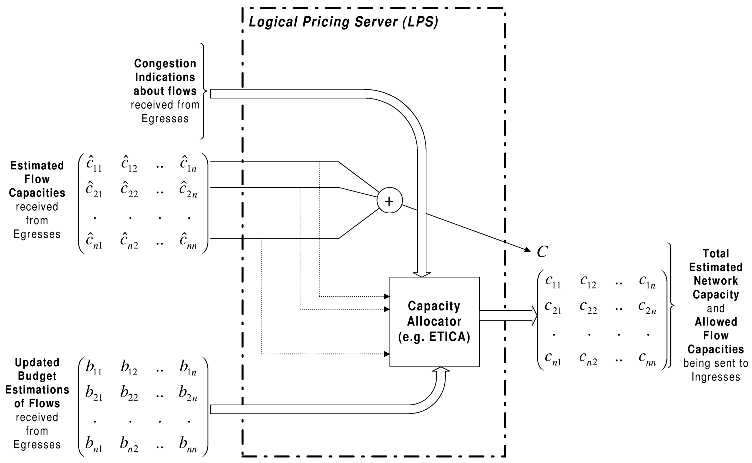


Figure 6.5: Major functions of LPS.

crucial property of Congestion-Based Capacity Estimator is that, it estimates capacity in a congestion-based manner, i.e. it decreases the capacity estimation when there is congestion indication and increases when there is no congestion indication. This makes the prices *congestion-sensitive*, since Pricing Scheme at Ingress calculates prices based on the estimated capacity. An example algorithm for Congestion-Based Capacity Estimator will be described later in Section 6.2.4.2.

Flow Cost Analyzer determines cost of each traffic flow (e.g. number of links traversed by the flow, number of bottlenecks traversed by the flow, amount of queuing delay caused by the flow) exiting at Egress j. Cost incurred by each flow can be several things: number of traversed links, number of traversed bottlenecks, amount of queuing delay caused. We assume that number of bottlenecks is a good representation of the cost incurred by a flow. In Appendix D, we define an algorithm ARBE, which estimates number of bottleneck traversed by a flow. ARBE outputs estimated number of bottlenecks $\hat{r}_{ij}$ traversed by the flow from ingress i to egress j.

LPS, as will be described in the next section, allocates capacity to edge-to-edge flows based on their budgets. The flows with higher budgets are given more capacity than the others. So, Egress j can penalize/favor a flow by increasing/decreasing its budget $\hat{b}_{ij}$. Fairness Tuner is the component that updates $\hat{b}_{ij}$. So, Fairness Tuner penalizes or favors the flow from ingress i by updating its estimated budget value, i.e. $b_{ij} = f(\hat{b}_{ij}, \hat{r}_{ij}, < parameters >)$ where $< parameters >$ are other optional

Table 6.1: List of parameters in Distributed-DCC framework.

Parameter	Symbol	Description
Contract Length (sec)	T	Length of contracts
Observation Interval(sec)	O	Time-scale of observations at Egress about congestion
LPS Interval (sec)	L	Time-scale of communication between LPS and provider stations
Edge-to-Edge Price ($/Mb)	p_{ij}	Unit price for traffic flow from i to j
Budget Estimation ($)	$\hat{b}_{ij}$	Estimation for budget of flow from i to j
Updated Budget Estimation ($)	b_{ij}	Budget Estimation for flow from i to j adjusted by Fairness Tuner
Estimated Network Capacity (Mb/s)	C	Estimation for total network capacity
Estimated Capacity (Mb/s)	$\hat{c}_{ij}$	Estimation of available capacity for flow i to j
Allowed Capacity (Mb/s)	c_{ij}	Capacity given by Capacity Allocator to flow i to j
Flow Input Rate at Ingress (Mb/s)	x_{ij}	Arrival rate of flow i to j at Ingress i
Flow Output Rate at Egress (Mb/s)	μ_{ij}	Departing rate of flow i to j at Egress j
Estimated Flow Cost	$\hat{r}_{ij}$	Estimation for amount of cost incurred by flow i to j
-	$\hat{k}$	Holding time of "congested" state in ETICA algorithm
Fairness Coefficient	α	Tuner for fairness type of Fairness Tuner

parameters that may be used for deciding how much to penalize or favor the flow. For example, if the flow ingress i is passing through more congested areas than the other flows, Fairness Tuner can penalize this flow by reducing its budget estimation $\hat{b}_{ij}$. We will describe an algorithm for Fairness Tuner later in Section 6.2.4.4.

Egress j sends $\hat{c}_{ij}$s (calculated by Congestion-Based Capacity Estimator) and b_{ij}s (calculated by Fairness Tuner) to LPS.

6.2.3 Logical Pricing Server (LPS)

Figure 6.5 illustrates basic functions of LPS in the framework. LPS receives information from egresses and calculates *allowed capacity* c_{ij} for each edge-to-edge flow. The communication between LPS and the stations take place at every *LPS interval* L. There is only one major sub-component in LPS: Capacity Allocator.

Capacity Allocator receives $\hat{c}_{ij}$s, bijs and congestion indications from Egress Stations. It calculates allowed capacity c_{ij} for each flow. Calculation of c_{ij} values is a complicated task which depends on internal topology. In general, the flows should share capacity of the same bottleneck in proportion to their budgets. We will later define a generic algorithm ETICA for Capacity Allocator in Section 6.2.4.3.

Other than functions of Capacity Allocator, LPS also calculates total available network capacity C, which is necessary for determining the contract parameter V_{max} at Ingresses. LPS simply sums $\hat{c}_{ij}$ to calculate C.

LPS can be implemented in a centralized or distributed manner (see Section 6.4.1).

6.2.4 Sub-Components

6.2.4.1 Budget Estimator

At Ingress i, Budget Estimator performs a very trivial operation to estimate budgets $\hat{b}_{ij}$ of each flow starting at Ingress i. The ingress i basically knows its current price for each flow, p_{ij}. When it receives a packet it just needs to determine which egress station the packet is going to. Given that Ingress i has the addresses of all the egress stations of the same diff-serv domain, it can find out which egress the packet is going to. So, by monitoring the packets transmitted for each flow, the ingress can estimate the budget of each flow. Let x_{ij} be the total number of packets transmitted for flow i to j in unit time, then the budget estimate for the flow i to j is $\hat{b}_{ij} = x_{ij}p_{ij}$. Notice that this operation must be done at the ingress rather than egress, because some of the packets might be dropped before arriving at the egress. This causes x_{ij} to be measured less, and hence causes $\hat{b}_{ij}$ to be less than it is supposed to be.

6.2.4.2 Congestion-Based Capacity Estimator

The essence of Congestion-Based Capacity Estimator is to decrease the capacity estimation when there is congestion indication(s) and to increase it when there is no congestion indication. In this sense, several capacity estimation algorithms can be used, e.g. Additive Increase Additive Decrease (AIAD), Additive Increase Multiplicative Decrease (AIMD). We now provide a full description of such an algorithm.

At Egress j, given congestion indications from Congestion Detector and output rate μ_{ij} of flows, Congestion-Based Capacity Estimator implements the following algorithm for each flow from Ingress i: Let O be *observation intervals* at which the estimator makes an observation about congestion status of the network. The estimator identifies each observation interval as *congested* or *non-congested*. Basically, an observation interval is congested if a congestion indication was received from Congestion Detector during that observation interval. At the end of each observation interval t, the estimator updates the estimated capacity $\hat{c}_{ij}$ as follows:

$$\hat{c}_{ij}(t) = \begin{cases} \beta * \mu_{ij}(t), & \textit{congested} \\ \hat{c}_{ij}(t-1) + \triangle\hat{c}, & \textit{non-congested} \end{cases}$$

where β is in (0,1), $\mu_{ij}(t)$ is the measured output rate of flow i to j during observation interval t, and $\triangle\hat{c}$ is a pre-defined increase parameter. This algorithm is a variant of well-known AIMD.

6.2.4.3 ETICA: Edge-to-edge, Topology-Independent Capacity Allocation

Firstly, note that LPS is going to implement ETICA algorithm as a Capacity Allocator (see Figure 6.5). So, we will refer to LPS throughout the description of ETICA below.

At LPS, we introduce a new information about each edge-to-edge flow f_{ij}. A flow f_{ij} is *congested*, if egress j has been receiving congestion indications from that flow recently (we will later define what "recent" is).

Again at LPS, let K_{ij} determine the state of f_{ij}. If $K_{ij} > 0$, LPS determines

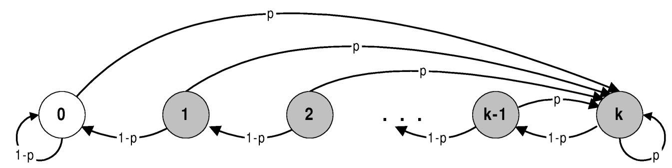


Figure 6.6: States of an edge-to-edge flow in ETICA algorithm: The states $i > 0$ are "congested" states and the state $i = 0$ is the "non-congested" state, represented with gray and white colors respectively.

f_{ij} as congested. If not, it determines f_{ij} as non-congested. At every LPS interval t, LPS calculates K_{ij} as follows:

$$K_{ij}(t) = \begin{cases} \hat{k}, & \text{congestion in } t-1 \\ max(0, K_{ij}(t-1) - 1), & \text{no congestion in } t-1 \end{cases} \tag{6.1}$$

where $\hat{k}$ is a positive integer. Notice that $\hat{k}$ parameter defines how long a flow will stay in "congested" state after the last congestion indication. So, in other words, $\hat{k}$ defines the time-line to determine if a congestion indication is "recent" or not. According to these considerations in ETICA algorithm, Figure 6.6 illustrates states of an edge-to-edge flow given that probability of receiving a congestion indication in the last LPS interval is p. Gray states are the states in which the flow is "congested", and the single white state is the "non-congested" state. Observe that number of congested states (i.e. gray states) is equal to $\hat{k}$ which defines to what extent a congestion indication is "recent". [11]

Given the above method to determine whether a flow is congested or not, we now describe the algorithm to allocate capacity to the flows. Let F be the set of all edge-to-edge flows in the diff-serv domain, and F_c be the set of *congested* edge-to-edge flows. Let C_c be the accumulation of $\hat{c}_{ij}$s where $f_{ij} \in F_c$. Further, let B_c be the accumulation of b_{ij}s where $f_{ij} \in F_c$. Then, LPS calculates the allowed capacity

[11] Note that instead of setting K_{ij} to $\hat{k}$ at every congestion indication, several different methods can be used for this purpose, but we proceed with the method in (6.1).

for f_{ij} as follows:

$$c_{ij} = \begin{cases} \frac{b_{ij}}{B_c} C_c, & K_{ij} > 0 \\ \hat{c}_{ij}, & otherwise \end{cases}$$

The intuition is that if a flow is congested, then it must be competing with other congested flows. So, a congested flow is allowed a capacity in proportion to its budget relative to budgets of all congested flows. Since we assume no knowledge about the interior topology, we can *approximate* the situation by considering these congested flows as if they are passing through a single bottleneck. If knowledge about the interior topology is provided, one can easily develop better algorithms by sub-grouping the congested flows that are passing through the same bottleneck.

In short, the ETICA algorithm basically says that a flow in one of its "congested" states gets a share[12] of the total capacity of the congested flows (i.e. C_c). If the flow is in its in "non-congested" state, then it uses its own capacity.

If a flow is not congested, then it is allowed to use its own estimated capacity, which will give enough freedom to utilize capacity available to that particular flow. Dynamics of the algorithm will be understood more clearly after the simulation experiments in Section 6.5.

6.2.4.4 Fairness Tuner

We examine the issues regarding fairness in two main cases. We first determine these two cases and then provide solutions within Distributed-DCC framework.

- *Single-bottleneck case:* The pricing protocol should charge *the same price to the users of the same bottleneck*. In this way, among the customers using the same bottleneck, the ones who have more budget will be given more rate than the others. The intuition behind this reasoning is that the cost of providing capacity to each customer is the same.

[12]Note that in this definition of ETICA, we defined this "share" as the ratio of b_{ij}/B_c which is based on f_{ij}'s monetary value with respect to monetary value of all congested flows F_c. This is because our main goal is to "price" effectively. However, one can define this share according to other criteria (such as equal to all congested flows), which makes it possible to use ETICA as a rate allocation algorithm.

- *Multi-bottleneck case:* The pricing protocol should *charge more to the customers whose traffic passes through more bottlenecks* and cause more costs to the provider. So, other than proportionality to customer budgets, we also want to allocate less rate to the customers whose flows are passing through more bottlenecks than the other customers.

  For multi-bottleneck networks, two main types of fairness have been defined: max-min fairness [44], proportional fairness [42]. In max-min fair rate allocation, all flows get equal share of the bottlenecks, while in proportional fair rate allocation flows get penalized according to the number of traversed bottlenecks. Depending on the cost structure and user's utilities, for some cases the provider may want to choose max-min or proportional rate allocation. So, we would like to have ability of tuning the pricing protocol such that fairness of its rate allocation is in the way the provider wants.

For a better understanding of proportional fairness and max-min fairness, we study them in terms of social welfare maximization with a canonical example in Appendix E.

To achieve the fairness objectives defined in the above itemized list, we introduce new parameters for tuning rate allocation to flows. In order to penalize flow i to j, the egress j can reduce $\hat{b}_{ij}$ while updating the flow's estimated budget. It uses the following formula to do so:

$$b_{ij} = f(\hat{b}_{ij}, r(t), \alpha, r_{min}) = \frac{\hat{b}_{ij}}{r_{min} + (r_{ij}(t) - r_{min}) * \alpha}$$

where $r_{ij}(t)$ is the congestion cost caused by the flow i to j, r_{min} is the minimum possible congestion cost for the flow, and α is *fairness coefficient.* Instead of $\hat{b}_{ij}$, the egress j now sends b_{ij} to LPS. When α is 0, Fairness Tuner is employing max-min fairness. As it gets larger, the flow gets penalized more and rate allocation gets closer to proportional fairness. However, if it is too large, then the rate allocation will move away from proportional fairness. Let α^* be the α value where the rate allocation is proportionally fair. If the estimation $r_{ij}(t)$ is absolutely correct, then $\alpha^* = 1$. Otherwise, it depends on how accurate $r_{ij}(t)$ is.

Assuming that each bottleneck has the same amount of congestion and capacity. Then, in order to calculate $r_{ij}(t)$ and r_{min}, we can directly use the number of bottlenecks the flow i to j is passing through. In such a case, r_{min} will be 1 and $r_{ij}(t)$ should be number of bottlenecks the flow is passing through. ARBE, in Appendix A, calculates an estimation for r_{ij}.

6.3 Edge-to-Edge Pricing Scheme (EEP)

For flow f_{ij}, Distributed-DCC framework provides an allowed capacity c_{ij} and an estimation of total user budget $\hat{b}_{ij}$ at ingress i. So, the provider station at ingress i can use these two information to calculate price. We propose a simple price formula to balance supply and demand:

$$\hat{p}_{ij} = \frac{\hat{b}_{ij}}{c_{ij}} \tag{6.2}$$

Here, $\hat{b}_{ij}$ represents user demand and c_{ij} is the available supply.

In Chapter 8, we will provide a detailed optimization analysis of this EEP pricing scheme in Distributed-DCC framework. We will show that the price calculation formula in (6.2) is optimal for the well-known total user utility maximization problem. We will also consider effect of different utility functions and elasticities of users on optimal prices.

6.4 Adaptation of Distributed-DCC to PFCC Architecture

In order to adapt Distributed-DCC to PFCC architecture, LPS must operate on very low time-scales. In other words, LPS interval must be small enough to maintain control over congestion, since PFCC assumes no underlying congestion control mechanism. This raises two issues to be addressed:

- In order to maintain human involvement into the system,intermediate agents between customers and Ingress stations must be implemented.

- Since LPS must operate at very small time-scales, scalability issues regarding LPS must be solved.

As we previously said earlier in Section 1.3, we do not focus on the first problem since it cannot be addressed within this thesis because of its large size and complexity. So, we assume that customers are willing to undertake high price variations, and leave development of necessary intermediate agents for future research. We address the second problem in the following sub-section.

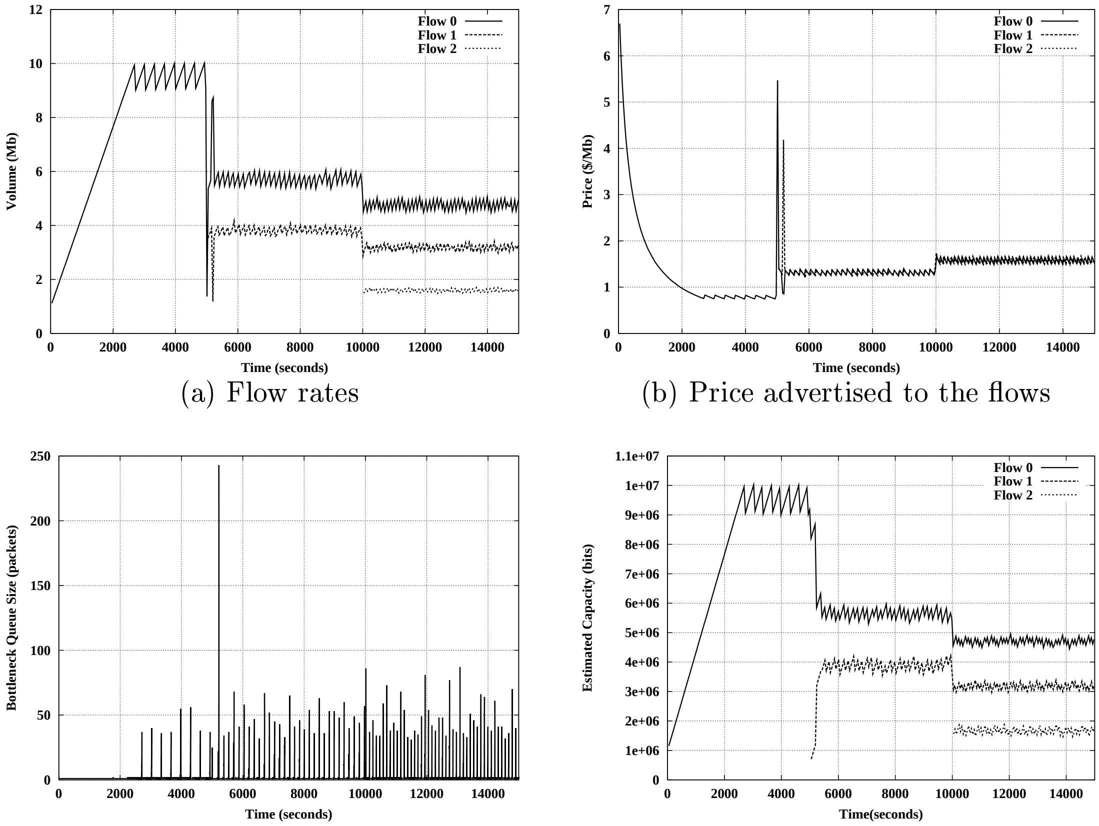
6.4.1 Scalability

Distributed-DCC operates on per edge-to-edge flow basis. There are mainly two issues regarding scalability: LPS, the number of flows. First of all, the flows are not per-connection basis, i.e. all the traffic going from edge router i to j is counted as only one flow. This actually relieves the scalability problem for operations that happen on per-flow basis. The number of flows in the system will be $n(n-1)$ where n is the number of edge routers in the diff-serv domain. So, indeed, scalability of the flows is not a problem for the current Internet since number of edge routers for a single diff-serv domain is very small. If it becomes so large in future, then aggregation techniques can be used to overcome this scalability issue, of course, by sacrificing some optimality.

Scalability of LPS can be done in two ways. First idea is to implement LPS in a fully distributed manner. The edge stations exchange information with each other (similar to link-state routing). Basically, each station will send total of $n-1$ messages, each of which headed to other stations. So, this will increase the overhead on the network because of the extra messages, i.e. the complexity will increase from $O(n)$ to $O(n^2)$ in terms of number of messages.

Alternatively, LPS can be divided into multiple local LPSs which synchronize among themselves to maintain consistency. This way the complexity of number of messages will reduce. However, this will be at a cost of some optimality again.

Since these above-defined scaling techniques are very well-known, we do not focus on detailed description of them.



(a) Flow rates

(b) Price advertised to the flows

(c) Bottleneck queue length

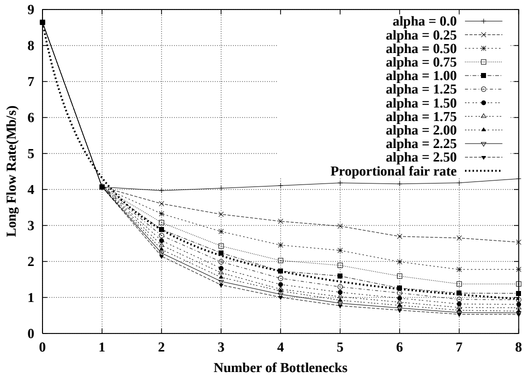
(d) Estimated Capacity, $\hat{c}_{ij}$, of flows

Figure 6.7: Results of single-bottleneck experiment for EEP.

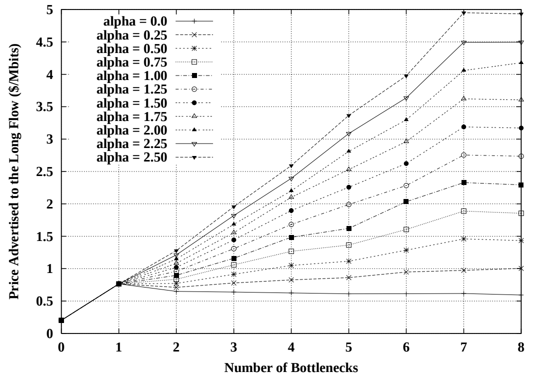
6.5 Simulation Experiments and Results

We now present *ns* [1] simulation experiments of EEP on single-bottleneck and multi-bottleneck topology. Our goals are to illustrate fairness and stability properties of the scheme.

