

# How to Market-Manage a QoS Network<sup>1</sup>

Jörn Altmann, Hans Daanen, Huw Oliver, Alfonso Sánchez-Beato Suárez

Market-Managed Multi-Service Internet Group, Internet and Mobile Systems Department, CMSL  
Hewlett-Packard Laboratories

email: [jorn\_almann, hans\_daanen, huw\_oliver, alfonso\_sanchez-beato]@hpl.hp.com

**Abstract-** This paper describes a control mechanism for a future Internet. It is an economic mechanism that enables users to choose different pairs of price/QoS priority levels for network services at the user time scale. User time scale means that prices vary at a rate suitable for human beings to respond to those price changes. The changes might happen on an hourly, daily, or weekly basis. We believe that such an economic control mechanism at medium time scale, which is coupled with a technical rate control mechanism at a short time scale, is a simple and cost-reducing system that can provide quality of service to end-users. We show that the approach proposed enables network service providers to react to network congestion appropriately and provides end-users with high flexibility in service selection. After discussing different Internet pricing principles, we describe the market-managed test network, the QoS pricing and charging software for network services, and the experiments run on a DiffServ network. We present a theoretical model based on Markov chains that can represent the proposed market-managed network. Finally, we present results from the analysis of the theoretical model as well as measurement results from the DiffServ network. We show that the behavior of such a system depends on the user's net benefit, the prices for the network services, as well as on the heterogeneity of the user pool.

## I. INTRODUCTION

In the future, network service providers will be forced to offer more flexibility to their customers in order to survive in the highly competitive Internet services market. Network service providers will have to provide highly customized network services, which allow end-users to change QoS/price choices any time. Flat-rated, tiered pricing plans will not be adequate to deal with these requirements. A larger variety of pricing plans will be necessary.

Clearly, any successful solution for supporting multiple services cannot rely on technical solutions alone but also has to take into account the economic aspects. Different qualities of service must be priced differently. If not, people will always use the best one. Of course, user behavior can be influenced by contracts and policing but this leads to complicated monitoring and policing systems. The economically efficient way to influence user behavior is by giving them the correct financial incentives.

The challenge of designing a market-managed network with service differentiation is to provide an approach based both on pricing and on technical rate control that moves choice to the end-customer and tackles the network problems of traffic management. In addition to this, the system has to be simple and has to generate predictable prices.

One of the main objectives followed during this research has been to determine to what extent our theoretical results

are confirmed by the behavior of real networks. We compare results from a Markov model with measurement results from a DiffServ network. In particular, we investigate the behavior of the market-managed network with respect to the user's net benefit, the prices for the network services, and the heterogeneity of the user pool.

Little work has been published on modeling pricing of service-differentiated networks. Park and Chen [9] analyzed a system where packets get marked according to customer's QoS requirement. Therefore, the costs are only performance related. Kelly [5] analyzed a system where packets get marked according to the congestion on the network. Since end-users have to pay for the number of marked packets, the system provides proportional fairness. The theoretical framework that we consider here is similar to Odlyzko's Paris Metro Pricing [8] for providing QoS on the Internet. The network under Paris Metro Pricing is partitioned into several logically separate channels. The channels only differ in the price to be paid for using them. Marbach [6] analyzed the equilibrium of such a system, using non-cooperative game theory. The theoretical model considered in this paper is based on queuing theory. We are using Markov models to simulate the performance of the network and compare it to real measurements. Compared to the non-cooperative game theory approach, this approach is more realistic to real router behavior. Paschalidis and Tsitsiklis [10] showed that a system with a small number of QoS priorities would be sufficient to express user preferences.

The next section describes the problem inherent with a market-managed network more fully, discussing pricing principles in the Internet services market and giving design motivations of our user time scale approach. Section III explains the design of our approach, the implementation choices, as well as the hardware and software architecture. After describing the theoretical model in section IV, we close by presenting the experimental results and analysis results in section V.

## II. PRICING PRINCIPLES

The principles that we have adopted as requirements for our pricing system have emerged from User Studies conducted in M3I [7] and INDEX [2].

### A. *The End-User Knows the Value*

Traditional approaches to providing QoS have defined various network service classes according to the needs of some categorization of Internet applications [3]. This moves from the current 'one size fits all' to differentiated processing

<sup>1</sup> This research was partly funded by the European Union's Fifth Framework Program: Project M3I – RTD No IST-1999-11429.

of packets. It still treats packets identically even if those packets are of very different values to the end-users. We want to leave the choice of how valuable a packet is up to the end-user. It will depend on what specific task they are attempting to complete as to how valuable the packet is.

### B. Control over Service and Price Selection

If network service providers do not control or even know what content their users transmit they can only differentiate themselves by customizing their network services. Customization means that they have to differentiate their offered network services and offer their customers flexibility in selecting qualities of network service.

Our studies have shown that users value this choice. There is a preference for a cheap, best effort service but users want to be able to choose a high quality of service for important tasks (and are willing to pay for it) [1].

If different levels of quality of network service are offered then they have to be priced differently (otherwise everyone will always use the best one).

By offering QoS selection on demand, end-users benefit from a broader selection of services. Network service providers also benefit from the surplus end-users are willing to pay if they can meet their needs.

### C. Usage-Based Charging

In the future we foresee an “always-on” Internet. Duration based charges will make no sense. Pure flat fee, “all you can eat” approaches have also been shown to be economically inefficient and tend to subsidize heavy users.

Usage-based Charging is a more appropriate means of allocating scarce resources among those end-users who value the service more. If end-users are charged a usage-based fee for their Internet usage, information will only be downloaded that is of higher utility to the user than the incurred costs (i.e. money spent on downloading or time spent waiting for the download to finish). Since the end-users’ incentive is to maximize their utility when using the Internet, each information download will be evaluated with regard to its net benefit.

Considering these aspects, we foresee that higher priority services will always be charged on a usage basis, even in the presence of a flat-rated, best-effort basic service.

### D. Price Predictability

Users don’t like unpredictable bills. Important design aspects of a QoS network include price predictability for the services offered. This is best done by giving the user direct access to a price/QoS selection that offers a service at a price that varies only over long time scales. That this is an essential aspect of pricing is shown in Altmann [2].

### E. Frequency of Price Changes

Related to predictability is the frequency at which prices for services change. The frequency of price changes can be classified according to the time scales at which they change. The time scales we distinguish are *network time scale*

(changing each second or faster), *user time scale* (changing hourly, daily, or weekly), and *long-term time scale* (only changing every month or year).

Methods that operate on network time scales change prices in real time as transient network overloads occur. While this allows congestion costs to be conveyed to the user at a fine grain and in real time, we see two drawbacks:

- The user cannot (and would find it tiresome to) respond sufficiently quickly to such rapid changes.
- It introduces greater unpredictability in the final charge for the service usage.

