Congestion Pricing Overlaid on Edge-to-Edge Congestion Control

Murat Yuksel, Shivkumar Kalyanaraman and Anuj Goel
Rensselaer Polytechnic Institute, Troy, NY
Emails: yuksem@ecse.rpi.edu, shivkuma@ecse.rpi.edu, goela@cs.rpi.edu

***Abstract*— One of the biggest obstacles for implementing congestion pricing is the pricing time-scale. The Internet traffic is highly variant and hard to control without a mechanism that operates on very low time-scales, i.e. on the order of round-trip-times (RTTs). However, pricing naturally operates on very large time-scales because of human involvement. So, in order to put tight control on congestion through pricing, new implementation methods and architectures are needed for congestion pricing. In order to solve this problem, we propose a novel approach Pricing over Congestion Control (POCC). The essence of POCC is to overlay congestion pricing on top of an underlying congestion control scheme which enforces a much tighter control than pricing. This way congestion in the interior network is controlled very tightly, while pricing is done at time-scales large enough to incorporate human involvement.**

I. Introduction

Implementation of congestion pricing still remains a challenge, although several proposals have been made, e.g. [1], [2], [3]. Among many others, one major implementation obstacle can be defined as the need for *frequent price updates*. This is relatively very hard to achieve in a wide area network such as the Internet, since users need to be informed about every price update. In [4], the authors showed that users do need feedback about charging of the network service (such as current price and prediction of service quality in near future). However, in our recent work [5], we illustrated that congestion control through pricing cannot be achieved if price changes are performed at a time-scale larger than roughly 40 round-trip-times (RTTs). This means that in order to achieve congestion control through pricing, service prices must be updated very frequently, i.e. 2-3 secs.

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

We particularly focus on diff-serv [6] architecture. We use an available edge-to-edge pricing mechamism (Distributed-DCC [7]) and edge-to-edge congestion control mechanism (Riviera [8]) in order to present the idea of pricing overlay over congestion control. We present simulation results for Distributed-DCC over Riviera, and illustrate benefits of overlaying pricing on top of congestion control.

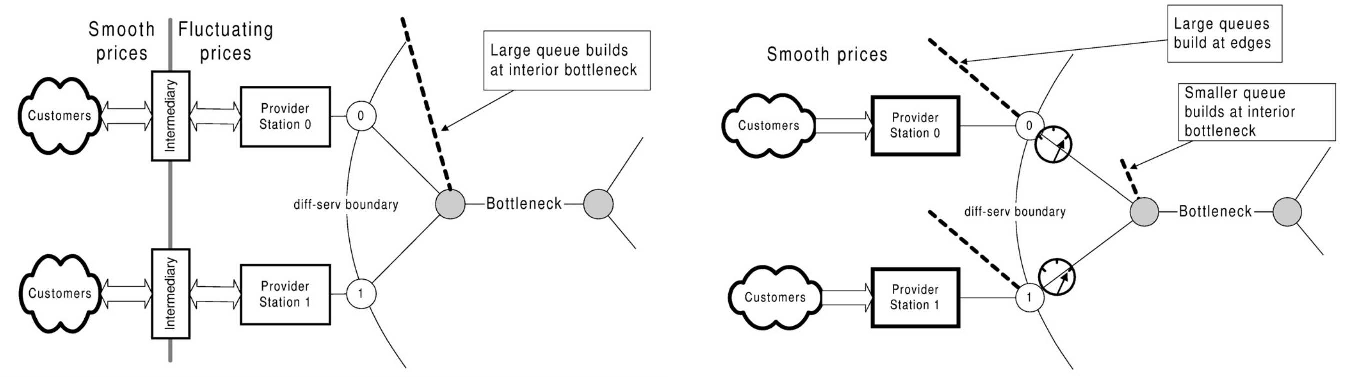
The paper is organized as follows: In the next section, we briefly survey the literature in the area of Internet pricing. In Section III, we present POCC ideas in detail and describe solutions to potential problems. Next in Sections III-B and III-C, we briefly describe an edge-to-edge pricing framework (Distributed-DCC) and an edge-to-edge congestion control mechanism (Riviera), which we will use later in simulation experiments. In Section IV, we present simulation experiments of Distributed-DCC over Riviera and illustrate POCC ideas. We finalize with summary and discussions.