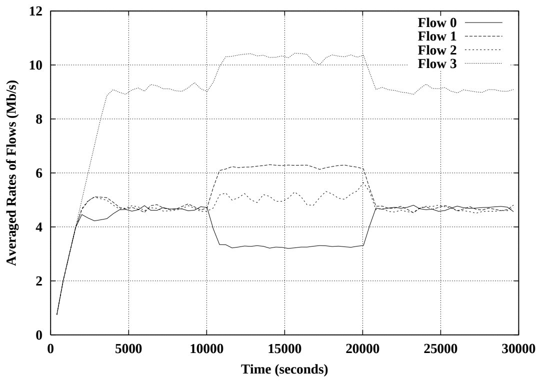
The single-bottleneck topology has a bottleneck link, which is connected to n edge nodes at each side where n is the number of users. The multi-bottleneck topology has $n-1$ bottleneck links, that are connected to each other serially. There are again n ingress and n egress edge nodes. Each ingress edge node is mutually connected to the beginning of a bottleneck link, and each egress node is mutually connected to the end of a bottleneck link. All bottleneck links have a capacity of 10Mb/s and all other links have 15Mb/s. Propagation delay on each link is 5ms, and users send UDP traffic with an average packet size of 1000B. To ease



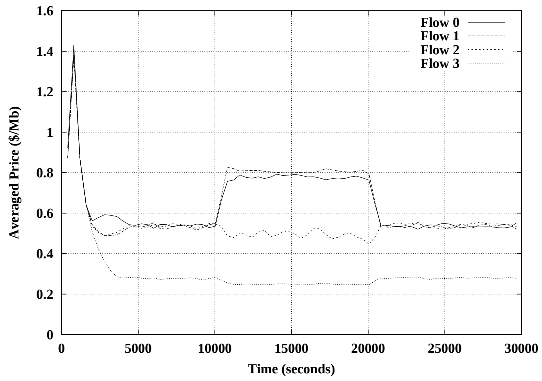
(a) Rate of long flow



(b) Price advertised to the long flow



(c) Fow rates



(d) Prices advertised to flows

Figure 6.8: Results of EEP experiments on multi-bottleneck topology.

understanding the experiments, each user sends its traffic to a separate egress. For the multi-bottleneck topology, one user sends through all the bottlenecks (i.e. long flow) while the others cross that user's long flow. The queues at the interior nodes (i.e. nodes that stand at the tips of bottleneck links) mark the packets when their local queue size exceeds 30 packets. Buffer size is assumed to be infinite. In the multi-bottleneck topology they increment a header field instead of just marking. Figure 3.2-a shows a single-bottleneck topology with $n = 3$. Figure 3.2-b shows multi-bottleneck topology with $n = 4$. The white nodes are edge nodes and the gray nodes are interior nodes. These figures also show the traffic flow of users on the topology.

The user flow tries to maximize its surplus by contracting for b/p amount of capacity, where b is its budget and p is price. The flows's budgets are randomized according to truncated-Normal distribution with a given mean value. This mean value is what we will refer to as flows's budget in our simulation experiments.

Ingresses send budget estimations to corresponding egresses at every *observation interval*. LPS sends information to ingresses at every *LPS interval*. Contracting takes place at every 4s, observation interval is 0.8s, and LPS interval is 0.16s. The parameter $\hat{k}$ is set to 200, which means a flow is determined to be non-congested at least after (please see Section 6.2.4.3) 200 LPS intervals equivalent to 8 contracting intervals.

The parameter $\triangle\hat{c}$ is set to 1 packet (i.e. 1000B), the initial value of $\hat{c}_{ij}$ for each flow f_{ij} is set to 0.1Mb/s, and β is set to 0.95.

6.5.1 Experiment on Single-bottleneck Topology

We run simulation an experiment for EEP on the single-bottleneck topology, which is represented in Figure 3.2-a. In this experiment, there are 3 users with budgets of 30, 20, 10 respectively for users 1, 2, 3. Total simulation time is 15000s, and at the beginning only the user 1 is active in the system. After 5000s, the user 2 gets active. Again after 5000s at simulation time 10000, the user 3 gets active.

In terms of results, each flow's rate is very important. Figure 6.7-a shows the flow rates. We see the flows are sharing the bottleneck capacity in proportion to their budgets. Also, Figure 6.7-d shows the estimated capacity $\hat{c}_{ij}$ of flows at LPS. Observe the correspondence between flow rates in Figure 6.7-a and the estimated capacities in Figure 6.7-d.

Figure 6.7-b shows the price being advertised to flows. As the new users join in, EEP increases the price in order to balance supply and demand. Also, we can see the same dynamic as in the volume allocation graphs.

Figure 6.7-c shows the bottleneck queue size. Notice that queue sizes make peaks transiently at the times when new users gets active. Otherwise, the queue size is controlled reasonably and the system is stable. The reason behind the transient peaks is that the parameter V_{max} is not restricted which allows the newly joining flow to contract up to all the available capacity. Since multiple flows have this ability to contract up to the whole capacity, they can potentially contract for the

whole capacity simultaneously which causes large queues. As we will also mention in discussions of future work, this is basically an admission control problem, which is out-of-scope for this particular work.

During the simulation, average utilization of the bottleneck link was more than 90%, and no packet drops were allowed.

6.5.2 Experiments on Multi-bottleneck Topology

On a multi-bottleneck network, we would like illustrate two properties for EEP:

- *Property 1:* provision of various fairness in rate allocation by changing the fairness coefficient α of Distributed-DCC framework

- *Property 2:* performance of the capacity allocation algorithm in terms of adaptiveness (see Section 6.2.4.3)

In order to illustrate Property 1, we run a series of experiments for EEP with different α values. We use a larger version of the topology represented in Figure 3.2-b. In the multi-bottleneck topology there are 10 users and 9 bottleneck links. Total simulation time is 10,000s. At the beginning, the user with the long flow is active. After each 1000s, one of these other users gets active. So, as the time passes the number of bottlenecks in the system increases since new users with crossing flows join in. We are interested in the rate of the long flow, since it is the one that cause more congestion costs than the other user flows.

Figure 6.8-a shows the average rate of the long flow versus the number of bottlenecks in the system. As expected the long flow gets less and less capacity as α increases. When $\alpha = 0$, the scheme achieves max-min fairness. Observe that when $\alpha = 1$, rate allocation goes along with proportionally fair rate allocation. This variation in fairness is basically achieved by advertisement of different prices to the user flows. Figure 6.8-b shows the average price that is advertised to the long flow as the number of bottlenecks in the system increases. We can see that the price advertised to the long flow increases as the number of bottlenecks increases. As α increases, the scheme becomes more responsive to the long flow by increasing its price more sharply.

Finally, to illustrate Property 2, we ran an experiment on the topology in Figure 3.2-b with small changes. We increased capacity of the bottleneck at node D from 10 Mb/s to 15Mb/s. There are four flows and three bottlenecks in the network as represented in Figure 3.2-b. Initially, all the flows have an equal budget of 10. Total simulation time is 30000s. Between times 10000 and 20000, budget of flow 1 is temporarily increased to 20. The fairness coefficient α is set to 0, and the parameter $\hat{k}$ is set to 25 which is equal to one contracting period. All the other parameters are exactly the same as in the single-bottleneck experiments of the previous section.

Figure 6.8-c shows the given volumes averaged over 200 contracting periods. Similarly, Figure 6.8-d shows the advertised prices averaged over 200 contracting periods. Until time 10000s, flows 0, 1, and 2 share the bottleneck capacities equally presenting a max-min fair allocation because α was set to 0. However, flow 3 is getting more than the others because of the extra capacity at bottleneck node D. This flexibility is achieved by the freedom given to individual flows by the ETICA capacity allocation algorithm (see Section 6.2.4.3). Note that the parameter $\hat{k}$ plays a crucial role in terms of functioning of the ETICA algorithm. In general, $\hat{k}$ should be small for multi-bottleneck topologies and large for single-bottleneck topologies. We will later show importance of $\hat{k}$ by extensive simulations in Chapter 9.

Between times 10000 and 20000, flow 2 gets a step increase in its allocated volume because of the step increase in its budget. In result of this, flow 0 gets a step decrease in its volume. Also, flows 2 and 3 adapt themselves to the new situation by attempting to utilize the extra capacity leftover from the reduction in flow 0's volume. So, flow 2 and 3 gets a step decrease in their volumes. After time 20000, flows restore to their original volume allocations, illustrating the adaptiveness of the scheme.

6.6 Summary

In this chapter, we presented Distributed-DCC framework, which is an extension of DCC in Chapter 4, for congestion pricing in a single diff-serv domain. Particularly, we considered the PFCC architecture within Distributed-DCC framework. We will study the POCC architecture in the next chapter.

Distributed-DCC can provide a contracting framework based on *short-term* contracts between user application and the service provider. By simulation, we showed stability of Distributed-DCC framework. We investigated fairness issues within Distributed-DCC and illustrated ways of achieving a *range of fairness types* (i.e. from max-min to proportional) through congestion pricing under certain conditions. The fact that it is possible to achieve various fairness types within a single framework is very encouraging. By using the EEP pricing scheme within the Distributed-DCC framework, we presented several simulation experiments showing the framework's performance.

To prevent confusions, we would like to stress that Distributed-DCC framework does not aim to replace end-based congestion control or traffic engineering techniques. These end-based traffic engineering techniques operate on the order of RTTs, i.e. milliseconds. However, Distributed-DCC operates on the order of short-term contracts, i.e. minutes. Also, Distributed-DCC operates at network "edge"s rather than network "end"s. Notice that this does not conflict with already deployed end-based congestion control algorithms such as TCP.

Future work should include investigation of issues related to extending Distributed-DCC on multiple diff-serv domains. Another future work item is to introduce soft admission control techniques in the framework by properly tuning the contract parameter V_{max}. Currently, V_{max} is set to total network capacity, which allows individual flows to contract for significantly larger than the network can handle. Several other improvements are possible to the framework such as better congestion-based capacity estimation techniques (see Section 6.2.4.2), better budget estimation techniques (see Section 6.2.4.1).

CHAPTER 7
DISTRIBUTED-DCC:
Pricing over Congestion Control (POCC)

7.1 Introduction

In the previous chapter we presented basics of Distributed-DCC along with experimental proof for its fairness and stability. There, in the previous chapter, we described Distributed-DCC framework based on the PFCC architecture. In this chapter we focus on overlaying Distributed-DCC over edge-to-edge congestion control schemes, i.e. the POCC architecture.

Among many others, one major implementation obstacle can be defined as the need for *frequent price updates*. This is relatively very hard to achieve in a wide area network such as the Internet, since users need to be informed about every price update. In [11], the authors showed that users do need feedback about charging of the network service (such as current price and prediction of service quality in near future). However, in Chapter 5, we illustrated that congestion control by pricing cannot be achieved if price changes are performed at a time-scale larger than roughly 40 round-trip-times (RTTs). This means that in order to achieve congestion control by pricing, service prices must be updated very frequently (i.e. 2-3 seconds since RTT is expressed in terms of milliseconds for most cases in the Internet).

We propose a new pricing architecture as a novel solution: Pricing over Congestion Control (POCC). POCC overlays pricing on top of an underlying congestion control mechanism to make sure congestion is controlled at low time-scales. This way the pricing mechanism on top can operate at larger time-scales, which makes human involvement possible.

Since Distributed-DCC is a pricing framework specifically designed for edge-to-edge structure, we particularly focus on diff-serv [27] architecture. So, we use an available "edge-to-edge" congestion control mechanism (Riviera [36, 37]) in order to present the idea of pricing overlay over congestion control. We present simulation results for Distributed-DCC over Riviera, and illustrate benefits of overlay pricing on top of congestion control.

This chapter is organized as follows: In Section 7.2, we briefly describe an edge-to-edge congestion control mechanism (Riviera), which we will use later in simulation experiments. Next in Section 7.3, we present POCC ideas in detail and describe solutions to potential problems. In Section 7.4, we present simulation experiments of Distributed-DCC over Riviera and evaluate POCC ideas by comparison to PFCC. We finalize with summary and discussions.

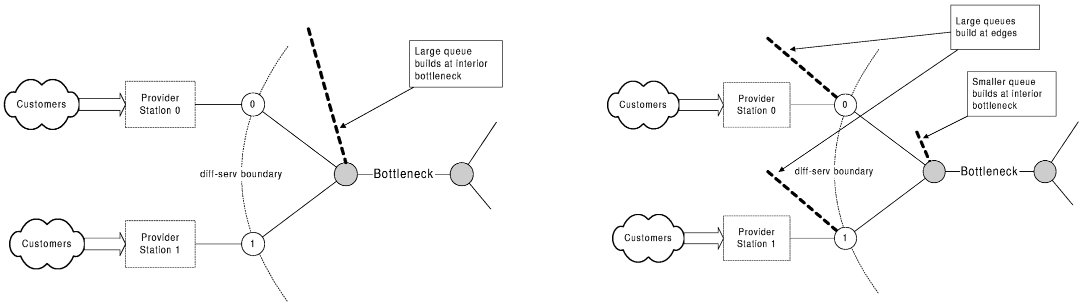
7.2 Edge-to-Edge Congestion Control: Riviera

We now describe overall properties of an edge-to-edge congestion control scheme, Riviera [36, 37], which we will also use in our experiments later in the chapter.

Riviera takes advantage of two-way communication between ingress and egress edge routers in a diff-serv network. Ingress sends a *forward* feedback to egress in response to feedback from egress, and egress sends *backward* feedback to ingress in response to feedback from ingress. So, ingress and egress of a traffic flow keep bouncing feedback to each other. Ignoring loss of data packets, the egress of a traffic flow measures the accumulation, a, caused by the flow by using the bounced feedbacks and RTT estimations.

The egress node keeps two threshold parameters to detect congestion: max_thresh and min_thresh. For each flow, the egress keeps a variable that says whether the flow is congested or not. When a for a particular flow exceeds max_thresh, the egress updates the variable to *congested*. Similarly, when a is less than min_thresh, it updates the variable to *not-congested*. It does not update the variable if a is in between max_thresh and min_thresh. The ingress node gets informed about the congestion detection by backward feedbacks and employs AIMD-ER (i.e. a variant of regular AIMD) to adjust the sending rate.

In a single-bottleneck network, Riviera can be tuned such that each flow gets weighted share of the bottleneck capacity. The ingress nodes maintain an additive increase parameter, α, and a multiplicative decrease parameter, β, for each edge-to-edge flow. These parameters are used in AIMD-ER. Among the edge-to-edge flows, by setting the increase parameters (α) at the ingresses and the threshold parameters



(a) without congestion control (b) with congestion control

Figure 7.1: Different pricing architectures with/without edge-to-edge congestion control.

(max_thresh and min_thresh) at the egresses in ratio of desired rate allocation, it is possible to make sure that the flows get the desired rate allocation. For example, assume there are two flows 1 and 2 competing for a bottleneck (similar to Figure 3.2-a). If we want flow 1 to get a capacity of w times more than flow 2, then the following conditions must be hold:

1. $\alpha_2 = w\ \alpha_1$
2. $max\_thresh_2 = w\ max\_thresh_1$
3. $min\_thresh_2 = w\ min\_thresh_1$

7.3 Pricing over Congestion Control (POCC)

The essence of POCC is to overlay pricing on top of congestion control, which is a novel approach. Assuming that there is an underlying edge-to-edge congestion control scheme, we can set the parameters of that underlying scheme such that it leads to fairness and better control of congestion. The pricing scheme on top can determine users's willingness-to-pay and set the parameters of the underlying edge-to-edge congestion control scheme accordingly. This way, it will be possible to favor some traffic flows with higher willingness-to-pay (i.e. budget) than the others. Furthermore, the pricing scheme will also bring benefits such as an indirect control on user demand by price, which will in turn help the underlying edge-to-edge congestion control scheme to operate more smoothly. However the overall system performance (e.g. fairness, utilization, throughput) will be dependent on the flexibility of the underlying congestion control mechanism.

Figure 7.1 illustrates the difference between a POCC architecture and a regular pricing architecture without underlying congestion control. We now first describe the problems raised by POCC architecture in diff-serv environment, then describe Distributed-DCC (i.e. an edge-to-edge pricing mechanism) and Riviera (i.e. an edge-to-edge congestion control mechanism), and then provide solutions to the problems for overlaying Distributed-DCC over Riviera.

7.3.1 POCC: Problems

In diff-serv environment, overlaying pricing on top of congestion control raises two major problems:

1. *Parameter mapping:* Since the pricing scheme wants to allocate network capacity according to the users's willingness-to-pay (i.e. the users with greater budget should get more capacity) that changes dynamically over time, it is a required ability to set corresponding parameters of the underlying edge-to-edge congestion control mechanism such that it allocates the capacity to the user flows according to their willingness-to-pay. So, this raises need for a method of mapping parameters of the pricing scheme to the parameters of the underlying congestion control mechanism. Notice that this type of mapping requires the congestion control mechanism to be able to provide parameters that tunes the rate being given to the edge-to-edge flows.

2. *Edge queues:* The underlying congestion control scheme will not always allow all the traffic admitted by the pricing scheme, which will cause queues to build up at the network edges. So, management of these edge queues is necessary in POCC architecture. Figures 7.1-a and 7.1-b compare the situation of the edge queues in the two cases when there is an underlying congestion control scheme and when there is not. Notice that this problem is very similar to inventory management problem. The packets in the edge queue correspond

Table 7.1: Differences between Distributed-DCC's PFCC and POCC versions.

DISTRIBUTED-DCC: PFCC	DISTRIBUTED-DCC: POCC
LPS must operate at small time-scales	LPS may operate at large time-scales
LPS must be scaled because of its operational time-scale	It is not necessary to scale LPS
Framework can achieve a range of fairness in rate allocation	Fairness of rate allocation is limited and depends on the underlying congestion control mechanism
Bottleneck queues at network core are large	Bottleneck queues at network core are small
Does not need to manage queues at network edges	Need to manage queues at network edges

to items in the inventory. Customer(s) is the variable capacity pipe, provided by the underlying congestion control scheme, that consumes the packets by transmitting them. Ordering of new items corresponds to reducing the price of the network service.

Other than the above two major problems, another problem is that the overall performance of the system will be dependent on not only the pricing scheme's performance, but also the performance of the underlying congestion control scheme. For instance, if the underlying congestion control scheme does not allow the network to be utilized more than 80%, then the utilization provided by the overall system will be limited by 80%. Also, if the underlying congestion control scheme is unable to provide some fairness types (e.g. max-min fairness) in the rate allocation, then the overall system will not be able to provide those fairness types.

7.3.2 POCC: Solutions for Distributed-DCC over Riviera

Adaptation of Distributed-DCC to POCC architecture has differences from adapting it to PFCC architecture. Table 7.1 lists the major differences of two. Similarly, Figure 7.2 shows the time-scales for PFCC and POCC versions of Distributed-DCC. Observe that LPS interval is supposed to be less than contracting interval in the case of PFCC, while it can be larger in the case of POCC.

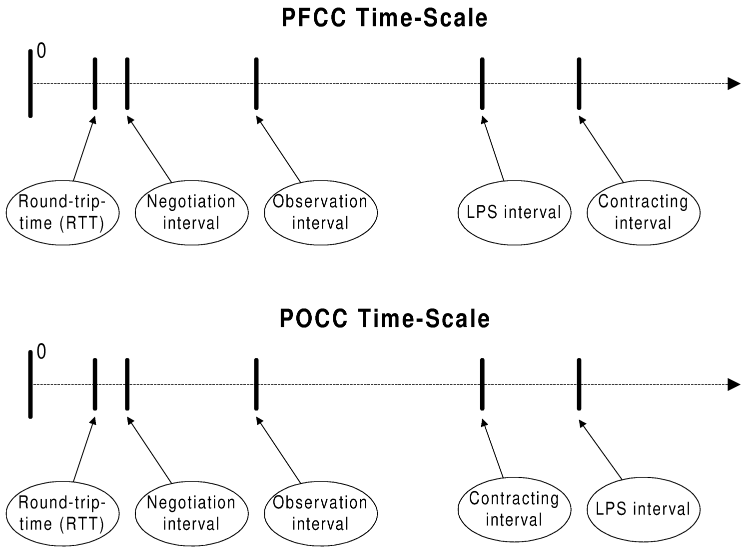


Figure 7.2: Time-scales of various parameters in Distributed-DCC for PFCC and POCC architectures.

For the case of Distributed-DCC over Riviera, we now provide solutions to the two major problems raised by overlaying pricing over edge-to-edge congestion control:

1. *Parameter mapping:* For each edge-to-edge flow, Distributed-DCC can calculate the capacity share of that flow out of the total network capacity. Let $\gamma_{ij} = c_{ij}/C$ be the fraction of network capacity that must be given to the flow i to j. Distributed-DCC can convey γ_{ij}s to the ingress stations, and they can multiply the increase parameter α_{ij} with γ_{ij}. Also, Distributed-DCC can communicate γ_{ij}s to the egresses, and they can multiply max_thresh_{ij} and min_thresh_{ij} with γ_{ij}. This solves the parameter mapping problem defined in Section 7.3.1.

2. *Edge queues:* We now propose solutions to the second problem, i.e. management of edge queues. In Distributed-DCC, ingress stations maintain an estimation of available capacity for each edge-to-edge flow. So, one intuitive way of making sure that the user will not contract for more than the amount that the network can handle is to subtract necessary capacity to drain the already built edge queue from the estimated edge-to-edge capacity c_{ij}, and then

make contracts accordingly. In other words, the ingress station updates the estimated capacity for flow i to j by the following formula $c'_{ij} = c_{ij} - Q_{ij}/T$, and uses c'_{ij} for price calculation. Note that Q is the actual edge queue length, and T is the length of the contract.

Within Distributed-DCC framework, one can also employ another technique to manage the edge queues. Remember that the egress nodes are making capacity estimation depending on if marked packets have arrived or not. Specifically, they reduce the estimated capacity of a flow to a fraction of its current output rate, when a marked packet was received in the last observation interval. So, the provider station at the ingress can mark the packets if size of the edge queue exceeds a threshold. This will indirectly reduce the capacity estimation, and hence drain the edge queue. Notice that it is also possible to employ this method simultaneously with the method described in the previous paragraph. In the simulation experiments of the next section we employ both of them simultaneously.

7.4 Simulation Experiments and Results

We now present *ns* [1] simulation experiments of Distributed-DCC over Riviera on single-bottleneck topology, in order to illustrate POCC ideas. We also make comparative evaluation of PFCC and POCC by using the PFCC results in Section 6.5.

We run simulation experiments for POCC on the single-bottleneck topology, which is represented in Figure 3.2-a. The experiment configuration is exactly the same as the experiment configuration for the single bottleneck topology in Section 6.5. However, there is an additional component in the simulation: edge queues. The edge queues mark the packets when queue size exceeds 200 packets. So, in order to manage the edge queues in this simulation experiment, we simultaneously employ the two techniques defined in the previous section.

There are 3 users with budgets of 30, 20, 10 respectively for users 1, 2, 3. Total simulation time is 15000s, and at the beginning only the user 1 is active in the system. After 5000s, the user 2 gets active. Again after 5000s at simulation time

10000, the user 3 gets active.

In terms of results, the volume given to each flow is very important. Figures 6.7-a and 7.3-a show the flow rates in PFCC and POCC respectively. We see the flows are sharing the bottleneck capacity almost in proportion to their budgets. In comparison to POCC, PFCC allocates the rate more smoothly to the flows but with the same proportionality. The noisy volume allocation in POCC is caused by coordination issues (i.e. parameter mapping, edge queues) investigated in Section 7.3.1.

