R U T C O R
R ESEARCH
R E P O R T

A DYNAMIC MODEL FOR PRICING THE PRIORITY-BASED NETWORK SERVICES

Sunju Park[a] Seungjae Han[b]
Michael H. Rothkopf[c]

RRR 1-2004, JANUARY 2004

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/~rrr

[a] Management Science and Information Systems Department, Rutgers Business School,Newark, NJ 07102
[b] Bell Labs, Lucent Technologies, Murray Hill, NJ 07974
[c] Management Science and Information Systems Department and Rutgers Center for Operation Research, Piscataway, NJ 08854-8003

RUTCOR RESEARCH REPORT
RRR 1-2004, JANUARY, 2004

A DYNAMIC MODEL FOR PRICING THE PRIORITY-BASED NETWORK SERVICES

Sunju Park Seungjae Han Michael H. Rothkopf

Abstract. Although today's Internet predominantly uses the flat-fee pricing scheme, the need for more sophisticated pricing has grown, especially for domains where bandwidth resources are scarce and expensive. This paper focuses on how to determine the price vector that maximizes the system revenue in priority-based networks. Similar problems have been examined using equilibrium analysis, but the assumptions needed to produce a static equilibrium may not hold in some realistic network environments. We propose a new method that does not require such assumptions. First, we have developed a method for computing the expected revenue for a given price vector. Then, instead of exhaustively trying out all the possible price vectors, we reduce the search space based on a set of pruning rules and apply a heuristic search to the remaining search space to find the near-optimal price vector. We demonstrate that the proposed scheme finds a near-optimal price vector efficiently, and compare our method with the equilibrium analysis.

Acknowledgements: The first author is partially supported by Rutgers Research Resource Council Grant.

1 Introduction

Pricing of network services has been an issue of continuous debate since the early stage of the Internet commercialization. Historically, the flat-fee scheme where the user is charged a fixed monthly fee irrespective of usage has been the predominant pricing structure of Internet access services, as both service providers and consumers prefer this simple pricing to usage-sensitive pricing (Odlyzko 2001). Under the flat-fee pricing, the service providers have been able to increase the network usage, receive predictable money flow, and have simpler billing and accounting. The consumers have also liked the simpler choices offered by the flat-fee scheme. Partially thanks to this pricing structure, the growth of Internet has been phenomenal and the penetration rate of Internet services by 2002 exceeded 50% in many countries including the USA, Korea, and Finland (ITU 2002).

Recently, however, pricing schemes other than the flat-fee scheme are beginning to emerge in domains, such as 3G cellular data services and next-generation Internet services. For cellular data services, wireless bandwidth is too scarce and expensive to tolerate the potential waste of resources that is typically associated with flat-fee pricing. On the next-generation Internet, new breeds of QoS-sensitive (i.e., delay, loss, or bandwidth sensitive) applications that require service differentiation are emerging. The coexistence of QoS-sensitive applications (such as multimedia and mission-critical medical applications) with traditional best-effort applications (such as emails and file transfers) may well lead the network not only to handle the corresponding traffic differently but also to *charge* the applications differently. Without an appropriate pricing scheme, prioritizing traffic based on service differentiation would become futile because all users will choose the highest possible service level.

Various pricing schemes, including priority pricing (Cocchi 1991, Cocchi 1993), smart-market pricing (Mackie-Mason 1995), Paris-metro pricing (Odlyzko 1997), responsive pricing (Mackie-Mason 1997), edge pricing (Shenker 1996), and proportional fairness pricing (Kelly 1997, Kelly 1998) have been proposed. All these schemes assume that users are inherently price-sensitive, and the service provider can use prices to influence their behavior as a means of congestion control. 'Which pricing scheme should be adopted?' is not the question we address in this paper. We have chosen the priority pricing as the target pricing structure and focus on the price selection problem, i.e., finding the price vector that maximizes the service provider's revenue. The priority pricing we have chosen is most suitable for IETF differentiated services (Blake 1998), the preferred Internet QoS architecture today. The IETF differentiated services provide a way of differentiating traffic based on the traffic class (level) information in the IP packet headers (called markers), and one way to achieve this differentiation is through priority-based scheduling (and thus priority pricing).

Price selection is particularly important in priority pricing since improper prices may not provide any gain over flat pricing (Cocchi 1991, Gupta 1997). Under priority pricing, the service provider offers a choice of {service level, price} pairs, and users, who best know the value of their jobs, each selects the service level he wants to use. To select the optimal price vector, therefore, the service provider needs to know which service level the users will choose. Note that a user's choice depends on not only the prices and the utility of his job but also the degree of congestion at each level (which in turn is determined by the aggregate result of all users' choices). When a lower-level service becomes more congested, for example, some users might

decide to use a higher-level service (with a higher fee) for their jobs. When a higher-level service becomes congested as a result, it becomes less attractive (i.e., not worth the money), and some users may decide to use lower priority levels, and so on. The price selections based on equilibrium analysis (Mendelson and Whang 1990, Mendelson 1985, Mandjes 2003), however, often do not properly account for the dynamics in some realistic network environments.

In comparison, we propose a new method that captures such dynamics explicitly when selecting the optimal price vector. Our method is based on a new model for computing the expected revenue for a given price vector. Since trying out all the possible price vectors (and choosing the one with the highest expected revenue) would be time consuming, we have developed a set of pruning rules to reduce the search space and applied a heuristic search to the remaining search space.

The remainder of the paper is organized as follows. In Section 2, we describe the general framework of the target pricing scheme and formally define the price-selection problem. In Section 3, we motivate our approach by explaining why the equilibrium analysis method does not deal properly with some practical constraints. In Section 4, we present our approach. In Section 5, we provide numerical examples and compare our method with the equilibrium analysis. Section 6 concludes the paper.

2 Framework for the Priority Pricing Scheme

We consider the priority-pricing scheme between the network service provider and the end users. The network offers I different service levels to users. Without loss of generality, we assign the highest level as level 1, the second highest as level 2, and so on. For service level i, the service provider charges P_i per data unit (e.g., byte). Once set, the prices do not change.

