ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Networks xxx (2004) xxx–xxx

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Pricing and admission control for QoS-enabled Internet

Tianshu Li [*], Youssef Iraqi, Raouf Boutaba

School of Computer Science, University of Waterloo, 200 University Ave W, Waterloo, Ont., Canada N2L 3G1

Abstract

Over the past ten years, many pricing schemes have been proposed for a QoS-enabled network. Most of the proposed QoS-pricing schemes focus on congestion-sensitive pricing and optimal pricing solutions. Integrating pricing and admission control has not been studied in details. In this paper, we pay more attention to the interrelation between pricing and admission control in QoS-enabled networks and propose a tariff-based architecture framework that flexibly integrates pricing and admission control for multi-domain DiffServ networks. We study the pricing and user behaviors in detail and design a market-regulated pricing and admission control scheme in our framework. We model the system as a market so that the price of a service class reflects the resource availability inside the network and is regulated by the market itself. We also evaluate our pricing strategy and admission control scheme through simulations.
© 2004 Published by Elsevier B.V.

Keywords: QoS-enabled; Internet; Pricing; DiffServ; Admission control; Economic

1. Introduction

With the Internet evolving into a multi-service network, the static pricing scheme, flat-rate pricing, used in the current Internet, is no longer sufficient nor effective. The study of dynamic pricing in a QoS-enabled network environment has become one of the hottest research areas in recent years. To support QoS in the future Internet, two QoS architectures, Integrated Services (IntServ) and Differentiated Services (DiffServ), have been standardized by the IETF. In the IntServ paradigm [1], deterministic QoS guarantee is provided by reserving resources along the path for each QoS-sensitive flow. Each network element has to maintain per-flow state information which causes serious scalability problems especially in an environment such as the Internet. This led to the development of the DiffServ architecture [2], where a small number of service classes are offered to the users and service differentiation can be realized by the different pre-defined and agreed forwarding behaviors (i.e., Per-Hop Behavior (PHB)) inside the DiffServ networks. Since there is no explicit resource reservation, what DiffServ can provide is a statistical QoS guarantee. However, it is generally believed that DiffServ is more likely to be implemented in the Internet core because of its simplicity and scalability.

Unlike IntServ, which can charge users based on the allocated resources, pricing for DiffServ networks is more complicated and has drawn a lot of attention in the networking community. Before the wide deployment of DiffServ, an effective and

[*] Corresponding author.
E-mail addresses: dtianshu@bbcr.uwaterloo.ca (T. Li), iraqi@bbcr.uwaterloo.ca (Y. Iraqi), rboutaba@bbcr.uwaterloo.ca (R. Boutaba).

1389-1286/$ - see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.comnet.2004.03.020

efficient pricing scheme has to be developed. Meanwhile, since price is such an important economic incentive for the end-users, pricing is often considered as an effective mechanism for congestion control and admission control, which in turn can improve the level of QoS guarantees. Indeed, QoS pricing schemes proposed so far often entail either congestion control or admission control or even both.

However, since pricing for QoS-enabled networks is such a large problem, it is extremely difficult to address all the issues in one scheme. We categorize the proposed approaches according to the following three underlying objectives:

- Economic Efficiency and Optimality.
- Simplicity and Scalability.
- Quality of Service.

The first goal is to achieve the optimal overall net value of network usage. The second goal is to design a simple and scalable architecture/scheme. The last goal is allowing the service providers to provide better QoS guarantee to the customers. These objectives are often conflicting or even contradictory. Most existing approaches try to address one or more of these issues. So far, there has not been a well-accepted solution. In this work, we try to address as many of the issues as possible with an emphasis on the two last ones. We believe that optimal pricing solutions are hard or even impossible to achieve as the scale of the problem is large (e.g., the Internet). Furthermore, many researchers have pointed out that in practice maximal simplicity is often more important than maximal efficiency. In our approach, we model the system as a market so that the price of network usage is regulated by market forces rather than focusing on setting an optimal price.

In this paper, we study the pricing and QoS management from various angles and propose a scheme that is more suitable for the future QoS-enabled Internet. We first study the relationship between pricing and two traffic management functions: congestion control and admission control, and propose a tariff-based pricing architecture that integrates pricing and admission control for DiffServ networks. The proposed architecture maintains domain and global price tables for core networks only. In this way, we decouple the pricing for the core networks from the end-to-end pricing, which fits well into the DiffServ paradigm. This pricing architecture was initially proposed in our previous work [3]. In this paper, we enhance our price setting strategy and study the user reaction to price change through economic market model. An associated admission control and class promotion scheme is then introduced and evaluated in detail.

The remainder of this paper is organized as follows: Section 2 reviews some background and related work in this area. In Section 3, we discuss the motivation of this work and state the design choices for our architecture. Our pricing architecture and the construction of pricing tables will also be presented in this section. Section 4 discusses our price setting strategy and the market model, and Section 5 describes the end-to-end pricing and the associated admission control scheme in our framework. In Section 6, we show the simulation results and their analysis. Finally, Section 7 concludes this paper.

2. Background and related work

Pricing for the Internet in general has long been an active research area. Example approaches such as paris metro pricing [6], responsive pricing [7], proportional fairness pricing [30], smart-market pricing [8], two-tier market pricing [9], edge pricing [10], and many others study the pricing for networks from various angles. More detailed review of pricing schemes can be found in [4,5]. In general, most of approaches try to address the problem from the three aspects mentioned earlier. In the following subsections, we will review the approaches proposed in the literature following this categorization.

2.1. Economic efficiency and optimality

Since pricing itself is an economic problem, how to achieve economic efficiency drew a lot of attention. An interesting approach in this category is to create an auctioning environment and let users bid for the use of the network resources. In [8], a second-price auctioning model was used in the network where each packet carries a bid in its header. The transmitted packets will be charged for the market-clearing price rather than the actual bid where the market-clearing price is the highest rejected bid. In [9], authors proposed a two-tier auctioning mechanism for supporting DiffServ in a multi-domain environment. In their model, the network is decomposed hierarchically into subnetworks and each subnetwork is abstracted into a single bottleneck capacity. In order to provide a particular class of service to users, this service class will compete not only for the resources locally with other service classes inside the subnetwork but also for the resources in the neighboring subnetworks. Although auctioning does not require a prior knowledge of the user traffic characteristics and has been generally considered as the one that achieves economic efficiency, it has unfortunately significant implementation overhead.

In the literature, the most commonly used approach is to create an optimization model and find the optimal price for the use of network resources. Many optimal pricing schemes have been proposed in the past few years. Most of them either assume a well-known user utility function or user demand function and establish an optimization model to maximize either the social welfare or provider's revenue.

This approach can be formally stated as follows. Let R denote the set of users. Let $U_r(x)$ denote the utility function of a user r using x network resources, and $C_r(x)$ the cost of user r using x network resources. Let $R_r(x)$ denote the revenue generated by user r using x network resources and $C_p(x)$ the cost of service provider providing x network resources. Then the problem is modelled by the following optimization systems:

From the User's Perspective: (Case 1)

$$\text{Maximize}: \quad \left\{ \sum_{r\in R} (U_r(x) - C_r(x)) \right\} \tag{1}$$

$$\begin{aligned} \text{Subject to}: \quad & \text{a set of constraints} \\ & (\text{e.g., budget constraint} \\ & C_r(x) \leqslant b_r, \ \text{for all } r); \end{aligned}$$

From the Provider's Perspective: (Case 2)

$$\text{Maximize}: \quad \left\{ \sum_{r\in R} (R_r(x) - C_p(x)) \right\} \quad \text{or} \quad \left\{ \sum_{r\in R} R_r(x) \right\}$$

$$\begin{aligned} \text{Subject to}: \quad & \text{a set of constraints} \\ & (\text{e.g., capacity constraint } \textstyle\sum x \leqslant C, \\ & \text{QoS constraint, or fairness constraint}); \end{aligned}$$

where b_r denotes the budget of user r and C denotes the total capacity. Notice that the description is simplified here for ease of understanding. In the following, we will describe these models briefly (more detailed model descriptions can be found in [11,13,15,16,20,28,33]).

Although there have been many optimal pricing approaches proposed in the literature, these two general models show the basics in most of them. The differences among the existing approaches lie in the different models used to compute $U_r(x)$, $C_r(x)$, and $R_r(x)$, the set of constraints incorporated in the optimization models, or the mechanism used to solve the optimization problem.

In Case 1, many utility functions $U_r(x)$ and cost functions $C_r(x)$ have been proposed. One commonly adopted utility function is the logarithmic utility function as used in [11]. The cost function is usually a function of the price and the utilized resources $f(p,x)$. The solution of this kind of model is the optimal price that maximizes the social welfare (or surplus). In Case 2, various demand functions have been used to model the expected traffic and hence obtain the generated revenue. Demand functions are usually modelled as a decreasing function of the price and the objective is to find the optimal price that maximizes the provider's revenue or profit. Notice that the provider's cost of providing all resources $\sum C_p(x)$ is sometimes considered as a constant, thus the revenue maximization model is equivalent to the profit maximization model in this case.

While deriving the optimal price for their models, many schemes often incorporate various constraints such as the budget constraint, capacity constraint, QoS guarantee (e.g., delay or loss), and fairness index and try to obtain other desirable results including optimal resource allocation and max–min/proportional fair resource sharing while achieving economic efficiency [13,28,30].

Other than the different sets of constraints incorporated in the optimization models, various techniques are used to solve the optimization problem. One trend is to model the interaction among users and service providers (through pricing and resource allocation) as either a cooperative or a non-cooperative game and use game theory to analyze the system. Example approaches can be found in [28–31,34]. Other mechanisms include linear and non-linear programming techniques as used in [13,15].

However, whether the optimal price is achievable is still under debate. For example, in [10], Shenker et al. argue that utility functions could not be well defined in short-term and sometimes even very difficult in a long-term time scale. Therefore, the effectiveness of such schemes is still questionable.

Furthermore, optimal pricing schemes require a centralized architecture to collect all the required information and perform an optimal price setting or resource allocation. Hence, most of the optimal pricing schemes do not scale well. Also, it is not feasible to adopt a centralized approach when multiple domains are involved. If the centralized approach is used only within a domain, then it is not clear if the locally optimal solution will lead to a globally optimal one.

2.2. Simplicity and scalability

As explained above, although economic efficiency and optimality have been the main focus for a lot of studies, many researchers argued that optimal pricing cannot be achieved in practice. These concerns resulted in a shift of research focus to the Simplicity and Scalability issues. Those issues may include architectural aspect, signalling overhead, and implementation feasibility.