Long-term time scale approaches are valuable in order to make investment decisions for upgrading the network, defining long-term QoS contracts. The goal is to predict the growth rate of the network traffic and find a compromise between times of congestion and times of under-utilization. The basis of these approaches is the analysis of aggregated usage patterns. The network is upgraded whenever the predicted needs exceed the current capacity. Prices of network services only change very rarely. Although this approach helps to limit the cost of network over-provisioning, the disadvantage of this approach is that it does not consider the daily/hourly fluctuations of traffic. Considering the network traffic of residential users, the difference between the traffic peaks during daytime and the lows at night is a factor of 4 [2].

The user time scale approach addresses these issues. User time scale means that prices for Internet services can change on an hourly, daily, or weekly basis, so that end-users can respond to those prices. Under this approach, end-users can express the value they place on the network service. For example, end-users could postpone ftp downloads until prices for network services go down (e.g. time of day pricing), or they could reduce the peak transmission rate, or the amount of data sent. However, prices of network services are known to the end-user in advance and vary slow enough for them to keep track. From the network service provider’s perspective, the user time scale approach reveals more information about the usage pattern of their customers compared to the long-term time scale approach.

## III. OVERVIEW OF OUR APPROACH

In our basic scheme (Figure 1), the user is offered a selection of priority level/price pairs. The traffic will be sent at the priority level chosen for the offered price. However, the absolute quality of service of each priority level is not guaranteed. The quality of service depends on the current state of the network. Therefore, the differences are relative and may change in real-time. Based on that, the user may choose to switch to higher or lower priority levels. As lower priority levels become congested there will be more incentive to move up the priority levels. As congestion eases there will be lower incentives to pay the higher prices at the higher levels and, therefore, the user will switch to lower priorities.

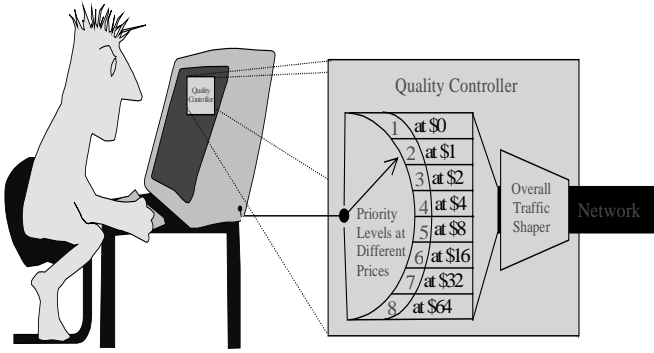


Figure 1. User Chooses Priority-Level/Price Pair

The basic pricing plan is quite straightforward. Each priority level is priced at a different rate. The prices are strictly increasing with regard to the priority. Initially, prices are set based on historic data, but they can be varied over longer time scales according to the usage patterns observed (i.e. with regard to the price elasticity of end-users). The perceived network load is different at each priority level.

By changing the prices for the different priority levels, the demand for each priority level can be controlled. In order to get feedback on the demand, the usage of the different priority levels will be monitored and recorded. Usage will be accounted, for example, as the number of bytes transmitted, the number of packets transmitted, or the amount of time connected to the network. More complex pricing plans can be generated based on these basic pricing plans.

The end-user may wish to use a traffic shaper that will tailor his traffic according to the charging scheme. For example, if the pricing plan charges for peak transmission rates, the user may want to shape the traffic of the individual applications and the overall traffic he generates in order to reduce the costs. Charging for peak transmission rates will be very important as soon as end-users will have high capacity, always-on physical links to the Internet, capable of transmitting data at rates higher than 1.5 Mbit/sec. It will stop end-users from injecting large bursts of traffic into the network, which could cause its collapse.

There are different alternatives for traffic shaping. If the individual applications are aware of the pricing plan they can adapt their own traffic, otherwise a per application traffic shaper needs to be available. The same goes for aggregate traffic. If the user needs to keep his overall traffic within certain profiles this can be done either by monitoring the aggregate traffic and giving feedback to the individual applications or by an overall traffic shaper.

#### A. Pricing and Charging Architecture

The components of the pricing and charging software can be grouped into two parts (Figure 2). One part runs on the end-user's computer and the other part runs on the network service provider's network.

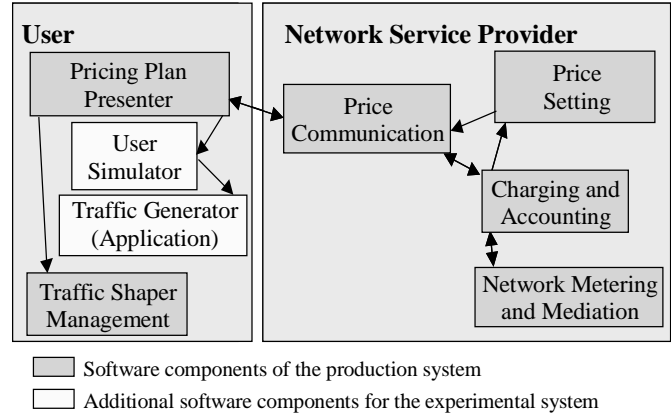


Figure 2. Software Architecture

#### Network Service Provider Software Components

The software running on the network side comprises components for **Network Metering and Mediation**, **Charging and Accounting**, **Price Setting**, and **Price Communication**.

- *Network-Metering-and-Mediation Component:* The Network-Mediation component, which feeds into the Charging-and-Accounting component, aggregates usage data according to criteria defined by the user-selected pricing plan. HP's Internet Usage Manager (IUM 2.02) is used for the mediation system. The Network-Mediation component's input comes from the network monitor NeTraMet (44b8-ecn-1.2), running on routers and metering the DiffServ CodePoint of every packet.

- *Charging-and-Accounting Component:* The Charging-and-Accounting component [11] is responsible for accounting for network services consumed by each user. It also rates the end-user usage data according to the choice of priority level.

The accounting records generated by IUM are interpreted by the accounting component and stored in a database. These usage records are processed by the charging component by applying the appropriate tariff (as communicated by the Price-Setting component). This will generate charges that are stored in a database as well to determine the input for an eventual bill.

- *Price-Setting Component:* The Price-Setting component [4] calculates and sets the prices for network services.

Its input comes from the pricing policy specified by the network service provider and the Charging-and-Accounting Component. The pricing policy specifies the overall pricing structure. The Charging-and-Accounting component provides usage information. The output (tariffs and characteristics of a session) is communicated to the Charging-and-Accounting component.