Each user may run one or more applications that need to send data through network, and we call each application data a 'job.' Examples of a job is sending an email message or accessing a WWW page. We assume that there exist J types of jobs and any user is allowed to have any type of job. (This is more general than classifying jobs based on user types.) The probability that a job is of type j is denoted by $Pr(j)$. We assume that the size of a type-j job follows an exponential distribution with a mean value C_j. For each job, the user decides whether to send the job, and if he decides to send it, which service level to use. The user pays the price associated with the selected level. We assume that the user does not change his decision on the service level until his job is completed. Each type-j job has a corresponding utility function, $U_j(t)$, which represents the value of a job as a function of the job completion delay, t. In general, users prefer to send their jobs as quickly as possible, so we assume that the utility function is decreasing with respect to the time spent until the job has completed. A linear utility function, shown in Figure 1-(a), implies uniform delay cost per time unit. A non-linear utility function, in Figure 1-(b), captures the urgency of time-critical applications.

(a) Linear utility function (b) Non-linear utility function

Figure 1: Examples of the utility function of a job.

We assume the network handles jobs based on "non-preemptive Strict Priority Scheduling (SPS)." Under SPS, jobs at a higher level are served ahead of those at a lower level. That is, a lower-level job will start service only when there is no higher-level job waiting. Within the same level, FIFO is used. 'Non preemptive' means that a newly-arriving higher-level job does not interrupt the lower-level jobs currently being served. Note that under SPS, jobs at a higher level will start service before ones at a lower level, but being at a specific service level does not guarantee any exact time of completion. Strictly speaking, this assumption of FIFO among the same level jobs is an approximation, since the schedule unit of the priority packet scheduling in the real networks is a packet not a job (which can be a train of multiple packets

The network serves jobs at the rate of μ. The current Internet architecture supports multiple tiers of service providers, each of which has a link (or several links)—called a *backhaul link*—to one or more upper-tier networks. The capacity of the backhaul link determines the job service rate, μ. We assume that job arrivals at each service level follow a Poisson process. Let the arrival rate at level 0, λ_0, denote the rate that some users opt not to submit their jobs. The sum of all job arrival rates, λ_i, is equal to the total *potential* job arrival rate, λ (i.e., $\sum_{i=0}^{I} \lambda_i = \lambda$). The *actual* job arrival rate, $\lambda - \lambda_0$, is always smaller than the service rate, μ. This will be true even when λ is higher than μ, because some job(s) would not be submitted to the network. If λ is higher than μ, the job completion time of some level(s) would become infinite due to congestion, which would result in zero or negative utility for some jobs, and thus such jobs would not be sent to the network.

The service provider faces a price-selection problem. It needs to determine the price vector for I service levels, $\{P_i\}$, that maximizes its revenue. The service provider's decision problem is formulated as follows. Equation (1) maximizes the sum of revenues generated at all service levels minus the cost (i.e., the fee the service provider pays to her upper-level service provider(s)).

$$\text{Maximize } \sum_{i \in I} (\mathrm{P_i} \times \lambda_i \times C) - \mathrm{P_B} \times \mu \times \mathrm{C}, \quad (1)$$

where

P_i is the per-data-unit price of level-i service (while $P_0 = 0$),

λ_i is the job arrival rate at level i,

C is the average job size, which is $\Sigma_{j \in J} C_j \times Pr(j)$,

P_B is the per-data-unit price of network backhaul capacity, and

μ is the service rate.

Strictly speaking, equation (1) maximizes the *profit*. We call it revenue maximization, however, because the second term (cost) is constant (assuming P_B and μ are static). Therefore, equation (1) in fact finds the revenue-maximizing price vector. Note that equation (1) assumes a fixed number of service levels, I, but the number of levels offered by the service provider also has an impact on the revenue. We address the impact of the number of service levels on revenue maximization in Section 5. Also note that while the definition of C is exact for all jobs, it may be approximate when some jobs are not sent. However, when the jobs not sent have an average duration similar to the jobs sent or when, as in our examples below, almost all jobs are sent, it is a good approximation.

3 System Convergence vs. Oscillation

In the literature (Mendelson and Whang 1990, Mendelson 1985, Mandjes 2003), equilibrium analysis has been applied to the price-selection problems similar to ours. We argue, however, that equilibrium analysis makes assumptions that are either practically infeasible or too expensive to implement under certain circumstances.

One of the potentially unrealistic assumptions of equilibrium analysis is that each job has an infinitesimal impact on the total system. This implies the presence of an infinite (or very large) number of active users, which may not be true. At the edge network (which directly provides the Internet access to the end users), for example, the number of active users is typically limited. Furthermore, some networks may have a physical limit on the number of users. For example, in 3G cellular networks, the most advanced form of wireless data networks today, the number of simultaneous sessions per cell (carrier) is limited to 64 because of the availability of Walsh codes (Harte, Kitka, and Levine 2002). If the impact of each job is not infinitesimal, the arrival of a job in fact changes the service completion time of other jobs, and as a result the system may fail to reach a static equilibrium but oscillate instead.

To understand the potential of oscillation, we examine the user's decision-making process and its impact on the job arrival rate at each service level. It is reasonable to assume that all users select the service level of their jobs solely based on self-interest. That is, a user with a type-j job selects the service level i that maximizes the expected utility value minus the associated price. The following expression gives the decision analysis of a user with a type-j job (If the result of the equation is less than zero, the user will not send the job.):

$$Argmax_{\ i}\ [U_j(T_{ij}) - P_i \times C_j], \qquad (2)$$

where P_i is the price of level-i service, and T_{ij} is the expected time to complete a type-j job at level i. Since no guarantee is made on the exact completion time under SPS, we use the *expected* completion time. Under Poisson job-arrival and exponential job-length assumptions, the expected completion time of a type-j job at level k, T_{kj}, under SPS can be computed using the *M/M/1* queuing model as follows (Bertsekas 1992).

$$T_{kj} = \frac{C_j}{\mu} + \frac{C^2/\mu^2 \cdot \sum_{i=1}^{I} \lambda_i}{(1 - C/\mu \cdot \sum_{i=1}^{k-1} \lambda_i)(1 - C/\mu \cdot \sum_{i=1}^{k} \lambda_i)} \quad (3)$$

Under SPS, the user's decision on which level to send his job depends on T_{kj}, which in turn depends on the job arrival rates of higher levels, λ_i, $1 \le i \le k$, which in turn depends on the other users' decisions. Figure 2 illustrates the oscillation in a system with two priority levels. If there were little level-1 traffic and the price of the level-1 were reasonable, many users would choose level-1. This trend would result in more traffic at level-1, which would increase T_{1j} and make the level-1 service less attractive. Then, some users may decide to choose level-2 for their new jobs, which would decrease λ_1 and make the level-1 attractive again (see Figure 2). This cycle may force the incoming jobs to flock from one level to another (oscillating), rather than making the system settling to a stable state (equilibrium).