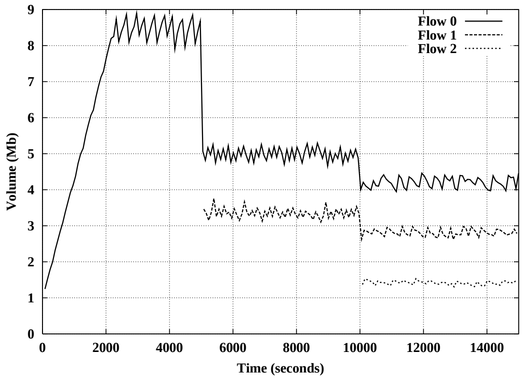
Figure 7.3-b shows the price being advertised to flows in POCC. As the new users join in, the pricing scheme increases the price in order to balance supply and demand.

Figures 6.7-c and 7.3-c shows the bottleneck queue size in PFCC and POCC respectively. Notice that queue sizes make peaks transiently at the times when new users gets active. Otherwise, the queue size is controlled reasonably and the system is stable. In comparison to PFCC, POCC manages the bottleneck queue much better because of the tight control enforced by the underlying edge-to-edge congestion control algorithm Riviera. The results follows with the big picture presented in Figure 7.1.

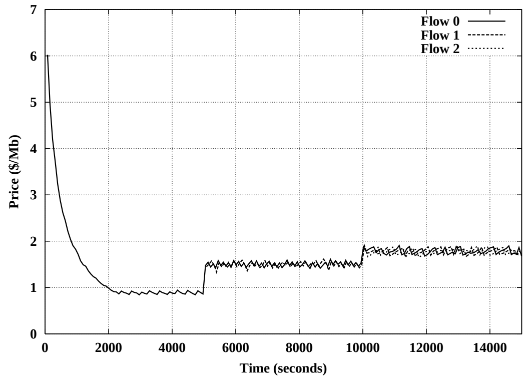
Figures 6.7-d and 7.3-d show the instantaneous estimated capacity $\hat{c}_{ij}$ parameter for the flows respectively in PFCC and POCC. We observe more variance in the case of POCC, which is again caused by the interaction between the pricing scheme and the underlying congestion control mechanism.

Figures from 7.4-a to 7.4-c show the sizes of edge queues in Distributed-DCC over Riviera. We can observe that users get active at 5000s of intervals. We observe stable behavior but with oscillations larger than the bottleneck queue illustrated in Figure 7.3-c. This is because of the tight edge-to-edge congestion control, which pushes backlog to the edges.

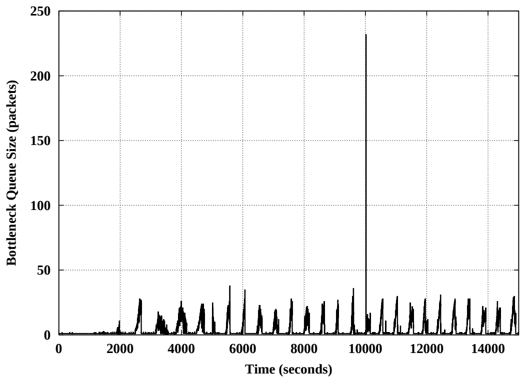
Also, observe that the edge queues are generally much lower than the threshold of 200 packets. This means that the packets were marked at the edge queues very rarely. So, the technique of marking the packets at the edges and reducing the estimated capacity indirectly was not dominant in this simulation. Rather, the



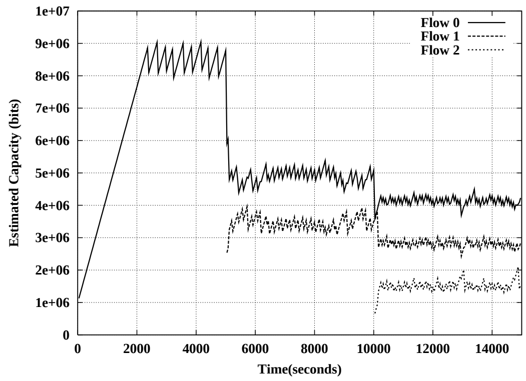
(a) Averaged flow rates



(b) Averaged prices



(c) Bottleneck queue length



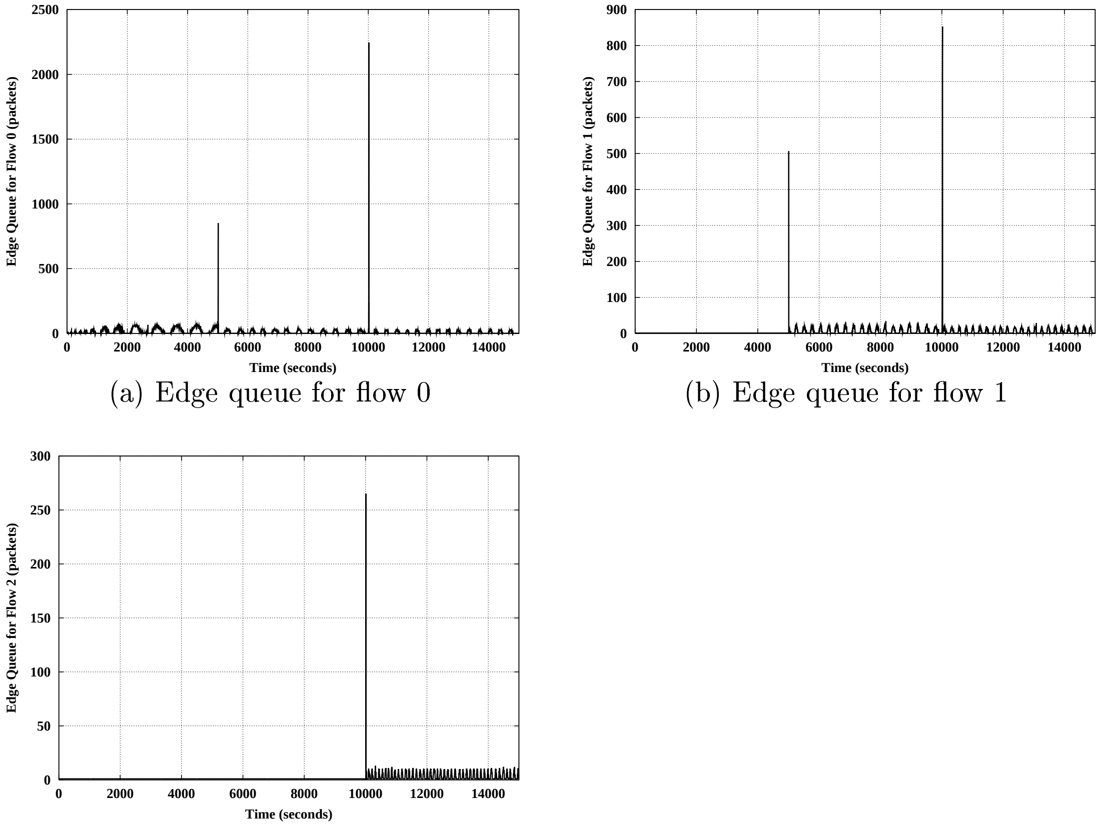
(d) Estimated Capacity, $\hat{c}_{ij}$, of flows

Figure 7.3: Results of single-bottleneck experiment for POCC.

technique of reducing the estimated capacity directly at the ingress was dominant in terms of handling of edge queues (please refer to the previous section for full understanding of these two techniques). This raises the issue of what exactly should the edge queue threshold be? This is a topic for further research.

7.5 Summary

In this chapter, we presented a new architecture to implement congestion pricing in large networks. We proposed Pricing over Congestion Control (POCC) as a novel approach in order to solve the time-scale problem of pricing. By comparative evaluation, we showed that POCC performs better in terms of managing congestion in network core because of the tight (low time-scale) control enforced by the underlying congestion control mechanism.



(a) Edge queue for flow 0

(b) Edge queue for flow 1

(c) Edge queue for flow 2

Figure 7.4: Edge queues in the single-bottleneck experiment for POCC.

Given that edge-to-edge congestion control mechanisms are available and can manage congestion in finer time-scales than pricing, we have studied issues related to overlaying pricing over edge-to-edge congestion control schemes. We defined two major problems: First one is parameter mapping between the overlaid pricing scheme and the underlying edge-to-edge congestion control scheme. The second one is management of edge queues raised by excessive traffic that is not allowed into the network because of tight control of the underlying congestion control scheme.

We provided solutions to the outlined problems within Distributed-DCC framework. In order to overlay Distributed-DCC on top of an edge-to-edge congestion control scheme, it is necessary that the edge-to-edge congestion control scheme provides parameter(s) that can be set dynamically for tuning the rate allocation to edge-to-edge flows. We finally presented simulations of Distributed-DCC over Riviera [36, 37], which is an edge-to-edge congestion control scheme developed by Harrison et al.

Future research should focus on finding better algorithms for management of edge queues, and investigating issues related to extending POCC ideas on multiple diff-serv domains. Also, POCC ideas should be tested with edge-to-edge pricing and congestion control schemes other than Distributed-DCC and Riviera.

CHAPTER 8
OPTIMIZATION ANALYSIS OF EDGE-TO-EDGE PRICING (EEP)

8.1 Introduction

In Chapters 4 and 6, we described and used a pricing scheme, EEP, within DCC and Distributed-DCC frameworks respectively. The main idea of the EEP is to balance supply and demand by equating price to the ratio of users' budget (i.e. demand) B by available capacity C. We developed methods of estimating users' budget and available capacity. Based on that, we used the pricing formula:

$$p = \frac{\hat{B}}{\hat{C}} \tag{8.1}$$

where $\hat{B}$ is the users' estimated budget and $\hat{C}$ is the estimated available network capacity. The capacity estimation is performed based on congestion level in the network, and this makes the EEP scheme a congestion-sensitive pricing scheme (see Section 6.2.4.2).

In this chapter, we will provide theoretical proof that (8.1) is optimal in the case of *logarithmic* user utilities. Further we will also show how to calculate optimal prices in the case of *non-logarithmic*[13] concave utilities.

We will also investigate users' elasticity to price and bandwidth. Specifically, we will first define different types of user elasticities, and then look at effect of these elasticities on optimal prices.

The chapter is organized as follows: First in Section 8.2, we define the optimization problem of total user utility maximization and split it into two sub-problems by following Kelly et al.'s [42] work. Next in Section 8.3, we solve the sub-problems for the case of logarithmic utility functions for user flows. Then we define utility-bandwidth elasticity and its relationship to demand-price elasticity in

[13] Note that non-logarithmic does not mean convex utility functions. Our proofs are valid only for concave utility functions.

Section 8.4. In Section 8.5, based on the elasticity definitions in Section 8.4, we define a general non-logarithmic utility function and re-solve the optimization problem for this utility function of flows. Finally, we summarize the work in this chapter in Section 8.6.

8.2 Problem Formulation

We now formulate the problem of *total user utility maximization* for a multi-user multi-bottleneck network. Another problem formulation for such a network is the *social welfare maximization*. These two formulations aim to optimize different things, but we show that both result in the same optimal price formulas under certain conditions. In this chapter we formulate and solve the former problem. Since formulation and solution to the latter problem is very similar, they are provided in Appendix B.

Also, note that optimization problem being solved is based on the assumption that each link in the network has an associated local price, just like in Low et al.'s [47] pricing framework. Notice that this violates the fundamental design principles of Distributed-DCC framework. This means our optimization study of EEP here is theoretically correct while Distributed-DCC framework trades off some optimality for implementation purposes. We now return back to formulation of the optimization problem in order to address the price calculation in EEP, which is a pricing scheme that was proposed in Chapter 6 for Distributed-DCC framework.

Let $F = \{1, ..., F\}$ be the set of flows and $L = \{1, ..., L\}$ be the set of links in the network. Also, let $L(f)$ be the set of links the flow f passes through and $F(l)$ be the set of flows passing through the link l. Let c_l be the capacity of link l. Let λ be the vector of flow rates and λ_f be the rate of flow f. We can formulate the total user utility maximization problem as follows:

$SYSTEM$:

$$\max_{\lambda} \sum_{f} U_f(\lambda_f)$$

subject to

$$\sum_{f \in F(l)} \lambda_f \leq c_l, \quad l = 1, ..., L \tag{8.2}$$

This problem can be divided into two separate problems by employing monetary exchange between user flows and the network provider. Following Kelly's [41] methodology we split the system problem into two:

The first problem is solved at the user side. Given accumulation of link prices on the flow f's route, p^f, what is the optimal sending rate in order to *maximize surplus*.

$FLOW_f(p^f)$:

$$\max_{\lambda_f} \left\{ U_f(\lambda_f) - \sum_{l \in L(f)} p_l \lambda_f \right\}$$

over

$$\lambda_f \geq 0 \tag{8.3}$$

The second problem is solved at the provider's side. Given sending rate of user flows (which are dependent on the link prices), what is the optimal price to advertise in order to *maximize revenue*.

$NETWORK(\lambda(p^f))$:

$$\max_{p} \sum_{f} \sum_{l \in L(f)} p_l \lambda_f$$

subject to

$$\sum_{f \in F(l)} \lambda_f \leq c_l, \quad l = 1, ..., L$$

over

$$p \geq 0 \tag{8.4}$$

Let the total price paid by flow f be $p^f = \sum_{l \in L(f)} p_l$. Then, solution to $FLOW_f(p^f)$ will be:

$$U'_f(\lambda_f) = p^f$$

$$\lambda_f(p^f) = U'^{-1}_f(p^f) \tag{8.5}$$

When it comes to the $NETWORK(\lambda(p^f))$ problem, the solution will be dependent on user flows utility functions since their sending rate is based on their utility functions as shown in the solution of $FLOW_f(p^f)$. So, in the next sections we will solve the $NETWORK(\lambda(p^f))$ problem for the cases of logarithmic and non-logarithmic utility functions.

8.3 Optimal Prices: Logarithmic Utility Functions

We model customer i's utility with the well-known function [14] [42, 44, 58, 47]

$$u_i(x) = w_i log(x) \tag{8.6}$$

where x is the allocated bandwidth to the customer and w_i is customer i's budget (or bandwidth sensitivity).

Now, we set up a vectorized notation, then solve the revenue maximization problem $NETWORK(\lambda(p^f))$ described in the previous section. Assume the network includes n flows and m links. Let λ be row vector of the flow rates (λ_f for $f \in F$), P be column vector of the price at each link (p_l for $l \in L$). Define the $n \times n$ matrix P^* in which the diagonal element P^*_{jj} is the aggregate price being advertised to flow j (i.e. $p^j = \sum_{l \in L(j)} p_l$) and all the other elements are 0. Also, let A be the $n \times m$ routing matrix in which the element A_{ij} is 1 if ith flow is passing though jth link and the element A_{ij} is 0, if not, C be the column vector of link capacities (c_l for $l \in L$). Finally, define the $n \times n$ matrix $\hat{\lambda}$ in which the diagonal element $\hat{\lambda}_{jj}$ is the rate of flow j (i.e. $\hat{\lambda}_{jj} = \lambda_j$) and all the other elements are 0.

Given the above notation, relationship between the link price vector P and the flow aggregate price matrix P^* can be written as:

$$AP = P^* e \tag{8.7}$$

$$\lambda = (\hat{\lambda} e)^T = e^T \hat{\lambda}$$

where e is the column unit vector.

[14]Wang and Schulzrinne introduced a more complex version in [79].

We use the utility function of (8.6) in our analysis. By plugging (8.6) in (8.5) we obtain flow's demand function in vectorized notation:

$$\lambda(P^*) = W{P^*}^{-1} \tag{8.8}$$

where W is row vector of the weights w_i in flow's utility function (8.6). Similarly, we can write derivative of (8.8) as:

$$\lambda'(P^*) = -W(P^{*2})^{-1} \tag{8.9}$$

Also, we can write the utility function (8.6) and its derivative in vectorized notation as follows:

$$U(\lambda) = W log(\hat{\lambda}) \tag{8.10}$$

$$U'(\lambda) = W\hat{\lambda}^{-1} \tag{8.11}$$

The revenue maximization of (8.4) can be re-written as follows:

$$\max_P R = \lambda A P$$

subject to

$$\lambda A \ \leq \ C^T. \tag{8.12}$$

So, we write the Lagrangian as follows:

$$L = \lambda A P + (C^T - \lambda A)\gamma \tag{8.13}$$

where γ is column vector of the Lagrange multipliers for the link capacity constrain.

By plugging (8.8) and (8.9) in appropriate places, the optimality conditions for (8.13) can be written as:

$$L_\gamma : C^T - W{P^*}^{-1}A = 0 \tag{8.14}$$

$$L_{P^*} : -W(P^{*2})^{-1}P^*e + W{P^*}^{-1}e - W(P^{*2})^{-1}A\gamma = 0 \tag{8.15}$$

By solving 8.15 for P^*, we obtain:

$$-{P^*}^{-1}P^*e + Ie - {P^*}^{-1}A\gamma = 0 \tag{8.16}$$

$$-{P^*}^{-1}A\gamma = 0 \tag{8.17}$$

$$P^* = 0 \tag{8.18}$$

Now, solve (8.14) for P^*:

$$C^T - W{P^*}^{-1}A = 0 \tag{8.19}$$

$$C^T A^{-1} P^* = W \tag{8.20}$$

$$P^* = A(C^T)^{-1}W \tag{8.21}$$

Apparently, the optimization problem has two solutions as shown in (8.18) and (8.21). Since (8.18) violates the condition $P > 0$, we accept the solution in (8.21).

We finally derive P by using (8.7):

$$AP = P^*e = A(C^T)^{-1}We \tag{8.22}$$

$$P = (C^T)^{-1}We \tag{8.23}$$

Since $P^* = (P^*)^T$, we can derive another solution:

$$AP = P^*e = W^T C^{-1} A^T e \tag{8.24}$$

$$P = A^{-1}W^T C^{-1} A^T e \tag{8.25}$$

Notice that the result in (8.23) holds for a single-bottleneck (i.e. single-link) network. In non-vectorized notation, this results translates to:

$$p = \frac{\sum_{f \in F} w_f}{c}$$

The result in (8.25) holds for a multi-bottleneck network. This result means that each link's optimal price is dependent on the routes of each flow passing through that link. More specifically, the optimal price for link l is accumulation of budgets of flows passing through link l (i.e. $W^T A^T$ in the formula) divided by total capacity of the links that are traversed by the flows traversing the link l (i.e. $A^{-1}C^{-1}$ in the formula). In non-vectorized notation, price of link l can be written as:

$$p_l = \frac{\sum_{f \in F(l)} w_f}{\sum_{f \in F(l)} \sum_{k \in L(f)} c_k}$$

8.4 Elasticity

The term *elastic* was first introduced to the networking research community by Shenker [69]. Shenker called applications that adjust their sending rates according to the available bandwidth as “elastic applications”, and the traffic generated by such applications as “elastic traffic”. An example of such traffic is the well-known TCP traffic, which is adjusted according to the congestion indications representing decrease in the available bandwidth. Shenker, further, called applications that do not change their sending rates according to the available bandwidth as “inelastic”. So, this interpretation of *elasticity* is the same as *adaptiveness*, i.e. an application is elastic if it adapts its rate according to the network conditions, it is inelastic if it does not.

The concept of elasticity originates from the theory of economics. In economics, demand elasticity according to price[15] is defined as *percent change in demand in response to a percent change in price* [75]. In other words, demand elasticity is the responsiveness of the demand to price changes. A formal definition of demand elasticity can be written as [75]:

$$\varepsilon = \frac{\triangle X(p)/X(p)}{\triangle p/p} \tag{8.26}$$

where p is price, $\triangle p$ is the change in the price, $X(p)$ is user's demand function, and $\triangle X(p)$ is the change in user's demand. (8.26) can be re-written as:

$$\varepsilon = \frac{p}{X(p)} \frac{dX(p)}{dp} \tag{8.27}$$

[15]Demand elasticity can be defined according to several things other than price (e.g. time of service, delay of service). In the rest of the text, we will refer to demand elasticity to price when we say demand elasticity.

Given ε, the characteristic L_ε of user demand is made according to the following functional definition [75]:

$$L_\varepsilon = \begin{cases} \textit{elastic}, & |\varepsilon| > 1 \\ \textit{unit elastic}, & |\varepsilon| = 1 \\ \textit{inelastic}, & |\varepsilon| < 1 \end{cases}$$

So, Shenker's interpretation of elasticity for user utility is actually different from the real meaning of elasticity in economics. Note that Shenker defined elasticity of user utility (or application utility) according to bandwidth, let's call it ϵ. Let $u(x)$ be user's utility if he is given x amount of bandwidth. Then, following the argument in (8.27), we can write ϵ as:

$$\epsilon = \frac{x}{u(x)} \frac{du(x)}{dx} \tag{8.28}$$

According to Shenker's interpretation, the functional definition for L_ϵ (i.e. characteristic of user's utility according to bandwidth) will be as follows:

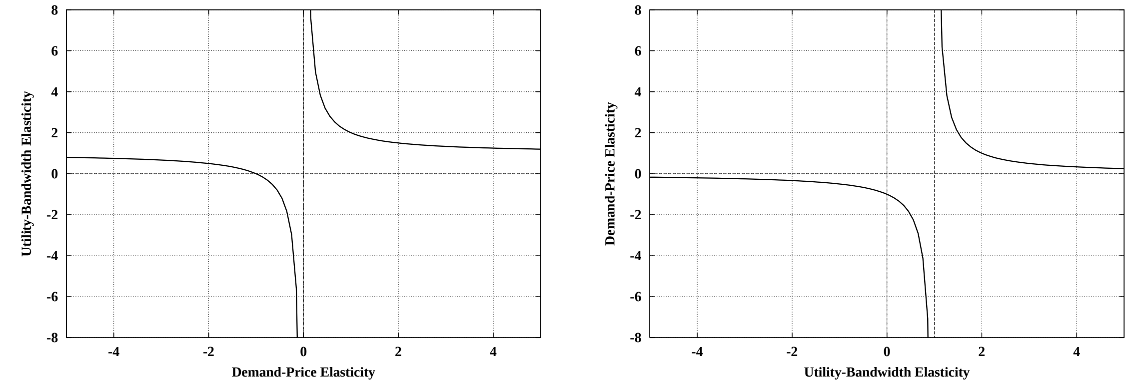
$$L_\epsilon = \begin{cases} \textit{inelastic}, & \epsilon = 0 \\ \textit{elastic}, & \epsilon \neq 0 \text{ \& user utility is concave} \\ \text{not defined}, & \epsilon \neq 0 \text{ \& user utility is convex} \end{cases}$$

Obviously, L_ϵ is a lot different than L_ε. Basically, L_ε interprets elasticity as *responsiveness* while L_ϵ does it as *adaptiveness.*

We can construct the relationship between ϵ and ε, given that the user solves the well-known maximization problem:

$$\max_x \{u(x) - xp\}$$

The solution to the above problem is $u'(x) = p$. So, given a price p, the user selects his demand such that his marginal utility equals to p. Based on that relationship between the utility function $u(x)$ and the demand function $X(p)$, we can construct the relationship between the demand-price elasticity ε and the utility-bandwidth ϵ elasticity. In the next sub-sections we will formulate the relationship between these elasticities.