II. Literature Survey

There has been several pricing proposals, which can be classified in many ways: *static* vs. *dynamic*, *per-packet* charging vs. *per-contract* charging, and charging *prior* to service vs. *posterior* to service. Although there are opponents to dynamic pricing in the area (e.g. [9], [10]), most of the proposals have been for dynamic pricing (specifically congestion pricing) of networks. Examples of dynamic pricing proposals are MacKie-Mason and Varian's Smart Market [1], Gupta et al.'s Priority Pricing [11], Kelly et al.'s Proportional Fair Pricing (PFP) [12], Semret et al.'s Market Pricing [3], and Wang and Schulzrinne's Resource Negotiation and Pricing (RNAP) [2]. Odlyzko's Paris Metro Pricing (PMP) [13] is an example of static pricing proposal. Clark's Expected Capacity [14] and Cocchi et al.'s Edge Pricing [15] allow both static and dynamic pricing. In terms of granularity, Smart Market, Priority Pricing, PFP and Edge Pricing employ *per-packet* charging, whilst RNAP and Expected Capacity employ *per-contract* charging.

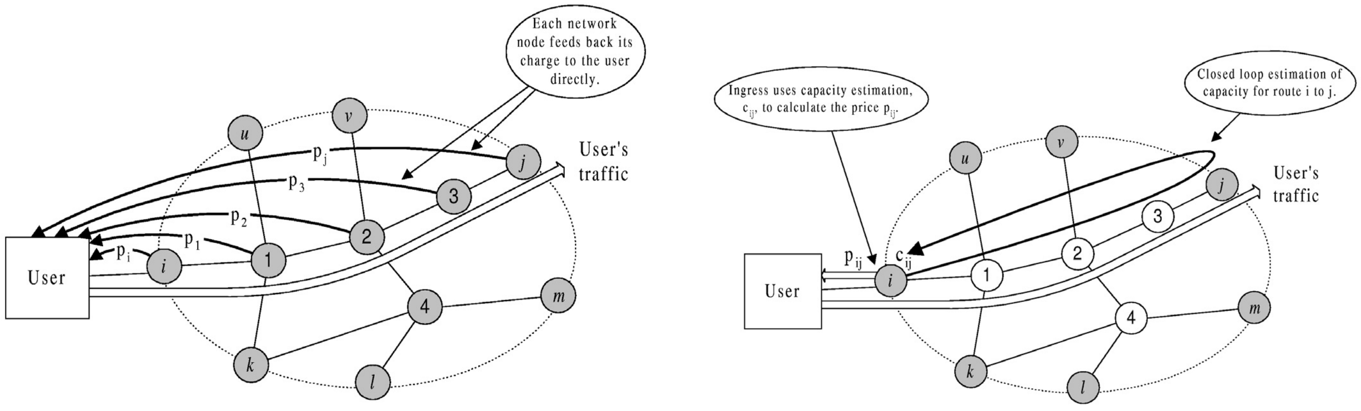
Smart Market is based primarily on imposing per-packet congestion prices, which makes it ideal because of its finest granularity [16]. While Smart Market holds one extreme in terms of granularity, Expected Capacity holds the other extreme. Expected Capacity proposes to use *long-term* contracts for statistical capacity allocation and pricing. Prices are updated at the beginning of each long-term contract, which incorporates little dynamism to prices.

An important recent work mainly focusing on implementation issues is RNAP. Although RNAP provides a complete picture for incorporation of admission control and congestion pricing, it has excessive implementation overhead since it requires all network routers to participate in determination of congestion prices. This requires upgrades to all routers similar to the case of Smart Market. We believe that pricing schemes that require upgrades to all routers will eventually fail in implementation phase. Because, Internet routers are owned by different entities who may or may not be willing to cooperate in upgrading routers.



(a) Pricing with no underlying edge-to-edge congestion control (b) Pricing over edge-to-edge congestion control

Fig. 1. Comparison of POCC with the case of no underlying congestion control.



Each network node feeds back its charge to the user directly.
User's traffic
User
Ingress uses capacity estimation, c_{ij}, to calculate the price p_{ij}.
Closed loop estimation of capacity for route i to j.
p_j
p_3
p_2
p_1
p_i
p_{ij}
c_{ij}

(a) Traditional pricing architecture (b) POCC

Fig. 2. Comparison of POCC with traditional pricing architectures in terms of participation of network routers into price calculation.

III. Pricing over Congestion Control (POCC)

Fro the rest of the paper, we will present POCC ideas by considering diff-serv environment. So, we will refer to edge-to-edge pricing and edge-to-edge congestion control during the dicussions. Note that the ideas can be generalized to the case where diff-serv is not under consideration.