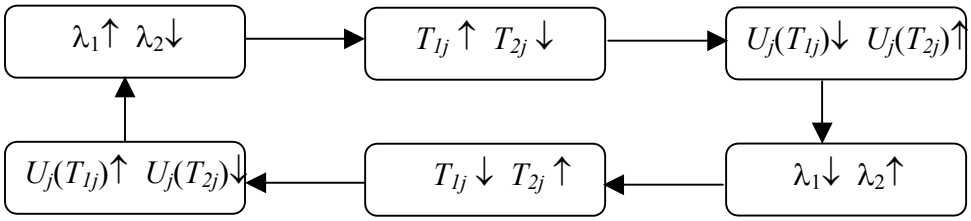


Figure 2: Oscillation in a two-level system.

Another questionable assumption is that all users have completely accurate knowledge about the current job arrival rates. In many real networks, to maintain such up-to-date knowledge is infeasible or very costly. For example, each user may need to sniff all of the network traffic to figure out which job has chosen which service level. Usually, such sniffing is infeasible for security reasons or due to limitations inherent to the routing structure. Alternatively, a user may ask the network for the current job arrival-rate information whenever he needs to send a job, and the network waits (i.e., does not accept any other job request) until that user makes his decision. This is undesirable for efficiency reasons. A more realistic alternative is that network *periodically* broadcasts the recent arrival rate vector, $\{\lambda_i\}$. If we assume periodic broadcasts and that all users between the periodic updates make decisions based on the latest broadcast $\{\lambda_i\}$ instead of the current (but unknown) actual arrival rates, this may increase the potential for oscillation. In some extreme cases, this may cause all jobs to flock to level-1 during one update interval, to level-2 at the next interval, and to level-1 in the following interval, and so on.

4 Solution Approach

Because of the potential for oscillation, we propose a new method for finding a near-optimal pricing vector. First, we have developed a method for modeling potential oscillation and

computing the expected revenue for a given price vector $\{P_i\}$. If the expected revenue for each price vector were computed, then the optimization would be simple; one would select the price vector with the highest revenue. A brute-force search of exhaustively trying out all possible price vectors, however, is computationally too expensive. Therefore, we have developed a set of rules for pruning the search space and a heuristic search for obtaining the near-optimal pricing vector.

4.1 Computation of $\{\lambda_i\}$ under given $\{P_i\}$

In this subsection, we focus on computing the expected revenue for each price vector $\{P_i\}$, $\Re_{\{Pi\}}$. As $\Re_{\{Pi\}}$ is a function of arrival rates, $\{\lambda_i\}$, we need to model the mapping from $\{P_i\}$ to corresponding $\{\lambda_i\}$. The computation of $\{\lambda_i\}$ under given $\{P_i\}$ can be divided into two classes: (i) the trivial cases where the arrival rates always converge, and (ii) the non-trivial cases where the arrival rates *may* oscillate.

We use the two-level system as an example, and later generalize it to *I*-level systems. In the system with two service levels, six scenarios are possible: (1) all jobs choose level-1, (2) all jobs choose level-2, (3) some jobs choose level-0 (i.e., not submitted) while the rest choose level-1, (4) some choose level-0 while the rest choose level-2, (5) some choose level-1 while the rest choose level-2, and (6) a mixture of level-0, level-1, and level-2. Computing $\{\lambda_i\}$ from $\{P_i\}$ is trivial for scenario-1 and 2, while non-trivial for the rest.

4.1.1 Trivial Cases

Scenario-1 occurs if all users have incentives to send their jobs at level-1 (condition A1) and selecting level-1 over level-2 is better for all users (condition A2)

.

$$U_j(T_{1j}) - P_1 \times C_j \geq 0, \text{ for } \forall j \in J, \quad \text{(A1)}$$

$$U_j(T_{1j}) - P_1 \times C_j \geq U_j(T_{2j}) - P_2 \times C_j, \text{ for } \forall j \in J. \quad \text{(A2)}$$

The conditions for scenario-2 are defined in a similar manner. All jobs are sent at level-2, when all users have incentives to send their jobs at level-2 (A3) and selecting level-2 over level-1 is better for all users (A4).

$$U_j(T_{2j}) - P_2 \times C_j \geq 0, \text{ for } \forall j \in J, \quad \text{(A3)}$$

$$U_j(T_{1j}) - P_1 \times C_j \leq U_j(T_{2j}) - P_2 \times C_j, \text{ for } \forall j \in J. \quad \text{(A4)}$$

Let $\{\lambda_0, \lambda_1, \lambda_2\}$ denote the job arrival rates at 0-, 1-, and 2- service levels, respectively. (Remember that the 0-service level indicates the job is not sent to the network.) When the two sets of conditions for scenario-1 (or scenario-2) are met, the arrival rates always converge to $\{0, \lambda, 0\}$ (or $\{0, 0, \lambda\}$). Once $\{\lambda_i\}$ is computed, the expected revenue for scenario-1 (or scenario-2), $\Re_{s1}$ (or $\Re_{s2}$), can be computed using equation (1) as follows:

$$\Re_{s1} = P_1 \times \lambda \times C - P_B \times \mu \times C, \quad (4)$$
$$\Re_{s2} = P_2 \times \lambda \times C - P_B \times \mu \times C. \quad (5)$$

When generalized, the conditions that all jobs select k-th level in the system with I number of priority levels are as follows:

$$U_j(T_{kj}) - P_k \times C_j \geq 0, \text{ for } \forall j \in J, \quad (A5)$$
$$U_j(T_{kj}) - P_k \times C_j \geq U_j(T_{ij}) - P_i \times C_j, \text{ for } \forall j \in J, \forall i \in I, i \neq k. \quad (A6)$$

Let λ_k in $\{\lambda_0, \ldots, \lambda_k, \ldots, \lambda_I\}$ denote the job arrival rate at k-th service level. When the two sets of conditions for scenario-k are met, all jobs will be sent at level-k (i.e, the arrival rates always converge to $\{0, \ldots, \lambda, \ldots, 0\}$, where λ appears at the k-th position. The expected revenue of the general case is as follows:

$$P_k \times \lambda \times C - P_B \times \mu \times C. \quad (6)$$

4.1.2 Non-Trivial Cases

Unlike scenario-1 or scenario-2, the rest of the scenarios do not guarantee convergence to a stable state due to the possibility of oscillation and thus do not have any simple way of deriving $\{\lambda_i\}$ for given $\{P_i\}$. Instead of assuming away the potential oscillation, we model explicitly the transition of $\{\lambda_i\}$ through time and use it to compute the expected revenue.