(a) Utility-bandwidth elasticity ϵ. (b) Demand-price elasticity ε.

Figure 8.1: Utility-bandwidth elasticity ϵ and demand-price elasticity ε with respect to each other.

8.4.1 Utility-Bandwidth Elasticity ϵ

Let $X(p) = Ap^{\varepsilon}$ where $\varepsilon \neq 0$ and $\varepsilon \neq -1$. Then,

$$p = u'(x) = A^{-1/\varepsilon} x^{1/\varepsilon}$$

$$u(x) = A^{-1/\varepsilon} \left(\frac{1}{\varepsilon} + 1\right) x^{1/\varepsilon + 1}$$

So,

$$\epsilon = \frac{1}{\varepsilon} + 1, \;\; \varepsilon \neq 0 \,\&\, \varepsilon \neq -1$$

Figure 8.1-a plots ϵ with respect to ε.

8.4.2 Demand-Price Elasticity ε

Let $u(x) = Bx^{\epsilon}$ where $\epsilon \neq 1$. Then,

$$u'(x) = p = A\epsilon x^{\epsilon - 1}$$

$$X(p) = \left(\frac{p}{A\epsilon}\right)^{\frac{1}{\epsilon - 1}}$$

So,

$$\varepsilon = \frac{1}{\epsilon - 1}, \;\; \epsilon \neq 1$$

Figure 8.1-b plots ε with respect to ϵ.

8.5 Optimal Prices: Non-Logarithmic Utility Functions

In Section 8.3, we derived optimal prices for the revenue maximization problem $NETWORK(\lambda(p^f))$. In that derivation users demand-price elasticity ε was -1 (see (8.8)), which means users had *unit elastic* demands. Now, we re-perform the derivation by assuming that users have a utility-bandwidth elasticity of ϵ, where users' demand-price elasticity is $\varepsilon = 1/(\epsilon - 1)$ based on the study in the previous section. Also, *note that* $0 < \epsilon < 1$ *must be satisfied in order to make sure concavity of the utility function.*

First, let B be row vector of the weights that are different for each flow's utility function, and $\hat{B}$ be an $(n \times n)$ matrix in which the element $\hat{B}_{jj}$ is the weight of flow j and all the other elements are zero.

We use a generic utility function. The function and its derivative is as follows:

$$U(\lambda) = B\hat{\lambda}^{\epsilon} \tag{8.29}$$

$$U'(\lambda) = B\epsilon\hat{\lambda}^{\epsilon-1} \tag{8.30}$$

According to the relationship between ϵ and ε described in Section 8.4.1, we can write the demand function and its derivative as follows:

$$\lambda(P^*) = \epsilon^{-\varepsilon} e^T \hat{B}^{-\varepsilon} P^{*\varepsilon} \tag{8.31}$$

Similarly, we can write derivative of (8.31) as:

$$\lambda'(P^*) = \epsilon^{-\varepsilon} \varepsilon e^T \hat{B}^{-\varepsilon} P^{*\varepsilon-1} \tag{8.32}$$

For the revenue maximization problem, we again solve the Lagrangian in (8.13) but for the new demand function of (8.31). By plugging (8.31) and (8.32) in appropriate places, the optimality conditions for (8.13) can be written as:

$$L_\gamma : C^T - \epsilon^{-\varepsilon} e^T \hat{B}^{-\varepsilon} P^{*\varepsilon} A = 0 \tag{8.33}$$

$$L_{P^*} : \epsilon^{-\varepsilon}\varepsilon e^T \hat{B}^{-\varepsilon} P^{*\varepsilon-1}(P^*e - A\gamma) + \epsilon^{-\varepsilon} e^T \hat{B}^{-\varepsilon} P^{*\varepsilon} e = 0 \tag{8.34}$$

By solving (8.34) for P^*, we obtain:

$$\varepsilon e^T \hat{B}^{-\varepsilon} P^{*\varepsilon-1}(P^*e - A\gamma) + e^T \hat{B}^{-\varepsilon} P^{*\varepsilon} e = 0 \tag{8.35}$$

$$\varepsilon P^{*\varepsilon-1}(P^*e - A\gamma) + P^{*\varepsilon} e = 0 \tag{8.36}$$

$$\varepsilon P^{*-1}(P^*e - A\gamma) + Ie = 0 \tag{8.37}$$

$$\varepsilon Ie - \varepsilon P^{*-1}A\gamma + Ie = 0 \tag{8.38}$$

$$(\varepsilon + 1)Ie = \varepsilon P^{*-1}A\gamma \tag{8.39}$$

$$P^*e = \frac{\varepsilon}{\varepsilon + 1}A\gamma \tag{8.40}$$

$$P^* = \frac{1}{\epsilon}A\gamma e^{-1} \tag{8.41}$$

Now, apply (8.41) into (8.33) and solve for γ:

$$C^T = \epsilon^{-\varepsilon} e^T \hat{B}^{-\varepsilon} \left(\frac{1}{\epsilon}A\gamma e^{-1}\right)^{\varepsilon} A \tag{8.42}$$

$$\epsilon^{\varepsilon} \hat{B}^{\varepsilon} (e^T)^{-1} C^T A^{-1} = \left(\frac{1}{\epsilon}A\gamma e^{-1}\right)^{\varepsilon} \tag{8.43}$$

$$\frac{1}{\epsilon}A\gamma e^{-1} = \epsilon A^{-1/\varepsilon}(C^T)^{1/\varepsilon}(e^T)^{-1/\varepsilon}\hat{B} \tag{8.44}$$

Substitute (8.44) into (8.41) and we obtain P^*:

$$P^* = \epsilon A^{-1/\varepsilon}(C^T)^{1/\varepsilon}(e^T)^{-1/\varepsilon}\hat{B} \tag{8.45}$$

From (8.45) we obtain P:

$$AP = P^*e = \epsilon A^{-1/\varepsilon}(C^T)^{1/\varepsilon}(e^T)^{-1/\varepsilon}\hat{B}e \tag{8.46}$$

$$P = \epsilon A^{-1} A^{-1/\varepsilon}(C^T)^{1/\varepsilon}(e^T)^{-1/\varepsilon}\hat{B}e \tag{8.47}$$

$$P = \epsilon A^{-1} A^{|1/\varepsilon|} \left((C^T)^{|1/\varepsilon|}\right)^{-1} (e^T)^{|1/\varepsilon|}\hat{B}e \tag{8.48}$$

$$P = \epsilon A^{-1} A^{|1/\varepsilon|} \left((C^T)^{|1/\varepsilon|}\right)^{-1} (e^T)^{|1/\varepsilon|} \left(\hat{B}^{|\varepsilon|}\right)^{|1/\varepsilon|} e \tag{8.49}$$

The result in (8.48) implies the same thing as in the case of logarithmic utility functions except that the link capacities must be taken more conservatively depending on the elasticity (ϵ or ε by choice) of flows. Observe that as flows demand-price elasticity ε gets higher, the capacity must be taken more conservatively based on the formula $(C^T)^{|1/\varepsilon|}$. Also observe that as flows utility-bandwidth elasticity ϵ gets higher, the capacity must be taken more conservatively based on the formula $(C^T)^{|1/\varepsilon|} = (C^T)^{|\epsilon-1|}$.

Based on (8.49) we can write the optimal price formulas for single-bottleneck and multi-bottleneck cases respectively as follows in non-vectorized form:

$$p = \epsilon \left(\frac{\sum_{f \in F} w_f^{|\varepsilon|}}{c} \right)^{|1/\varepsilon|}$$

$$p_l = \epsilon \left(\frac{\sum_{f \in F(l)} w_f^{|\varepsilon|}}{\sum_{f \in F(l)} \sum_{k \in L(f)} c_k} \right)^{|1/\varepsilon|}$$

8.6 Summary

In this chapter, we analyzed the optimality of EEP pricing scheme which we developed within Distributed-DCC framework. We proved that EEP is optimal in the case of logarithmic user utility functions by solving the total user utility maximization problem for the system.

Then, we revised the term *elasticity* in the area of networking, and defined utility-bandwidth elasticity. We also determined relationship between utility-bandwidth elasticity and the well-known demand-price elasticity. Based on the investigation of elasticity, we then defined a non-logarithmic form of utility functions which include elasticity as a parameter. Considering the newly defined non-logarithmic utility functions, we re-solved the optimization problem of total user utility maximization, and showed that EEP is still optimal with minor changes to its price calculation.

We also showed the major differences between Distributed-DCC framework and well-known optimal pricing frameworks (e.g. Low's [47], Kelly et al.'s [42]) in terms of trade-off between optimality and deployability.

CHAPTER 9
OPTIMIZATION ANALYSIS OF DISTRIBUTED-DCC

9.1 Introduction

Previously in Chapter 6, we presented the Distributed-DCC framework. In this chapter, we focus on analyzing Distributed-DCC in terms of various aspects.

In Chapter 6, we experimentally showed that Distributed-DCC has ability to achieve multiple fairness types in rate allocation under certain conditions. Here in this chapter, we will provide the theoretical reasoning behind this ability. Also, we will look at how much effect each key parameter has on Distributed-DCC's performance.

The chapter is organized as follows: In Section 9.2, we study theoretical reasoning behind Distributed-DCC's fairness capabilities. In Section 9.3, by several simulations, we investigate Distributed-DCC's sensitivity to several parameters, such as contract length, LPS interval, budget ratio of flows. Finally, we summarize the chapter in Section 9.4.

9.2 Fairness

In Chapter 6, we provided experimental proof that a provider in Distributed-DCC framework can achieve various types of fairness by tuning the parameter α. In this section, we will explain the theory behind this fairness ability in Distributed-DCC. Note that the theoretical analysis in this section will be based on the assumption of logarithmic utility functions and the provider employs the EEP pricing scheme. The analysis can be extended to all concave utility functions given that the provider employs an appropriate pricing scheme.

Let $u_i(\lambda_i) = w_i log(\lambda_i)$ be flow i's utility when it is given a bandwidth of λ_i. Let a_i be the actual effective capacity for flow i on its route. Let r_i be the real number of bottlenecks flow i passes through. Also, let $\rho_i(t)$ be the actual probability of flow i being congested[16]. According to the notation of Distributed-DCC, we can write

[16] According to the Distributed-DCC algorithms, notice that $\rho_i(t) \to 1$ when $\beta \to 1$ or $\hat{k} \to \infty$.

the followings for flow i:

$$\begin{aligned}
\hat{b}_i(t) &\rightarrow w_i && (9.1)\\
\hat{c}_i(t) &\rightarrow \beta a_i && (9.2)\\
\hat{r}_i(t) &\rightarrow r_i && (9.3)\\
b_i(t) &= \frac{\hat{b}_i(t)}{1+\alpha(r_i(t)-1)} && (9.4)\\
c_i(t) &= \rho_i(t)\frac{b_i(t)}{B_c(t)}C_c(t) + [1-\rho_i(t)]\hat{c}_i(t) && (9.5)\\
C_c(t) &= \sum_{i\in F_c} \hat{c}_i(t) && (9.6)\\
B_c(t) &= \sum_{i\in F_c} b_i(t) && (9.7)
\end{aligned}$$

According to algorithms in Distributed-DCC, the network will calculate flow i's actual price as follows:

$$p_i(t+1) = \frac{\hat{b}_i(t)}{c_i(t)} \tag{9.8}$$

Notice that this above price formulation is not optimal because it may not be an exact solution to problem (8.4). However, it is the formulation based on edge-to-edge capabilities of Distributed-DCC. For further understanding of this formulation reader should refer to the Distributed-DCC paper.

Given the price formulated in (9.8), flow i solves the problem (8.3). This leads to:

$$\lambda_i(t+1) = \rho_i(t)\frac{\hat{b}_i(t)}{1+\alpha(r_i(t)-1)}\frac{C_c(t)}{B_c(t)} + [1-\rho_i(t)]\hat{c}_i(t) \tag{9.9}$$

On steady-state, (9.1)-(9.7) hold. So, we can re-write (9.9) accordingly:

$$\lambda_i(t+1) = \rho_i(t)\frac{w_i}{1+\alpha(r_i-1)}\frac{C_c(t)}{B_c(t)} + [1-\rho_i(t)]\beta a_i \tag{9.10}$$

Notice that Distributed-DCC will operate in a state very close to the case where $\rho_i(t) \rightarrow 1$. This is because β is normally selected very close to 1. An alternative way to achieve $\rho_i(t) \rightarrow 1$ is to increase the parameter $\hat{k}$, i.e. $\rho_i(t) \rightarrow 1$ as $\hat{k} \rightarrow \infty$. So, we continue by analyzing the case when $\rho_i(t) \rightarrow 1$.

As $\rho_i(t) \to 1$, the followings hold:

$$C_c(t) \to \sum_i \beta a_i$$

$$B_c(t) \to \sum_i \frac{w_i}{1 + \alpha(r_i - 1)}$$

In other words, the former one is *effective total capacity* and the latter one is *effective total budget*.

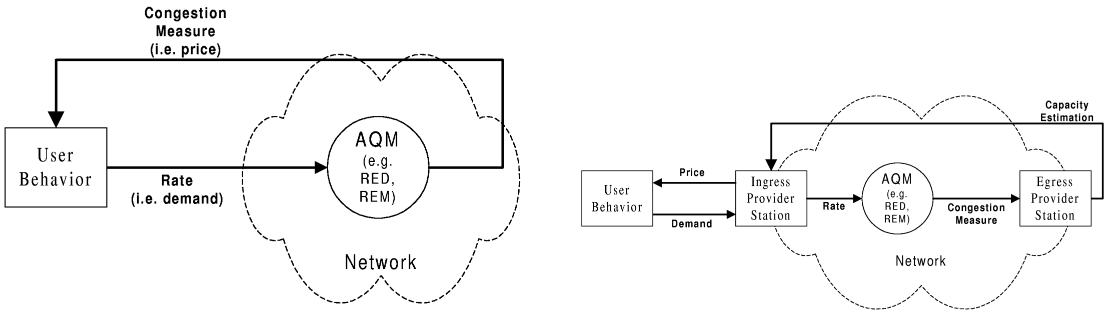
Then,

$$\lambda_i \to \frac{w_i}{1 + \alpha(r_i - 1)} \frac{\sum_j \beta a_j}{\sum_j \frac{w_j}{1+\alpha(r_j - 1)}} \tag{9.11}$$

1. When $\alpha = 0$:
   (a) If $w_i = 1$, $\lambda_i \to \frac{\sum_i \beta a_i}{N}$ where N is the number of edge-to-edge flows. This means all flows get equal share, i.e. *max-min fairness*.
   (b) If $w_i \neq 1$, $\lambda_i \to \frac{w_i}{\sum_i w_i} \sum_i \beta a_i$. This means each flow gets a share in proportion to its budget w_i, i.e. *weighted max-min fairness*.
2. When $\alpha = 1$: Each flows gets a non-negative bandwidth based on how many bottlenecks (i.e. r_i) it passes through. Rate allocation to flow i will be:
   (a) If $w_i = 1$, $\lambda_i \to \frac{1}{r_i} \frac{\sum_j \beta a_j}{\sum_j w_j / r_j}$. This means each flow gets penalized according to the number of bottlenecks it passes through r_i, i.e. *proportional fairness*.
   (b) If $w_i \neq 1$, $\lambda_i \to \frac{w_i}{r_i} \frac{\sum_j \beta a_j}{\sum_j w_j / r_j}$. This means each flow gets a share in proportion to its budget w_i and also gets penalized according to the number of bottlenecks it passes through r_i, i.e. *weighted proportional fairness*.

   Also, by following the well-known arguments[17], it is easy to show that the bandwidth allocation vector λ will be *weighted proportional fair* as long as it solves the problem (8.2).
3. When $0 < \alpha < 1$: Following the discussion in the previous item, the rate allocation will be between max-min fair and proportional fair.

[17] Convexity of the feasible region for λ and concavity of utility functions.



(a) Low's pricing framework. (b) Distributed-DCC framework.

Figure 9.1: Big-picture comparison of Distributed-DCC with Low's pricing framework: Distriubted-DCC is able to provide a range of fairness types by taking advantage of edge-placed provider stations, while Low's framework is only able to provide proportional fairness.

4. When $\alpha > 1$: Again, following the arguments in the previous item, this time each flow will be penalized more than the number of bottlenecks it passes through. For this case, there is no defined fairness term in the literature.

Notice that all the above abilities of Distributed-DCC are achieved because of the fact that provider has a lot of administrative privileges. The fairness abilities of Distributed-DCC must not be compared with abilities of Kelly et al.'s or Low's schemes. In their frameworks [42, 47, 48, 46], it is assumed that the provider does not intervene the rate allocation and the rate allocation is totally left to end-user behavior (which is based on user utilities) and Active Queue Management (AQM) schemes being used in the network queues (which basically communicates congestion measures to the end-user in some form such as ECN, packet drop). Figures 9.1-a and 9.1-b show the big-pictures of our Distributed-DCC framework and Low's framework respectively. Observe that provider is involved in defining the rate allocation in Distributed-DCC while rate allocation is totally left to user's behavior and the queue behavior.

Also, note that α parameter in Distributed-DCC is different than the α parameter in Mo and Walrand's definition of (p, α) fairness [58]. Mo and Walrand defined, similar to Low's work without consideration of the service provider in the scenario, a user utility function such that rate allocation is (p, α) fair. They showed that fairness of the rate allocation is proportional when $\alpha = 1$, and is max-min fair when $\alpha \to \infty$. Notice that α in Mo and Walrand's work is a parameter for user utility function, whereas it is a parameter for provider in Distributed-DCC.

9.3 Sensitivity

In Distributed-DCC, there are several parameters that effect system performance. For example, in Chapter 6 we showed that the fairness coefficient α affects the rate allocation significantly. In this section, we will investigate four parameters: contract length T, observation interval O, LPS interval L, and the parameter $\hat{k}$. In fact, the last one $\hat{k}$ is a parameter of the ETICA capacity allocation algorithm, but we will investigate it since it has crucial role in performance of the whole system. Also, we only investigate the PFCC architecture, since performance of POCC architecture depend on the underlying edge-to-edge congestion control scheme.

To see sensitivity of system performance to the four parameters, we run several simulation experiments on the same single-bottleneck topology that we experimented in Chapter 6, except that there are only two flows in the network. Experiment parameters are again the same as the ones that we used in the single-bottleneck experiments of Chapter 6. The only difference is that we vary each parameter T, O, L, $\hat{k}$ one at a time. So, the initial experimental set-up is that $T = 100RTT$, $O = 20RTT$, and $L = 4RTT$ where $RTT = 0.04sec$.

We look at three metrics in system performance: average bottleneck queue length, average utilization, and service differentiation. To measure the service differentiation ability of the system, we set ratio of budgets of the two flows and observe if the system really allocates capacity in proportion to flows' budgets. Let R be the ratio of the two flows' budgets set before the simulation. We vary R from 1 to 100. Also, to see the effect of $\hat{k}$, we vary $\hat{k}$ from 5 to 175.

So, for each $(\hat{k}, R)$ pair, we vary each of the three parameters (i.e. T, O and L) one at a time starting from the initial set-up $T = 100RTT$, $O = 20RTT$, and $L = 4RTT$. The following sub-sections present the results of these simulation experiments and observations made from them.

9.3.1 Effect of Contract Length

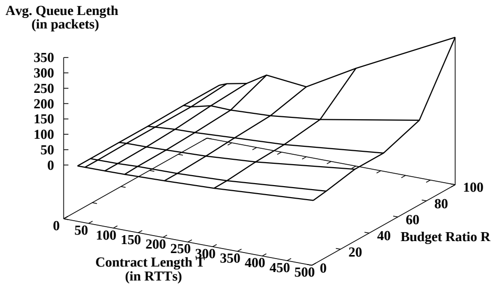
For each $(\hat{k}, R)$, we vary the contract length T from $25RTT$ to $500RTT$. Figures 9.2-a to 9.2-h show behavior of average queue length for different values of $\hat{k}$. Figure 9.2-a, for instance, plots average queue length as T and R vary when $\hat{k} = 5$.

In all the graphs from Figure 9.2-a to Figure 9.2-h, we observe that average queue length increases steadily as T increases. This is simply because the pricing framework looses control as pricing interval increases, remembering the discussion in Chapter 5. Also, observe that R effects average queue length negatively. This is because of the unpredictability caused by ETICA's frequent state changes (see Section 6.2.4.3 for details). Note that state changes happen when $\hat{k}$ is small, i.e. the flow goes back to "non-congested" state after staying at "congested" state for short amount of time. But, when $\hat{k}$ is large, the flow stays in the "congested" state longer. We can observe this by following the graphs from Figure 9.2-a to Figure 9.2-h as $\hat{k}$ increases. Observe that effect of R gets smaller as $\hat{k}$ increases and vanishes at $\hat{k} = 125$.

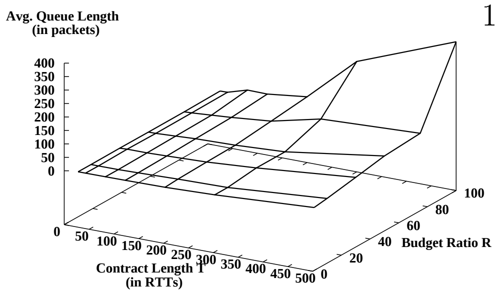
Figures 9.3-a to 9.3-d plot the utilization of bottleneck link during the simulations for different values of $\hat{k}$. We observe that neither changes in T nor changes in R effect the bottleneck utilization.

Finally, Figures 9.4-a to 9.4-f show service differentiation ability of the system for different values of the budget ratio R. Figure 9.4-b, for example, plots observed ratios of the two flows' rates for different values of $\hat{k}$ as the contract length T increases. It also plots the initially set ratio of flow budgets, shown as "optimal".