- *Price-Communication Component:* This component offers services at the calculated prices to end-users. After an offer has been accepted by the end-user, the Price Communication component forwards charging information to end-users and the Charging-and-Accounting component, that needs the information to initialize the accounting and rating engine for this user accordingly.

### End-User Software Components

The components of the software running on the end-user's computer are the **Traffic Shaper Management, Pricing Plan Presenter**, and for the purpose of our simulation experiments the **User Simulator**, and the **Traffic Generator**.

- *Traffic-Shaper-Management Component*: The Traffic-Shaper-Management component handles the interaction with the traffic shaper. It could, for example, set the maximum transmission rate.

- *Pricing-Plan-Presenter Component*: The Pricing Plan Presenter presents the prices for the Internet services to the end-user (as well as the accumulated charges). The user sees a simple selector for the priority level. After a selection has been made, the acceptance of the offer is sent to the Price-Communication component of the network service provider.

- *User Simulator*: The user simulator, which would not be deployed in any production system, was only implemented in order to conduct our experiments. The user simulator represents a number of end-users with a certain demand for network services. The demand functions (respectively utility functions) differ in nature according to the application and the ultimate value depends on the task in hand. We have experimented with populations of traffic profile parameters.

For a file transfer, for example, the user simulator observes the available bandwidth at different priority levels and, then, based on the observation, calculates an estimate of the time of transmission of a file. Afterwards, the user simulator can calculate the utility versus the cost of transmission. Depending on the result, it will either not start the transmission or will choose the optimal priority and send to the traffic generator the calculated traffic profile parameters for this flow.

An issue is how to get current transmission rates for each priority level if there has not been a transmission over a longer time period. Solutions include keeping a database of historic data or probing the network.

- *Traffic Generator*: The traffic generators generate TCP or UDP packet flows. A new process is created for each flow. It also inserts the priority level chosen by the user into the IP header.

### B. QoS Implementation

The implementation alternatives come from QoS technologies such as IntServ or DiffServ. DiffServ with pre-marking provides a natural way of implementing our scheme. We are using ALTQ3.0 as our queue management software on the routing machines. ALTQ is configured to use a strict priority queuing discipline. The packets with the highest priority in the queue get processed first. If no highest priority packets are left in the queue, packets with the next highest priority are sent.

The traffic shaper (or the IntServ router, which works as a traffic shaper) is located between the customer's computer and the DiffServ router. For a production system, this raises the question whether it should reside on the NSP network or on the end-user's premises. For our experimental system, it does not matter. In either case, for setting the preferred transmission rate, the end-user has to have access to configure the traffic shaper.

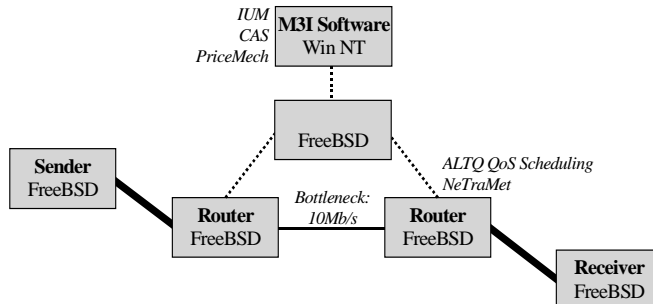


Figure 3. Test Network Architecture

### C. Network Architecture

All components of the test network run on eight standard PCs that are connected via a 100 Mbit/s Ethernet switch.

As shown in Figure 3, The WinNT machine runs the M3I software such as the mediation software (IUM), the charging and accounting software (CAS), and the price setting and communication software (PriceMech). Two machines, which run ALTQ, are used as routers for the experiments. In addition to this, one of them runs the network monitor NeTraMet so that data sent from the Sender machine to the Receiver machine can be metered at the network node.

## IV. THE THEORETICAL MODEL OF THE MARKET-MANAGED NETWORK

### A. General

We assume a system in which jobs arrive with exponential inter-arrival distribution and different levels of QoS are provided. An index attached to each job describes the relative importance between jobs. Depending on that index, the service received by each job will be different. When there is an arrival, the job decides which priority to choose (job priority level) or to exit the system without being executed.

There will be  $M$  different levels of QoS, where  $1$  is the lowest precedence and  $M$  is the highest. Only one type of job occupies the processor at a given time. We use pre-emptive priorities and processor sharing (for jobs with the same priority level). Three cases have to be considered under pre-emptive priorities, if the server is occupied with priority  $k$  jobs and another job arrives that chooses priority  $l$ :

- a)  $l > k$ : priority  $k$  jobs will leave the server until all jobs with higher priority are finished executed.

- b)  $l < k$ : the job with priority  $l$  will wait until the server has executed all higher precedence jobs.

- c)  $l = k$ : all jobs that are already in the system and the new arrival will share the processor equally.

This is a close approximation to the strict priority queuing discipline that we use for the packet scheduling at the routers, since we assume that jobs are composed of a high number of packets.

An exponential distribution for the length of the jobs is assumed. The distribution has a mean  $\mu$  for the time that the server takes to finish one transfer, assuming only one transfer in the system and there are no new arrivals during the service time. As the nature of the jobs is the same independent of the

priority level they choose, we will not have a different distribution for jobs at different priorities. The processor is shared between jobs with the same priority in such a way that if there are  $n$  jobs with that priority, each job occupies a server with rate  $\mu/n$ . This means that the distribution of the time in the system for a job varies depending on the state. However, the total output rate for each state where there is at least one job present is  $\mu$ , as that is the sum of the output rates for all the servers in that state.

We define also  $K$  different types of jobs. The arrival of each job type into the system follows a Poisson process. The rates are  $(\lambda_1, \lambda_2, \dots, \lambda_K)$ . Jobs that belong to the same type have the same utility function, representing the sender's willingness to pay for the execution of the job. The job is either executed with a certain priority or not, depending on the utility function.