Let the state of the system at time t, S^t, represent the $\{\lambda_i\}$ values *being used* at time t, $\{\lambda_i\}^t$. Since we allow for the periodic information update, the state transition is discrete and occurs only when new $\{\lambda_i\}$ is broadcast, as shown in Figure 3. Note that as the user's decision is based on the expected completion time, T_{ij} (which is a function of λ_i), his decision between the last and the new updates is not affected by the actual job arrivals in-between.

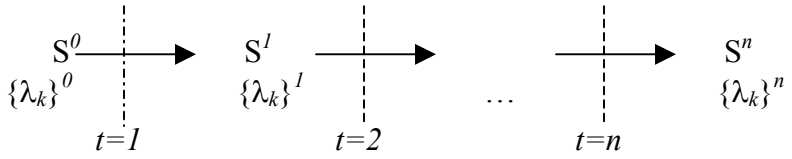


Figure 3: $\{\lambda_i\}$ transition over time.

Let T_{kj}^t be the expected completion time of a type-j job at level-k when the system is at S^t. Let $X_{k,j}^t$ take the value of 1 if the user finds it most profitable to send his type-j job at level-k when the system is at S^t. That is,

$$X_{k,j}^t = \begin{cases} 1, \text{ if } U_j(T_{kj}^t) - P_k \cdot C_j \geq 0, \text{ and } U_j(T_{kj}^t) - P_k \cdot C_j \geq U_j(T_{ij}^t) - P_i \cdot C_j \text{ for } \forall i \in I, i \neq k \\ 0, \text{ otherwise} \end{cases}. \quad (7)$$

Then, the probability that a newly-arriving job selects level-k at S^t, $Pr(k, S^t)$, is computed as follows, where $Pr(j)$ is the probability that a job is of type j:

$$Pr(k, S^t) = \sum_j \Pr(j) \cdot X^t_{k,j} \ . \quad (8)$$

Using *Pr(k, S^t)*, we compute the arrival rates at S^{t+1}, $\{\lambda_k^{t+1}\}$, as follows:

$\lambda_k^{t+1} = \lambda \times Pr(k, S^t)$ for $\forall k \in I$. (9)

Under given $\{P_i\}$, the transition of $\{\lambda_i\}$ over time may converge to a single state or may oscillate among a set of states. Once the transition path of $\{\lambda_i\}$ is identified, the expected system revenue is calculated as follows:

$$\Re_{\{P_i\}} = \sum_S \left(W(S) \times \sum_i \left(P_i \times \lambda_i(S) \times C \right) \right) - P_B \times \mu \times C, \quad (10)$$

where *W(S)* is the probability of being in state *S* ($\Sigma_S W(S) = 1$), and $\lambda_i(S)$ is the job arrival rates at level *i* in state *S*.

4.2 Heuristic Search with Pruning

In the previous subsection, we examined how to compute the expected revenue for a given price vector. Even when one can compute the expected revenue for a given price vector, however, it is impractical to try out all the possible price vectors to find the optimal pricing. Instead, we use the following two procedures to obtain a near-optimal pricing: (1) reduce the search space by pruning, and (2) apply a heuristic search algorithm to the remaining search space.

For the possible prices of each level *i*, P_i, we can prune out two types of regions, Π_i. and Φ_i, where Π_i is the range of P_i where no job selects level-*i*, and Φ_i is the range of P_i where all jobs choose level-*i*. Because all P_i's in Π_i generate the same revenue as the lowest P_i that guarantees no job selects level-*i*, one only needs to compute the revenue for the lowest P_i, and therefore can prune Π_i. On the other hand, the system always converges to a stable state for P_i's in Φ_i, and the revenues for the Φ_i region can be easily computed using equation (6).

We explain the pruning rules using a two-level system as an example and describe the general case later. For the system with two service levels, we use the following four rules to reduce the search space.

[Rule 1: pruning Π_1] Prune all P_1's larger than $min[U_j(C_j/\mu)/C_j]$ for $\forall j \in J$.
[Rule 2: pruning Π_2] For a given P_1, prune all P_2's larger than P_1.
[Rule 3: pruning Φ_1] Prune all P_1's smaller than $min[U_j(T_{1j}^*)/C_j]$ for $\forall j \in J$, where T_{1j}^* is the expected completion time of a type-*j* job at level-1 when the arrival rate is $\{0, \lambda, 0\}$.
[Rule 4: pruning Φ_2] For a given P_1, prune all P_2's smaller than $min[U_j(T_{2j}^*)/C_j , U_j(T_{2j}^*)/C_j - U_j(C_j/\mu)/C_j + P_1]$ for $\forall j \in J$, where T_{2j}^* is the expected completion time of a type-*j* job at level-2 when the arrival rate is $\{0, 0, \lambda\}$.

Rule 1 and Rule 2 are obvious. Since the minimum expected completion time is $1/\mu$ (when there is no queuing delay), the highest possible utility value of a type-*j* job is $U_j(C_j/\mu)$. So, if the price of the level-1 service is higher than the highest possible utility value for all jobs ($P_1 \times C_j > U_j(C_j/\mu)$ for all $j \in J$), no job will select level-1. If P_2 is higher than P_1, on the other

hand, no job chooses level-2 (since jobs at level-1 has priority over jobs at level-2 under SPS, no one will pay more for level-2 service).