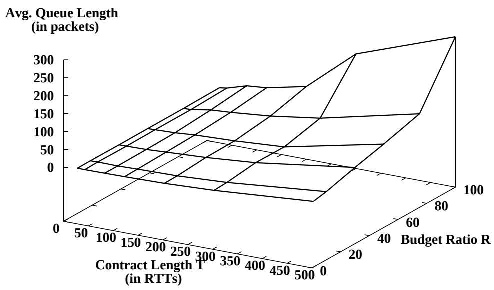
We can observe that as $\hat{k}$ gets larger the observed ratio gets closer to the optimal ratio. This is mainly because of the topology and dynamics of the ETICA algorithm. As we discussed earlier, for single-bottleneck topology larger $\hat{k}$ values are better, which will allow less freedom to individual flows in sharing the bottleneck capacity. A "free" (which corresponds to "non-congested" state in ETICA) flow tends to get an equal share of the bottleneck capacity. In "congested" state, however, Distributed-DCC allocates capacity proportional to flows' budgets. So, a flow's rate goes back an forth between the *equal share* and the *proportional share*.



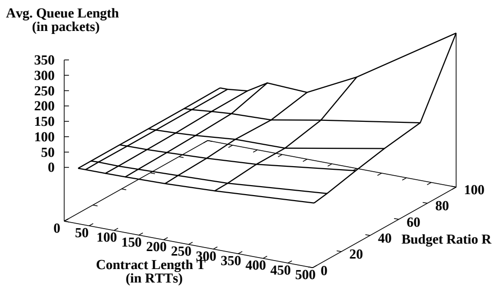
(a) $\hat{k} = 5$



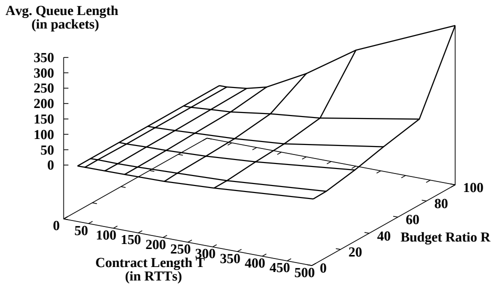
(b) $\hat{k} = 15$



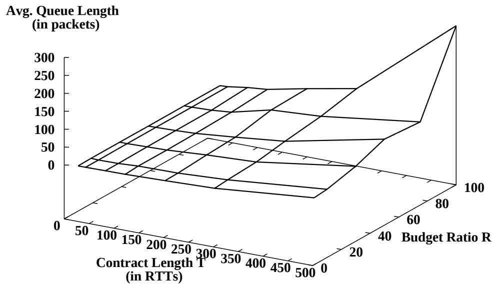
(c) $\hat{k} = 45$



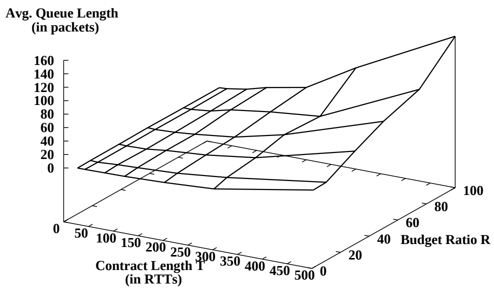
(d) $\hat{k} = 60$



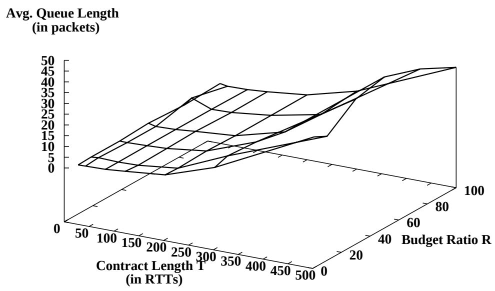
(e) $\hat{k} = 75$



(f) $\hat{k} = 100$

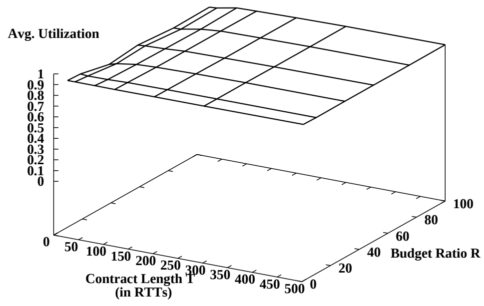


(g) $\hat{k} = 125$

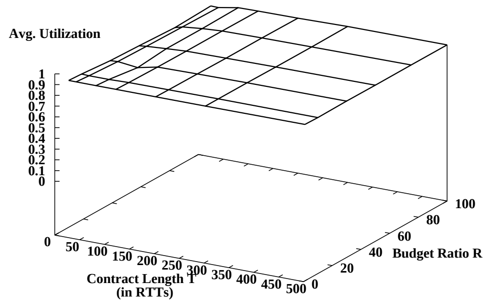


(h) $\hat{k} = 150$

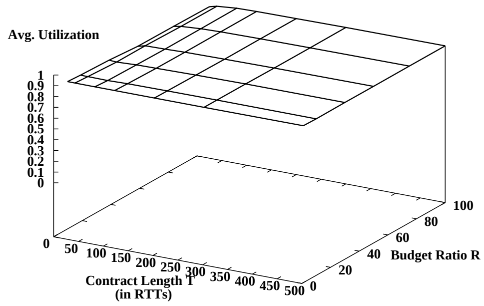
Figure 9.2: Effect of contract length on bottleneck queue length: Increasing T or increasing budget ratio R of flows causes larger queues.



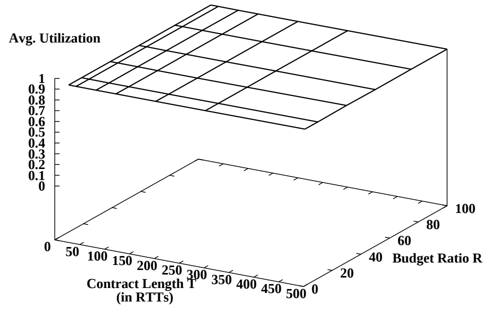
(a) $\hat{k} = 15$



(b) $\hat{k} = 45$



(c) $\hat{k} = 75$



(d) $\hat{k} = 150$

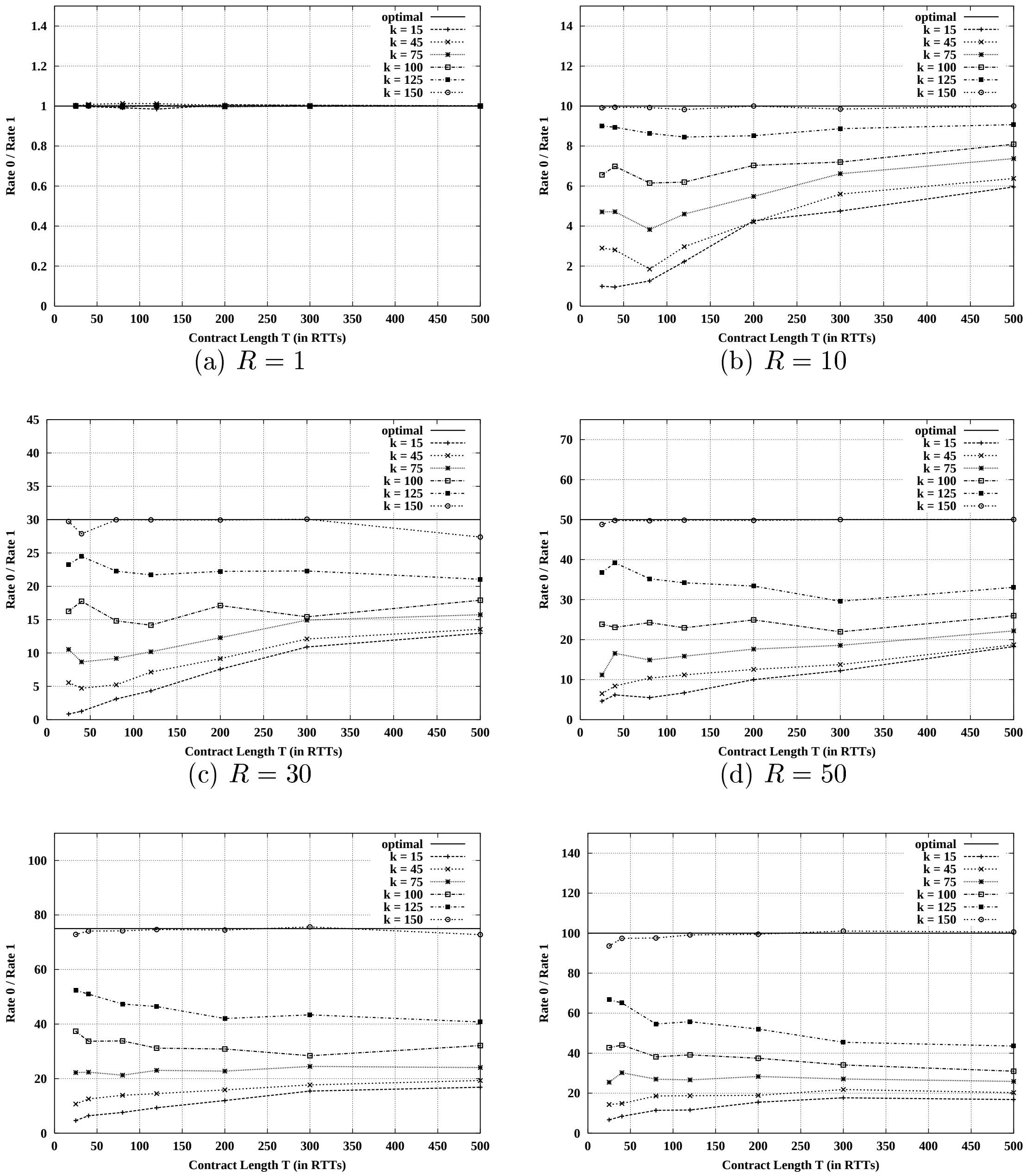
Figure 9.3: Effect of contract length on bottleneck utilization: Neither T nor R has effect on bottleneck utilization.

As R gets larger we see importance of $\hat{k}$ on the service differentiation ability. Actually, when $R = 1$ all $\hat{k}$ values perform equally, since the equal share and the proportional share are equivalent. However, when R gets larger the difference between $\hat{k}$ curves gets larger too.

Another observation from the Figures 9.4-a to 9.4-f is that service differentiation gets slightly better when T increases. This is because of the averaging effects of larger contracting periods. However, this costs larger queues as we observed in Figure 9.2.

9.3.2 Effect of LPS Interval

For each $(\hat{k}, R)$, we vary the LPS interval L from $5RTT$ to $100RTT$. Note that we vary L only up to a contracting period $T = 100RTT$. Apparently, increasing L to values larger than a contracting period is going to reduce system performance. So, we do not investigate the case $L > T$.



(a) $R = 1$ (b) $R = 10$

(c) $R = 30$ (d) $R = 50$

(e) $R = 75$ (f) $R = 100$

Figure 9.4: Effect of contract length on service differentiation: Increasing T improves service differentiation very slightly.

Figures 9.5-a to 9.5-h show behavior of average queue length for different values of $\hat{k}$. Figure 9.2-a, for instance, plots average queue length as L and R vary when $\hat{k} = 5$. For small values of $\hat{k}$, we observe that average queue length increases as R increases, which is again because of the frequent state changes of flows in ETICA algorithm. However, as $\hat{k}$ increases, the effect of changes in R becomes less important and do not effect the average queue length.

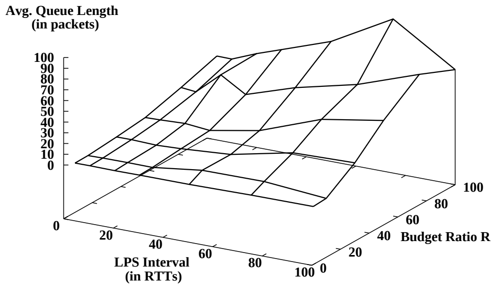
Another interesting observation is that average queue length is high for larger R values regardless of $\hat{k}$, when L is less than the observation interval, i.e. $L < O = 20RTT$. This is because of two things: First, LPS cannot get observations from all stations. This causes temporary inaccuracies in its calculations, such as calculation of estimated capacity. Second, smaller L means that flows's state transitions will occur more frequently since the parameter $\hat{k}$ is defined in terms of Ls, i.e. if $\hat{k} = 10$, then the flow will go back to "non-congested" state after 10 *LPS interval*s. So, this reveals a non-intuitive fact about dynamics of Distributed-DCC, i.e. *LPS interval L should be set to a value in between the observation interval and the contracting period.* In other words, the LPS interval should satisfy the condition $O < L < T$ to obtain better performance in Distributed-DCC.

Overall, we observe that changes in L does not affect average queue length as long as number of state transitions of flows in ETICA is small.

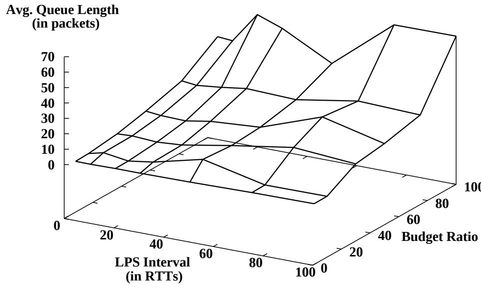
Figures 9.6-a to 9.6-d plot the utilization of bottleneck link during the simulations for different values of $\hat{k}$. We observe that neither changes in L nor changes in R effect the bottleneck utilization.

Finally, Figures 9.7-a to 9.7-f show service differentiation ability of the system for different values of the budget ratio R. Figure 9.7-b, for example, plots observed ratios of the two flows' rates for different values of $\hat{k}$ as the contract length L varies. It also plots the initially set ratio of flow budgets, shown as "optimal".

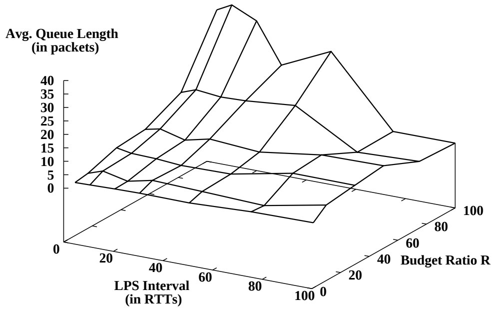
We observe a similar behavior in service differentiation graphs as we did in the average queue length graphs in Figure 9.5. Service differentiation up to the observation interval $O = 20RTT$ significantly worse than the service differentiation in between the observation interval $20RTT$ and the contract length $100RTT$. So again, we observe that L should be larger than the observation interval, and less



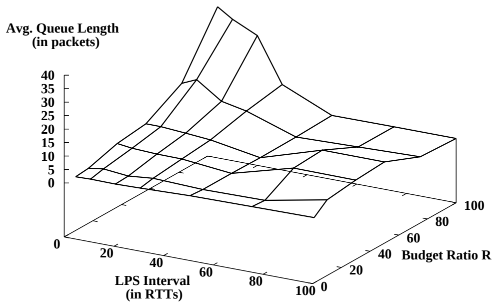
(a) $\hat{k} = 5$



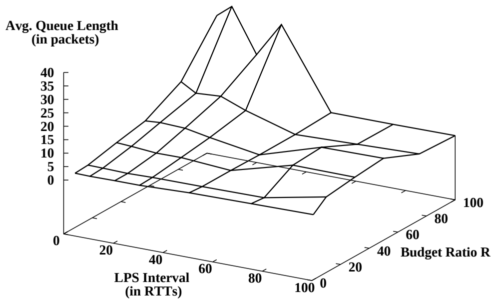
(b) $\hat{k} = 15$



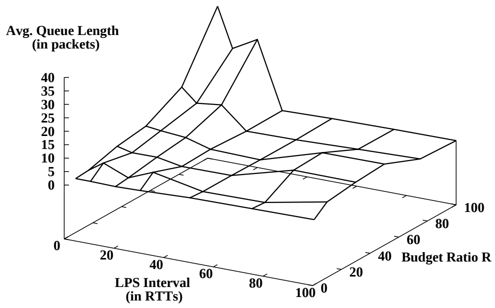
(c) $\hat{k} = 45$



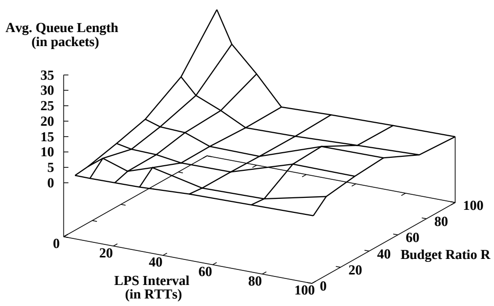
(d) $\hat{k} = 60$



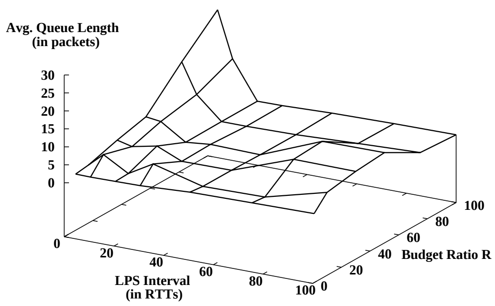
(e) $\hat{k} = 75$



(f) $\hat{k} = 100$

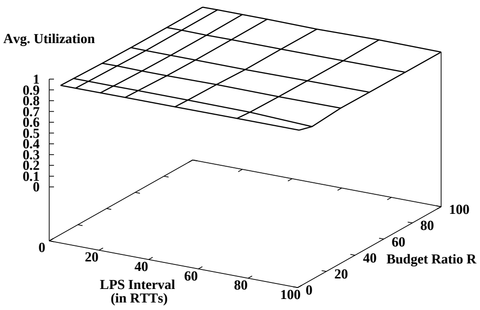


(g) $\hat{k} = 125$

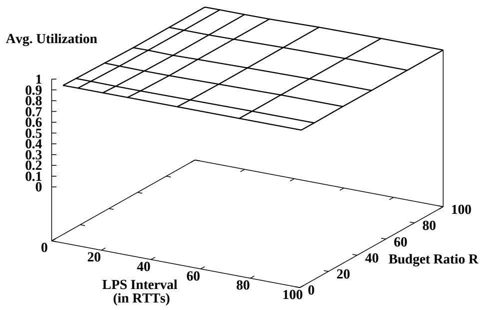


(h) $\hat{k} = 150$

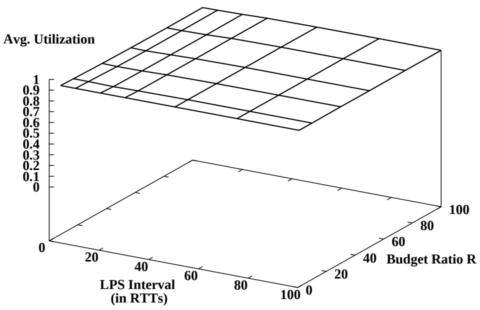
Figure 9.5: Effect of LPS interval on bottleneck queue length: When L is less than observation interval ($O = 20RTT$) budget ratio R effects queue length negatively.



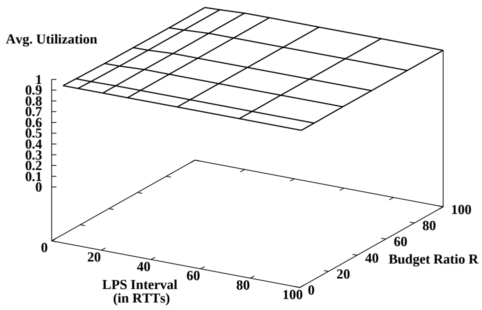
(a) $\hat{k} = 15$



(b) $\hat{k} = 45$



(c) $\hat{k} = 75$



(d) $\hat{k} = 150$

Figure 9.6: Effect of LPS interval on bottleneck utilization: Neither L nor the budget ratio R of flows has effect on utilization.

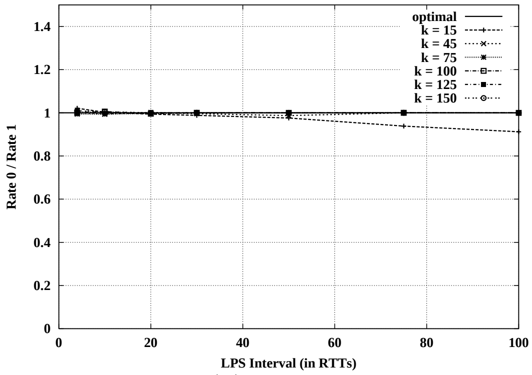
than the contract length.

Also, as we did in Section 9.3.1, we again see that effect of $\hat{k}$ on service differentiation becomes more important as the ratio R increases.

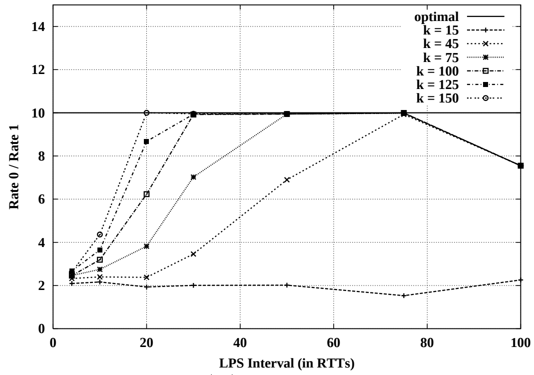
9.3.3 Effect of Observation Interval

For each $(\hat{k}, R)$, we vary the observation interval O from $4RTT$ to $100RTT$. Note that we vary O only up to a contracting period $T = 100RTT$. Apparently, increasing O to values larger than a contracting period is going to reduce system performance. So, we do not investigate the case $O > T$.