In [10], Shenker et al. suggest an edge pricing scheme that charges users based on the estimated path and estimated cost so that pricing is pushed to the edge. In [12], a resource negotiation and pricing (RNAP) protocol and a distributed price setting strategy are proposed to support pricing in a large scale network. In this framework, each flow negotiates the price and transmission rate through a RSVP-like signalling protocol. In [15–17], authors take advantage of the explicit congestion notification (ECN) bit in the packet header and propose pricing schemes that charge users based on the ECN marks. In [6], Odlyzko proposes the PMP pricing scheme where the network is partitioned into several logical subnetworks and a different fixed price is charged for each subnetwork. Odlyzko also argues that one should strive for maximal simplicity even at the expense of maximal efficiency in use of the network capacity. All these schemes try to address the implementation side of the pricing problem by emphasizing simplicity and scalability. We believe that a flexible and scalable pricing architecture is crucial especially when the problem is of the scale of the Internet. We share the same concerns with these authors and propose a pricing scheme that emphasizes the simplicity and scalability issues.

2.3. Quality of service

Last but not the least is the Quality of Service aspect. Since price is such an important economic incentive to users, it is often considered as an effective mechanism for traffic management. Indeed, many proposed pricing schemes entail either congestion control or admission control or even both. The relationship between pricing and these two traffic management functions is discussed in the following sub-sections.

2.3.1. Congestion-sensitive pricing and user adaptation

Several approaches assume that users are rational regarding the price signals and use the pricing as a main mechanism for congestion control. When congestion occurs, extra costs are charged in order to address the externality issue (i.e., the impact that transmitting a packet may have on other user's traffic during the congestion). Since users are expected to react to the price signals, congestion-sensitive pricing schemes often emphasize user adaptation where users adjust their sending rate in case of congestion or price change.

Some approaches that fall into this category can be found in [7,11,12,14].

However, in a strict sense, congestion-sensitive pricing does not address the QoS guarantee. When congestion occurs, QoS is no longer guaranteed. To provide a better QoS guarantee, what we want is to avoid the congestion, not to act after the congestion occurs. A proactive approach is preferred to a reactive approach in this case. Furthermore, QoS guarantee is a commitment that service providers made to the end users. Asking users to adapt to a price change or even terminate the service is undesirable. We believe that a proper interpretation of user adaptation in a DiffServ environment is the choice of a service class at the beginning of the service session rather than adjusting user sending rate in the middle of the service session. In other words, an elastic request would likely choose a lower class of service while an inelastic request may choose a higher service class if the budget is sufficient. If the budget is not sufficient, then the request can either lower the service class requirement (if it is tolerable) or decide not to enter the network at all. Of course, users are free to adjust their sending-rate during a service session but this should not be mandatory.

2.3.2. Admission control and pricing

Admission control (AC) is used to control the network load by restricting the access to the network and hence improve the level of QoS guarantee. Admission control approaches can be categorized in a number of ways such as parameter-based approaches versus measurement-based approaches and edge/end-point admission control versus hop-by-hop admission control.

Parameter-based approaches assume some traffic pattern and try to maintain the aggregated resource consumption below the total capacity. They often lead to conservative resource allocation and hence low network utilization especially in the case of bursty traffic. Measurement-based admission control relies on the measurement of current network load and therefore responds faster to the network status and consequently improves the network utilization. Example approaches can be found in [14,17,18].

As being done in the IntServ/RSVP architecture, admission control is traditionally performed on a hop-by-hop basis [1]. Each intermediate network element along the path has to decide whether the new request can be accommodated or not and reserves resources accordingly. However, adding admission control functionality to all the core elements violate the DiffServ principle of leaving the core simple. End-point/edge admission control that pushes the admission control functionality to the edge of the network seems more suitable in this case and has a number of advantages over the hop-by-hop approach such as faster response time and less implementation overhead. Studies also show that simple admission control algorithms based on estimated or measured network status are generally robust [17]. Most of the approaches in this category use probing [19,22] or explicit congestion notification (ECN) [17,19] to convey the network status back to the end points. The admission decision is then made based on this feedback information. A comprehensive study on end point admission control can be found in [19].

However, so far, most of the studies consider the pricing and admission control as two separate management functions. In other words, admission decisions are made solely on the load measurement or estimation and have no direct relation with the pricing. Using price as a primary admission criterion has not been studied sufficiently. In [17], the authors suggest that the admission decision could be made based on the user's willingness to pay for the ECN mark. However, it is not very clear how users should pay for the mark (i.e., what is the price for the mark?). In [11,12], Wang and Schulzrinne presented a comprehensive pricing framework and a signaling protocol called RNAP that integrates the admission control, congestion control, and pricing for DiffServ networks. However, they focus mainly on the congestion-sensitive pricing and do not study the admission control sufficiently. Admission control in their framework is performed hop-by-hop and is independent from pricing. Our architecture has similarities with the latter one but also differs in a number of ways. These will be discussed in the subsequent sections.

3. Pricing architecture

3.1. Motivation and design choices

From the discussion so far and the implication of our view of user adaptation, it is more desirable to tie the pricing with admission control in a DiffServ environment. Indeed, in a QoS-enabled network, there are two major concerns from both user's and provider's points of views: are there enough resources available for a particular traffic flow and what is the price for this flow? These two questions are exactly what admission control and pricing try to answer. An architecture that integrates pricing and admission control is therefore very promising.

The design of our architecture is based on the following considerations:

- *Decouple the pricing for core networks from the access networks and end-users:*
  As discussed in [21,23], due to the scale and complexity of the Internet, a practical QoS implementation in the Internet is to deploy DiffServ in the core and give the access networks the freedom of implementing IntServ, DiffServ, or other QoS techniques. This implies that a tightly integrated single end-to-end pricing scheme is unlikely to be accepted. Furthermore, a large number of service providers are involved in the pricing of the Internet and each of them should have the choice of their own pricing scheme. A single pricing model that suits all is very unlikely. Due to these two observations, we devise a separate pricing scheme for the core networks that implement DiffServ technology.
- *Price should reflect the availability of the network resources in the core:*
  Congestion cost is introduced to address the externality issue. However, as mentioned above, the ultimate goal of QoS management is to avoid congestion. Therefore, it is more meaningful to charge users based on the remaining resources rather than the congestion cost, and hence tie the pricing with admission control instead of congestion control.
  Since we are more interested in the implementation side of the problem, we do not assume a well known utility function for setting the optimal price. Instead of trying to find the optimal price through solving an optimization problem, we model the system as a market and let the market force regulate the price and maintain a certain level of QoS guarantee. In other words, each network element sets the price merely based on its own load. The rarer the resource, the higher the price. We also follow a simple and intuitive rule about price setting: price changes very slowly when there are plenty of resources available and increases drastically when the resources become scarce. When the price is very high it simply indicates that there are few or no resources left for new requests.
- *Per-flow messaging is not acceptable in a large network such as the Internet:*
  In order to aggregate and convey this price information to the access networks or even the end-users, we need a flexible and efficient architecture to accumulate the price along the path. Wang and Schulzrinne proposed a signaling protocol called resource negotiation and pricing protocol (RNAP) [12] which accumulates the price that is set by each network element and negotiates the price and resource along the path. The major drawback of such an approach is the per-flow messaging. In order to obtain the price for each flow along the path, pricing messages are sent back and forth between the source and destination. Although the possibility of aggregating the messages has been investigated, control messages overhead remains significant. Our architecture avoids his problem by maintaining the global price tables for the core networks at the access networks. This reduces the number of pricing messages significantly.
- *Edge/end-point admission control fits well in a DiffServ environment:*
  Since the price reflects the availability of the resources in the core, admission decisions can mainly or even purely be based on the price (in this case, the price is the only admission criterion). Thanks to the decoupling of pricing for the core networks, we are able to maintain the global price table at the access networks via domain abstraction. This enables us to push the

admission control to the access networks. In our architecture, it is the end-user or access network that decide whether a flow should enter the network or not. Inter-domain admission control is also possible using a global price table.

3.2. Domain abstraction

There are two types of price tables in our architecture: the domain price table and the global price table. A price entry in the domain price table represents the price for a service class from one edge node to another edge node within a domain, where a price entry in the global price table represents the price for a service class from one domain to another domain. Maintaining price information at such a scale may sound very impractical at a first glance. However, by abstracting an entire domain into a single node in a multi-domain network, constructing a global price table becomes feasible. Fig. 1 depicts the concept of domain abstraction.

Semret et al. [9] also use a similar type of abstraction in their pricing scheme. A global market is created for all the domains to bid for the capacity in order to provide a QoS guaranteed service. In their abstraction, the overall capacity requirement in a domain is abstracted into a single bottleneck capacity. Although this is a valid assumption, some inaccuracy can be still introduced into the abstraction. In our abstraction, we do not make any simplification or assumption but rather accumulate the price for each ingress–egress path which will be used to build the domain and global price tables.

3.3. Domain price table

Once each network element has calculated the price locally, the price information is exchanged and aggregated over the entire domain. The ultimate goal here is to accumulate the price for each ingress–egress pair. To compute the total price for a ingress–egress pair, we need to know the route from the ingress to the egress. This requires some knowledge of the domain topology and routing table information within the domain. The choice of using a centralized or a decentralized approach depends on the routing strategy used in the domain.

If a link state routing approach such as OSPF or IS-IS is used, a centralized approach is preferred since all the information required to construct the domain price table is available immediately. The centralized approach uses a pricing station for each domain or autonomous system. The pricing station will communicate with all network elements within the domain and collect the price information. As a result, the pricing station will eventually maintain a price table for each ingress–egress pair within the domain. When an ingress–egress pair has multiple possible routes,

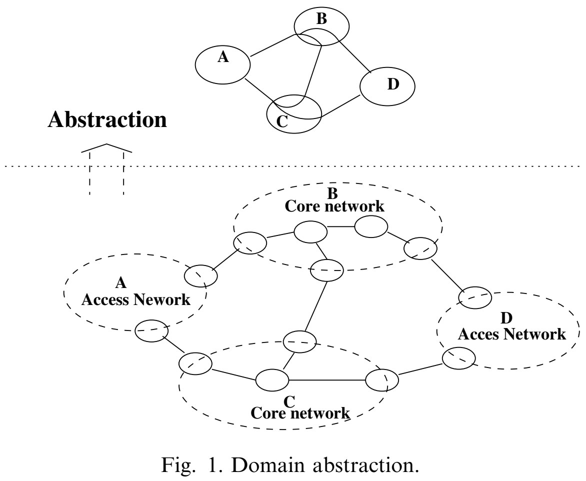


Fig. 1. Domain abstraction.

the domain routing table is consulted and the route from the routing table will be used.