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

Figure 1 illustrates the difference between POCC architecture and a pricing architecture without underlying congestion control. Observe that in POCC, the pricing protocol will be able to operate at larger time-scale and hence provide smooth prices to customers. However, when there is no underlying congestion control, the pricing protocol will have to operate at small time-scale and update prices frequently, in order to enforce control over congestion. This will cause highly fluctuating prices and hence there will be need for an intermediary (i.e. a software or hardware agent) that undertakes the job of smoothing prices.

As another benefit from POCC, amount of effort to calculate prices will be less. To calculate congestion-based prices, the provider has to gather congestion information across its network. So, if the pricing protocol is responsible for controlling congestion, then congestion at any node in the network must immediately effect prices. Traditionally [12], [17], this problem of communicating congestion information to user is resolved by involving each network node in price calculation as shown in Figure 2-a. The idea is to let each network node convey its congestion price (i.e. congestion measure) to the user directly, and charge the user based on each of these conveyed prices. However, this approach requires upgrades to all current Internet routers which are typically owned by different entites who may not cooperate in upgrading routers. POCC solves this implementation problem by pricing at large time-scales since control of congestion is left the underlying edge-to-edge congestion control mechanism. Pricing at larger time-scales allow provider to gather congestion information at larger time-scales which does not require participation of all network nodes. So, it becomes possible to employ closed-loop edge-to-edge techniques to obtain congestion information. An example is shown in Figure 2-b, where congestion information is obtained by closed-loop estimation of available capacity. Observe that traditional pricing architectures require participation of all network nodes in price calculation, while POCC requires particiation of only a subset of them. Figure 2 shows the difference by representing participating nodes in gray and the others in white.

We now first describe the problems raised by POCC architecture in diff-serv environment, then describe Distributed-DCC (i.e. an edge-to-edge pricing mechanism) and Riviera (i.e. an edge-to-edge congestion control mechanism), and then provide solutions to the problems for overlaying Distributed-DCC over Riviera.

A. POCC: Problems

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

1) *Parameter mapping:* Since the pricing scheme wants to allocate network capacity according to the user incentives (i.e. the users with larger budget should get more capacity) that changes dynamically over time, it is a required ability to set corresponding parameters of the underlying edge-to-edge congestion control mechanism such that it allocates the capacity to the user flows according to their incentives. So, this necessitates a method of mapping parameters of the pricing scheme to the parameters of the congestion control mechanism.
2) *Edge queues:* The underlying edge-to-edge congestion control scheme will not always allow all the traffic admitted by the pricing scheme, which will cause queues to build up at the network edges. So, management of these edge queues is necessary in POCC architecture. Figures 1-a and 1-b compare the situation of the edge queues in the two cases when there is an underlying congestion control scheme and when there is not.

B. An Edge-to-Edge Pricing Framework: Distributed-Dynamic Capacity Contracting (Distributed-DCC)

Distributed-DCC models a *short-term* contract for a given traffic class as a function of price per unit traffic volume P_v, maximum volume V_{max} (maximum number of bytes that can be sent during the contract) and the term of the contract T (length of the contract):

$$Contract = f(P_v, V_{max}, T) \qquad (1)$$

In the Distributed-DCC framework, customers can only access network core by making contracts with the provider stations placed at the edge routers. Access to available contracts can be done in different ways, which is known as *edge strategy*. Two basic edge strategies are “bidding” (many users bids for an available contract) or “contracting” (users negotiate with the provider for an available contract). So, edge strategy is the decision-making mechanism to identify which customer gets an available contract at the provider station.

Stations can advertise congestion-based prices if they have actual information about the congestion level in the network core. This congestion information can come from the interior routers or from the egress edge routers depending on the congestion-detection mechanism being used. DCC assumes that the congestion detection mechanism is able to give congestion information in time scales (i.e. observation intervals) smaller than contracts. The reader can find more details about Distributed-DCC in [7].