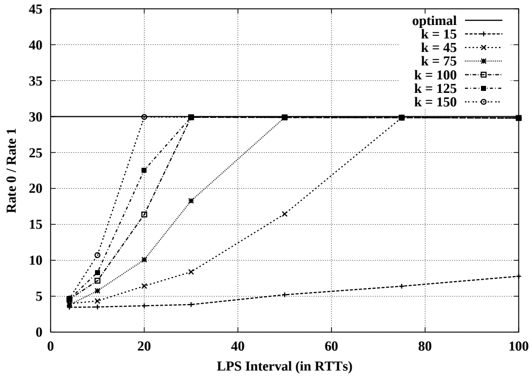
Figures 9.8-a to 9.8-h show behavior of average queue length for different values of $\hat{k}$. Figure 9.8-a, for instance, plots average queue length as O and R vary when $\hat{k} = 5$. Observe that for small $\hat{k}$ values, average queue length increases as R increases. Again, this is because of the large number of state transitions in ETICA



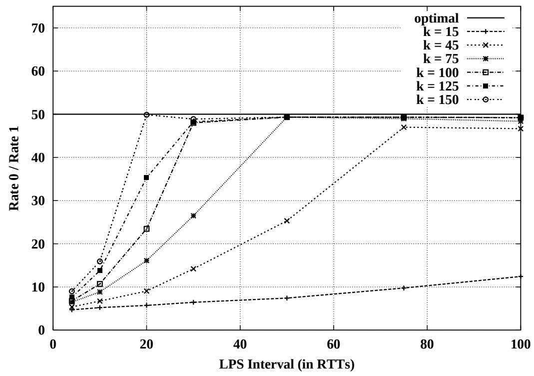
(a) $R = 1$



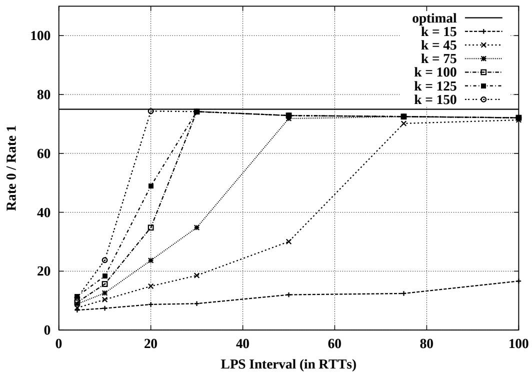
(b) $R = 10$



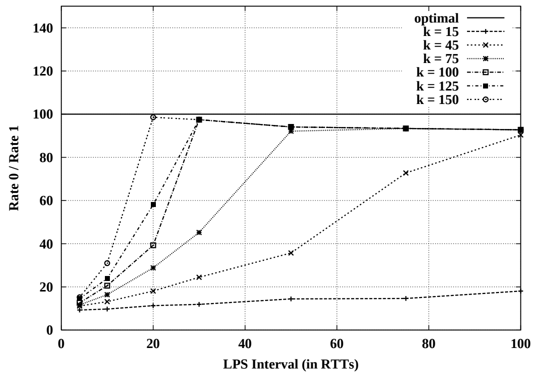
(c) $R = 30$



(d) $R = 50$

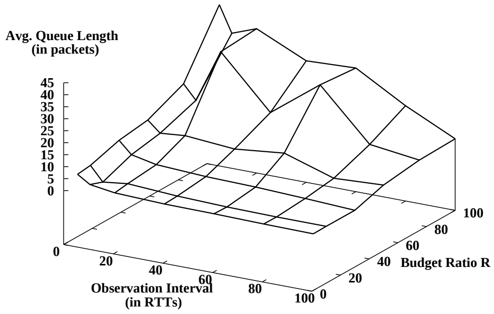


(e) $R = 75$

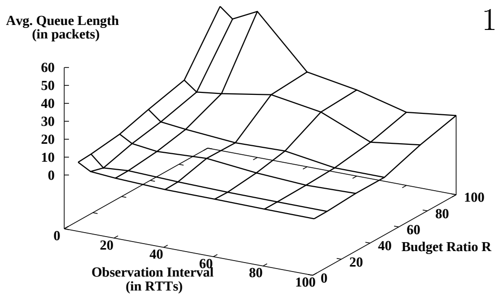


(f) $R = 100$

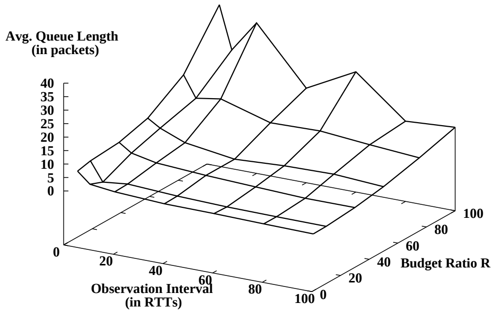
Figure 9.7: Effect of LPS interval on service differentiation: L values larger than the observation interval $O = 20RTT$ performs significantly better in service differentiation.



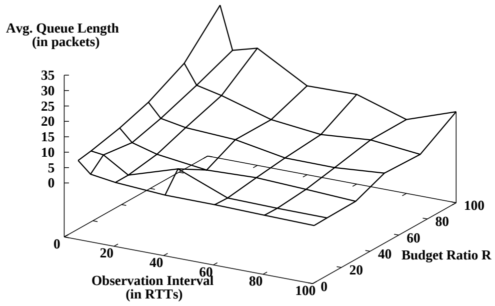
(a) $\hat{k} = 5$



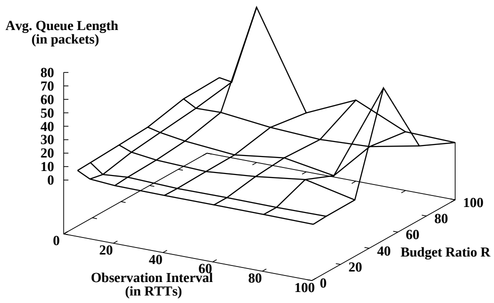
(b) $\hat{k} = 15$



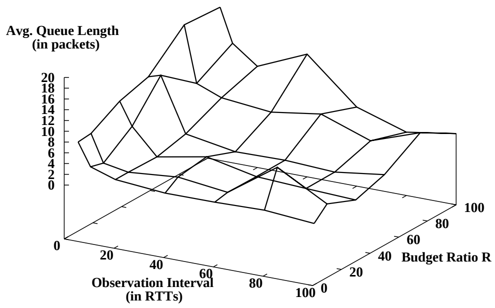
(c) $\hat{k} = 45$



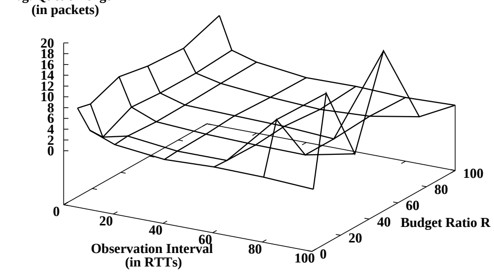
(d) $\hat{k} = 75$



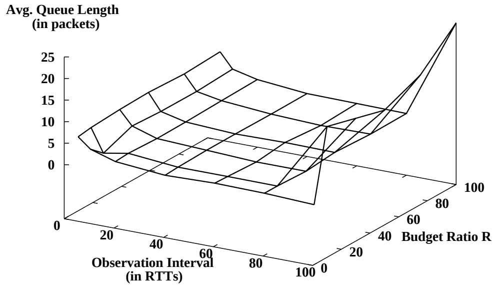
(e) $\hat{k} = 100$



(f) $\hat{k} = 125$

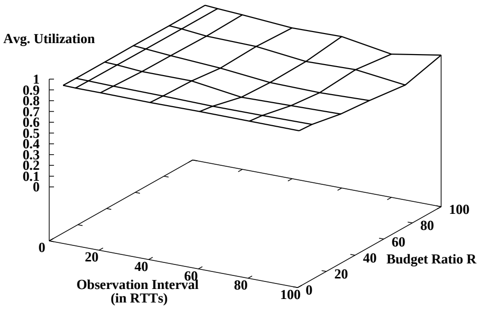


(g) $\hat{k} = 150$

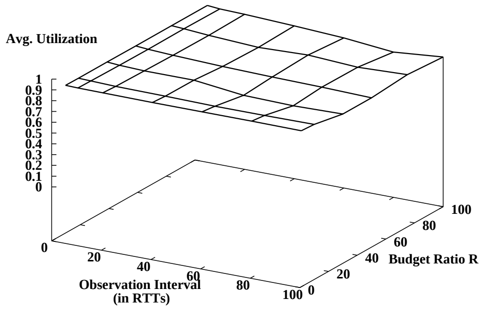


(h) $\hat{k} = 175$

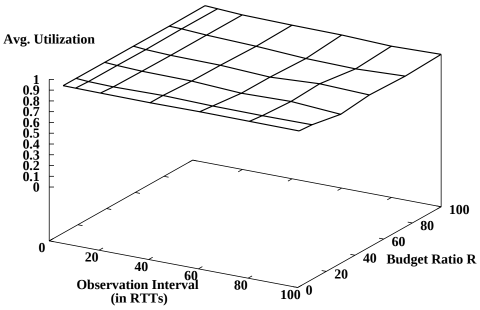
Figure 9.8: Effect of observation interval on bottleneck queue length: Larger O values perform better for small $\hat{k}$, medium O values perform better for large $\hat{k}$.



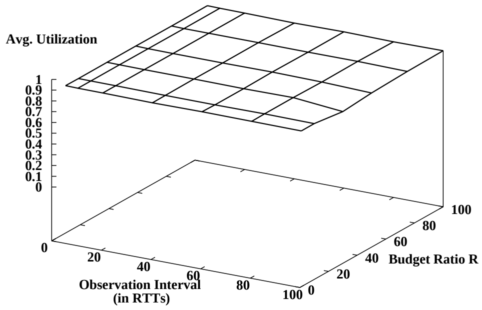
(a) $\hat{k} = 15$



(b) $\hat{k} = 45$



(c) $\hat{k} = 75$



(d) $\hat{k} = 150$

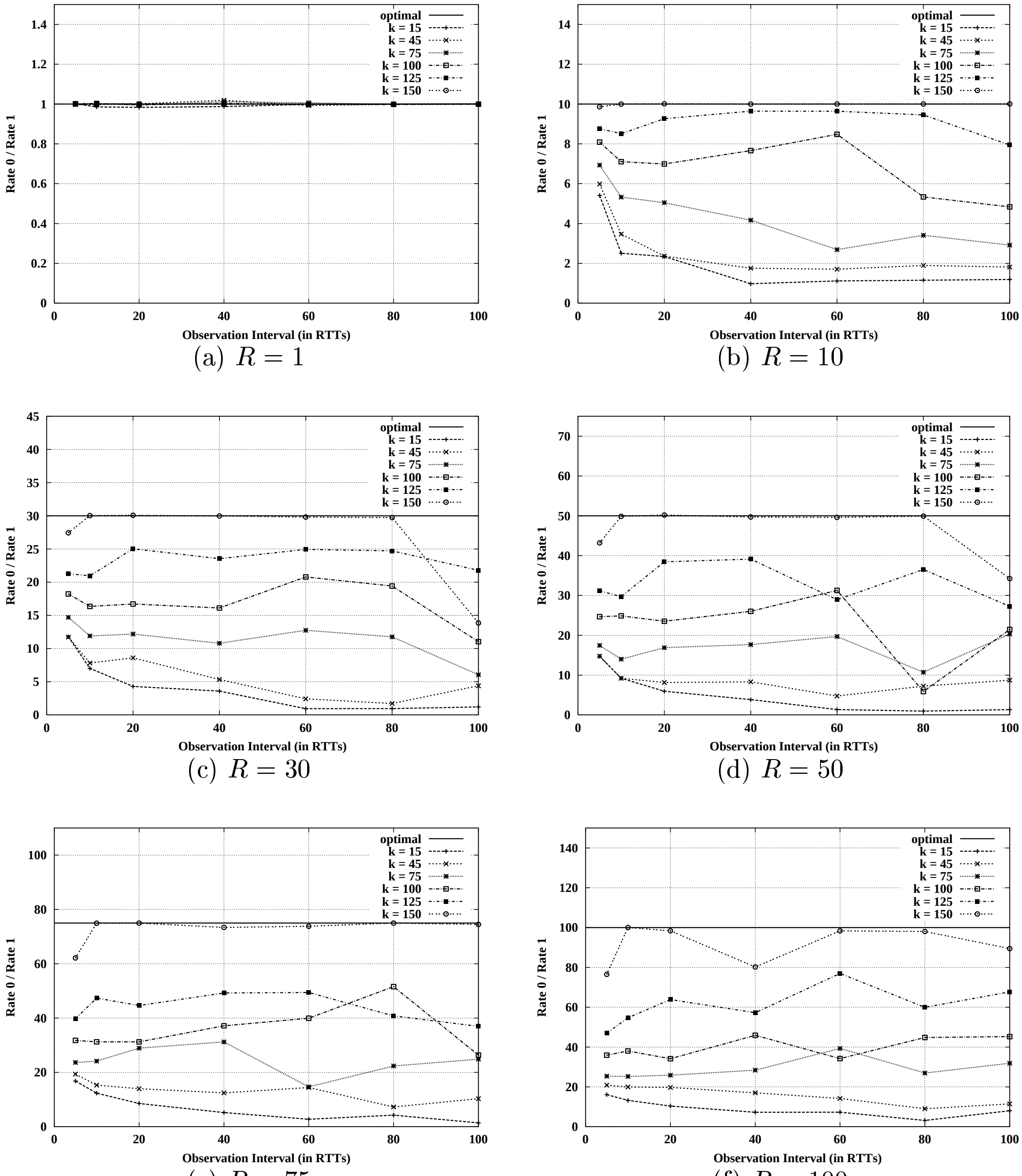
Figure 9.9: Effect of observation interval on bottleneck utilization: Neither O nor the budget ration R of flows has effect on utilization.

algorithm. However, for large values of $\hat{k}$ we do not see any effect of R at all. This is because ETICA causes small number of state transitions for large $\hat{k}$ values in single-bottleneck topologies.

Also, for large $\hat{k}$ values, we observe that average queue length increases sharply as observation interval gets closer to the contract length $100RTT$. This is simply because accuracy of capacity estimation (which is dependent on number of observations made during a contract) deteriorates. We do not see this effect in the graphs for small $\hat{k}$ values, because the effect of state changes is dominant for those cases.

Figures 9.9-a to 9.9-d plot the utilization of bottleneck link during the simulations for different values of $\hat{k}$. We observe that neither changes in O nor changes in R effect the bottleneck utilization.

Figures 9.10-a to 9.10-f show service differentiation ability of Distributed-DCC for varying values of the observation interval O. Figure 9.10-f, for example, plots



(a) $R = 1$ (b) $R = 10$ (c) $R = 30$ (d) $R = 50$ (e) $R = 75$ (f) $R = 100$

Figure 9.10: Effect of observation interval on service differentiation: Changes in O value do not seem to effect service differentiation significantly.

Finally, Figures 9.10-a to 9.10-f show service differentiation ability of the system for different values of the budget ratio R. Figure 9.10-b, for example, plots observed ratios of the two flows' rates for different values of $\hat{k}$ as the contract length O varies when $R = 10$. It also plots the initially set ratio of flow budgets, shown as "optimal".

Other than small deterioration when O gets closer to the contract length $100RTT$, we do not really see any significant effect of the observation interval O on service differentiation. This is mainly because service differentiation of Distributed-DCC is heavily dependent on accuracy of capacity estimation, for which one observation per contract is enough for our simulated system. In our simulated system, capacity estimation uses only the latest observation (see Section 6.2.4.2). If we had used all the observations (e.g. average them), then effect of the observation interval on service differentiation would be more significant.

Also, as we did in Sections 9.3.1 and 9.3.2, we again see that effect of $\hat{k}$ on service differentiation becomes more important as the ratio R increases.

9.4 Summary

Regarding Distributed-DCC in PFCC architecture, we investigated fairness, stability and parameter sensitivity issues. We showed the reasons why Distributed-DCC is able to provide multiple fairness types in rate allocation. By comparison to Low et al.'s [47] framework, we demonstrated that Distributed-DCC takes advantage of provider's administrative privileges in order to achieve various fairness types in rate allocation.

By extensive simulations, we also investigated effects of different parameters on Distributed-DCC's performance. We demonstrated that ETICA's dynamics has dominant effects of Distributed-DCC's performance especially when ratio of flows' budgets R is large. This is because of the fact that ETICA's capacity allocation to a individual flow in "congested" state is based on the ratio of that flow's budget to the total budget of all congested flows. So, when the difference between flows' budgets is large (i.e. R is large), an individual flow's allocated capacity varies a lot between the "congested" and "non-congested" states. This characteristic of ETICA dominates in the system performance when flows make frequent state transitions.

Also, we illustrated that the parameter $\hat{k}$ in ETICA algorithm is very important in terms of system performance, since it mainly defines how frequent the flows will make transitions among the congested and non-congested states. We showed for a single-bottleneck topology that small $\hat{k}$ values effect system performance negatively.

We also investigated the time-scale parameters contract length T, observation interval O, and LPS interval L. We demonstrated effect of these time-scale parameters on three performance metrics: average queue, utilization, and service differentiation ability. We found that the best setting for the three time-scale parameters is: $O < L < T$.

CHAPTER 10
SUMMARY AND FUTURE WORK

10.1 Thesis Summary

One of the main problems of the current Internet is lack of better economic models and tools. Currently, only the traditional hierarchical (i.e. 1-tier, 2-tier, ...) model is employed in the Internet. This thesis is an effort to enable better economic models (e.g. brokers, exchanges) by developing some of the necessary building blocks. Particularly, dynamic pricing tools are fundamental to development of new economic models to the Internet.

The thesis focused on several issues ranging from deployability of dynamic pricing (and particularly congestion pricing) to fairness of rate allocation. First in Chapter 3, we investigated a well-known congestion pricing proposal, Smart Market [50], in terms of deployability. Even though Smart Market is ideal theoretically, we found that it is hard to deploy because of its per-packet granularity, i.e. charging per-packet.

Motivated by granularity level, we proposed a new dynamic pricing framework, Dynamic Capacity Contracting (DCC) in Chapter 4, and developed a congestion pricing scheme, Edge-to-Edge Pricing (EEP), within the framework. Later in Chapter 8, we analyzed optimality of EEP by considering parameters (such as elasticity) related to user behavior. For proof-of-concept, we assumed that all provider stations (placed at network edges) advertise the same price, and showed by simulations that DCC can perform as close as Smart Market. Later in Chapter 6, we relaxed the assumption of same price advertisement at provider stations, and called the new version of DCC as Distributed-DCC.

Distributed-DCC's granularity level is short-term contracts, which is a larger granularity than per-packet pricing as in Smart Market and a smaller granularity than long-term contracts as in Expected Capacity [18]. However, this raised the issue of human involvement in contract negotiations since short-term contracts will require negotiation between user and the provider frequently. So, in Chapter 5, we wanted to answer the question how short these contract should be in order to maintain control over network congestion. We found that the contracts must be very short in order to maintain the control over congestion whereas human involvement is possible only at larger contracts.

So, we developed two pricing architectures based on this time-scale problem: Pricing for Congestion Control (PFCC) and Pricing over Congestion Control (POCC). PFCC uses small time-scale pricing directly for controlling congestion and employs end-placed software/hardware agents which take inputs from human user at large time-scale while negotiating with the provider at small time-scale on behalf of the user. POCC uses an underlying edge-to-edge congestion control mechanism by overlaying pricing on top of it. This way, POCC controls congestion at small time-scales while allowing pricing at time-scales large enough for human involvement. In Chapters 6 and 7, we illustrated how to adapt Distributed-DCC to the PFCC and POCC architectures respectively.

Also in Chapter 6, we showed that Distributed-DCC can achieve a range of weighted fairness types from max-min to proportional. We showed how the provider can set Distributed-DCC's parameter "fairness coefficient" in order to employ different fairness in rate allocation. For example, if the provider wants to penalize users whose traffic traversing more congested areas, then he/she can set fairness coefficient to higher values. If he/she wants to treat all users equally regardless of traffic routes, then he/she can set fairness coefficient to zero. We provided theory behind this capability of Distributed-DCC later in Chapter 9, along with sensitivity analysis of Distributed-DCC.

The Distributed-DCC framework we proposed in this thesis has potential to enable more complex economic models in the Internet. Within the framework, it is possible to implement dynamic pricing, particularly congestion pricing. It can be used as a building-block for a variety of network services which may have value to different sets of users.

10.2 Contributions

- A detailed survey of major pricing proposals for the Internet. (Chapter 2)

- A detailed study of the well-known pricing scheme Smart Market. ([82], Chapter 3)
- An implementation strategy for the Smart Market on diff-serv architecture and a packet-based simulation of it. ([86], Chapter 3)
- A new pricing framework, Dynamic Capacity Contracting (DCC), to implement congestion-sensitive pricing. ([72, 6], Chapter 4)
- Investigation of steady-state dynamics of congestion-sensitive pricing in a customer-provider network. ([87, 84], Chapter 5)
- An approximate model for the correlation between congestion-sensitive prices and actual network congestion level. ([87, 85], Chapter 5)
- Extension of DCC to Distributed-DCC in order to implement it on wide area networks. ([83, 84], Chapter 6)
- A new macro edge-to-edge, topology-independent capacity allocation algorithm: ETICA. ([84], Chapter 6)
- An algorithm for routing-sensitive bottleneck-count estimation: ARBE. ([84], Appendix D)
- A new edge-to-edge pricing scheme: EEP. ([83, 84], Chapters 4, 6 and 8)
- A new pricing architecture: Pricing over Congestion Control (POCC). ([84], Chapter 7)
- Investigation of elasticity in the area of networking and definition for "utility-bandwidth" elasticity. ([84], Chapter 8)
- Methods of achieving multiple fairness types within a single framework. ([83, 84], Chapters 6 and 9)
- A method for estimating ratio of continuous Gamma functions. ([85], Appendix C)

10.3 Future Research

This thesis mainly focused on problems associated with adaptive pricing of network services within a single diff-serv domain. Future research should develop schemes and techniques to solve inter-domain problems with a special focus on constructing new economic models for the Internet.

Distributed-DCC, which is an adaptive pricing framework for single diff-serv domain, has several possible future research agenda itself. We can list possible future research items for Distributed-DCC as follows:

- Using soft admission control techniques to set V_{max} parameter of the contracts in order to make the scheme more robust and conservative, i.e. do not temporarily allow users to contract more than available capacity which causes large transient queues
- In POCC architecture, finding better algorithms to manage the edge queues.
- Expansion of the concept of contracting to point-to-anywhere contracts.
- Exploring inter-domain pricing issues in diff-serv environment:
  - Exploring the concept of bandwidth intermediary to facilitate the mediation between customer and multiple providers by leveraging the DCC framework.
  - Finding good pricing strategies for the provider in different market environments, e.g. monopoly, competitive.

Since Distributed-DCC charges each edge-to-edge flow separately, an interesting future research is to investigate user strategies to select which edge-to-edge route to send his/her traffic. Within the Distributed-DCC framework, if the provider can route traffic according to user's selection, then users will have multiple edge-to-edge routes to send their traffic through. This way user can select among multiple edge-to-edge routes made available by the provider, can maximize his/her utility. Also, at the provider side, the provider can perform route-based pricing and can distribute user traffic to the non-utilized parts of its network by increasing price of heavily use routes.

LITERATURE CITED

[1] UCB/LBLN/VINT network simulator - ns (version 2). http://www-mash.cs.berkeley.edu/ns, 1997.

[2] D. Allen. Will GigaBit Ethernet WAN services make us forget about SONET? *Network Magazine*, 16(7):68–73, July 2001.

[3] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. *IETF RFC 2581*, April 1999.

[4] J. Altman and K. Chu. A proposal for a flexible service plan that is attrative to users and Internet service providers. In *Proceedings of Conference on Computer Communications (INFOCOM)*, 2001.

[5] J. Altmann, H. Oliver, H. Daanen, and A. S.-B. Suarez. How to market-manage a QoS network. In *Proceedings of Conference on Computer Communications (INFOCOM)*, 2002.