Price setting in a network element can be enforced using a policy-based approach. In this case, a signalling protocol such as COPS can be used between pricing station (acting as the Policy Decision Point) and network elements (acting as the Policy Enforcement Point). COPS messaging can be used to instruct the network elements about the pricing policy. A direct application for that is the implementation of time of day pricing where the pricing station can change the pricing policy for the different time periods of the day and informs the network element when and how to set the price. Fig. 2 depicts the domain price table maintained in a centralized pricing station (PS).

It is relatively complicated if a distance-vector routing approach such as RIP is used within the domain. A distributed approach can be considered in this case. One approach is to add price as a metric into the routing table and modify the routing so that price accumulation will be performed automatically. However, the major drawback of this solution is to bind the price update with the route update. A route update implies a price update but not the other way around. This may generate significant overhead if the price update is more frequent than the route update. Furthermore, to propagate the price information among domains and construct the global price table, it's relatively easier when a global view of the entire domain (e.g., a complete set of ingress–egress pairs) is available. One naive centralized approach in this case is to ask each element to send its routing table to the pricing station whenever a route update occurs. The pricing station is then able to collect all the route information and generate the routes for each ingress–egress pair. A centralized pricing station approach is also in line with the Bandwidth Broker (BB) [32] approach. In fact, the pricing station could be part of the BB functionality.

We assume a pricing interval in our pricing architecture for both domain and global price tables. The update of the domain price table is not done periodically but rather change-driven to reduce the control-message overhead. The network element sends a price update to the Pricing Station only if there is a noticeable price change for that particular network element. A noticeable price change means that the change of price is larger than a pre-defined value.

Domain Price Table

Route			Price	
Ingress	Egress	Intermediate		
a1	a5	a2 , a4	class 1	45
			class 2	...
			...	...
a1	a3	a2	...	
...	...	...	...	

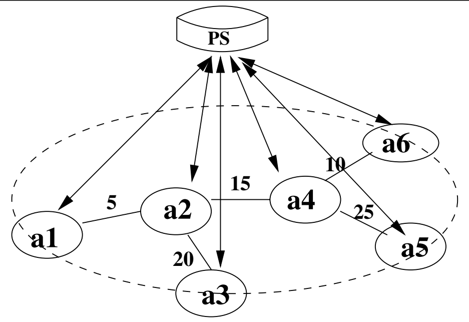


Fig. 2. A domain price table.

3.4. Global price table

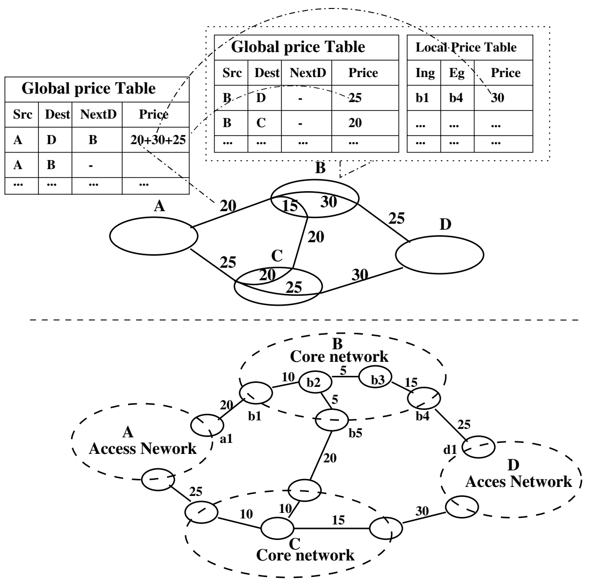
Because of the abstraction mentioned earlier, we are now able to construct the global price table by propagating the price among all of the pricing stations. The price for an ingress–egress pair inside a domain becomes the price for passing through the corresponding node in the abstracted global network. In this way, we can view the whole core networks as a single network that contains a limited number of nodes and links. Of course, further hierarchical decomposition can be applied if the size of global price table is still too large. In this paper, for the purpose of clarity, we assume only one level of abstraction and one level of global price table is constructed.

Unlike the domain price table, a global price table has to be constructed and maintained in a distributed manner. The update of the global price tables is also different. Periodical advertisement is used to propagate the price information in the same fashion as done in the Border Gateway Protocol (BGP). Each pricing station will advertise the price of a ingress–egress pair to its interested neighboring pricing station. Upon receiving such update information, each pricing station will update the global price table accordingly and propagate the update information to its interested neighbors at next pricing interval. For example, in Fig. 3, domain A and D are access networks. At time unit 1, D will first advertise the price of $b4 \rightarrow d1$ to B. At time unit 2, B will propagate the price to A as well as the price of $b1 \rightarrow b4$ and the price of $a1 \rightarrow b1$. From A's point of views, this is the price from A to D.

To deal with the issue of multiple routes through different domains, inter-domain routing tables are consulted during the price propagation. In this case, we believe that using the price of the best path in the routing table is sufficient. Notice that we do not require a complete view of the route because of the distributed nature global price table construction. Only the knowledge of the next domain or autonomous system is required. This could be easily obtained from the BGP routing table stored in the border gateways (assuming that BGP is used for inter-domain routing). In the example of Fig. 3, A will receive two prices for destination D ($B \rightarrow D$ and $C \rightarrow D$), but since the next domain or AS for destination D is B in the inter-domain routing table, the price from $B \rightarrow D$ is used in the global price table.

It is possible that paths in the routing table change after the advertisement had an impact on the pricing. An admission decision based on the price of a route may become invalid if the path in the routing table changes (packets will eventually take a different route than the expected one and upon which the admission decision was based). In this case, service quality may not be guaranteed especially if the new path does not have enough resources. Another concern is that the price agreed by user and service provider is no longer the real-price in the network.

One possible solution is to loosen the constraint on the service commitment and allow access networks to notify the end-users if the service should be terminated. If there are enough resources in the new route, then we believe that service providers should absorb this price difference because of the commitment they made to the end-users and simply continue the service. If the route does not have enough resources, since the price for the new path will be available "immediately", then further loosening of the service commitment is required so



Global price Table

Src	Dest	NextD	Price
A	D	B	20+30+25
A	B	-	
...	...	...	...

Global price Table

Src	Dest	NextD	Price
B	D	-	25
B	C	-	20
...	...	...	...

Local Price Table

Ing	Eg	Price
b1	b4	30
...	...	...
...	...	...

Fig. 3. Global price table in access networks.

that renegotiation of the service can be made. However, this will violate our interpretation of the user adaptation. We suggest that in this case, service providers should take the responsibility and terminate the service session without any charge. Note that even in the IntServ/RSVP approach, if a link failure happens, service quality can not be guaranteed unless re-negotiation and new resource reservation are enforced.

4. The market model

4.1. Pricing strategy

As discussed in Section 2, optimal pricing usually requires certain knowledge about user utility or demand to find the optimal price that either maximizes the user utility (based on the user utility function) or provider's revenue (based on the demand information). However, as mentioned before, most of the optimal pricing schemes require a centralized architecture. This is not scalable or even not feasible especially in an environment like the Internet. We believe that a distributed dynamic pricing setting strategy is more desirable in this case. Instead of trying to find an optimal price, we adopt the market-based approach which is to let the market force regulate the price based on the supply and demand. This type of approach is also studied in [24,27]. However, existing approaches often consider over-simplified network models (e.g., an end-to-end path has a single capacity, all traffic inside a domain traverse the same bottleneck link, or other assumptions that are often not very realistic) and focus on finding the optimal price in the market or optimal resource provisioning inside the network. Our approach shares the same market-managed philosophy but emphasizes implementation feasibility and scalability.

In our model, we assume each network element incorporates a load monitor so that price can be set based on its current load level. How to monitor and measure the network load in a router/switch is closely related to the different queuing and scheduling techniques used in a router/switch. If class-based queuing is used, then sampling the queue length could be a simple solution. We do not study the network monitoring in depth since it is out of the scope of this paper. In this paper, we assume the existence of a method to monitor the traffic load for each service class.

An iterative tatonnement process is used to set the price locally at a network element in several distributed pricing models [11,12,27]. In these models, price at time $t+1$ is a function of price at time t. However, price adjustment is usually some small constant, hence the price changes gradually and can not successfully reflect the real traffic condition inside the network. We believe that when the network is severely stressed the price should increase much faster. We follow an intuitive approach for price setting. Fig. 4 illustrates our price setting strategy in general.

It is worth noting that all the prices we mention in this paper are the price per bandwidth unit. In Fig. 4, P^i_{base} denotes the base price for service class i. This basic unit price reflects the equipment costs, maintenance/administrative costs, other service related costs, and revenue consideration for a network element. Let f^i be the fill factor for service class i. We will discuss how to set these two parameters latter in this section. When the load for service class i is lower than its fill factor, a static pricing can be used. In this case, the price is simply

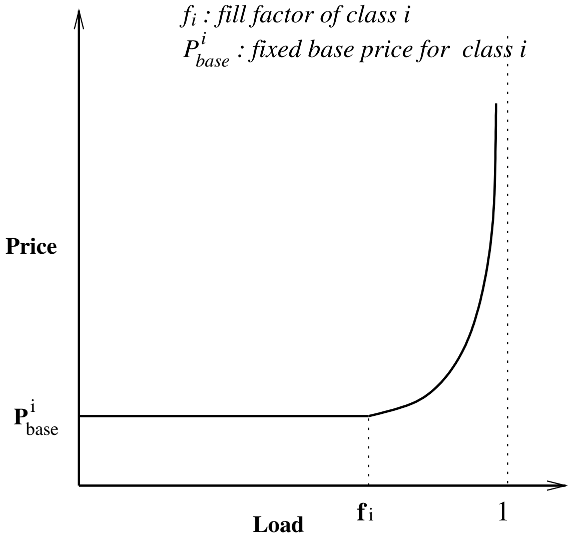


Fig. 4. General pricing strategy.

the base price P^i_{base} for service class i. When the load exceeds its fill factor the price will be increased rapidly and even dramatically when the load is close to the maximum capacity. We adopted the hyperbolic growth in this case to reflect our price setting strategy. When the network element is heavily loaded, the price skyrockets and only few users will be willing to accept the price.

Let $P_j^i(t)$ be the price at time t for service class i at link j:

$$P^i_j(t) = \begin{cases} P^i_{\text{base}} & \text{if } l^i_j(t) \leqslant f^i_j, \\ P^i_{\text{base}} \times \left(\dfrac{1-f^i_j}{1-l^i_j(t)} \right)^n & \text{otherwise,} \end{cases} \tag{2}$$

where $l^i_j(t)$ denotes the load of service class i at time t for a link j. Here n is a factor used to control the steepness of the price curve and $n \geqslant 1$. As we can see, the denominator is actually the residual capacity. For a particular path in the network, the total price $P^i_{\text{total}}(t)$ for traversing the path is the sum of the prices across all the links on the path:

$$\begin{aligned} P^i_{\text{total}}(t) &= \sum_j P^i_j(t) \\ &= \sum_g P^i_{\text{base}} + \sum_k P^i_{\text{base}} \times \left(\frac{1-f^i_k}{1-l^i_k(t)} \right)^n \end{aligned} \tag{3}$$

where $l^i_g(t) \leqslant f^i_g$ and $l^i_k(t) > f^i_k$.