C. An Edge-to-Edge Congestion Control Mechanism: Riviera

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

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

In a single-bottleneck network, Riviera can be tuned such that each flow gets weighted share of the bottleneck capacity. The ingress nodes maintain an additive increase parameter, α, and a multiplicative decrease parameter, β, for each edge-to-edge flow. These parameters are used in AIMD-ER. Among the edge-to-edge flows, by setting the increase parameters (α) at the ingresses and the threshold parameters (max_thresh and min_thresh) at the egresses in ratio of desired rate allocation, it is possible to make sure that the flows get the desired rate allocation. For example, assume there are two flows 1 and 2 competing for a bottleneck (similar to Figure 3). If we want flow 1 to get a capacity of w times more than flow 2, then the following conditions must be hold:

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

D. POCC: Solutions for Distributed-DCC over Riviera

1) *Parameter mapping:* For each edge-to-edge flow, Distributed-DCC can calculate the capacity share of that flow out of the total network capacity. Let $\gamma_{ij} = c_{ij}/C$ be the fraction of network capacity that must be given to the flow i to j, where C is the total network capacity and c_{ij} is Distributed-DCC’s target capacity for flow i to j. Distributed-DCC can convey γ_{ij}s to the ingress stations,

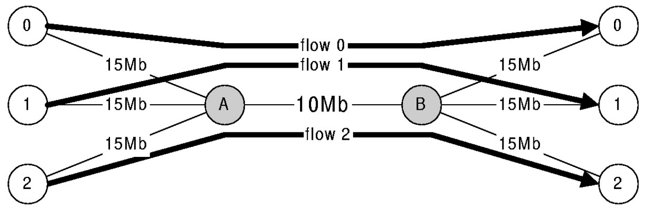


Fig. 3. Experimental single-bottleneck network.

and they can multiply Riviera's increase parameter α_{ij} with γ_{ij}. Also, Distributed-DCC can communicate γ_{ij}s to egresses, and they can multiply Riviera's threshold parameters with γ_{ij}. This solves the parameter mapping problem defined in Section III-A.

2) *Edge queues:* In Distributed-DCC, ingress stations maintain an estimation of available capacity for each edge-to-edge flow. So, one intuitive way of making sure that the user will not contract for more than the amount that the network can handle is to subtract necessary capacity to drain the already built edge queue from the estimated edge-to-edge capacity c_{ij}, and then make contracts accordingly. In other words, the ingress station updates the estimated capacity for flow i to j by the following formula $c'_{ij} = c_{ij} - Q_{ij}/T$, and uses c'_{ij} for price calculation. Note that Q is the actual edge queue length, and T is the length of the contract.

IV. Simulation Experiments and Results

We now present *ns* [18] simulation experiments of Distributed-DCC over Riviera on single-bottleneck topology. The single-bottleneck topology has a bottleneck link, which is connected to n edge nodes at each side where n is the number of users. The bottleneck link has a capacity of 10Mb/s and all other links have 15Mb/s. Propagation delay on each link is 5ms, and users send UDP traffic with an average packet size of 1000B. To ease understanding of experiments, each user sends its traffic to a separate egress. Figure 3 shows a single-bottleneck topology with $n = 3$. The white nodes are edge nodes and the gray nodes are interior nodes. The figure also shows the traffic flow of users on the topology. Buffer size is assumed to be infinite, so no packet drop is allowed.

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

We run simulation experiments for POCC on the single-bottleneck topology, which is represented in Figure 3. We also run experiment for Distributed-DCC with exactly the same parameters in order to see the effect of using an underlying congestion control mechanism. In these experiments, there are 3 users with budgets of 30, 20, 10 respectively for users 1, 2, 3. Total simulation time is 15000s, and at the beginning only the user 1 is active in the system. After 5000s, the user 2 gets active. Again after 5000s at simulation time 10000, the user 3 gets active.

In terms of results, the volume given to each flow is very important. Figures 4-a and 5-a show the flow rates in Distriuted-DCC only and Distributed-DCC over Riviera respectively. We see the flows are sharing the bottleneck capacity in proportion to their budgets. In comparison to Distributed-DCC over Riviera, Distributed-DCC only allocates the rate more smoothly but with the same average proportionality to the flows. The noisy volume allocation in Distributed-DCC over Rivera is caused by coordination issues (i.e. parameter mapping, edge queues) investigated in Section III-A.

Figure 4-b and 5-b show the price being advertised to flows in Distributed-DCC only and Distributed-DCC over Riviera respectively. As the new users join in, the pricing scheme increases the price in order to balance supply and demand.

Figures 4-c and 5-c shows the bottleneck queue size in Distributed-DCC only and Distributed-DCC over Riviera respectively. Notice that queue sizes make peaks transiently at the times when new users gets active. Otherwise, the queue size is controlled reasonably and the system is stable. In comparison to Distributed-DCC only, Distributed-DCC over Riviera manages the bottleneck queue much better because of the tight control enforced by the underlying edge-to-edge congestion control algorithm Riviera. The results follows with the big picture presented in Figure 1.