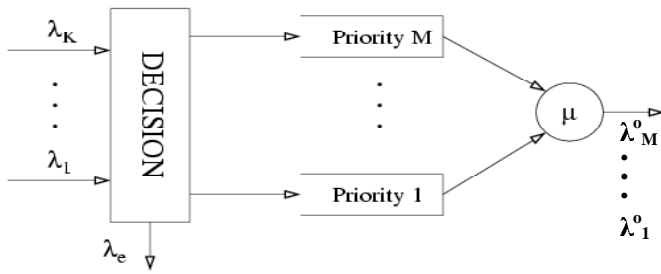


Figure 4. System Model

Finally, let  $\tilde{N} = (n_1, n_2, \dots, n_M)$  be the number of jobs in the system for each priority. This vector defines the state space for the Markov chain. The possible transitions between the states are the arrivals defined by  $(\lambda_1, \lambda_2, \dots, \lambda_K)$  and the departure rate  $\mu$ . The priority of the job that occupies the server in a given state  $\tilde{N}$ , where  $\tilde{N} \neq 0$ , will be:

$$s(\tilde{N}) = \max \left\{ j = 1, \dots, M \mid n_j > 0 \right\} \quad (1)$$

Upon an arrival of a new job in the system, its priority has to be chosen. A job of type  $i$  has a utility function  $U_i(t)$ , where  $t$  is the time spent until the job has finished. This function is monotonously decreasing with  $t$ . Let  $(p_1, p_2, \dots, p_M)$  be the price of the job execution at each priority, where  $p_1 < p_2 < \dots < p_M$ . We will also define the residual utility for class  $i$  jobs and priority  $j$  in state  $\tilde{N}$  as

$$U_{i,j}^R(\tilde{N}) = U_i(\tilde{T}_j(\tilde{N})) - p_j \quad (2)$$

where  $\tilde{T}_j(\tilde{N})$  is an estimation of the job completion time at precedence  $j$  in state  $\tilde{N}$ .

The priority choice for a job of type  $i$  in state  $\tilde{N}$  can be expressed by  $d_i(\tilde{N})$ . It will be 0 if  $U_{i,j}^R(\tilde{N}) \leq 0$  for all  $j$ , indicating that the transfer is not executed, as no surplus from sending the file would result. In any other case,  $d_i(\tilde{N})$  will represent the lowest priority level at which the highest utility is gained from sending the file.

$$d_i(\tilde{N}) = \begin{cases} 0 & \text{if } U_i^R(\tilde{N}) \leq 0 \\ \min \left\{ j \mid U_{i,j}^R(\tilde{N}) \in U_i^R(\tilde{N}) \right\} & \text{otherwise} \end{cases}$$

$$\text{where } U_i^R(\tilde{N}) = \max \left\{ U_{i,j}^R(\tilde{N}) \mid j = 1, \dots, M \right\} \quad (3)$$

The problem to be solved now is the calculation of the estimated completion times  $\tilde{T}_j(\tilde{N})$  if the job chooses priority  $j$ . In order to do this, we need to introduce three definitions.

**a)** We define two random variables  $t_j(\tilde{N})$  and  $\tau(\tilde{N})$ .  $t_j(\tilde{N})$  is the remaining system time for a job with priority  $j$  when the system is in state  $\tilde{N}$ .  $\tau(\tilde{N})$  which is the remaining time in state  $\tilde{N}$  until the next transition. Because the system is memory-less these values are independent of how long we are in state  $\tilde{N}$ .

**b)** We define  $P_i^{arr}(\tilde{N})$  as the probability that the first transition that takes place in  $\tilde{N}$  is an arrival of a class  $i$  job and  $P^{dep}(\tilde{N})$  as the probability that the first transition is caused by a departure. These probabilities can be easily obtained through the frequencies of the transitions that go out of  $\tilde{N}$ . Besides, we define the departure probability  $P_j^{out}(\tilde{N})$  of a specific job with priority  $j$  in state  $\tilde{N}$  if there is a departure in that state:

$$P_j^{out}(\tilde{N}) = \begin{cases} \frac{1}{n_j} & \text{if } s(\tilde{N}) = j \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

**c)** As the last definition we introduce  $e_j$ , the unitary vector with a 1 in position  $j$  and 0 elsewhere.

The equation for  $t_j(\tilde{N})$  can easily be derived, since there are only three possibilities to depart from state  $\tilde{N}$ : arrival of a new job, completion of another job or completion of the specific job. Therefore, we get

$$\begin{aligned} t_j(\tilde{N}) = & \sum_{i=1}^K P_i^{arr}(\tilde{N}) \cdot \left( \tau(\tilde{N}) + t_j(\tilde{N} + e_{d_i(\tilde{N})}) \right) + \\ & \left( 1 - P_j^{out}(\tilde{N}) \right) \cdot P^{dep}(\tilde{N}) \cdot \left( \tau(\tilde{N}) + t_j(\tilde{N} - e_{s(\tilde{N})}) \right) + \\ & P_j^{out}(\tilde{N}) \cdot P^{dep}(\tilde{N}) \cdot \tau(\tilde{N}) \end{aligned} \quad (5)$$

As an estimate for the completion time  $\tilde{T}_j(\tilde{N})$  at a priority  $j$  we use the mean of the job completion time  $T_j$  on the condition that the system is in state  $\tilde{N}$ . This is equal to the average remaining system time  $t_j$  after the system moved to the state  $\tilde{N} + e_j$ .

$$\tilde{T}_j(\tilde{N}) \stackrel{\Delta}{=} E[T_j \mid \tilde{N}] = E[t_j(\tilde{N} + e_j)] \quad (6)$$

Using this definition in combination with equation (5) results in:

$$\begin{aligned}
E[T_j | \tilde{N}] &= \sum_{i=1}^K P_i^{arr}(\tilde{N} + e_j) \cdot \left( \Gamma(\tilde{N} + e_j) + E[T_j | \tilde{N} + e_{d_i(\tilde{N} + e_j)}] \right) + \\
&\left( 1 - P_j^{out}(\tilde{N} + e_j) \right) \cdot P^{dep}(\tilde{N} + e_j) \cdot \left( \Gamma(\tilde{N} + e_j) + E[T_j | \tilde{N} - e_{s(\tilde{N} + e_j)}] \right) + \\
&P_j^{out}(\tilde{N} + e_j) \cdot P^{dep}(\tilde{N} + e_j) \cdot \Gamma(\tilde{N} + e_j) \quad (7)
\end{aligned}$$

where  $\Gamma(\tilde{N}) = E[\tau(\tilde{N})]$ , the mean of the remaining time in state  $\tilde{N}$  until the next transition.

Note all these values are dependent on the specific vector of arrival rates.

### B. Finding the Solution for a Simple System

In this part we will describe how to find a solution for the Markov chain for a system with 2 priorities and 1 type of job (one utility function).

To obtain the solution for the estimations we have to formulate (7) for each state and solve the linear system. In general, no analytical solution can be found. This means we have to resort to numerical methods. There are two problems that complicate solving this system.

The first problem in solving the linear system is that it is infinite. Fortunately, charging in combination with strict priorities helps us here. As there are more and more jobs in the system  $T_j(\tilde{N})$  will grow unbounded. Since the utility functions are decreasing and the prices are fixed, equation (2) states that there is a moment in which the system is so full that it only makes sense to send at the highest priority and finally not to send at all.