[6] G. S. Arora, M. Yuksel, S. Kalyanaraman, T. Ravichandran, and A. Gupta. Price discovery at network edges. In *Proceedings of International Symposium on Performance Evaluation of Telecommunication Systems (SPECTS)*, July 2002.

[7] D. Axner. Scoping out the GigE MAN vendors. *Network World*, September 2001.

[8] J. Bailey, J. Nagel, and S. Raghavan. Ex-Post Internet pricing. In *MIT/Tufts Internet Service Quality Economics Workshop*, December 1999.

[9] T. Basar and R. Srikant. Revenue-maximizing pricing and capacity expansion in a many-users regime. In *Proceedings of Conference on Computer Communications (INFOCOM)*, 2002.

[10] E. Blanton and M. Allman. On making TCP more robust to packet reordering. *ACM Computer Communication Review*, 32(1), January 2002.

[11] A. Bouch and M. A. Sasse. Why value is everything?: A user-centered approach to Internet quality of service and pricing. In *Proceedings of IEEE/IFIP International Workshop on Quality of Service (IWQoS)*, 2001.

[12] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet architecture: An overview. *IETF RFC 1633*, June 1994.

[13] L. Brakmo and L. Peterson. TCP Vegas: End-to-end congestion avoidance on a global Internet. *IEEE Journal of Selected Areas in Communications*, 13(8):1465–1480, October 1995.

[14] A. C. Chiang. *Fundamental Methods of Mathematical Economics.* McGraw Hill, Inc., 1984.

[15] D. G. Childers. *Probability and Random Processes.* McGraw Hill, Inc., 1997.

[16] D. M. Chiu. Some observations on fairness of bandwidth sharing. Technical Report TR-99-80, Sun Microsystems Laboratories, September 1999.

[17] D. M. Chiu and R. Jain. Analysis of increase/decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, June 1989.

[18] D. Clark. A model for cost allocation and pricing in the Internet. Technical report, MIT, 1995.

[19] D. Clark. *Internet cost allocation and pricing.* Eds McKnight and Bailey, MIT Press, 1997.

[20] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang. Pricing in computer networks: Motivation, formulation and example. *IEEE/ACM Transactions on Networking*, 1, December 1993.

[21] M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes formulation and example. *IEEE/ACM Transactions on Networking*, 5(6):835–846, December 1997.

[22] J. L. Devore. *Probability and Statistics for Engineeting and the Sciences.* Brooks/Cole Publishing Company, 1995.

[23] R. J. Dolan and H. Simon. *Power Pricing: How Managing Price Transforms the Bottom Line.* The Free Press, 1996.

[24] P. Dube, V. Borkar, and D. Manjunath. Differential join prices for parallel queues: Social optimality, dynamic pricing algorithms and application to Internet pricing. In *Proceedings of Conference on Computer Communications (INFOCOM)*, 2002.

[25] R. J. Edell and P. P. Varaiya. Providing Internet access: What we learnt from the INDEX trial. Technical Report 99-010W, University of California, Berkeley, 1999.

[26] J. Boyle et. al. The COPS(Common Open Policy Service) Protocol. IETF Internet Draft, draft-ietf-rap-cops-02.txt, August 1998.

[27] S. Blake et. al. An architecture for Differentiated Services. *IETF RFC 2475*, December 1998.

[28] S. Floyd and T. Henderson. The NewReno modification to TCP's fast recovery algorithm. *IETF RFC 2582*, April 1999.

[29] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[30] N. Foukia, D. Billard, P. Reichl, and B. Stiller. User behavior for a pricing scheme in a multi-provider scenario. In *MIT/Tufts Internet Service Quality Economics Workshop*, December 1999.

[31] R. J. Gibbens and F. P. Kelly. Resource pricing and evolution of congestion control. *Automatica*, 35:1969–1985, 1999.

[32] A. Gupta, D. O. Stahl, and A. B. Whinston. Pricing of services on the Internet. Technical report, University of Texas, 1997.

[33] A. Gupta, D. O. Stahl, and A. B. Whinston. *Priority pricing of Integrated Services networks.* Eds McKnight and Bailey, MIT Press, 1997.

[34] A. Gupta, D. O. Stahl, and A. B. Whinston. The economics of network management. *Communications of ACM*, 42:57–63, 1999.

[35] D. Harrison and S. Kalyanaraman. Edge-to-edge traffic control for the Internet. Technical Report ECSE-NET-2000-2, Rensselaer Polytechnic Institute, Networks Lab, July 2000.

[36] D. Harrison, S. Kalyanaraman, and S. Ramakrishnan. Overlay bandwidth services: Basic framework and edge-to-edge closed-loop building block. Poster in the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), 2001.

[37] D. Harrison, S. Kalyanaraman, and S. Ramakrishnan. Congestion control as a bulding-block for QoS. *Computer Communication Review*, 32:71–71, 2002.

[38] J. Hoe. Improving the start-up behavior of a congestion control scheme for TCP. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, August 1996.

[39] V. Jacobson. Congestion avoidance and control. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, August 1988.

[40] R. Jain, T. Mullen, and R. Hausman. Analysis of Paris Metro Pricing strategy for QoS with a single service provider. In *Proceedings of IEEE/IFIP International Workshop on Quality of Service (IWQoS)*, 2001.

[41] F. P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[42] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control in communication networks: Shadow prices, proportional fairness and stability. *Journal of Operations Research Society*, 49:237–252, 1998.

[43] L. Kleinrock. *Queueing Systems, Volume I: Theory*. John Wiley and Sons, 1975.

[44] S. Kunniyur and R. Srikant. End-to-end congestion control: Utility functions, random losses and ecn marks. In *Proceedings of Conference on Computer Communications (INFOCOM)*, 2000.

[45] X. Lin and N. Shroff. Simplification of network dynamics in large systems. In *Proceedings of IEEE/IFIP International Workshop on Quality of Service (IWQoS)*, pages 54–63, 2002.

[46] S. H. Low. A duality model of TCP flow controls. In *Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, September 2000.

[47] S. H. Low and D. E. Lapsley. Optimization flow control – I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–875, 1999.

[48] S. H. Low, L. L. Peterson, and L. Wang. Understanding Vegas: A duality model. In *Proceedings of ACM SIGMETRICS*, 2001.

[49] J. K. MacKie-Mason, L. Murphy, and J. Murphy. *Responsive pricing in the Internet*. Eds McKnight and Bailey, MIT Press, 1997.

[50] J. K. MacKie-Mason and H. R. Varian. *Pricing the Internet*. Kahin, Brian and Keller, James, 1993.

[51] J. K. MacKie-Mason and H. R. Varian. Pricing the congestible network resources. *IEEE Journal of Selected Areas in Communications*, 13:1141–1149, 1995.

[52] P. Marbach. Pricing Differentiated Services networks: Bursty traffic. In *Proceedings of Conference on Computer Communications (INFOCOM)*, pages 184–193, 2001.

[53] P. Marbach. Priority service and max-min fairness. In *Proceedings of Conference on Computer Communications (INFOCOM)*, 2002.

[54] M. Martin. Yipes extends Ethernet line. *Network World*, September 2001.

[55] M. Mathis and J. Mahdavi. Forward Acknowledgement: Refining TCP congestion control. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, August 1996.

[56] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanov. TCP Selective Acknownledgment options. *IETF RFC 2018*, October 1996.

[57] B. M. Mitchell and I. Vogelsang. *Telecommunications Pricing: Theory and Practice*. Cambridge University Press, 1991.

[58] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, October 2000.

[59] A. M. Odlyzko. A modest proposal for preventing Internet congestion. Technical report, AT & T Research Lab, 1997.

[60] A. M. Odlyzko. The economics of the Internet: Utility, utilization, pricing, and quality of service. Technical report, AT & T Research Lab, 1998.

[61] A. M. Odlyzko. Internet pricing and history of communications. Technical report, AT & T Research Lab, 2000.

[62] A. Orda and N. Shimkin. Incentive pricing in multi-class communication networks. In *Proceedings of Conference on Computer Communications (INFOCOM)*, 1997.

[63] D. Pappalardo. AT&T dives into metro Ethernet. *Network World*, September 2001.

[64] I. Ch. Paschalidis and Y. Liu. Pricing in multiservice loss networks: Static pricing, asymptotic optimality, and demand substitution effects. *IEEE/ACM Transactions on Networking – in print*, 2002.

[65] I. Ch. Paschalidis and J. N. Tsitsiklis. Congestion-dependent pricing of network services. *IEEE/ACM Transactions on Networking*, 8(2):171–184, 2000.

[66] S. D. Patek and E. Campos-Nanez. Pricing of dial-up services: An example of congestion-dependent pricing. University of Virginia, 2000.

[67] N. Semret, R. R.-F. Liao, A. T. Campbell, and A. A. Lazar. Market pricing of differentiated Internet services. In *Proceedings of IEEE/IFIP International Workshop on Qualityof Service (IWQoS)*, pages 184–193, 1999.

[68] N. Semret, R. R.-F. Liao, A. T. Campbell, and A. A. Lazar. Pricing, provisioning and peering: Dynamic markets for differentiated Internet services and implications for network interconnections. *IEEE Journal of Selected Areas in Communications*, 18, 2000.

[69] S. Shenker. Fundamental design issues for the future Internet. *IEEE Journal of Selected Areas in Communications*, 13:1176–1188, 1995.

[70] S. Shenker, D. Clark, D. Estrin, and S. Herzog. Pricing in computer networks: Reshaping the research agenda. *Telecommunications Policy*, 10(3):183–201, 1996.

[71] R. Simon, W. S. Chang, and B. Jukic. Network Path Pricing: A QoS-based model. In *International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 457–465, August 2000.

[72] R. Singh, M. Yuksel, S. Kalyanaraman, and T. Ravichandran. A comparative evaluation of Internet pricing models: Smart Market and Dynamic Capacity Contracting. In *Proceedings of Workshop on Information Technologies and Systems (WITS)*, 2000.

[73] W. R. Stevens. *UNIX Network Programming, Volume 1, Second Edition, Networking APIs: Sockets and API.* Prentice Hall, 1998.

[74] H. R. Varian. Estimating the demand for bandwidth. In *MIT/Tufts Internet Service Quality Economics Workshop*, December 1999.

[75] H. R. Varian. *Intermediate Microeconomics: A Modern Approach.* W. W. Norton and Company, 1999.

[76] X. Wang and H. Schulzrinne. RNAP: A resource negotiation and pricing protocol. In *International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 77–93, 1999.

[77] X. Wang and H. Schulzrinne. An integrated resource negotiation, pricing, and QoS adaptation framework for multimedia applications. *IEEE Journal of Selected Areas in Communications*, 18, 2000.

[78] X. Wang and H. Schulzrinne. Performance study of congestion price based adaptive service. In *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 1–10, 2000.

[79] X. Wang and H. Schulzrinne. Pricing network resources for adaptive applications in a differentiated services network. In *Proceedings of Conference on Computer Communications (INFOCOM)*, 2001.

[80] R. B. Wilson. *Nonlinear Pricing.* Oxford University Press, 1993.

[81] Y. Yemini, A. Dailianas, and D. Florissi. MARKETNET: A market-based architecture for survivable large-scale information systems. In *MIT/Tufts Internet Service Quality Economics Workshop*, December 1999.

[82] M. Yuksel and S. Kalyanaraman. Simulating the Smart Market pricing scheme on Differentiated Services architecture. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS) part of Western Multi-Conference (WMC)*, 2001.

[83] M. Yuksel and S. Kalyanaraman. Distributed Dynamic Capacity Contracting: A congestion pricing framework for diff-serv. In *Proceedings of International Conference on Management of Multimedia Networks and Services (MMNS)*, 2002.

[84] M. Yuksel and S. Kalyanaraman. Distributed Dynamic Capacity Contracting: An overlay congestion pricing framework. Submitted to Journal of Computer Communications special issue on Internet Pricing and Charging, 2002.

[85] M. Yuksel and S. Kalyanaraman. Pricing granularity for congestion-sensitive pricing. Submitted to Journal of Computer Communications special issue on Internet Pricing and Charging, 2002.

[86] M. Yuksel and S. Kalyanaraman. A strategy for implementing Smart Market pricing scheme on diff-serv. In *Proceedings of Communication Quality and Reliability Symposium part of GLOBECOM*, 2002.

[87] M. Yuksel, S. Kalyanaraman, and Biplab Sikdar. Effect of pricing intervals on congestion-sensitivity of network service prices. Technical Report ECSE-NET-2002-1, Rensselaer Polytechnic Institute, ECSE Networks Lab, 2002.

APPENDIX A
Maximization of Total User Utility in a Multi-User Single-Bottleneck Network

Suppose n users are using a network with a capacity of c. $u_i(x) = b_i logx$ represents the utility function of user i, where x is the amount of capacity given to the user i and b_i is the user i's sensitivity to network capacity, i.e. willingness-to-pay for the service. Let $x_1, x_2, ..., x_n$ be the capacities given to the users correspondingly from 1 to n. In order to maximize total user utility, what is the optimum capacity allocation to the users with the constraint that $\sum_{i=1}^{n} x_i \leq c$?

The total user utility will be:

$$U = \sum_{i=1}^{n} u_i(x_i)$$

Since user utilities are non-decreasing functions of capacity, we need to allocate all the available capacity to the users in order to maximize U. So for the optimum capacity allocation, the following equation holds:

$$\sum_{i=1}^{n} x_i = c$$

By applying Lagrange-Multiplier Method [14], we first convert U to the following:

$$U = Z = \sum_{i=1}^{n} u_i(x_i) + \lambda(c - \sum_{i=1}^{n} x_i)$$

We can get the following system of equations by equating partial derivative of Z to zero for each unknown variable:

$$\begin{aligned} Z_\lambda &= c - \sum_{i=1}^{n} x_i = 0 \\ Z_i &= \frac{b_i}{x_i} - \lambda = 0, \quad i = 1..n \end{aligned} \tag{A.1}$$

After solving system equations A.1, we get the solution as follows:

$$\begin{aligned} \lambda &= \frac{\sum_{i=1}^{n} b_i}{c} \\ x_i &= \frac{b_i}{\sum_{j=1}^{n} b_j} c, \quad i = 1..n \end{aligned} \tag{A.2}$$

This result shows that maximization of user utilities can be done only by allocating the limited capacity to the users proportional to their sensitivity to bandwidth, i.e. willingness-to-pay.

APPENDIX B
Optimal Prices for Social Welfare Maximization

In Chapter 8, we formulated and solved the problem of total user utility maximization for a network. In this appendix, we formulate and solve the problem of social welfare maximization, which has subtle differences. We will show that the two optimization problems in fact result in the same optimal prices.

B.1 Problem Formulation

Additional to the notation used in Chapter 8, let $K_l(x) = k_l x$ be the cost of providing the capacity x at any link l where k_l is a non-negative constant[18]. We can formulate the social welfare maximization problem as follows:

$SYSTEM$:

$$\max_{\lambda} \sum_{f} \left[U_f(\lambda_f) - \sum_{l \in L(f)} K(\lambda_f) \right]$$

subject to

$$\sum_{f \in F(l)} \lambda_f \;\leq\; c_l, \quad l = 1, ..., L \tag{B.1}$$

We again split this problem two separate problems:

The first problem is solved at the user side. Given accumulation of link prices on the flow f's route, p^f, what is the optimal sending rate in order to *maximize surplus.*

$FLOW_f(p^f)$:

$$\max_{\lambda_f} \left\{ U_f(\lambda_f) - \sum_{l \in L(f)} p_l \lambda_f \right\}$$

over

[18]Note that results of the analysis will be effected by this assumption linear cost function. A more realistic model would be to use a concave cost function.

$$\lambda_f \geq 0 \tag{B.2}$$

The second problem is solved at the provider's side. Given sending rate of user flows (which are dependent on the link prices), what is the optimal price to advertise in order to *maximize profit*. Notice that in the social welfare paximization problem the provider solves profit maximization problem rather than revenue maximization problem which was the case in the total user utility maximization problem in Chapter 8. This is because, in social welfare maximization, cost function is introduced into the system model.

$NETWORK(\lambda(p^f))$:

$$\max_p \sum_f \left[\sum_{l \in L(f)} p_l \lambda_f - \sum_{l \in L(f)} K(\lambda_f) \right]$$

subject to

$$\sum_{f \in F(l)} \lambda_f \;\; \leq \;\; c_l, \quad l = 1, ..., L$$

over

$$p \geq 0 \tag{B.3}$$

Let the total price paid by flow f be $p^f = \sum_{l \in L(f)} p_l$. Then, solution to $FLOW_f(p^f)$ will be:

$$U_f'(\lambda_f) = p^f$$

$$\lambda_f(p^f) = U_f'^{-1}(p^f) \tag{B.4}$$

When it comes to the $NETWORK(\lambda(p^f))$ problem, the solution will be dependent on user flows utility functions since their sending rate is based on their utility functions as shown in the solution of $FLOW_f(p^f)$. So, in the next sections we will solve the $NETWORK(\lambda(p^f))$ problem for the cases of logarithmic and non-logarithmic utility functions.

B.2 Optimal Prices: Logarithmic Utility Functions

We again model customer i's utility with the well-known function

$$u_i(x) = w_i log(x) \tag{B.5}$$

where x is the allocated bandwidth to the customer and w_i is customer i's budget (or bandwidth sensitivity).

Additional to the vectorized notation in Chapter 8, let K be the column vector of the cost of providing unit link capacities. Given this notation, we can write the following relationships:

$$AP = P^* e \tag{B.6}$$

$$\lambda = (\hat{\lambda} e)^T = e^T \hat{\lambda}$$

where e is the column unit vector.

We use the utility function of (B.5) in our analysis. By plugging (B.5) in (B.4) we obtain flow's demand function in vectorized notation:

$$\lambda(P^*) = W P^{*-1} \tag{B.7}$$

where W is row vector of the weights w_i in flow's utility function (B.5). Similarly, we can write derivative of (B.7) as:

$$\lambda'(P^*) = -W(P^{*2})^{-1} \tag{B.8}$$

Also, we can write the utility function (B.5) and its derivative in vectorized notation as follows:

$$U(\lambda) = W log(\hat{\lambda}) \tag{B.9}$$

$$U'(\lambda) = W \hat{\lambda}^{-1} \tag{B.10}$$

The profit maximization of (B.3) can be re-written as follows:

$$\max_P R = \lambda AP - \lambda AK$$

subject to

$$\lambda A \leq C^T. \tag{B.11}$$

So, we write the Lagrangian as follows:

$$L = \lambda AP - \lambda AK + (C^T - \lambda A)\gamma \tag{B.12}$$

where γ is column vector of the Lagrange multipliers for the link capacity constrain.

By plugging (B.7) and (B.8) in appropriate places, the optimality conditions for (B.12) can be written as:

$$L_\gamma : C^T - W{P^*}^{-1}A = 0 \tag{B.13}$$

$$L_{P^*} : -W({P^*}^2)^{-1}(P^*e - AK) + W{P^*}^{-1}e - W({P^*}^2)^{-1}A\gamma = 0 \tag{B.14}$$

By solving B.14 for P^*, we obtain:

$$-{P^*}^{-1}(P^*e - AK) + Ie - {P^*}^{-1}A\gamma = 0 \tag{B.15}$$

$${P^*}^{-1}AK - {P^*}^{-1}A\gamma = 0 \tag{B.16}$$

$${P^*}^{-1}(AK - A\gamma) = 0 \tag{B.17}$$

$$P^* = 0 \tag{B.18}$$

Now, solve (B.13) for P^*:

$$C^T - W{P^*}^{-1}A = 0 \tag{B.19}$$

$$C^T A^{-1} P^* = W \tag{B.20}$$

$$P^* = A(C^T)^{-1}W \tag{B.21}$$

Apparently, the optimization problem has two solutions as shown in (B.18) and (B.21). Since (B.18) violates the condition $P > 0$, we accept the solution in (B.21).

We finally derive P by using (B.6):

$$AP = P^{*}e = A(C^{T})^{-1}We \tag{B.22}$$

$$P = (C^{T})^{-1}We \tag{B.23}$$

Since $P^{*} = (P^{*})^{T}$, we can derive another solution:

$$AP = P^{*}e = W^{T}C^{-1}A^{T}e \tag{B.24}$$

$$P = A^{-1}W^{T}C^{-1}A^{T}e \tag{B.25}$$

Notice that the result in (B.25) is the same as in the case of total user utility maximization in (8.25).