Rule 3 prunes all P_1's that guarantee all jobs select level-1 (i.e., scenario-1). Remember there are two necessary conditions for scenario-1, A1 and A2. A1 says that the highest price of the level-1 service that guarantees all jobs select level-1 is $min[U_j(T_{1j}^*)/C_j]$ (i.e., $P_1 \times C_j < U_j(T_{1j}^*)$ for all $j \in J$), where T_{1j}^* is the expected completion time of a type-j job at level-1 when the arrival rate is $\{0, \lambda, 0\}$. T_{1j}^* can be easily computed from equation (3). A2 says $P_1 \times C_j < U_j(T_{1j}) - U_j(T_{2j}) + P_2 \times C_j$ for all j. When $P_1 = P_2$, A2 is always true, because T_{1j} is always smaller than T_{2j} under SPS. In other cases, as long as A2 can be satisfied, the revenue generated will be less than that of scenario-1, so one may not have to consider condition A2. Therefore, A1 defines the highest P_1 that guarantees scenario-1.

Rule 4 finds the highest price of the level-2 service for a given P_1 that guarantees scenario-2, and prunes all P_2's below it. A3 says $P_2 \times C_j < U_j(T_{2j}^*)$ for all $j \in J$, where T_{2j}^*, the expected completion time of a type-j job at level-2 when the arrival rates are $\{0, 0, \lambda\}$, can be computed by equation (3). A4 says $P_2 \times C_j < U_j(T_{2j}) - U_j(T_{1j}) + P_1 \times C_j$ for all j. T_{1j} for the first user who chooses level-1 is C_j/μ (i.e., zero queuing delay). Therefore, the biggest P_2 that guarantees scenario-2 is $min[U_j(T_{2j}^*)/C_j, U_j(T_{2j}^*)/C_j - U_j(C_j/\mu)/C_j + P_1]$ for all j.

When generalized, the pruning rules for the system with I levels are as follows.

[GenRule 1: pruning Π_1] Prune all P_1's larger than $min[U_j(C_j/\mu)/C_j]$ for $\forall j \in J$.
[GenRule 2: pruning Π_k] For a given P_{k-1}, prune all P_k's, $1 < k \le I$, larger than P_{k-1}.
[GenRule 3: pruning Φ_1] Prune all P_1's smaller than $min[U_j(T_{1j}^*)/C_j]$ for $\forall j \in J$, where T_{1j}^* is the expected completion time of a type-j job at level-1 when the arrival rate is $\{0, \lambda_1 = \lambda, 0, \ldots, 0\}$.
[GenRule 4: pruning Φ_k] For a given $P_1, \ldots, P_{k-1}$, prune all P_k's, $1 < k \le I$, smaller than $min[U_j(T_{kj}^*)/C_j, U_j(T_{kj}^*)/C_j - U_j(C_j/\mu)/C_j + P_m]$ for $\forall j \in J, \forall m \in I, m < k,$, where T_{kj}^* is the expected completion time of a type-j job at level-k when the arrival rate is $\{0, \ldots, \lambda_k = \lambda, \ldots, 0\}$.

The revenues for the price vectors in the remaining search space after pruning can be computed by the method in Section 4.1.2. Exhaustively evaluating all possible price vectors to find the best one, however, would be time consuming. Hill-climbing search can find the optimal solution as long as the objective function is unimodal.[1] This means that in our problem, the revenues should be unimodal with respect to the price vectors, which is not true in general (although it is true under the assumptions made in equilibrium analysis). One may use simulated annealing or repeatedly apply hill climbing from multiple starting points to avoid getting stuck in a local optimum, but we have decided not to use hill climbing. Instead, we use 'iterative-deepening' search (Russell and Norvig 1995).

Iterative-deepening search combines the benefits of depth-first search and breadth-first search. It begins with the price vectors coarsely selected from the remaining search space, and gradually tries out the other price vectors while decreasing the granularity of prices until the resulting improvement has become either negligibly small, the computational time limit has been

[1] Unimodality ensures the existence of a proper (i.e., single, relative) maximum in a given region.

exhausted, or the pre-set depth limit has been reached. Figure 4 presents our iterative deepening algorithm.

```
Algorithm Iterative Deepening Search
    Inputs: I, the number of service levels
        μ, the job service rate
        λ, the job arrival rate
        U_j(·), the utility function of type-j job, for all j
        G, the unit of granularity
        D, the depth limit
        Δ_max, the limit on the computational time
        Ω, the limit on improvement
    Outputs: {P_k}*, the best price vector
        ℜ*, the expected maximum revenue

    oldℜ ← 0, ℜ* ← 0;                          // Initialize
For k ← 1 to I do
    Prune the lower and upper bounds of P_k, Φ_k and Π_k, respectively;
End
    For d ← 1 to D do                          // Expand the next level until D
      For k ← 1 to I do
            inc[k] ← (Π_k – Φ_k) / (d × G);    // Set the granularity of prices
      End

      For P_1 ← Φ_1 to Π_1 incremented by inc[1] do
      For P_2 ← Φ_2 to Π_2 incremented by inc[2] do
      …
      For P_I ← Φ_I to Π_I incremented by inc[I] do
            Compute ℜ for {P_k};               // Use the method in Section 4.1.2
            If (ℜ > ℜ*) then ℜ* ← ℜ, {P_k}* ← {P_k}; end
      End
      …
      End
      End

      If (elapsed_time > Δ_max) then return; end   // If time limit has reached, stop.
      If (ℜ* – oldℜ < Ω) then return; end          // If the improvement is negligible, stop.
      oldℜ ← ℜ*;
    End
End_Algorithm
```

Figure 4: Iterative deepening search algorithm.

5 Experiments and Numerical Results

First, we present an example that illustrates how the proposed method works. Then, we examine the impact of various system parameters, including user-side parameters such as utility functions and job arrival rates, and network-side parameters such as number of service levels and job service rate, on the best price vector and the expected revenue. Finally, we compare the proposed scheme against an existing equilibrium analysis method.

5.1 An Illustrative Example

The following example demonstrates how the pruning rules reduce the search space and how the iterative-deepening search finds the near-optimal solution.