Figures from 6-a to 6-c show the sizes of edge queues in Distributed-DCC over Riviera. We observe stable behavior but with oscillations larger than the bottleneck queue illustrated in Figure 5-c. This is because of the tight edge-to-edge congestion control, which pushes backlog to the edges.

Also from Figures 4-a and 5-a, we observe that capacity being allocated to flows are slightly different. In the case of Distributed-DCC only, the flow rates are slightly higher than the ones in the case of Distributed-DCC over Riviera. This is mainly due to Riviera's tight control, which reduces the effective capacity of the bottleneck.

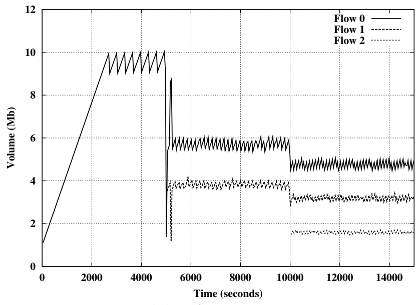
V. Summary

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

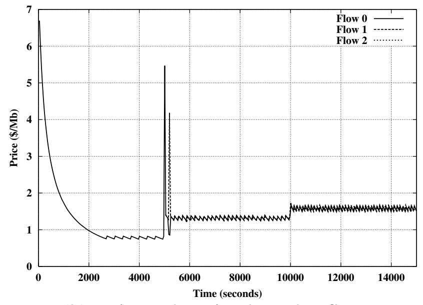
Future work should include investigation of issues related to extending POCC ideas on multiple diff-serv domains. Also, POCC ideas must be tested with edge-to-edge schemes other than Distributed-DCC and Riviera. Another interesting issue to look at is the end-to-end delay in POCC. POCC reduces the bottleneck queue length, but a thorough analysis of end-to-end delay is necessary for evaluating QoS in POCC architecture.

References

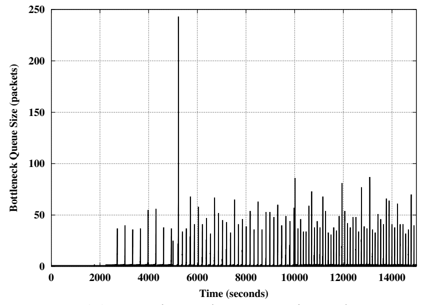
[1] J. K. MacKie-Mason and H. R. Varian, *Pricing the Internet.* Kahin, Brian and Keller, James, 1993.



(a) Flow rates

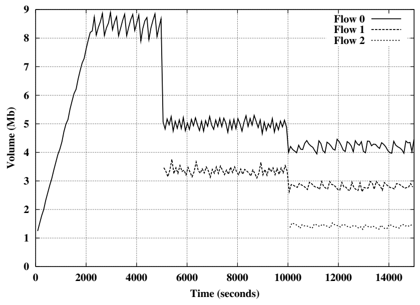


(b) Price advertised to the flows

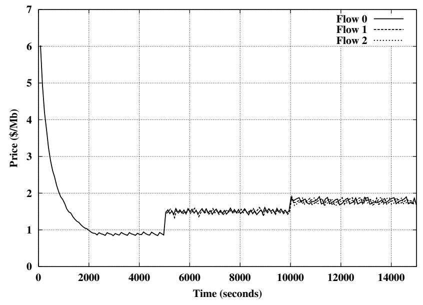


(c) Bottleneck queue length

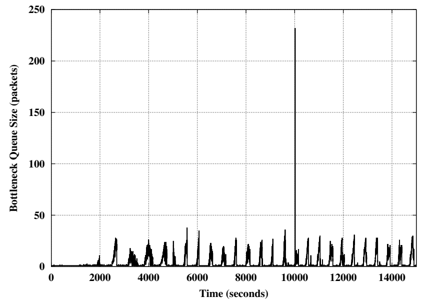
Fig. 4. Results of single-bottleneck experiment for Distributed-DCC without any underlying congestion control