B.3 Optimal Prices: Non-Logarithmic Utility Functions

As in Chapter 8, we use the same generic utility function. The function and its derivative are as follows:

$$U(\lambda) = B\hat{\lambda}^{\epsilon} \tag{B.26}$$

$$U'(\lambda) = B\epsilon\hat{\lambda}^{\epsilon-1} \tag{B.27}$$

According to the relationship between ϵ and ε described in Section 8.4.1, we can write the demand function and its derivative as follows:

$$\lambda(P^{*}) = \epsilon^{-\varepsilon}e^{T}\hat{B}^{-\varepsilon}P^{*\varepsilon} \tag{B.28}$$

Similarly, we can write derivative of (B.28) as:

$$\lambda'(P^{*}) = \epsilon^{-\varepsilon}\varepsilon e^{T}\hat{B}^{-\varepsilon}P^{*\varepsilon-1} \tag{B.29}$$

For the revenue maximization problem, we again solve the Lagrangian in (B.12) but for the new demand function of (B.28). By plugging (B.28) and (B.29) in appropriate places, the optimality conditions for (B.12) can be written as:

$$L_\gamma : C^T - \epsilon^{-\varepsilon} e^T \hat{B}^{-\varepsilon} P^{*\varepsilon} A = 0 \tag{B.30}$$

$$L_{P^*} : \epsilon^{-\varepsilon} \varepsilon e^T \hat{B}^{-\varepsilon} P^{*\varepsilon - 1}(P^* e - AK - A\gamma) + \epsilon^{-\varepsilon} e^T \hat{B}^{-\varepsilon} P^{*\varepsilon} e = 0 \tag{B.31}$$

By solving (B.31) for P^*, we obtain:

$$\varepsilon e^T \hat{B}^{-\varepsilon} P^{*\varepsilon - 1}(P^* e - AK - A\gamma) + e^T \hat{B}^{-\varepsilon} P^{*\varepsilon} e = 0 \tag{B.32}$$

$$\varepsilon P^{*\varepsilon - 1}(P^* e - AK - A\gamma) + P^{*\varepsilon} e = 0 \tag{B.33}$$

$$\varepsilon P^{*-1}(P^* e - AK - A\gamma) + Ie = 0 \tag{B.34}$$

$$\varepsilon Ie - \varepsilon P^{*-1}(AK + A\gamma) + Ie = 0 \tag{B.35}$$

$$(\varepsilon + 1)Ie = \varepsilon P^{*-1}(AK + A\gamma) \tag{B.36}$$

$$P^* e = \frac{\varepsilon}{\varepsilon + 1}(AK + A\gamma) \tag{B.37}$$

$$P^* = \frac{1}{\epsilon}(AK + A\gamma)e^{-1} \tag{B.38}$$

Now, apply (B.38) into (B.30) and solve for γ:

$$C^T = \epsilon^{-\varepsilon} e^T \hat{B}^{-\varepsilon} \left(\frac{1}{\epsilon}(AK + A\gamma)e^{-1}\right)^{\varepsilon} A \tag{B.39}$$

$$\epsilon^{\varepsilon} \hat{B}^{\varepsilon} (e^T)^{-1} C^T A^{-1} = \left(\frac{1}{\epsilon}(AK + A\gamma)e^{-1}\right)^{\varepsilon} \tag{B.40}$$

$$\frac{1}{\epsilon}(AK + A\gamma)e^{-1} = \epsilon A^{-1/\varepsilon} (C^T)^{1/\varepsilon} (e^T)^{-1/\varepsilon} \hat{B} \tag{B.41}$$

Substitute (B.41) into (B.38) and we obtain P^*:

$$P^* = \epsilon A^{-1/\varepsilon} (C^T)^{1/\varepsilon} (e^T)^{-1/\varepsilon} \hat{B} \tag{B.42}$$

From (B.42) we obtain P:

$$AP = P^* e = \epsilon A^{-1/\varepsilon} (C^T)^{1/\varepsilon} (e^T)^{-1/\varepsilon} \hat{B} e \tag{B.43}$$

$$P = \epsilon A^{-1} A^{-1/\varepsilon} (C^T)^{1/\varepsilon} (e^T)^{-1/\varepsilon} \hat{B} e \tag{B.44}$$

$$P = \epsilon A^{-1} A^{|1/\varepsilon|} \left((C^T)^{|1/\varepsilon|} \right)^{-1} (e^T)^{|1/\varepsilon|} \hat{B} e \tag{B.45}$$

$$P = \epsilon A^{-1} A^{|1/\varepsilon|} \left((C^T)^{|1/\varepsilon|} \right)^{-1} (e^T)^{|1/\varepsilon|} \left(\hat{B}^{|\varepsilon|} \right)^{|1/\varepsilon|} e \tag{B.46}$$

The result in (B.46) is again the same as in the case of total user utility maximization in (8.49).

APPENDIX C
Approximating Ratios of Complete or Incomplete Continuous Gamma Functions

C.1 The Gamma Function and Problem Definition

Gamma function has two versions: *complete*, *incomplete* [22]. Complete and incomplete continuous Gamma functions are respectively as follows:

$$\Gamma(x) = \int_{t=0}^{\infty} e^{-t} t^{x-1} dt \tag{C.1}$$

$$\Gamma(x, y) = \int_{t=y}^{\infty} e^{-t} t^{x-1} dt \tag{C.2}$$

Discrete version of the complete Gamma function is a simple factorial:

$$\Gamma(x) = (x-1)! \tag{C.3}$$

Let f be the function being integrated in the continuous Gamma functions, i.e.:

$$f(t, x) = e^{-t} t^{x-1} \tag{C.4}$$

Figure C.1 shows plot of the function $f(t, x)$ for various values of x. Notice that the Gamma function is nothing but the area under the curve of $f(t, x)$. Figures C.2 illustrates the difference between complete and incomplete Gamma functions in terms of area under the curve of $f(t, x)$. The area $A + B$ corresponds to the complete Gamma function $\Gamma(x)$, and A corresponds to the incomplete Gamma function $\Gamma(x, y)$.

Given the above information, we want to approximate ratio:

$$\frac{\Gamma(x, y)}{\Gamma(x)} = \frac{A}{A + B} \tag{C.5}$$

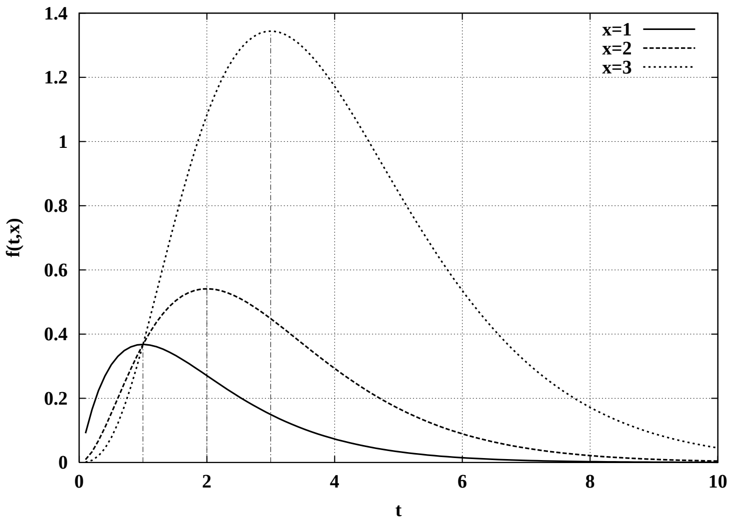


Figure C.1: The function $f(t,x)$ for various values of x.

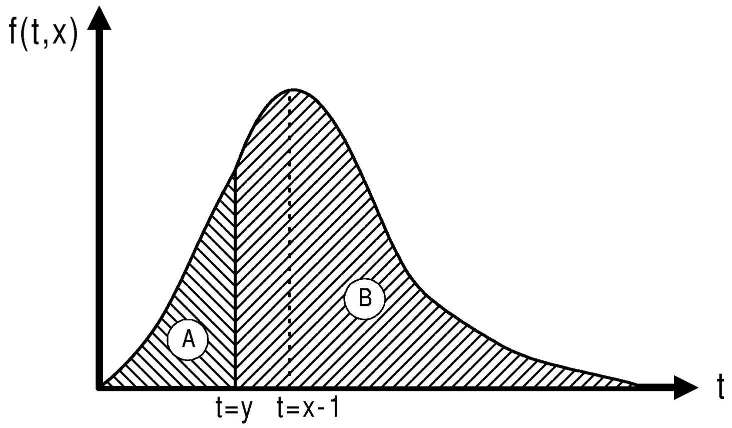


Figure C.2: Visualization of complete and incomplete Gamma functions: The area B is $\Gamma(x,y)$, and the area $A+B$ is $\Gamma(x)$.

C.2 Approximation Methodology

The intuition behind our approximations is the similarity of shape of $f(t,x)$ to *triangle*. Observe from Figure C.1 that as the parameter x gets larger the shape of $f(t,x)$ more triangular. We use this similarity in approximating the ratio in (C.5). Figure C.3-a shows an example triangle being matched to the $f(t,x)$ function. In that example we approximate the ratio of (C.5) as:

$$R = \frac{\Gamma(x,y)}{\Gamma(x)} = \frac{A}{A+B} \cong \frac{A'}{A'+B'}$$

Notice that the function $f(t,x)$'s maxima is the point at $t = x-1$, i.e. $f(x-1,x)$. Just to ease notation, let $t_m = x-1$ and $g(t) = f(t,x)$. Also, let's call the smaller piece of the triangle between $t = 0$ and $t = x-1$ as *left-piece triangle*, and the other piece of it as *right-piece triangle*. So, the left-piece triangle will have coordinates: $(0,0)$, $(0,t_m)$, $(t_m, g(t_m))$. For the right-piece triangle, we can consider various coordinates depending how well we want to approximate. Actually, the problem is to identify where should the hypotenus of the right-piece triangle intersect with $f(t,x)$. Since the shape of $f(t,x)$ gets similar to an equi-sided triangle as x gets larger, we choose to select this intersection point at $t = 2t_m = 2(x-1)$, which will resemble it more to an equi-sided triangle. With this consideration, we can calculate the coordinates of the right-piece triangle by simple geometry rules: $(0,t_m)$, $(t_m, g(t_m))$, $(t_m(g(t_m) - g(2t_m))/(g(t_m) - g(2t_m)), 0)$.

Since we know the function $f(t,x)$, we now can calculate areas A' and B'. However, this is dependent on whether y resides on the left of the right of $t_m = x-1$. So, we need to consider three cases:

C.2.1 Case I: $y = x - 1$

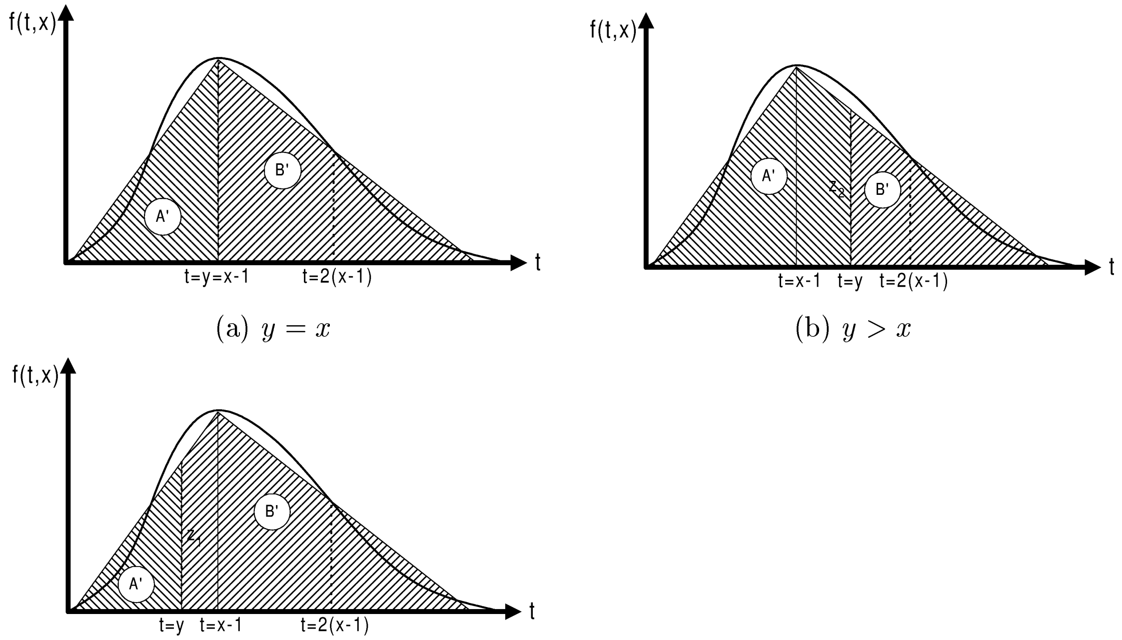
This case is shown in Figure C.3-a. Calculations of the triangular areas in the figure will be as follows:

$$A' = \frac{t_m g(t_m)}{2}$$

$$B' = \frac{\left(t_m + \frac{t_m g(2t_m)}{g(t_m) - g(2t_m)}\right) g(t_m)}{2}$$

So, the ratio R for this case will be:

$$R_1 = \frac{g(t_m) - g(2t_m)}{3g(t_m) - 2g(2t_m)}$$



(a) $y = x$ (b) $y > x$ (c) $y < x$

Figure C.3: Three possible cases for approximation of ratio $\Gamma(x,y)/\Gamma(x)$.

C.2.2 Case II: $y < x - 1$

This case is shown in Figure C.3-b. Calculations of the triangular areas in the figure will be as follows:

$$A' = \frac{yz_1}{2}$$

$$B' = \frac{\left(2t_m + \frac{t_m g(2t_m)}{g(t_m)-g(2t_m)}\right) g(t_m)}{2} - \frac{yz_1}{2}$$

where $z_1 = yg(t_m)/t_m$. So, the ratio R for this case will be:

$$R_2 = \frac{y^2}{t_m^2} \frac{g(t_m) - g(2t_m)}{2g(t_m) - g(2t_m)}$$

C.2.3 Case III: $y > x - 1$

This case is shown in Figure C.3-c. Calculations of the triangular areas in the figure will be as follows:

$$A' = \frac{\left(2t_m + \frac{t_m g(2t_m)}{g(t_m)-g(2t_m)}\right) g(t_m)}{2} - \frac{yz_2}{2}$$

$$B' = \frac{yz_2}{2}$$

where $z_2 = g(2t_m)$. So, the ratio R for this case will be:

$$R_3 = \frac{2t_m g(t_m)^2 - (t_m - y)g(t_m)g(2t_m) + yg(2t_m)^2}{t_m g(t_m)(2g(t_m) - g(2t_m))}$$

C.2.4 Integration of All Cases

In order to calculate the ratio $R = \Gamma(x,y)/\Gamma(x)$, we need to know if y is equal to, less than, or greater than $x-1$ as presented in the previous sections corresponding to each case.

We can put together an integrated formula for R by considering probability of each case happening. Let p_1 be the probability of being y equal to x-1 (i.e. Case I), p_2 be the probability of being y less than x-1 (i.e. Case II). Then, an integrated formula for R will be:

$$R = p_1 R_1 + p_2 R_2 + (1 - p_1 - p_2)R_3 \tag{C.6}$$

Since p_1 and p_2 will depend on distribution of y, the integrated approximation of R will change significantly based on that distribution. In modeling of the correlation between prices and congestion measures, in Section 5.3.1.2, we used the integrated formula by calculating the probabilities p_1 and p_2 based on the Possion distribution of the traffic.

Also note that in this particular appendix we only provided methodology for approximating the ratio $\Gamma(x,y)/\Gamma(x)$. It is possible to use the ideas in this appendix for approximating other possible ratios of Gamma functions, such as $\Gamma(x,y_1)/\Gamma(x,y_2)$, $\Gamma(y)/\Gamma(x)$.

APPENDIX D
Algorithm for Routing-Sensitive Bottleneck-Count Estimation (ARBE)

Given a diff-serv network, we would like to estimate number of bottlenecks each edge-to-edge flow is passing through. The algorithm ARBE presented in this appendix provides a solution to this problem.

Assuming that interior routers increment *bottleneck-count* header field of packets when congested, ARBE calculates the number of bottlenecks an edge-to-edge flow is passing through. ARBE operates at the *egress* edge router.

Assuming that each bottleneck has the same amount of congestion and also assume that they have the same capacity. Let $r_{ij}(t)$ be the number of bottlenecks the flow from ingress i to egress j, f_{ij}, is passing through at time t. ARBE operates on deterministic time intervals, and calculates $r_{ij}(t)$ as follows:

$$r_{ij}(t) = \begin{cases} \hat{r}_{ij}(t), & r_{ij}(t-1) \leq \hat{r}_{ij}(t) \\ r_{ij}(t-1) - \triangle r, & otherwise \end{cases} \tag{D.1}$$

where $\hat{r}_{ij}(t)$ is the highest number of bottlenecks that flow passed through in time interval t, $\triangle r$ is a pre-defined value. $\hat{r}_{ij}(t)$ is updated at each packet arrival by simply equating it to the maximum of its actual value and the bottleneck-count header field of the newly arrived packet. Algorithm 1 shows the pseudo-code for the algorithm.

Realize that the bottleneck-count header field of the packets are being incremented only if they are passing through a congested bottleneck. It is possible that some of the bottlenecks are not congested when a particular packet is passing through them. For example, the bottleneck-count header field of the packet may be incremented only three times, although it actually passed through six bottlenecks. So, it is necessary to bias the estimation to the largest number of bottlenecks the packets of that flow have passed recently.

Algorithm 1 Algorithm for Routing-Sensitive Bottleneck-Count Estimation

ARBE($BC(t)$, $\triangle r$)
$\{\triangle r$ is decaying step-size.$\}$
$\{BC(t)$ is the maximum bottleneck-count received in the last interval t.$\}$
$\{BC$ is the actual estimation for bottleneck-count.$\}$
if $BC(t) > BC$ **then**
  $BC \leftarrow BC(t)$
else
  $BC \leftarrow BC - \triangle r$
end if
return BC

Also as another issue, IP routing causes route of the flows to change dynamically. To consider the dynamic behavior of the routes, it is also necessary to decrease r_{ij} when $r_{ij}(t-1) > \hat{r}_{ij}(t)$. So, if the route of the flow has changed, then after some time (depending on how large the $\triangle r$ is) the value of r_{ij} will decrease to the actual number of bottlenecks the flow is passing through.

APPENDIX E
Max-Min Fairness, Proportional Fairness, and Social Welfare Maximization

Consider a multi-bottleneck network in which there is a long flow that is crossed by n parallel flows. An example of such a network is shown in Figure 3.2-b. Suppose all the bottlenecks are equivalent in capacity, C. Intuitively, whatever the long flow gets, all the parallel flows will get the rest of the capacity. Let x_0 be the capacity given to the long flow and x_1 be the capacity given to one of the parallel flows. Suppose that the utility of the long flow is $u_0(x_0) = w_0 log(x_0)$ and the utility of one of the parallel flows is $u_1(x_1) = w_1 log(x_1)$. Notice that w_0 and w_1 are the sensitivity of the flows to capacity (also interpreted as flow's budget). Since the long flow is passing through n bottlenecks, cost of providing capacity to the long flow is n times more than cost of providing capacity to one of the parallel flows. So, let cost of providing x_1 to one of the parallel flows be $K_1(x_1) = kx_1$, and let the cost of providing x_0 to the long flow be $K_0(x_0) = nkx_0$. Within this context, the social welfare, W, and its Lagrangian will be:

$$W = w_0 log(x_0) + nw_1 log(x_1) - nkx_0 - nkx_1$$

$$W \equiv Z = w_0 log(x_0) + nw_1 log(x_1) - nkx_0 - nkx_1 + \lambda(x_0 + x_1 - C)$$

After solving the above Lagrangian, we get the following solutions for x_0 and x_1 to maximize W:

$$x_0 = \frac{w_0 C}{w_0 + nw_1}$$
$$x_1 = \frac{nw_1 C}{w_0 + nw_1}$$

From the above result, we make two observations:

- First, if both the long flow and a parallel flow have equal bandwidth sensitivity, i.e. $w_0 = w_1$, then the optimal allocation will be $x_0 = C/(n+1)$ and $x_1 =$

$Cn/(n+1)$. This is the *proportional fair* case. So, proportional fairness is optimal only when all the flows have equal bandwidth sensitivity. As another interpretation, it is optimal only if all the flows have equal budget.

- Second, if the long flow is sensitive to bandwidth n times more than a parallel flow, i.e. $w_0 = nw_1$, then the optimal allocation will be $x_0 = x_1 = C/2$. This is the *max-min fair* case. So, max-min fairness is optimal only when the long flow's utility is sensitive to bandwidth in proportion to the cost of providing capacity to it. In other words, by interpreting bandwidth sensitivity as the flow's budget, max-min fairness is optimal only when the long flow has budget in proportion to the cost of providing capacity to it.

Observations similar to above have been made in the area, e.g. [42, 16].

APPENDIX F
Pseudo-Code for Distributed-DCC

Algorithm 2 Algorithm for Ingress i.

DDCC_INGRESS_i(T)
{*This procedure is executed at contracting time-scale.*}
{C *is the total estimated network capacity.*}
{c_i *is the vector of allowed capacities for flows starting at this Ingress.*}
{$\hat{b}_i$ *is the vector of estimated budgets of flows starting at this Ingress.*}
{x_i *is the vector of sending rates of flows starting at this Ingress.*}
{T *is the contracting time-scale.*}

{*Run Budget Estimator and update vector of budget estimations.*}
$\hat{b}_i = x_i p_i$
Send $\hat{b}_{ij}$s to corresponding Egresses.
{*Run Pricing Scheme and update contract parameters* p_i *and* V_{max}. *Below is according to EEP pricing scheme.*}
$p_i = \hat{b}_i / c_i$
$V_{maxi} = c_i * T$

Algorithm 3 Algorithm for Egress j.

DDCC_EGRESS_j(α, r_{min}, $\triangle r$, $\triangle \hat{c}$, β)
{*This procedure is executed at every LPS Interval L.*}
{α *is the fairness coefficient.*}
{r_{min} *is the minimum number of bottlenecks a flow may traverse.*}
{$\triangle r$ *is the decaying step-size for ARBE.*}
{$\triangle \hat{c}$ *is the additive increase parameter for AIMD-ER.*}
{β *is the multiplicative decrease parameter for AIMD-ER.*}
{BC_{ij} *is the highest number of bottlenecks traversed by a single packet of flow* f_{ij} *in the last L.*}

{*Run Flow Cost Analyzer and estimate number of bottlenecks each flow is traversing.*}
for $\forall i \neq j$ **do**
  $\hat{r}_{ij} = ARBE(BC_{ij}, \triangle r)$
end for
{*Run Fairness Tuner to penalize flows passing through more bottlenecks.*}
for $\forall i \neq j$ **do**
  $b_{ij} = \hat{b}_{ij} / \left[r_{min} + \alpha(\hat{r}_{ij} - r_{min})\right]$
end for
{*Run Congestion-Based Capacity Estimator to estimate available capacity for each flow. Below is the algorithm for AIMD-ER algorithm.*}
for $\forall i \neq j$ **do**
  if Received congestion indication from flow f_{ij} in the last L **then**
    $\hat{c}_{ij} = \beta \hat{\mu}_{ij}$
  else
    $\hat{c}_{ij} = \hat{c}_{ij} + \triangle \hat{c}$
  end if
end for
Send row vectors $\hat{c}_j$ and b_j to LPS.
Inform LPS about congestion indications generated by each flow.

Algorithm 4 Algorithm for LPS.

LPS($\hat{k}$)
{*This procedure is executed at every LPS Interval L.*}
{B_c *is total budget of congested flows.*}
{C_c *is total estimated capacity for congested flows.*}
{$\hat{k}$ *is the amount of holding time for flows in congested state of ETICA algorithm.*}

{*Estimate total network capacity.*}
$C = 0$
for $\forall i$ **do**
  for $\forall j$ **do**
    $C += \hat{c}_{ij}$
  end for
end for
{*Run Capacity Allocator to calculate allowed capacities to flows. Below is the ETICA algorithm.*}
$B_c = 0$
$C_c = 0$
for $\forall i$ **do**
  for $\forall j$ **do**
    if Flow f_{ij} generate congestion indication in the last L **then**
      $K_{ij} = \hat{k}$
    else
      $K_{ij} -= 1$
    end if
    if $K_{ij} > 0$ **then**
      $C_c += \hat{c}_{ij}$
      $B_c += b_{ij}$
    end if
  end for
end for
for $\forall i$ **do**
  for $\forall j$ **do**
    if $K_{ij} > 0$ **then**
      $c_{ij} = b_{ij} C_c / B_c$
    else
      $c_{ij} = \hat{c}_{ij}$
    end if
  end for
end for
Send row vectors C and $\hat{c}_{ij}$s to corresponding Ingresses.