Our example roughly mimics WWW access over the 3G wireless network. The network supports two priority levels ($I = 2$). We set $\mu = 6.0$ and $P_B = 1.0$. The average size of a web page is empirically measured as around 30 Kbytes or 240 Kbits (Molina 2000), and the 3G-1X-RTT (or 3G-1X-EVDO) networks provide the peak throughput of 144kbps (or 2.4Mbps), so setting μ to be 6.0 corresponds to the total capacity of 1.44 Mbps. Ten different types of jobs are in the system (J = 10). All utility functions are linear (i.e., $U_j(t) = \beta + \alpha \times t$) with the max value $\beta = 10$ and the slope α being uniformly distributed between $\alpha_{min} = -3$ and $\alpha_{max} = -1$. The total job arrival rate, λ, is 5.0, and the job size, C_j, is set to 1.0 for all job types. We set the frequency of broadcasting updates of $\{\lambda_i\}$ to be 10 times per second. The typical update interval of routing protocols (e.g., RIP, OSPF) is roughly once a second (Huitema 1995), but we chose a smaller interval because propagation and convergence are a lesser issue in our context. The parameters for iterative-deepening search are set as G = 10, D = 5, $\Delta_{max} = \infty$, and $\Omega = 0$.

Figure 5 shows the remaining search space after applying the pruning rules. Rule 1 ensures that P_1 cannot be larger than 9.4. Rule 2 ensures that P_2 cannot be larger than P_1. Rule 3 prunes P_1 up to 7.0. As shown in Figure 6-(a), when P_1 is less than or equal to 7.0, scenario-1 happens as P_2 increases. Rule 4 computes the lower bound of P_2 for each remaining P_1. For example, when P_1 is 9.0, the lower bound of P_2 is 6.5 in Figure 5. Figure 6-(b) shows that when P_1 is 9.0, all jobs are sent at level 2 (i.e., scenario-2) until P_2 reaches 6.5. Figure 6-(b) also shows that, as P_2 becomes greater than 6.5, the system enters non-trivial states, and the revenue fluctuates, which indicates that the strict hill climbing may end up with a local optimum.

Applying the iterative-deepening search on the remaining region in Figure 5, we have found the best price vector of $\{P_1 = 8.90, P_2 = 8.00\}$, and the corresponding expected revenue of 35.04. The result is close to the optimal price vector obtained by exhaustively searching all possible price vectors at the granularity of 0.01 (the lowest denominator of currency) without pruning, $\{P_1 = 8.93, P_2 = 8.15\}$.

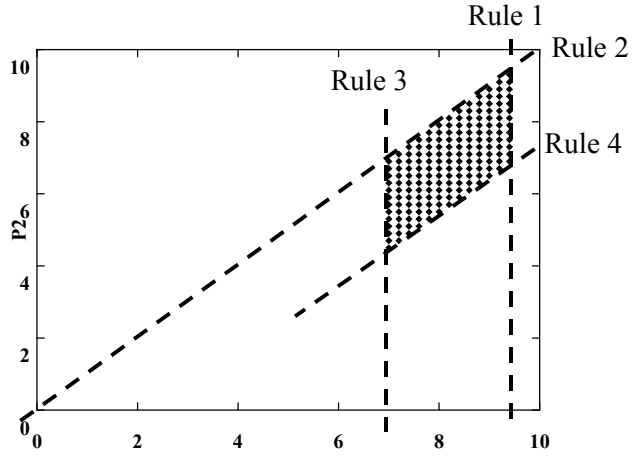
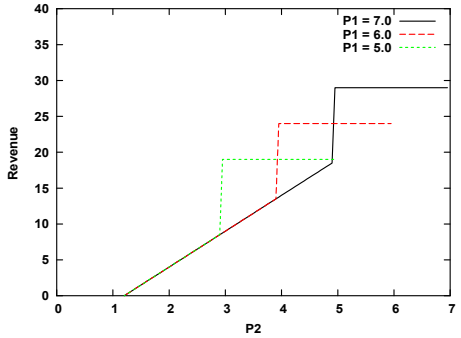
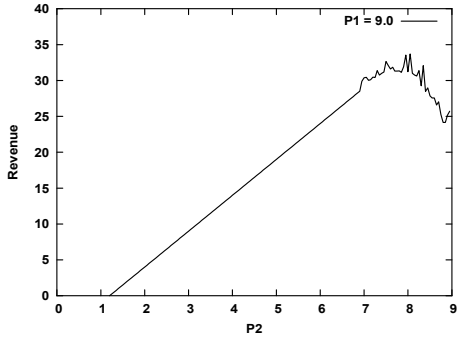


Figure 5: Pruning of the search space in a two-level system.



(a) The system converges to scenario-1



(b) The system does not converge to scenario-1

Figure 6: Changes in revenue when P_2 increases for a given P_1.

5.2 Impacts of System Parameters

To study the impact of the arrival rate, λ, and the service rate, μ, on the best price vector and the expected revenue, we change the values of λ and μ from the previous example. The results are summarized in Table 1 and 2.

When λ is fixed, the increase of μ (up to a certain value) results in an increased revenue and a higher P_i (see Table 1). This is because the increased μ reduces the job completion time and thus the users are willing to pay more. A bigger μ, however, means a higher backhaul cost in equation (1), and thus beyond a certain μ value the higher payments by users is not enough to compensate the higher backhaul cost. This results in the profit decrease after a certain point. The actual arrival rate, $\lambda - \lambda_0$, increases up to the potential maximum arrival rate.

λ	μ	$\{P_1, P_2\}$	ℜ	$(\lambda - \lambda_0)$	λ_0
5.0	6.0	{8.90, 8.00}	35.04	4.89	0.11
5.0	7.0	{9.20, 8.60}	36.44	4.95	0.05
5.0	8.0	{9.25, 8.90}	37.55	5.00	0.0
5.0	9.0	{9.35, 9.15}	37.35	5.00	0.0
5.0	10.0	{9.65, 9.40}	37.00	5.00	0.0

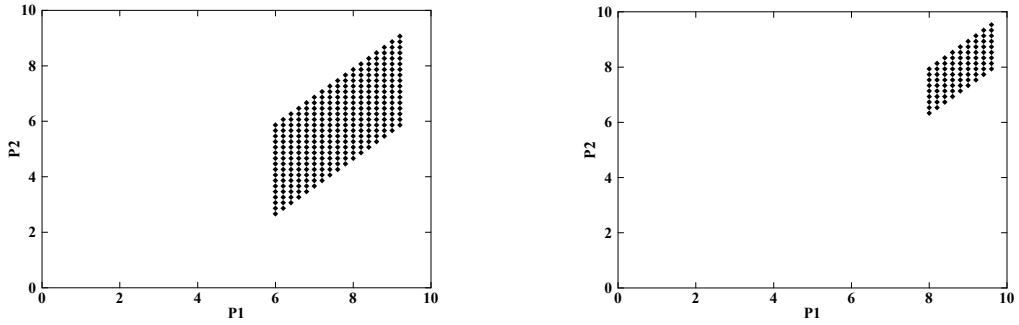
Table 1: The impact of μ while λ is fixed.