$P^i_{\text{total}}(t)$ can be interpreted as two parts, where the first part is the sum of all the base prices of links that are lightly loaded, and the second part is the sum of all the prices of links that are heavily loaded. One desirable and important property of this pricing strategy is that whenever there is a bottleneck link along the path, the price for this bottleneck link will dominate the total price because of the hyperbolic pricing function. This pricing strategy will give users the right economic incentive and reflects the current network condition effectively since the prices for non-congested links do not have the same significance as the price for congested links. In any distributed price setting scheme, if the price at a network element changes gradually, it will fail to do so since the price change perceived by users will be smoothed out due price aggregation.

Our pricing strategy also works well with multiple bottlenecks because from the end-user's point of view the incentive difference between a very high price and an even higher price is not significant.

One can view the network pricing as users purchasing network resources along a path. In order to use the network, the users have to purchase all resources along this path. If a link is heavily loaded and no resources are available then it is equivalent to say that the entire path is unavailable. Hence, a very high price is reasonable and simply indicates the entire path is unavailable. This justifies the use of the hyperbolic pricing function in our strategy.

In practice, service providers can also arbitrarily set the price to infinity when the load or the price for a link reaches some threshold, this way indicating that the resources are out of supply. We will describe the user reaction model next and discuss the market model in the following sections in detail.

4.2. User reaction

Several research studies investigate the issue of user behavior upon price change. These include [13,20,23,24] and [27]. This issue is usually referred to as the demand function in the literature. For example, in [23], user reaction is modelled as a probability function where the probability of the user accepting a certain price for a certain service class is a function of current price versus the normal price. Another interpretation of this model is the percentage of users that will accept a certain price for a particular service class. We adopt this model in our work with some modification tailored for DiffServ networks as follows:

$$\begin{gathered} D[P^i_{\text{total}}(t)] = \mathrm{e}^{-\alpha_i \times \left(P^i_{\text{total}}(t)/P^i_{\text{total\_base}} - 1\right)^m}, \quad m \geqslant 1, \ \alpha_i > 0, \\ P^i_{\text{total\_base}} = \sum P^i_{\text{base}}, \quad P^i_{\text{total}}(t) \geqslant P^i_{\text{total\_base}}, \end{gathered} \tag{4}$$

where $P^i_{\text{total\_base}}$ is the sum of all base prices for class i across the path and m is a factor used to control the steepness of the demand curve. α_i is a constant that represents the price-sensitivity for service class i. The higher the base price of a service class, the lower the price sensitivity. This is reasonable because if a user is willing to pay a high price for a service, then it is likely that the user will be less price-sensitive than a user with relatively tight budget. Hence, different price sensitivity factors can be set for different classes. Note that when $P^i_{\text{total}}(t)$ equals to $P^i_{\text{total\_base}}$, the probability $D[P^i_{\text{total}}(t)]$ of accepting a price is 1. This is because the base prices of all service classes are assumed to be acceptable to all the users. Also note that in both very high and very low price regions, the probability of accepting a price does not change significantly. This is because the user may be somewhat indifferent to price change in these regions.

Therefore, the actual traffic load of a service class is simply the product of the offered load $Q_i(t)$ of service class i and the probability $D[P_{\text{total}}(t)]$ of accepting the price $P_{\text{total}}(t)$:

$$L_i(t) = D[P^i_{\text{total}}(t)] \cdot Q_i(t). \tag{5}$$

4.3. Market model

Now, having the supply function (3) and the demand function (5), we can then model the system as a market as shown in Fig. 5.

Note that this is slightly different from a traditional economic market model. In the network environment, the authors in [27] consider a constant supply function since they observe that Network Providers provide a fixed amount of bandwidth on a link. They focus on finding the optimal price and optimal resource provisioning for individual users so that all the resources are utilized and total user utilities are maximized. In their case, the optimal price is when the link is fully utilized. In our model, price is used to influence the user behavior for the purpose of traffic control. We form the market differently where the supply function is the used (actually supplied) capacity instead of the entire capacity. Fig. 5 gives an example of the supply and demand functions. Therefore, given an offered load $Q(t)$ we can see that there is an equilibrium price P' when the supply meets the demand. In a real network environment, the offered load changes over time. Hence, this results in a set of equilibrium prices over time. Fig. 5 shows that different offered loads cause the shift of the demand function and consequently generate a set of equilibrium prices and loads for a service class.

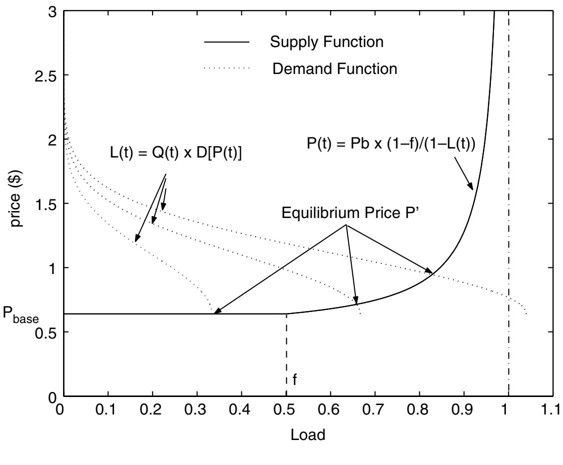


Fig. 5. Supply and demand in the market.

The stability of a market equilibrium has been well studied in economics [24–27]. The system is globally stable if it is continuous and the supply function is strictly upward sloping and the demand function is strictly downward sloping. This can be illustrated easily using excess demand analysis. When the price is lower than the equilibrium price, the excess demand is positive. This encourages the demand and therefore increases the price so that P moves towards P'. When the price is higher than the equilibrium price, the excess demand is negative, which decreases the demand and thus decreases the price so that P moves towards P' too. This means the price stability property [26] holds in a continuous model:

$$\lim_{t\to\infty} P(t) = P'. \tag{6}$$

This can be illustrated mathematically. For the ease of understanding, we show it for a single link case and simplify the equation in Eqs. (2) and (4). Reformulating the supply and demand function in our model we obtain:

$$\text{Demand}: \quad q^d(p) = Q_i \cdot \mathrm{e}^{-\alpha_i \times \left(P_i/P^i_{\text{base}}-1\right)^n}, \tag{7}$$

$$\text{Supply}: \quad q^s(p) = 1 - \left(\frac{P^i_{\text{base}}}{P_i}\right)^{1/n} * \ (1-f^i). \tag{8}$$

Notice that Eq. (8) is the inverse function of Eq. (2) when $l_j^i(t) > f_j^i$. Since when $l_j^i(t) \leqslant f_j^i$, the supply function is a constant, it will not affect the model. As stated in [16,24,25], the stability condition for a single market environment is:

1. $q^s(p)$ and $q^d(p)$ are continuous, and when $p = p'$, $q^s(p) = q^d(p)$.
2. $q^s(p)$ is strictly increasing and $q^d(p)$ is strictly decreasing.
3. $\frac{\mathrm{d}}{\mathrm{d}p}(q^d(p) - q^s(p)) < 0$.

It is easy to check that Condition 1 and 2 are true, and it implies the existence of a unique equilibrium in our system. Condition 3 can also be verified easily since in our system

$$\frac{\mathrm{d}}{\mathrm{d}p}(q^d(p)) < 0 \quad \text{and} \quad \frac{\mathrm{d}}{\mathrm{d}p}(q^s(p)) > 0,$$

$$\frac{\mathrm{d}}{\mathrm{d}p}(q^d(p) - q^s(p)) = \frac{\mathrm{d}}{\mathrm{d}p}(q^d(p)) - \frac{\mathrm{d}}{\mathrm{d}p}(q^s(p)) < 0.$$

It is worth noting that the equilibriums are not Pareto optimal in the traditional sense since not all the resources will be utilized. However, the idea underlying the DiffServ Model is to achieve service quality differentiation in an aggregated manner rather than quantitative guarantee for individual flows. As suggested in [6], one easy and feasible way to achieve this is to expect different traffic loads for different service classes so that classes with lower traffic load will have relatively higher service quality. A static differential pricing (i.e., PMP pricing) scheme was also proposed to attract different traffic loads for different classes. Although this approach is simple and easy to implement, the service quality guarantee is weak. Our approach uses a dynamic differential pricing scheme and aims at providing better QoS guarantee by means of an integrated admission control.

As shown in Fig. 5, the equilibrium price in our system is higher then the equilibrium price when constant supply function (vertical line at load 1) is considered. From economic point of view, this can be explained as users paying some extra charge for the unused capacity to obtain a relatively higher QoS level.

Now let us consider how service providers can set the values of P_{base}^i and f^i based on this model. Using the market model, and assuming the demand function is known, by adjusting the values of P_{base}^i and f^i, different equilibrium prices and loads of a service class can be obtained. In other words, different P_{base}^i and f^i will result in a different resource utilization. We use a heuristic approach for the mapping between the load of a service class and QoS guarantee (may include delay and packet loss probability) can be obtained from historical data or statistical analysis. Therefore, by selecting a desirable equilibrium traffic load, service providers can set these two variables easily. Fig. 6 shows that when user demand function is known, then by setting the values of P_{base}^i and f^i, service providers can set different expected traffic loads for different classes and maintain different QoS levels.

Although our goal is not to find an optimal pricing solution, our approach does not exclude the possibility of using an optimal pricing model to set the base price P_{base}^i. For example, we can use the optimal pricing and resource provisioning model used in [13]. In our model the link capacity used in the optimization model will not be the total capacity C but the $C * f^i$. Since P_{base}^i is a constant and can be computed offline, the optimization model used to find P_{base}^i can be based on long-term strategy and historical data hence avoiding scalability issues.

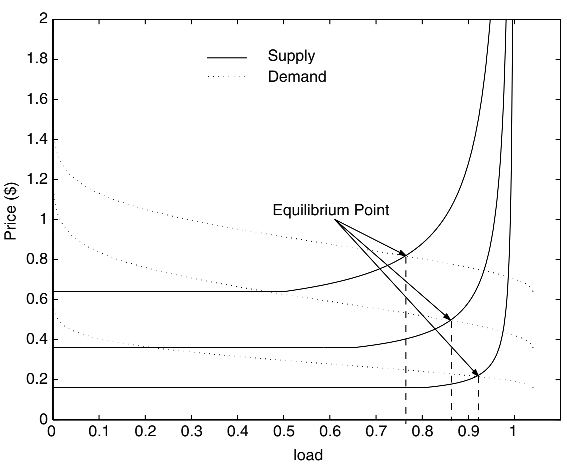


Fig. 6. Equilibrium prices and loads for three classes.