Considering our two-priority system, where states have the format  $(n_1, n_2)$ , with large  $n_1$  and small  $n_2$ , the decision will be to send at high priority only. If  $n_2$  gets even larger, the decision is not to send at all. For states with small  $n_1$  and large  $n_2$  the decision will be not to send either.

Thus, assuming a state  $(n_1, n_2)$  where  $n_1$  is infinite, we create a row representing decisions to send with high priority ( $n_2$  increases) and not to send if  $n_2$  gets large. Assuming that such a row in the Markov chain exists, we can generate the equations (7) for high priority for each state in this row. If  $d(n_1, n_2 + 1) = 2$  then we get (8). If  $d(n_1, n_2 + 1) = 0$  then we get (9).

$$\begin{aligned}
E[T_2 | n_1, n_2] &= \frac{\lambda}{\lambda + \mu} \left( \frac{1}{\lambda + \mu} + E[T_2 | n_1, n_2 + 1] \right) + \\
&\frac{\mu}{\lambda + \mu} \cdot \frac{n_2}{n_2 + 1} \cdot \left( \frac{1}{\lambda + \mu} + E[T_2 | n_1, n_2 - 1] \right) + \\
&\frac{\mu}{\lambda + \mu} \cdot \frac{1}{n_2 + 1} \cdot \frac{1}{\lambda + \mu} \quad (8)
\end{aligned}$$

$$E[T_2 | n_1, n_2] = \frac{n_2}{n_2 + 1} \cdot \left( \frac{1}{\mu} + E[T_2 | n_1, n_2 - 1] \right) + \frac{1}{n_2 + 1} \cdot \frac{1}{\mu} \quad (9)$$

It can be seen that these equations do not depend on states that are above or under the row created. Decisions taken in rows with large  $n_1$  are exactly the same, i.e.  $E[T_1 | n_1, n_2]$  will increase for larger  $n_1$  but  $E[T_2 | n_1, n_2]$  will remain the same. This leads to a Markov chain as shown in Figure 5, for states with large  $n_1$  and  $n_2$ . The number in each state is  $d(\tilde{N})$ .

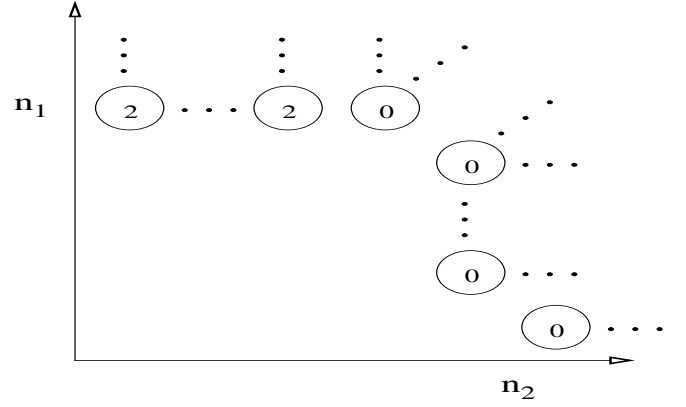


Figure 5. Markov Chain with a Large Number of Low Priority Jobs  $n_1$  and a Large Number of High Priority Jobs  $n_2$

The second problem is that a mutual dependency between the decisions and the mean completion times exists. To break this dependency we start with a Markov chain with certain transition decisions and, then, calculate the expected completion time for each state and priority. After that, we use (3) to check whether the transition decision was correct in each state. If the transitions were not correct, we change the transitions in the Markov chain to make them coherent with the optimal decision, and re-compute the times. We have to repeat this process until we find a Markov chain in which all the transitions are consistent with the optimal decision in each state.

In more detail, in order to find the exact shape of the Markov chain, we first need to generate the row that goes through the state of the format  $(n_1^*, n_2)$  with  $d(\tilde{N}) = 2$ , where  $n_1^*$  is the first  $n_1$  for which  $d(n_1, 0) = 2$ . We do that by starting with a row, where only the first state  $(n_1^*, 0)$  has  $d(\tilde{N}) = 2$  and all the other states in the row have  $d(\tilde{N}) = 0$ . We work out the completion times at high priority for these states, and check if the decisions are coherent with the optimal decisions. If the decisions taken are not optimal, we increase the number of states of the form  $(n_1^*, n_2)$  with  $d(\tilde{N}) = 2$  by one. We repeat this process until we obtain a chain where all decisions are optimal. The state  $\tilde{N}$  of the optimal chain, where  $d(\tilde{N}) = 0$ , is  $(n_1^*, n_2^*)$ .

To find the rest of the Markov chain we follow a similar iteration. Taking the Markov chain that we just obtained, we check the imposed decisions of each state in row  $l$ . If there are wrong decisions, we change the imposed decisions accordingly. In case the length of the chain ( $n_2^*$ ) is not sufficient, i.e. the decision in the last state is not zero, we add states to the Markov chain until the decisions are optimal. We repeat adding another row until we find a solution, i.e. the last row added is equal to the row  $(n_1^*, n_2)$ .

This method does not always give a consistent solution. It can oscillate between two solutions or give more than one solutions. In the latter case, we calculate the other extreme of the solution space by starting the iteration from large  $n_1$  instead of  $n_1 = 1$ .

### C. Calculating Stationary Output Rates

The solutions that are found by the described method have a similar structure. It is obvious that the whole state space does not form an irreducible Markov chain. In fact there is only one irreducible Markov chain. The states that are not in this chain can be seen as transitory states that have paths to the irreducible chain but no paths out of it, so they do not influence the stationary behavior. Considering the case with one type of utility function and two priorities, the states that form the irreducible Markov chain are:

- The states along the  $(n_1, 0)$  axis where  $d(\tilde{N}) = 1$
- The first state  $(n_1^*, 0)$  where  $d(\tilde{N}) = 2$
- The states  $(n_1^*, n_2)$  where  $d(\tilde{N}) = 2$
- The first state  $(n_1^*, n_2^*)$  where  $d(\tilde{N}) = 0$

The described chain has the same transitions as an M/M/1/K queue, where  $K = n_1^* + n_2^*$ , which makes it very easy to analyze. If  $\pi_{(n_1, n_2)}$  is the stationary probability of being in state  $(n_1, n_2)$  for the irreducible Markov chain, the output rates for priority 1 and 2 and the exit rate in the system are as follows:

$$\lambda_1^o = \mu \cdot \sum_{i=1}^{n_1^*} \pi(i, 0) \quad (10)$$

$$\lambda_2^o = \mu \cdot \sum_{i=1}^{n_2^*} \pi(n_1^*, i) \quad (11)$$

$$\lambda_e = \lambda \cdot \pi(n_1^*, n_2^*) \quad (12)$$

## V. SYSTEM BEHAVIOR ANALYSIS

Within this section, we analyze the behavior of the market-managed system with regard to residual utility, price, as well as user heterogeneity.