The increase of λ also results in higher revenues (see Table 2). A higher λ leads to higher system utilization (i.e., $(\lambda - \lambda_0)/\mu$), which results in higher revenue. The prices become lower with higher λ, because higher λ increases the job completion time and users are less willing to pay the high prices.

λ	μ	$\{P_1, P_2\}$	$\Re$	$(\lambda - \lambda_0)/\mu$	λ_0
1.0	6.0	{9.45, 9.40}	3.40	0.167	0.0
2.0	6.0	{9.45, 9.25}	12.50	0.333	0.0
3.0	6.0	{9.10, 8.90}	21.06	0.500	0.0
4.0	6.0	{8.75, 8.15}	28.52	0.667	0.0
5.0	6.0	{8.90, 8.00}	35.04	0.815	0.11

Table 2: The impact of λ while μ is fixed.

As most computational time is spent is spent on computing $\{\lambda_i\}$ from $\{P_i\}$ of the un-pruned space, the bigger the pruned search space, the more efficient the proposed scheme. One of the factors in determining the size of pruned search space is the distribution of utility functions. Figure 7 shows the impacts of the distribution of utility functions on the size of un-pruned search space. The same experiment setup as in Figure 5 is used for comparison. In general, the wider the distribution of utility functions is, the smaller the pruned search space is.



(a) $\alpha_{min} = -4$ and $\alpha_{max} = -1$ (b) $\alpha_{min} = -2$ and $\alpha_{max} = -1$

Figure 7: The impact of the distribution of utility functions on search space.

We have assumed so far that the number of service levels offered by the service provider is pre-determined, but it can be chosen based on the underlying technologies, the overhead of accounting & billing, and so on. Table 3 reports how the number of levels offered by the system affects the expected revenue. In general, as the number of levels increases, both expected revenue and the system utilization ratio increase.

Number of levels	$\{P_i\}$	$\Re$	$(\lambda - \lambda_0)/\mu$
1	{8.50}	30.12	0.708
2	{8.90, 8.00}	35.04	0.815
3	{9.20, 8.85, 8.30}	36.59	0.822

Table 3: The impact of the number of service levels.

5.3 Comparison with Equilibrium Analysis Method

We investigate the difference between our scheme and the equilibrium analysis method. For comparison, we choose the 'profit-center pricing structure' proposed by Mendelson (Mendelson 1985). In his model, Mendelson assumes the strictly concave and twice-differentiable *value* function, $V(\lambda^*)$, a standard assumption in equilibrium analysis. The value function, $V(\lambda^*)$, represents the expected gross value (per unit time) corresponding to arrival rate λ^* ($= \lambda - \lambda_0$). The marginal value function, $V'(\lambda^*)$, represents the value of a single job when the arrival rate is λ^*. In our investigation, we use the iso-elastic marginal value function, $V'(\lambda^*) = K/\sqrt{\lambda}$. We use a network with a single service level (I = 1) and only one type of job (J = 1) with the linear utility function (i.e., $U_j(t) = \beta + \alpha \times t$).

Let us briefly explain how the equilibrium method computes the optimal price and expected profit. Readers may refer to Mendelson's seminal work (1985) for more details. A Nash equilibrium is satisfied when the users indifferent to joining the system, which is represented by the condition of $V'(\lambda^*) = P_1 + \alpha \times T_{11}$. The expected profit per time unit is equal to $\lambda^* \times P_1 - \mu \times P_B = \lambda^* \times [V'(\lambda^*) - \alpha \times T_{11}]$. The first order conditions for maximizing the above objective function are as follows, where $f(\rho) = \rho/(1 - \rho)$:

$$V'(\lambda^*) + V''(\lambda^*) = (\alpha/\mu) \times f'(\lambda^*/\mu), \quad (11)$$

$$P_B = \alpha \times (\lambda^*/\mu^2) \times f'(\lambda^*/\mu). \quad (12)$$

By applying the assumed value function to the above equations, we get two equations which are $K^2 = 4\alpha \times P_B \times f'(\rho)$ and $\mu = K^2 \cdot \rho/(4 P_B^2)$. Then, we can compute the optimal ρ from the first equation and the optimal μ from the second equation. After computing T_{11} using the *M/M/1* queuing model, we can derive the optimal P_1 from the equilibrium condition. The expected profit can be directly computed from the objective function.

In Section 3, we pointed out that since the impact of each job is not infinitesimal and the arrival rates are broadcast periodically, the outputs of the equilibrium analysis may not match those of our scheme which explicitly models the system dynamics. We have tested our claim in two different environments. First, we have analyzed high-capacity networks (i.e., larger μ) with frequent arrival rate updates (i.e., $\{\lambda_i\}$ is broadcast 100 times per second). In this environment, the equilibrium approach should be able to model the system more accurately. Table 4 presents the results under various system parameters[2]. As expected, the outputs of two schemes are close. Second, we have analyzed relatively low-capacity networks (i.e., smaller μ) with infrequent arrival rate updates (i.e., $\{\lambda_i\}$ is broadcast 5 times per second). Table 5 shows that the outputs of the equilibrium method do not match those of our method. This is because the equilibrium method is not able to capture that the impact of each job is greater in low-capacity networks and because the system oscillation is greater with less frequent $\{\lambda_i\}$ broadcasts.

[2] Note that the μ values are determined based on P_B, K, and α in the profit-center pricing scheme, and we have used those computed μ values in our experiments.