There are two parameters, response time and pricing interval, that are implicitly defined in the model. The value of these two parameters affect the stability of the system. Response time is the time taken to convey the price signals back to users plus the time for users to react to the signals. Pricing interval is how often a price will be updated or recalculated. If the response time is too large, then users may react to a signal that is no longer true. Similarly, if the pricing interval is too large, then again it would not be able to reflect the network status successfully. However, since our pricing scheme is targeted for the core networks, the aggregation of traffic in the core and the ability of implementing very dynamic pricing schemes allow us to assume that these two parameters can be sufficiently small so that the current network status can be successfully conveyed through pricing and hence the system should remain stable.

Moreover, even if the model is indeed continuous and the market equilibrium is stable, the change of offered-load will cause the shift of equilibrium points and consequently price fluctuation as well. Choosing appropriate dynamic pricing intervals in relation to response time and price stability requires extensive real-world traffic analysis and is out of the scope of this paper. Here, we focus more on the study of using pricing as an effective traffic management mechanism. In the following sections, we will focus on admission control and end-to-end pricing to show that our pricing scheme is indeed an effective and efficient mean to provide better QoS guarantees.

5. End-to-end pricing and admission control

5.1. Admission control and class promotion

Another useful property of our system is that our price function ensures that the load $l_j(t)$ at link j never reaches 1:

$$\lim_{l^i_j(t)\to 1} P^i_j(t) \to \infty. \tag{9}$$

This property also holds for any end-to-end path as well since if there is a bottleneck link that is heavily loaded, then its price increases hyperbolicly towards infinity. Hence, the price for the particular link will dominate the price for the entire path. In other words, The following property holds as well:

$$\lim_{l^i_j(t)\to 1} P^i_{\text{total}}(t) \to \infty \text{ for any link } j \text{ along the path.} \tag{10}$$

This property is ideal for our distributed pricing and admission control scheme. More specifically, we are able to set the prices in a distributed fashion and the aggregated price still reflects resource availability inside the network effectively. Unlike traditional admission control approaches, the decision makers in our model are not the service providers but the end-users and the decision varies with the current price since the latter reflects the resource availability.

As shown in Fig. 6, service differentiation is realized by using our pricing scheme and the market model to obtain class traffic load differentiation. However, it is possible that a lower service class be very stressed and the price for this service class becomes higher than an upper service class. In this case, we propose what we call a class promotion where a request is promoted to a higher class of service. A similar load balancing technique has also been used in [20] but the criterion used to promote a request in their approach is whether there are available resources in other classes. However, to check if there are available resources for all links along the entire path for every request is difficult and not scalable. In our model, the price itself provides enough information and the class promotion is seamless and simple. Notice that a request will be promoted only to higher classes even if lower service classes are lightly loaded. This is because the lower service classes might become heavily loaded as well in the future and we do not want to sacrifice the service quality of the admitted request.

Based on the previous consideration, our admission control algorithm consists of the following steps as illustrated in the Fig. 7:

5.2. End-to-end pricing

One of the main advantages of our pricing framework is that it enables us to focus on the

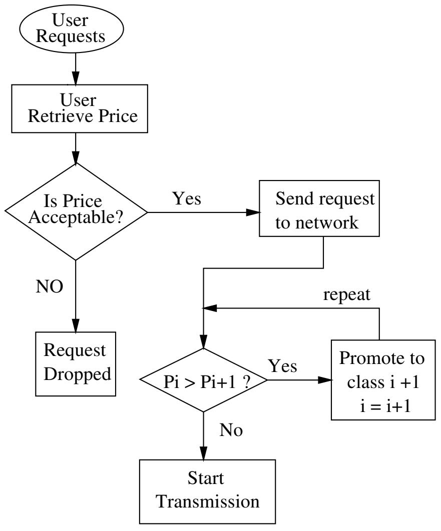


Fig. 7. Admission control flowchart.

pricing in the core networks only and pushes the end-to-end pricing to the access networks. Since the pricing here is restricted to the core networks and global price tables are available at the access networks, the pricing of the access networks can be implemented without involving the core at all. This eliminates possible scalability problems caused by pricing and admission control signaling and the need for keeping per flow state information inside the core. It also gives access networks the flexibility of implementing an end-to-end pricing scheme at different protocol layers. For example, a light-weight signaling protocol can be used between the sender and receiver access networks without any involvement of the core nodes. This protocol can be implemented at the network layer like the pricing for core networks or even at upper layers such as the transport or the application layers.

Various pricing schemes can be applied to achieve end-to-end pricing. For example, the access network can implement a flat-rate pricing or a time of day pricing for simplicity and predictability. In such case, the cost of accessing core networks can be absorbed by the access network and it is the access networks that choose the appropriate service class for the user traffic based on some policy defined by the access provider. Alternatively, access networks can charge users for the reserved resources if IntServ is implemented or use any dynamic pricing scheme such as the one proposed in this paper. In the next section we study one example of static pricing, Time Of Day pricing, which can be implemented at access networks.

5.3. Example scenario: Time of Day (TOD) pricing

The main idea of Time of Day (TOD) pricing is to charge users different fixed prices at different time periods during a day (typically, high price at peak-time and low price at off-time). This encourages the users to use the network rationally. It is essentially a compromise between flat-rate pricing and dynamic pricing in that it can provide incentives to the users and at the same time allows for price predictability as in flat-rate pricing. However, the incentive provided by TOD pricing is on a long-term time scale (e.g., hours). It does not address the QoS guarantee issue in particular.

In this section, we show a pricing framework where TOD pricing is used at the access networks and our dynamic pricing is used at the core and where both service providers and users requirements can be satisfied. The main components of this framework are described in the following:

- Core networks implement our dynamic pricing scheme where domain and global price tables are maintained as described in our pricing architecture. A price table that contains the prices for each service class using the core networks will be available at access networks. This enables a fast admission-control decision-making process at the edge.
- Access providers study the characteristics of user traffic and divide a day into a set of time segments. Each time segment with an average offered-load (with small variation) will correspond to an equilibrium price in the market model.
- A static pricing scheme is implemented where an end-to-end price based on the equilibrium price will be posted to end-users. Users will then be charged according to this price at the corresponding time segment. This set of TOD prices

is updated periodically (e.g., weekly or monthly).

- Access networks need to absorb the price difference between access networks and core networks. Being the users of the core networks, access networks perform the admission control shown in Fig. 7 with some minor differences as shown below:

```
service providers retrieve the price
from the global price table;
if ($P_i(t) > P_{\text{threshold}}$)
   service providers reject the request;
else
   while ($P_i(t) > P_{i+1}(t)$)
      promote request to class $i + 1$;
      $i = i + 1$;
   endwhile
endif
```

Note that access network providers can set a price threshold $P_{\text{threshold}}$ for the use of core networks. One rule for setting this price threshold could be that the service provider's profit must be greater than or equal to zero. In other words, the revenue generated by admitting a request must be at least the same as the cost of using the core networks so that service providers would not lose money.

Also, in this scenario, only the access network providers are involved in the admission control process where in the original admission control scheme both users and service providers are involved.

As we can see from this example scenario, our pricing framework allows integration of a dynamic pricing scheme of the core and a relatively static pricing scheme at the edge thanks to the separation of core pricing from access pricing. The core pricing scheme provides an effective control mechanism to service providers. In turn, the access pricing scheme offers simplicity and price predictability to end-users. This way the requirements of both providers and users are satisfied.

6. Simulation

6.1. Simulation model

In order to study the behavior of our pricing strategy, we setup a DiffServ network environment using the *ns-2* simulator. We have modified the DiffServ component developed by Nortel Networks to incorporate our pricing and admission control mechanisms. Since the goal of the simulation is to evaluate our price setting strategy and admission control scheme, we only simulate a single DiffServ domain and do not emphasize the construction and maintenance of the domain and global price tables.

Fig. 8 illustrates the network topology used in our simulation which consists of three core routers and eight edge routers in a single DiffServ domain. Three core routers implement the dsRED core queue which has no policing and marking functionality but only PHB forwarding. All edge routers implement the dsRED edge queue which

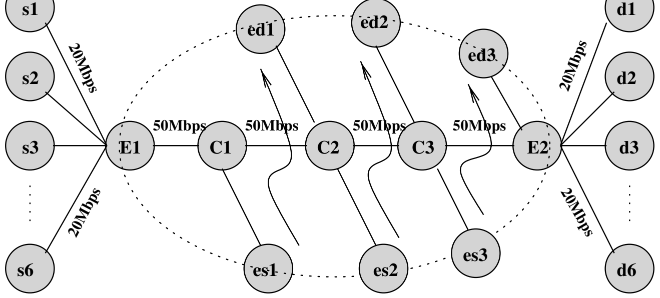


Fig. 8. Simulation network topology.

supports the DiffServ packet classifying, marking, and policing. Edge routers act as the pricing stations and handle user requests generated by the sources. Six source–destination pairs are configured in our simulation. Additionally, six extra edge routers are configured inside the DiffServ domain to create cross-traffic and simulate bottlenecks at links C1C2, C2C3, and C3E2. All the sources in the simulation implement our pricing and admission control algorithm so that the entire domain forms the system described in previous sections.

The capacity of each link from source nodes to the edge of the DiffServ domain is set to 20 Mbps. All links within the DiffServ domain are set to 50 Mbps. Propagation delay for all the links is set to 5ms. All links are full duplex outside the DiffServ domain and DropTail queue management is used. Inside the DiffServ domain, only the links that connect core routers are full duplex and the rest of the links are simplex links because different type of dsRED queuing techniques are used in different directions. Weighted Round Robin (WRR) scheduling is used at each link to manage the physical dsRED queues. In our simulations, we consider three service classes and the weights for the three classes are distributed as 3, 3, and 4, and the fill factor f^i for each class at all the links is set to 50%, 65%, and 80% respectively. The base prices P^i_{base} for each class are set as 0.09($), 0.06($), and 0.04($) per time unit respectively and the pricing interval is set to 2 s. A pricing agent is attached to each link in order to set the price locally and the agents communicate with each other to propagate the price information back to the edge. To simplify the result analysis, we set the steepness factor n in our price function to 2, the steepness factor m in our demand function to 2, and the price sensitivity α to 1 initially (see Sections 4.1 and 4.2).