We make two simplifying assumptions while analyzing the market-managed network. We consider only one or two different user groups that are characterized by having linear utility functions, and a system, which supports only two priorities (high (2) and low (1) priority).

On the test network the traffic generator creates files with exponential file length and exponential inter-arrival times. The rate of the server  $\mu$  is the quotient between the data transmission rate in the bottleneck link and the mean file size. When there is an arrival and the decision is to send the file, a new process is created and a transmission starts. The decision whether to send or not to send the file depends on previously measured values of throughput at different priorities. When a transfer has finished, the rate that has been obtained is saved. In addition to this, a mean rate is calculated every 0.5 seconds. The mean rate is based on the rates collected during the last 10 seconds. To predict the transmission rate, we calculate the weighted sum of the last mean rates. The linear weights are bigger for the latest rates.

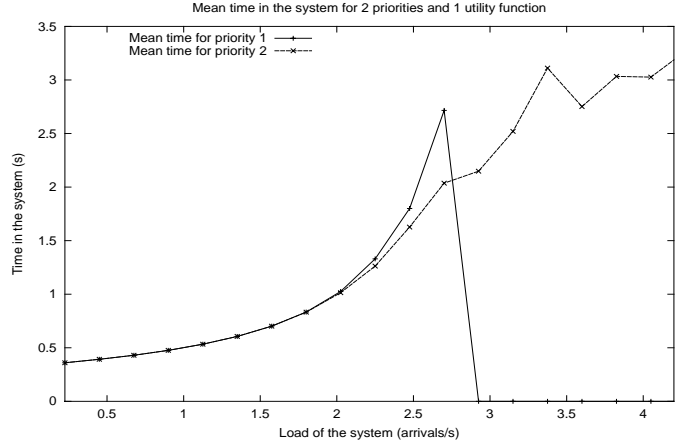


Figure 6. Predicted Transmission Time in the System

### A. Residual Utility and System Behavior

In order to demonstrate how the residual utility determines the behavior of the system, we analyze the average time of a job in the system in the theoretical model. Figure 6 displays the times for both priorities. For the analysis, we consider a system with the service rate  $\mu = 3$ , and the utility function  $U_1(t) = 10 - t$ , where  $t$  is the transmission time of a job in the system. The price for sending a file at low priority is  $p_1 = 3$ . The price for sending it at high priority is  $p_2 = 6$ .

Figure 6 shows that the times are almost the same until the load of the system is  $\lambda = 2$ . At this point, the times start to differ since the number of jobs at high priority became large enough in order to delay the execution of jobs in the low priority class. If the load of the system increases further (up to 2.7), no low priority jobs remain in the system (therefore, no values in the graph).

Note that the times for the high priority class stay below 4. This value is the threshold where the residual utility for that priority becomes zero. The maximum transmission time for low priority jobs is near to 3. This is due to the difference of residual utilities of the two priorities. When the transmission time is near 3, the residual utility for that priority is 4. That is equal to the maximum utility of high priority jobs. Consequently, as the residual utility for sending at high priority is approximately that for sending at low priority, some users decide to go for the high priority service. If the level increases even further, all users go for high priority.

In a best-effort system with one priority and no charging the results are totally different. The completion time of the job in the system increases to 10 second (according to the utility function  $U_1(t)$ ) if the arrival rate approaches the maximum service rate.

When we measure the transmission times for jobs on the DiffServ test network we get a similar result. For these measurements, we took the same parameter setting as for the theoretical model. Figure 7 shows the experimental times.

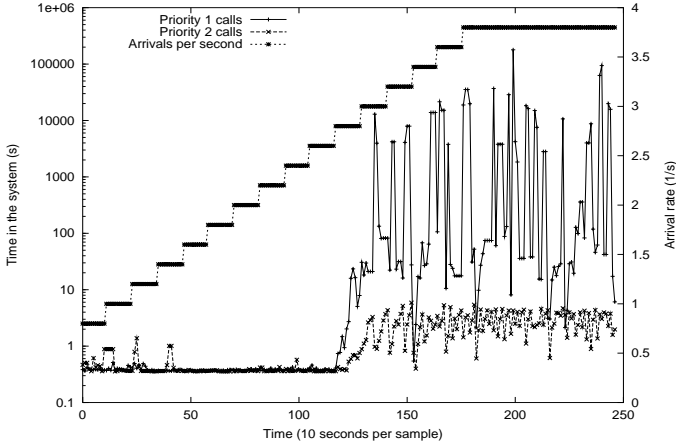


Figure 7. Measured Transmission Time in the System

Comparing the experimental transmission times (Figure 7) and those obtained from the theoretical model (Figure 6), the times for both priorities are similar under low system load, approximately 0.3-0.4 seconds. However, the experimental times do not increase smoothly with  $\lambda$  as in the model. There is a sudden increase of times if  $\lambda = 2.8$ . The times for high priority are around 2. If the load of the system increases further, the times for the high priority jobs stay in the range of 2 to 4, as predicted in the model.

Only in order to get regular estimates for the throughput at high and low priority, we randomly send some jobs at both priorities. These jobs are responsible for the high spikes in the transmission times of low priority traffic for loads larger than  $\lambda = 2.7$ . Without these probes, we would not get any current values for the throughput at all.

From these measurements we can state that if we consider residual utility in the decision making process before transmitting a file, we can make predictions about the system behavior. The transmission times of jobs in the system will not exceed the bounds given by the utility function.

### B. Prices and System Behavior

Figure 8 illustrates the theoretical output rates at low and high priority at different arrival rates  $\lambda$ , varying from 0.25 to 4.5. The price of the low priority is  $p_1 = 3$  and the price of the high priority is  $p_2 = 6$ . The rate of the server is  $\mu = 3$ . Since the network model has sometimes more than one Markov chain as a possible solution, we plotted the results obtained from the maximum and minimal Markov chain. However, the results of both Markov chains differ only slightly.

As the numerical solution in Figure 8 shows, arriving jobs choose the low priority service if the utilization of the system is low, but change to high priority service if the load increases. The user's choice of priority changes within a quite narrow range of the system load. At all other system loads, one priority is predominant. Another interesting fact is that jobs decide to send at high priority even if the system is not at the load where both low and high residual utility are equal. This is due to the stochastic nature of the network model. As the arrivals are randomly distributed, it is always possible to arrive at a full system even if the load is fairly low.