(a) Flow rates

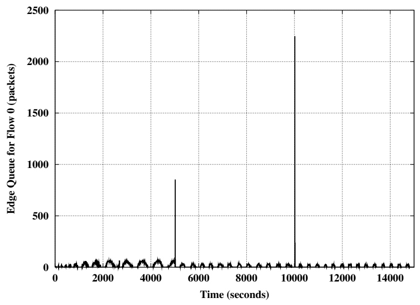


(b) Price advertised to the flows

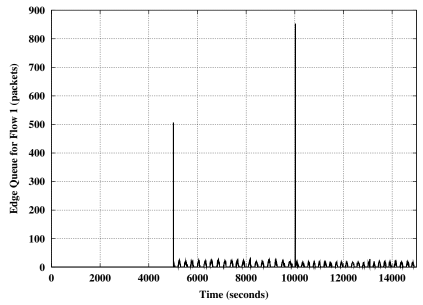


(c) Bottleneck queue length

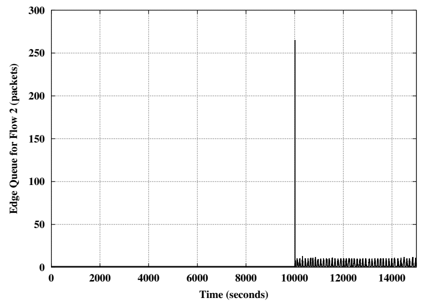
Fig. 5. Results of single-bottleneck experiment for POCC (Distributed-DCC over Riviera)



(a) Edge queue for flow 0



(b) Edge queue for flow 1



(c) Edge queue for flow 2

Fig. 6. Sizes of edge queues in the single-bottleneck experiment for POCC (Distributed-DCC over Riviera)

[2] X. Wang and H. Schulzrinne, "An integrated resource negotiation, pricing, and QoS adaptation framework for multimedia applications," *IEEE Journal of Selected Areas in Communications*, vol. 18, 2000.

[3] N. Semret, R. R.-F. Liao, A. T. Campbell, and A. A. Lazar, "Pricing, provisioning and peering: Dynamic markets for differentiated Internet services and implications for network interconnections," *IEEE Journal of Selected Areas in Communications*, vol. 18, 2000.

[4] A. Bouch and M. A. Sasse, "Why value is everything?: A user-centered approach to Internet quality of service and pricing," in *Proceedings of IEEE/IFIP International Workshop on Quality of Service (IWQoS)*, 2001.

[5] M. Yuksel, S. Kalyanaraman, and B. Sikdar, "Effect of pricing intervals on congestion-sensitivity of network service prices," Rensselaer Polytechnic Institute, ECSE Nets Lab, Tech. Rep. ECSE-NET-2002-1, 2002.

[6] S. B. et. al, "An architecture for Differentiated Services," *IETF RFC 2475*, December 1998.

[7] M. Yuksel and S. Kalyanaraman, "Distributed Dynamic Capacity Contracting: A congestion pricing framework for diff-serv," in *Proceedings of IFIP/IEEE MMNS*, 2002.

[8] D. Harrison, S. Kalyanaraman, and S. Ramakrishnan, "Overlay bandwidth services: Basic framework and edge-to-edge closed-loop building block," Poster in ACM SIGCOMM, 2001.

[9] A. M. Odlyzko, "Internet pricing and history of communications," AT & T Labs, Tech. Rep., 2000.

[10] I. C. Paschalidis and J. N. Tsitsiklis, "Congestion-dependent pricing of network services," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 171–184, 2000.

[11] A. Gupta, D. O. Stahl, and A. B. Whinston, *Priority pricing of Integrated Services networks*. Eds McKnight and Bailey, MIT Press, 1997.

[12] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control in communication networks: Shadow prices, proportional fairness and stability," *Journal of Operations Research Society*, vol. 49, pp. 237–252, 1998.

[13] A. M. Odlyzko, "A modest proposal for preventing Internet congestion," AT & T Labs, Tech. Rep., 1997.

[14] D. Clark, *Internet cost allocation and pricing*. Eds McKnight and Bailey, MIT Press, 1997.

[15] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang, "Pricing in computer networks: Motivation, formulation and example," *IEEE/ACM Transactions on Networking*, vol. 1, December 1993.

[16] M. Yuksel and S. Kalyanaraman, "A strategy for implementing Smart Market pricing scheme on diff-serv," in *Proc. of IEEE GLOBECOM'02*.

[17] S. H. Low and D. E. Lapsley, "Optimization flow control – I: Basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–875, 1999.

[18] "UCB/LBLN/VINT network simulator - ns (version 2)," http://www-mash.cs.berkeley.edu/ns, 1997.