For each class, there are two types of traffic sources in our simulation. CBR and Pareto on/off traffic are generated independently and flow requests are modelled by a Poisson arrival process. Note that, in our model, flows do not retry if the request is dropped. The holding time for each flow is exponentially distributed with a mean value of 100 s. The packet size for both CBR and Pareto traffic is set to the default value 210 bytes and the peak rate for CBR is 125 Kbps. For the Pareto traffic, the shape parameter is set to 1.5, where the on and off times are both set to 500 ms, and the peak rate is set to 125 Kbps. The rest of the parameters are set to the default values used in *ns-2*.

To simulate the bottlenecks inside the network, we set up three cross-traffic flows inside the DiffServ domain so that we can control the load of each link and the number of bottlenecks we can simulate. For the ease of result discussion, we let C2C3 be the link that always has the highest load even in the multiple bottlenecks situation. This is achieved by using CBR traffic source for the cross-traffic at C2C3 and the rate of the C2C3 is always the highest among all the cross-traffic inside the domain. We use Exponential on/off traffics for

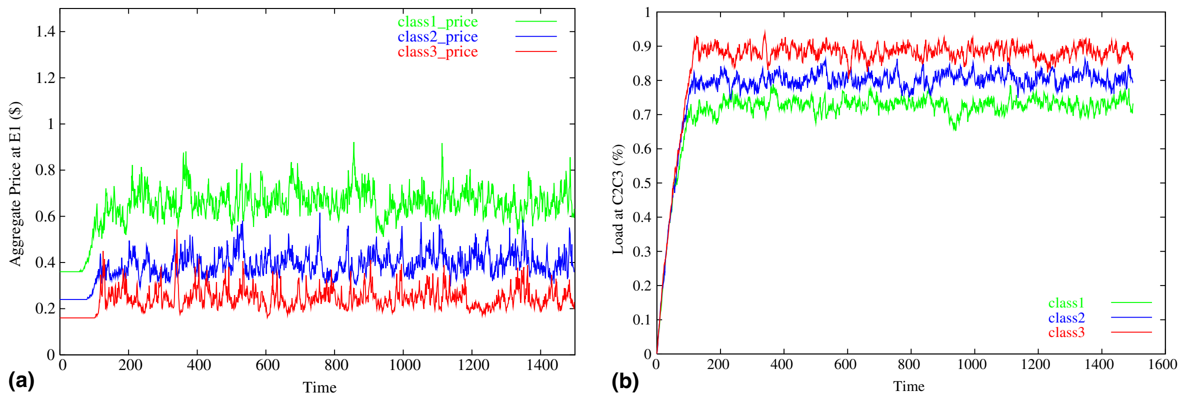


Fig. 9. Single bottleneck case with 1.2 offered-load: (a) total price at E1 and (b) load at C2C3.

cross-traffic at C1C2 and C3E2 to simulate highly dynamic network conditions.

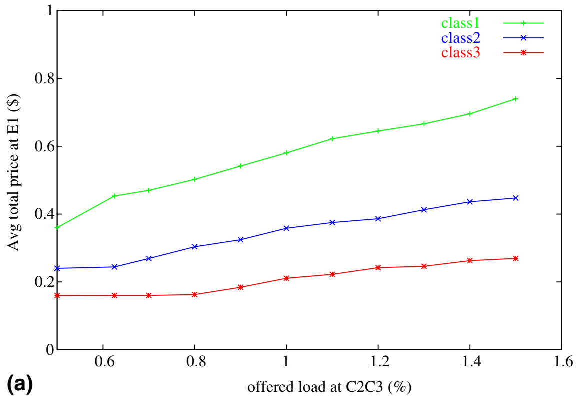
In our simulation, the goal is to price the resource according to the traffic load and control the load level for each class using our pricing and admission control approach. Therefore, we mainly concentrate on metrics such as traffic load at bottlenecks, user request blocking ratio (i.e., the number of requests being rejected or dropped versus the total number of requests), and advertised total price.

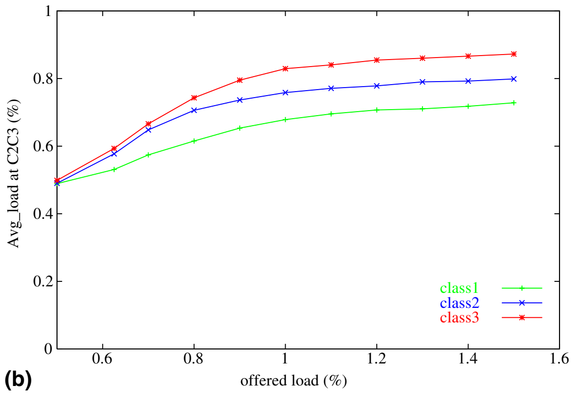
6.2. Analysis

6.2.1. Single bottleneck

We first consider the case of a single bottleneck to examine the basics of our approach. The rate of cross-traffic at C2C3 is set to at least twice of the traffic at C1C2 and C3E2. Therefore, C2C3 is the only bottleneck along the path.

We conduct the simulation for all classes at different offered loads ranging from 0.5 to 1.5 with a step size of 0.1. When the offered load is less than the class fill factor f^i, no price change occurs and hence all requests are accepted. When the offered load exceeds f^i, the market force regulates the price and hence maintains a certain price and traffic load in the network. Fig. 9(a) gives the aggregate price observed at E1 for each class with 1.2 offered load for each class and Fig. 9(b) shows the traffic load at the bottleneck link C2C3 with 1.2 offered load for each class. These two figures illustrate how the system behaves given a certain offered-load. Clearly, the system is well protected from the price oscillation and the traffic load is well controlled at the bottleneck.





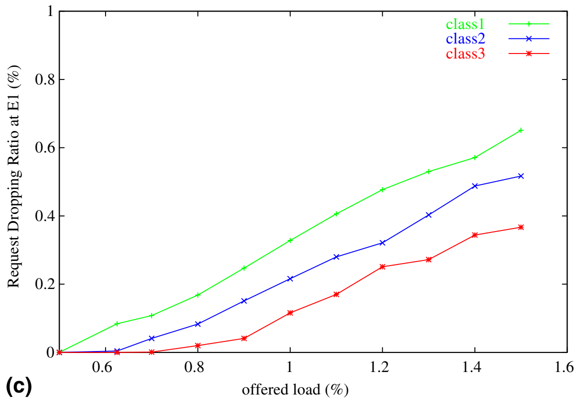


Fig. 10. Single bottleneck case: (a) average total price at E1 versus offered-load, (b) average load at C2C3 versus offered-load and (c) request dropping ratio versus offered-load.

For each offered load, the system has similar behavior at different equilibrium points. Fig. 10(a) shows the average total prices for different offered loads and Fig. 10(b) shows the average loads at the

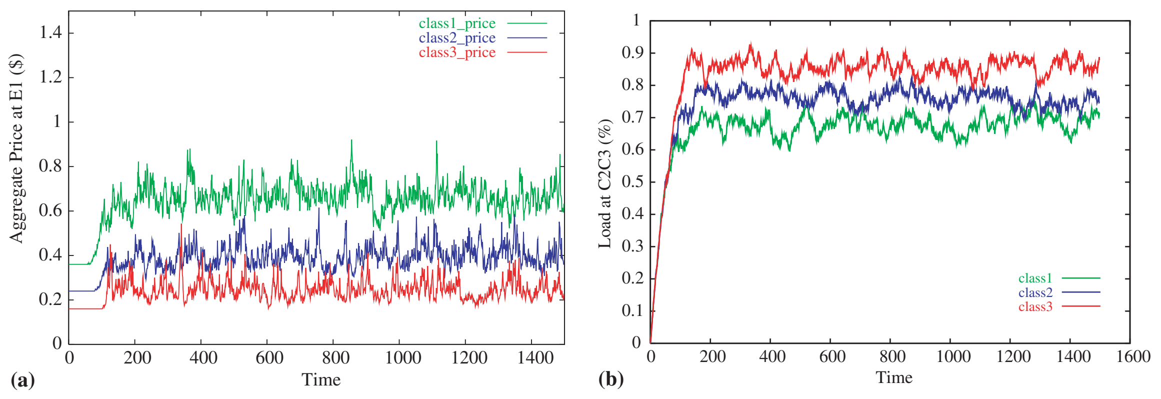


Fig. 11. Three bottlenecks case with 1.2 offered load: (a) total price at E1 and (b) load at C2C3.

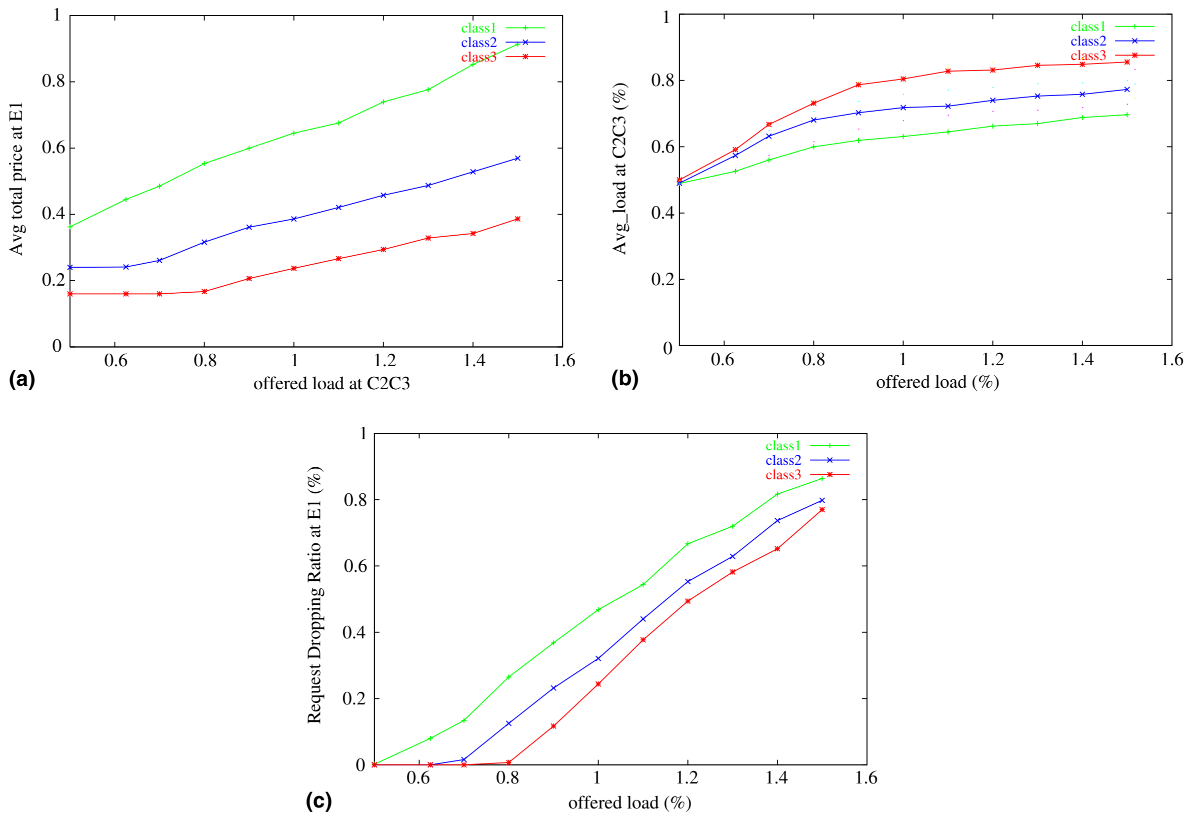


Fig. 12. Three bottlenecks case: (a) average total price at E1 versus offered-load, (b) average load at C2C3 versus offered-load and (c) request dropping ratio versus offered-load.

bottleneck C2C3 versus different offered loads. As we expected, the average price of a service class increases when the offered load increases. However, thanks to our price setting strategy and property (9), the increase of the average load at the bottleneck is very small, and the amount of increase declines as the offered load increases. This is especially desirable for the purpose of controlling the traffic inside the network. Fig. 10(c) depicts the ratio of requests that are dropped by users due to high prices versus the offered load. As the offered load increases, the blocking ratio increases as well so that the system remains in a controlled state.