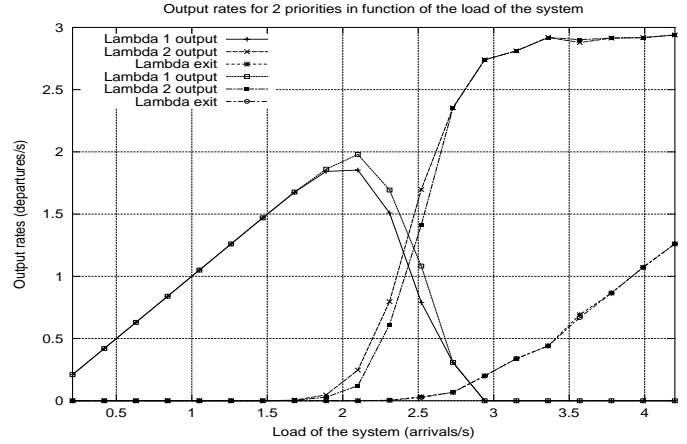


Figure 8. Numerical Result for a System at Different Loads

Figure 9 shows the experimental results for a system with the same parameters as used for the theoretical model. The switch between priority 1 and priority 2 is even more abrupt than predicted in the theoretical model. There is no load at which both priorities execute a significant amount of jobs simultaneously.

Considering these results, the question arises which factors affect the load of the system and the switching behavior of users. Since the choice of high or low priority depends on the residual utility received by the user, we start to examine the affect of prices of different priority classes.

As Figure 10 illustrates, the ratio between the slope of the utility curve and the slope of the cost for using the two priorities determines the user's choice of the priority. If the slope of the utility function is greater than the slope of the cost function, the user will choose high priority. If the slope is less the user will choose low priority. In case the two slopes are equal, either choice is optimal. This point is called crossover point.

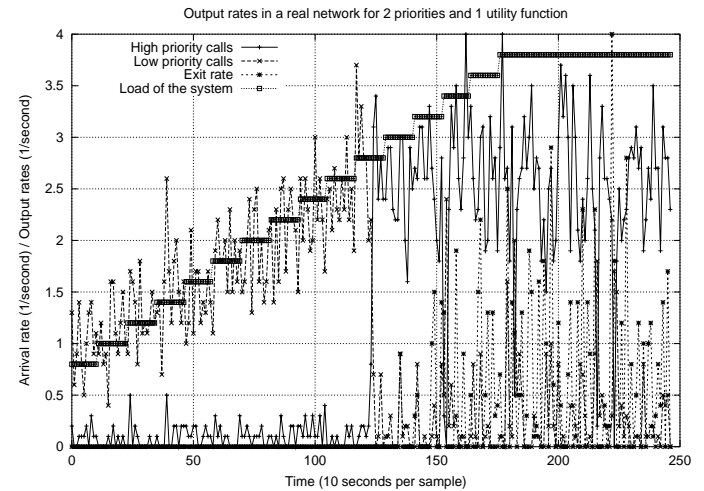


Figure 9. Measurement Result for a System under Loads

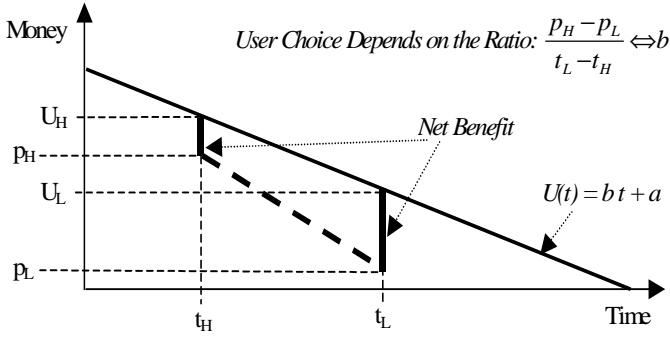


Figure 10. Interdependency between Utility and Prices

In order to simulate this, we take the parameter setting of Figure 8 and solve the Markov chain for different prices. Figure 11 shows the output rates for both priorities at different prices as a function of the load of the system. The price difference between high and low priority vary from 1 to 7, i.e. the price pairs go from (4.5, 5.5) to (1.5, 8.5). The crossover points, which is marked as a dot in the figure, determines the load at which the output rate for low and high priority are the same.

Figure 11 shows that the crossover points move to higher loads if the difference between the price for high priority and the price for low priority increases. That demonstrates that the load at a certain priority can be controlled by relative price differences between the priorities. Users stay longer within a low priority service if the cost for choosing the next higher priority level is relative high. In general, assuming fixed transmission times for high and low priority, the choice depends on the ratio  $(p_H - p_L) / b$ , where  $b$  is the slope of the utility function,  $p_H$  the price for the high priority service, and  $p_L$  the price for low priority (Figure 10).

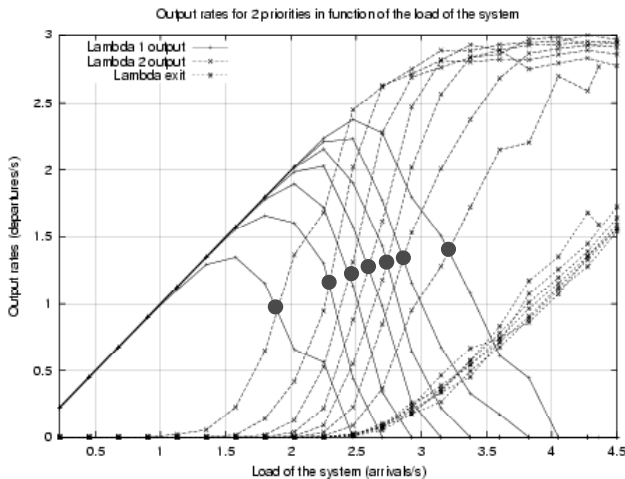


Figure 11. Numerical Results for the Crossover Points

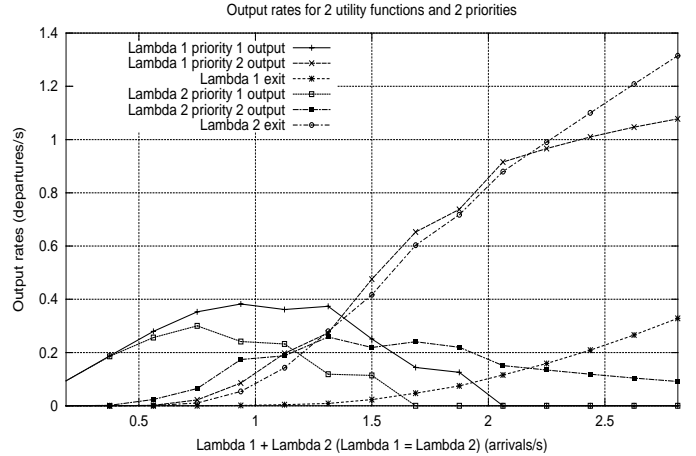


Figure 12. System Behavior under Two User Groups and Equally Increasing Arrival Rate for Both Queues

### C. User Heterogeneity and System Behavior

In order to illustrate how the system behavior changes if the pool of users is heterogeneous, we theoretically analyze a system with two different linear utility functions (i.e. two user groups). For this model, we use the service rate  $\mu = 1.25$ , the utility functions  $U_1(t) = 10 - t$  and  $U_2(t) = 10 - 2t$ , the price  $p_2 = 7$  for high priority service, and the price  $p_1 = 3$  for low priority service.