(P_B, λ, μ, K, α)	Equilibrium method	Our method
(0.5, 80.0, 86.0, 10, -1)	P_1 = 1.08, $\Re$ = 36. 86	P_1 = 1.10, $\Re$ = 37.13
(0.5, 70.0, 75.5, 10, -3)	P_1 = 1.16, $\Re$ = 28.50	P_1 = 1.15, $\Re$ = 29.06
(1.0, 85.0, 90.0, 20, -1)	P_1 = 2.11, $\Re$ = 81.00	P_1 = 2.15, $\Re$ = 81.78
(1.0, 75.0, 83.0, 20, -3)	P_1 = 2.21, $\Re$ = 68.36	P_1 = 2.25, $\Re$ = 68.87

Table 4: Large capacity networks with frequent arrival rate broadcast.

(P_B, λ, μ, K, α)	Equilibrium method	Our method
(1.5, 7.0, 8.4, 10, -1)	P_1 = 3.49, $\Re$ = 9.50	P_1 = 5.75, $\Re$ = 11.55
(1.5, 5.0, 6.4, 10, -3)	P_1 = 4.11, $\Re$ = 5.52	P_1 = 6.35, $\Re$ = 9.45
(2.0, 19.0, 21.5, 20, -1)	P_1 = 4.33, $\Re$ = 36.86	P_1 = 7.15, $\Re$ = 38.51
(2.0, 17.0, 19.0, 20, -3)	P_1 = 4.65, $\Re$ = 28.51	P_1 = 7.40, $\Re$ = 37.48

Table 5: Small capacity networks with infrequent arrival rate broadcast.

6 Conclusion

This paper investigates the price-selection problem for priority network services by modeling explicitly the system dynamics instead of assuming the existence of a static equilibrium. We have argued that the conditions needed to produce a static equilibrium may not hold in the realistic network service environments. Indeed, we have observed that when those conditions are not likely to be met, the system may oscillate among a number of states without reaching a static equilibrium, and that the equilibrium analysis method may produce sub-optimal price selection. As an alternative, we propose a dynamic modeling method that explicitly captures the state transition. In addition, we use the iterative-deepening search combined with a clever search-space pruning algorithm to obtain the near-optimal pricing. Accurate modeling of system dynamics is a key to better price selection.

Although we have specifically targeted the multi-level priority-pricing scheme and chosen revenue maximization as the system goal in this paper, the general issues and arguments here are applicable to both different pricing paradigms and different system objectives like economic efficiency maximization.

7 References

Bertsekas, D., and R. Gallager. (1992). *Data Networks*. Prentice-Hall.

Blake, F., D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. (1998). An architecture for differentiated services, IETF RFC 2475.

Cocchi, R., D. Estrin, S. Shenker, and L. Zhang. (1991). A Study of Priority Pricing in Multiple Service Class Networks. *Proc. ACM SIGCOMM*, pages 123-130.

Cocchi, R., S. Shenker , D. Estrin, and L. Zhang. (1993). Pricing in Computer Networks: Motivation, Formulation, and Example. *ACM/IEEE Transactions on Networking*, Vol 1, pages 614-627.

Gupta, A., D. Stahl, and A. Whinston. (1997). Priority Pricing of Integrated Service Networks. *Internet Economics*, MIT Press, pages 323-352.

Harte, L., R. Kitka, and R. Levine. (2002). *3G Wireless Demystified*. McGraw-Hill.

Huitema, C. (1995). *Routing in the Internet*. Prentice-Hall.
ITU. (2002). ICT Free Statistics on Information Technology. http://www.itu.int/ITU-D/ict/statistics/at_glance/Internet02.pdf
Kelly, F. (1997). Charging and Rate Control for Elastic Traffic, *European Transactions on Telecommunications*, Vol 8, pages 33-37.
Kelly, F., A. Maullo, and D. Tan. (1998). Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability, *Journal of the Operation Research Society*, Vol 49, pages 237-252.
Mackie-Mason, J., and H. Varian. (1995). Pricing Congestible Network Resources. *IEEE JSAC*, Vol. 13, No 7, pages 1141-1149.
Mackie-Mason, J., L. Murphy, and J. Murphy. (1997). The Role of Responsive Pricing in the Internet. *Internet Economics*, MIT Press, pages 279-303.
Mandjes, M. (2003). Pricing Strategies under Heterogeneous Service Requirements. *IEEE INFOCOM*.
Mendelson, H. (1985). Pricing Computer Services: Queuing Effects. *Communications of the ACM*, Vol. 28, No 3, pages 312-321.
Mendelson, H., and S. Whang. (1990). Optimal Incentive-Compatible Priority Pricing for the M/M/1 Queue. *Operations Research*, Vol. 38, No 5, pages 870-883.
Molina, M., P. Castelli, and G. Foddis. (2000). Web Traffic Modeling Exploiting TCP Connection's Temporal Clustering through HTML-REDUCE, *IEEE Network*, Vol 14, No 3, pages 46-55.
Odlyzko, A. (1997). A Modest Proposal for Preventing Internet Congestion. http://www.research.att.com/~amo.
Odlyzko, A. (2001). Internet Pricing and the History of Communications. http://www.research.att.com/~amo.
Russel, S., and P. Norvig. (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall.
Shenker, S., D. Clark, D. Estrin, and S. Herzog. (1996). Pricing in Computer Networks: Reshaping the Research Agenda, *ACM Computer Communication Review*, Vol 26, pages 19-43.

8 Appendix: Notations

μ	Job service rate.
λ_i	Job arrival rate at level *i*.
λ_0	Rate of jobs that are not submitted to the network.
$\{\lambda_i\}$	Arrival rate vector, where λ_k at the *k*-th position represents the arrival rate at *k*-th level.
λ	Total potential job arrival rate ($\sum_{i=0}^{I} \lambda_i = \lambda$).
C	Average job size.
C_j	The size of type-*j* job.
$Pr(j)$	The probability that a job is of type j.
$U_j(t)$	Utility function of a type-*j* job.
P_i	Price of level-*i* service.
$\{P_i\}$	Price vector for the system with *I* levels.
P_B	Price of the network backhaul capacity.
T_{ij}	Expected completion time of a type-*j* job at level *i*.
$\Re$	Revenue.
S^t	$\{\lambda_i\}$ values at time *t*, $\{\lambda_i\}^t$.
$X_{k,j}^{\ t}$	Boolean variable that takes 1 if it is most profitable to send the type-*j* job at level-*k* when the system is at S^t.

$\backslash Pr(k, S^t)$	Probability that a job is sent at level-k when the system is at S^t.
W(S)	Probability of being in state S (where $\Sigma_S W(S) = 1$).