6.2.2. Three bottlenecks

To simulate multiple bottlenecks inside the network, we set the sending rate of the other two cross-traffics (es1 and es3) to approximately the rate of es2. In this way, when the network is heavily loaded, there will be three bottlenecks for the traffic originating from E1.

Figs. 11 and 12 show the same set of diagrams as in the single bottleneck case. Fig. 11(a) and (b) show that even if there are multiple bottlenecks, the system is still protected with well controlled loads and prices.

Fig. 12(a) and (b) respectively depict the aggregate price at E1 and load at C2C3 for different offered-loads. Since the system property (10) holds for any number of bottlenecks, we expect the system to have a similar behavior as in the one bottleneck case. We observe a clear difference of average total prices at E1 in the two scenarios. However, the difference of average loads at C2C3 is very small. This indicates that the system is still well controlled and robust.

Fig. 12(c) gives the request dropping ratio in the three bottlenecks scenario. Notice that the request dropping ratio here is much higher than the one in the single bottleneck case. This is because in our simulation the path originated from E1 consists of four hops where three of them have very high prices (bottlenecks), and the path originated from the extra sources consists of three hops with only one of them being a bottleneck. Therefore, the dropping ratio at E1 will be much higher than the dropping ratio at extra sources. Consequently, this cause the dropping ratio at E1 in the three bottleneck case is much higher than the ratio in the single bottleneck case.

This actually illustrates a real network situation in that a flow that traverses several bottlenecks has more impact on the network and hence should be charged for a higher price than flows traversing a single bottleneck. Many oversimplified network models (for example, a single capacity or a single path abstraction for an entire network) can not capture this kind of behavior but our system does.

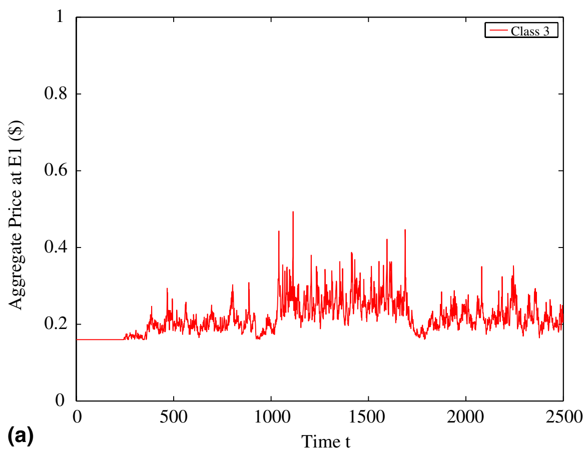
6.2.3. System dynamics and class promotion

In order to study the dynamics of our system and the effect of class promotion, we consider several scenarios where the offered load for a service class (in this case class 3) is initially relatively small (0.9 for class 3), and then at time 1000 s during the simulation increased to 1.5. At simulation time 1700 s, the offered load is set back to its initial value (i.e., 0.9).

Fig. 13(a) and (b) illustrate the behavior of the system at this situation where the price and load stabilize after a period of time and then at time 1000 s both start to change and reach another stability point. The amount of change between these two stability levels is not significant. This is due to the fact that the system is well protected and the difference of average load at C2C3 between offered load 0.9 and 1.5 is only about 0.07 (from Fig. 10(b)). However, from these two diagrams we can still observe that the system changes from one equilibrium state to another one.

To study the effect of class promotion, we create a scenario such that class 2 and 3 are initially lightly loaded. Then we increase the offered load of class 3 to 1.5 at test time 1000 s. We first consider a case without class promotion. From Fig. 14(a) and (b), we observe that class 3 will reach another equilibrium point and the prices for class 3 and class 2 overlap if no class promotion is used.

We then enable the promotion mechanism and repeat the simulation. From Fig. 15(a) and (b) we can see that the load balancing behavior between class 3 and class 2 indeed increases the load for class 2 and new equilibrium points are reached for both classes. Since in this case the load of class 1 is initially high already, no promotion from class 3 to



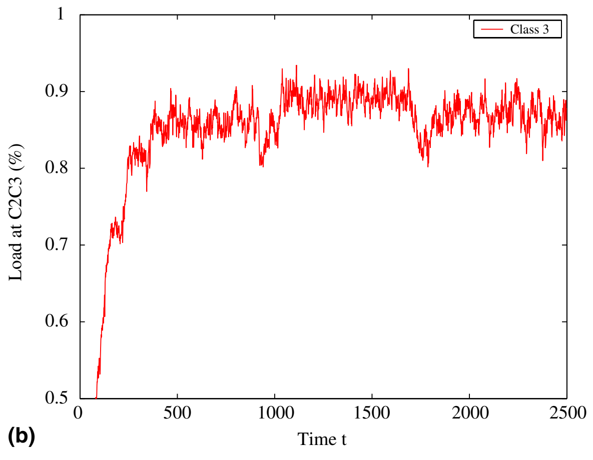
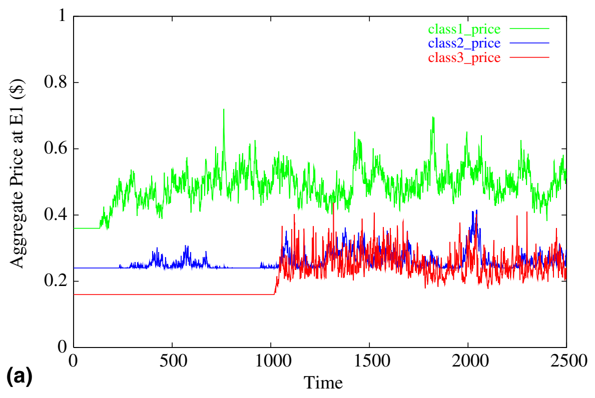


Fig. 13. System dynamics: (a) total price at E1 and (b) load at C2C3.



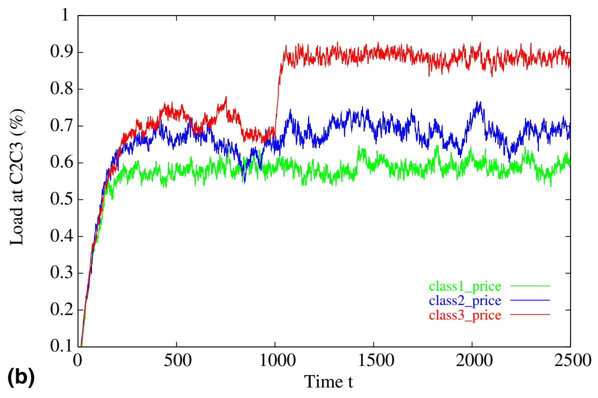
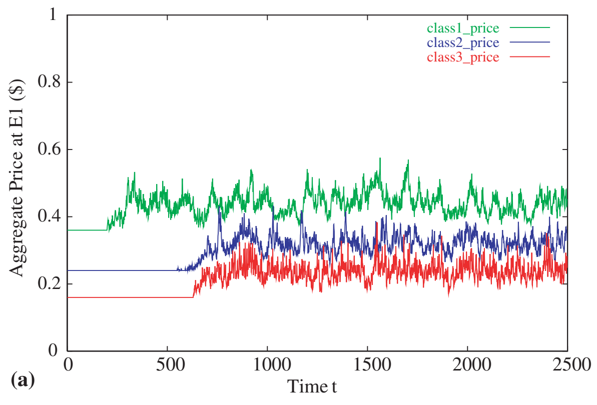


Fig. 14. Without class promotion: (a) total price at E1 and (b) load at C2C3.



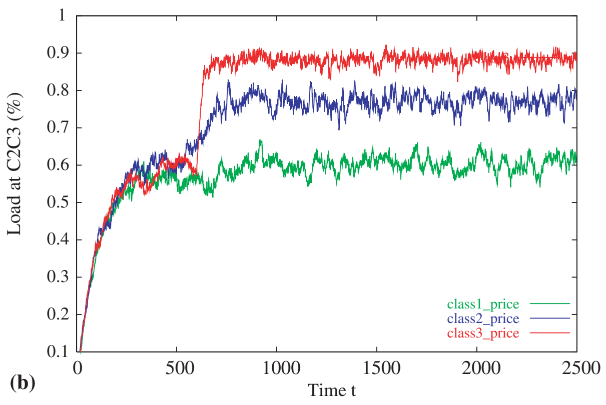


Fig. 15. With class promotion: (a) total price at E1 and (b) load at C2C3.

class 1 occurs but this can be expected too if class 1 is also lightly loaded.

7. Conclusion

In this paper, we have presented our integrated pricing and admission control approach for DiffServ networks. In particular, we have proposed a tariff-based pricing architecture that separates the pricing for core networks from end-to-end pricing through domain abstraction and maintains domain and global price tables at the access networks. The motivation of maintaining the global price table at the access networks is to enable an accurate and fast decision-making process and avoid the per-flow signalling overhead. We have also described our pricing scheme and adopted a market-based approach that uses price to regulate the traffic load for service differentiation and QoS guarantee. Since the price in our scheme effectively reflects the resource availability in the network, it is used not only as an economic incentive but also as a mean for admission control and load balancing.