For both utility functions, Figure 12 shows the exit rates, the output rates for high priority, and the output rates for low priority. The sum of the arrival rates  $\lambda_1$  and  $\lambda_2$  varies between 0.2 and 2.7, while  $\lambda_1 = \lambda_2$ .

Figure 12 shows that the output rates at high priority for the two utility functions differ significantly. While the output rate as a function of system load for utility  $U_1(t)$  increases, the output rate for utility  $U_2(t)$  decreases. That illustrates that the higher the arrival rates are, the more capacity is consumed by users with utility function  $U_1(t)$ . Comparing the utility functions, it becomes obvious that users with utility function  $U_1(t)$  receive a higher utility from using the service than users with utility  $U_2(t)$ . It also demonstrates that this system prefers users with high utility to users with low utility. The exactly same conclusions can be drawn, if the output rates at low priority for both user groups are compared.

This system behavior basically equals the behavior of an admission control system. The admission to this system depends on the valuation that a user places on the service. The resource is allocated to those users, who value the service most. The user's service valuation is also reflected in his willingness-to-pay. Users with utility  $U_1(t)$  are willing to pay more than users with utility  $U_2(t)$ .

Summarizing, Figure 12 shows that users who pay a higher price or value the service most are preferred in the resource allocation process.

## VI. CONCLUSION

This work studies a future Internet that is a multi-service network. It will support different kinds of applications (e.g. email, real-time audio, file transfer, and streaming video) as

well as different needs of users (e.g. quality of service and flexibility in service selection). The network that we propose is a market-managed network, which is based on an economic control mechanism. It enables end-users to express their preferences for different priorities of network services. Users choose between different priority levels, which are priced at a fixed rate. There is however no guarantee on the service quality delivered at each priority level.

We motivate this approach by discussing other possible solutions considering pricing. Following the description of the idea of our approach, we explain the experimental network as well as the software used for pricing and charging. Beside the measurement data obtained from the experimental network, data comes from a Markov model that we developed to model our approach. The analysis of these data focuses on three issues: residual utility, price, and user heterogeneity.

As our analysis shows, the system behavior changes significantly when the admission control process includes usage-based pricing of network services. For instance, the transmission times of jobs in the system will stay in certain bounds given by the utility function and the price of the service. Assuming that the utility function of users is fixed, the transmission times of jobs can directly be controlled by prices. Therefore, we can state that our market-managed approach is a simple mean to control the load of a network. Since this approach requires only a simple traffic shaper (to control the peak transmission rate) as a technical rate control mechanism, it is a simple and inexpensive solution to provide quality of service on the Internet.

The analysis also provided insight about the decision process of users. The understanding of user behavior is a necessity in order to calculate and set the correct prices for services of such a market-managed network. An interesting finding of our analysis is that the greater the relative price difference between the low and high priority service is the longer people use a low priority service. This knowledge will help to design pricing structures for market-managed networks, which will guarantee congestion-free service at the high priority levels.

Recapitulating the research work presented in this paper, we state that market-managed networks can be implemented and can provide sufficient means for service provider to control the load of their network. We showed that our market-managed network allocates resources to those users who have the highest valuation of services and are willing to pay higher prices. Users with a lower valuation of services leave the network.

#### ACKNOWLEDGEMENTS

The M3I Project (Market-Managed Multi-services Internet Project) is funded by the European Union [7]. It aims to design, implement, and trial a software infrastructure that supports charging of differentiated network services. The areas of research comprise accounting and charging, user behavior analysis, pricing mechanisms, and economics modeling. The partners are Telenor, BT, Hewlett-Packard, ETH Zürich, University of Darmstadt, and Athens University of Economics and Business.

#### REFERENCES

- [1] Jörn Altmann and Karyen Chu, "A Proposal for a Flexible Service Plan that is Attractive to Users and Internet Service Providers", *IEEE InfoCom2001, Conference on Computer Communications*, Anchorage, Alaska, USA, April 2001.
- [2] Jörn Altmann, Björn Rupp, and Pravin Varaiya, "Internet Demand under Different Pricing Schemes," *ACM EC'99, Conference on Electronic Commerce (SIGecom)*, Denver, Colorado, USA, November 1999.
- [3] R. Bohn, H.-W. Braun, K. Claffy, and S. Wolff, "Mitigating the Coming Internet Crunch: Multiple Service Levels via Precedence," *SDSC TR GA-A21530*, November 1993.
- [4] Martin Karsten, Jens Schmitt, Lars Wolf, and Ralf Steinmetz, "Provider-Oriented Linear Price Calculation for Integrated Services," *IEEE/IFIP IWQoS'99, International Workshop on Quality of Service*, London, June 1999.
- [5] Frank P. Kelly and Richard J. Gibbens "Resource Pricing and the Evolution of Congestion Control," *Automatica* 35, 1998.
- [6] Peter Marbach, "Pricing Differentiated Services Networks: Bursty Traffic," *IEEE InfoCom2001, Conference on Computer Communications*, Anchorage, Alaska, USA, April 2001.
- [7] M3I – Market-Managed Multi-services Internet Project, <http://www.m3i.org>.
- [8] Andrew Odlyzko, "Paris Metro Pricing for the Internet," *ACM EC'99, Conference on Electronic Commerce (SIGecom)*, Conference on Electronic Commerce, pp. 140-147, 1999.
- [9] Shaogang Chen and Kihong Park, "An architecture for Non-Cooperative QoS Provision in Many-Switch Systems," *IEEE InfoCom1999, Conference on Computer Communications*, 1999.
- [10] Ioannis C. Paschalidis and J.N. Tsitsiklis, "Congestion-Dependent Pricing of Network Services," *IEEE/ACM Transactions on Networking*, Vol. 8, No. 2, pp 171-184, August 2000.
- [11] Burkhard Stiller, Jan Gerke, Peter Reichl, and Placy Flury, "Management of Differentiated Service Usage by the Cumulus Pricing Scheme and a Generic Internet Charging System," *IEEE IM2001, Symposium on Integrated Network Management*, Seattle, U.S.A., May 2001.