Our architecture is flexible in that end-to-end pricing is decoupled from the core networks. It is scalable thanks to the domain abstraction and distributed nature of our pricing scheme. Admission control is pushed to the edge and no per-flow based messaging for either pricing or admission control is needed. This way, our architecture is in line with the philosophy of edge-pricing [10] and PMP pricing [6] but allows for better QoS support. Maintaining price tables for core networks also enables the split of revenue among service providers.

The simulation results shown that our pricing strategy is indeed effective in terms of providing the right economic incentives to the users and maintaining a certain level of traffic load for different service classes. Thanks to the class promotion used in our scheme, traffic load can be balanced among service classes seamlessly.

Also, as shown in the example of Time of Day (TOD) pricing, our framework provides a feasible and effective way for realizing a dynamic short-term pricing scheme inside the core and a relatively static and predictable long-term pricing scheme at the edge, and at the same time achieves better QoS guarantees through a simple and distributed pricing and admission control scheme.

All in all, in this paper, we have presented a simple but effective and scalable approach for realizing pricing and admission control in QoS-enabled DiffServ networks. We believe that with proper domain abstraction and careful price setting strategies, the benefit gained from maintaining the price tables can certainly overweigh the overhead it introduces.

As discussed at the beginning of this paper, pricing for QoS-enabled Internet is a very difficult problem due to the size and multi-domain nature of the Internet. Although we strived to address all issues related to the provisioning of a complete pricing scheme from all aspects of economical optimality, implementation feasibility, and QoS support, there is still space left for improvement. The following is a discussion of some open research issues:

One issue that has not been addressed in our scheme is fairness. Fairness (either proportional or max–min) is typically easier to achieve in the context of a centralized pricing and resource allocation architecture. How to incorporate the fairness aspect into our pricing strategy and address the fairness issue in a distributed architecture like the one proposed in this paper requires further investigation.

Another issue concerns the relationship between the pricing interval and the response time and how this will affect the effectiveness of a pricing scheme. This issue has not been studied sufficiently in the literature. For example, in the studied scenario, how to choose the time segment used for Time of Day (TOD) pricing properly and what is the relationship between this time segment and the short-term pricing interval?

Another possible future work is to investigate the applicability of our architecture in other contexts: For example, QoS routing using price as a constraint may be an interesting application of our pricing architecture since the price is available immediately and reflects the availability of resources in the network. The pricing architecture proposed in this paper supposes a routing approach as used in the current Internet. The relationship between pricing and QoS-routing has not been studied thoroughly and how to integrate them together properly requires further investigation.

Acknowledgements

The authors would like to thank Sonia Waharte and Jin Xiao for their useful comments and discussions.

References

[1] S. Shenker, C. Partridge, R. Guerin, Specification of Guaranteed Quality of Service, RFC 2212, Internet Engineering Task Force, September 1997.
[2] S. Blake et al., An Architecture for Differentiated Services, RFC 2475, Internet Engineering Task Force, December 1998.
[3] T. Li, Y. Iraqi, R. Boutaba, Tariff-based pricing and admission control for DiffServ networks, in: Proceedings of the 8th IFIP/IEEE International Symposium on Integrated Network Management (IM'2003), 2003, pp. 73–86.
[4] M. Falkner, M. Devetsikiotis, I. Lambadaris, An overview of pricing concepts for broadband IP networks, IEEE Communications Surveys & Tutorials, 2nd Quarter, 2000, pp. 9–13.
[5] L.A. DaSilva, Pricing for QoS-enabled networks: a survey, IEEE Communications Surveys & Tutorials, 2nd Quarter, 2000, pp. 2–8.
[6] A.M. Odlyzko, Paris metro pricing for the Internet, in: Proceedings of the ACM Conference on Electronic Commerce, 1999, pp. 140–147.
[7] J. MacKie-Mason, L. Murphy, J. Murphy, Responsive pricing in the Internet, in: L.W. McKnight, J.P. Bailey (Eds.), Internet Economics, MIT Press, Cambridge, MA, 1997, pp. 279–303.
[8] J.K. MacKie-Mason, H.R. Varian, Pricing congestible network resources, IEEE Journal on Selected Areas in Communications 13 (1995) 1141–1149.
[9] N. Semret, R.R.-F. Liao, A.T. Campbell, A.A. Lazar, Pricing, provisioning and peering: dynamic markets for differentiated Internet services and implications for network interconnections, IEEE Journal on Selected Areas in Communications 18 (12) (2000) 2499–2513.
[10] S. Shenker, D. Clark, D. Estrin, S. Herzog, Pricing in computer networks: reshaping the research agenda, ACM Computer Communication Review 26 (2) (1996) 19–43.
[11] X. Wang, H. Schulzrinne, Pricing network resources for adaptive applications in a differentiated services network, in: Proceedings of IEEE INFOCOM 2001, Anchorage, AK, April 2001.
[12] X. Wang, H. Schulzrinne, An integrated resource negotiation, pricing, and QoS adaptation framework for multimedia applications, IEEE Journal on Selected Areas in Communications 18 (12) (2000) 2514–2529.
[13] N.J. Keon, G. Anandalingam, Optimal pricing for multiple services in telecommunications networks offering quality-of-service guarantees, IEEE/ACM Transactions on Networking 11 (1) (2003) 66–80.
[14] A. Ganesh, K. Laevens, Congestion pricing and user adaptation, in: Proceedings of IEEE INFOCOM 2001, Anchorage, AK, April 2001.
[15] F.P. Kelly, Mathematical modelling of the Internet, in: B. Engquist, W. Schmid (Eds.), Mathematics Unlimited, Springer, Berlin, 2001, pp. 685–702.
[16] B. Hansen, E. Nævdal Competition in the Internet and dynamic pricing by ECN marks, in: QofIS'2000 Workshop on Internet Charging, Berlin, September 2000.
[17] F.P. Kelly, P.B. Key, S. Zachary, Distributed admission control, IEEE Journal on Selected Areas in Communications 18 (12) (2000) 2617–2628.
[18] R. Mortier, I. Pratt, C. Clark, S. Crosby, Implicit admission control, IEEE Journal on Selected Areas in Communications 18 (12) (2000) 2629–2639.
[19] L. Breslau, E.W. Knightly, S. Schenker, I. Stoica, H. Zhang, Endpoint admission control: architectural issues and performance, in: ACM SIGCOMM 2000, Stockholm, Sweden, August 2000.
[20] E.W. Fulp, D.S. Reeves, Optimal provisioning and pricing of Internet differentiated services in hierarchical markets, in: Proceedings of the IEEE International Conference on Networking, 2001.
[21] V. Fineberg, A practical architecture for implementing end-to-end QoS in an IP network, IEEE Communications Magazine 40 (1) (2002) 122–130.
[22] B. Pang, H. Shao, W. Zhu, W. Gao, An admission control scheme to provide end-to-end statistical QoS provision in IP networks, in: 21st IEEE International Performance, Computing, and Communications Conference, 2002, pp. 399–403.
[23] J. Hou, J. Yang, S. Papavassiliou, Integration of pricing with call admission control to meet QoS requirements in cellular networks, IEEE Transactions on Parallel and Distributed Systems 13 (9) (2002) 898–910.
[24] A. Takayama, Mathematical Economics, Cambridge University Press, Cambridge, 1985, pp. 295–301.
[25] R.G.D. Allen, Mathematical Economics, Macmillan St. Martin's Press, New York, 1959, pp. 325–329.
[26] H.R. Varian, Microeconomic Analysis, Norton, New York, 1992.
[27] E.W. Fulp, M. Ott, D. Reininger, D.S. Reeves, Paying for QoS: an optimal distributed algorithm for pricing network

resources, in: Sixth International Workshop on Quality of Service (IWQoS'98), 1998, pp. 75–84.
[28] P. Marbach, Priority service and max–min fairness, in: Proceedings of INFOCOM 2002, vol. 1, June 2002, pp. 23–27.
[29] H. Yache, R. Mazumdar, C. Rosenberg, A game theoretic framework for bandwidth allocation and pricing in broadband networks, IEEE/ACM Transactions on Networking 8 (5) (2000) 667–678.
[30] F.P. Kelly, A. Maulloo, D. Tan, Rate control for communication networks: shadow prices, proportional fairness and stability, Journal of the Operational Research Society 49 (1998) 237–252.
[31] X.-R. Cao, H.-X. Shen, R. Milito, P. Wirth, Internet pricing with a game theoretical approach: concepts and examples, IEEE/ACM Transactions on Networking 10 (2) (2002) 208–216.
[32] B. Teitelbaum, S. Hares, L. Dunn, R. Neilson, R.V. Narayan, F. Reichmeyer, Internet2 QBone: building a testbed for differentiated services, IEEE Network 13 (5) (1999) 8–17.
[33] I.Ch. Paschalidis, J.N. Tsitsiklis, Congestion-dependent pricing of network services, IEEE/ACM Transactions on Networking 8 (2) (2000) 171–184.
[34] P. Marbach, Pricing differentiated services networks: bursty traffic in: Proceedings of IEEE INFOCOM 2001, vol. 2, 2001, pp. 650–658.

Tianshu Li received the B.Sc. in Computer Science from University of Victoria, Canada, in 2001. He received his M.MATH degree in Computer Science from the University of Waterloo, Canada in 2003. His research interests include network and distributed systems management, resource management in wired and wireless networks.

Youssef Iraqi received the B.Sc. in Computer Engineering with high honors from Mohamed V University, Morocco, in 1995. He received his M.S. and Ph.D. degrees in Computer Science from the University of Montreal in 2000 and 2003. He is currently a research assistant professor at the School of Computer Science at the University of Waterloo. From 1996 to 1998, he was a research assistant at the Computer Science Research Institute of Montreal, Canada. His research interests include network and distributed systems management, resource management in multimedia wired and wireless networks, and wireless ad hoc networking.

Raouf Boutaba is currently an Associate Professor in the School of Computer Science of the University of Waterloo. Before that he was with the Department of Electrical and Computer Engineering of the University of Toronto. Before joining academia, he founded and was the Director of the Telecommunications and Distributed Systems Division of the Computer Science Research Institute of Montreal (CRIM). He conducts research in the areas of network and distributed systems management and resource management in multimedia wired and wireless networks. He has published more than a hundred papers in refereed journals and conference proceedings. He is the recipient of the Premier's Research Excellence Award, a fellow of the Faculty of Mathematics of the University of Waterloo, and a distinguished lecturer of the IEEE Computer Society. He is the Chairman of the IFIP Working Group on Networks and Distributed Systems, the Vice Chair of the IEEE Communications Society Technical Committee on Information Infrastructure, and the Chair of the IEEE Communications Society Committee on Standards. He is on the advisory editorial board of JNSM, the editorial board of JCN, and the editorial board of Computer Networks.