Dynamic Overlay Single-Domain Contracting for End-to-End Contract-Switching

Murat Yuksel and Aparna Gupta and Koushik Kar

Abstract The Internet's simple design resulted in huge success in basic telecommunication services. However, in terms of providing end-to-end QoS services, the Internet's architecture needs major shifts since it neither allows (i) users to indicate their *value* choices at sufficient granularity nor (ii) providers to manage *risks* involved in investment for new innovative QoS technologies and business relationships with other providers as well as users. To allow these much needed economic flexibilities, we envision *contract-switching* as a new paradigm for the design of future Internet architecture. Just like packet-switching enabled flexible and efficient multiplexing of data, a contract-switched inter-network will enable flexible and economically efficient management of risks and value flows with more tussle points.

We show that economic flexibilities can be embedded into the inter-domain designs by concatenating single-domain contracts and this framework can be used to compose end-to-end QoS-enabled contract paths. Within such a framework, we also show that financial engineering techniques (e.g. options pricing) can be used to manage risks involved in inter-domain business relationships. We address implementation issues for dynamic pricing over a single domain by outlining a congestion-sensitive pricing framework Distributed Dynamic Capacity Contracting (Distributed-DCC), which is able to provide a range of fairness (e.g. max-min, proportional) in rate allocation by using pricing as a tool.

1 Introduction

The Internet's simple best-effort packet-switched architecture lies at the core of its tremendous success and impact. Today, the Internet is firmly a commercial medium

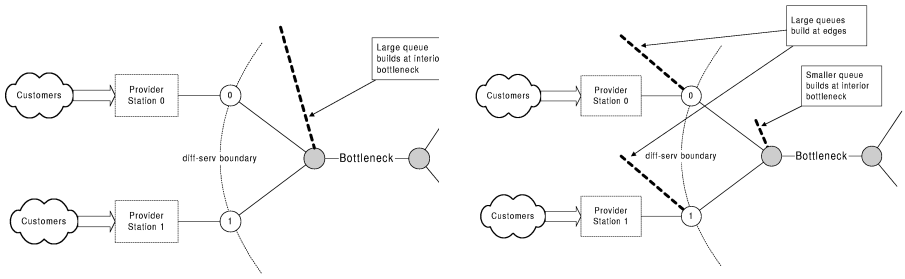
Murat Yuksel
University of Nevada - Reno, Reno, NV 89557 e-mail: yuksem@cse.unr.edu

Aparna Gupta
Rensselaer Polytechnic Institute, Troy, NY 12180 e-mail: guptaa@rpi.edu

Koushik Kar
Rensselaer Polytechnic Institute, Troy, NY 12180 e-mail: koushik@ecse.rpi.edu

involving several competitive service providers and content providers. However, current Internet architecture neither allows (i) users to indicate their *value* choices at sufficient granularity nor (ii) providers to manage *risks* involved in investment for new innovative QoS technologies and business relationships with other providers as well as users. Currently, users can only indicate their value choices at the access/link bandwidth level not at the routing level. End-to-end QoS contracts are possible today via virtual private networks, but with static and long-term contracts. Further, an enterprise that needs end-to-end capacity contracts between two arbitrary points on the Internet for a short period of time has no way of expressing its needs.

We envision an Internet architecture that allows flexible, fine-grained, dynamic contracting over multiple providers. With such capabilities, the Internet itself will be viewed as a *"contract-switched"* network beyond its current status as a "packet-switched" network. A contract-switched architecture will enable flexible and economically efficient management of risks and value flows in an Internet characterized by many tussle points [1] where competition for network resources takes place. Realization of such an architecture heavily depends on the capabilities of provisioning dynamic single-domain contracts which can be priced dynamically or based on intra-domain congestion. Implementation of dynamic pricing still remains a challenge, although several proposals have been made, e.g. [2, 3, 4]. Among many others, two major implementation obstacles can be defined: need for *timely feedback* to users about the price, determination of *congestion information* in an efficient, low-overhead manner.



(a) Pricing for Congestion Control (PFCC) (b) Pricing over Congestion Control (POCC)

Fig. 1 Different pricing architectures with/without edge-to-edge congestion control.

The first problem, timely feedback, is relatively hard to achieve in a wide area network such as the Internet. In [5], the authors showed that users do want feedback about charging of the network service (such as current price and prediction of service quality in near future). However, in [6], we illustrated that congestion control by pricing cannot be achieved if price changes are performed at a time-scale larger than roughly 40 round-trip-times (RTTs). This means that in order to achieve congestion control by pricing, service prices must be updated very frequently (i.e. 2-3 seconds since RTT is expressed in terms of milliseconds for most cases in the Internet). In order to solve this time-scale problem for dynamic pricing, we propose two solutions, which lead to two different pricing "architectures":

- *By placing intelligent intermediaries (i.e. software or hardware agents) between users and the provider.* This way it is possible for the provider to update prices frequently at low time-scales, since price negotiations will be made with a software/hardware agent rather than a human. Since the provider will not employ any congestion control mechanism for its network and try to control congestion by only pricing, we call this pricing architecture as *Pricing for Congestion Control (PFCC)*.
- *By overlaying pricing on top of an underlying congestion control mechanism.* This way it is possible to enforce tight control on congestion at small time-scale, while performing pricing at time-scales large enough for human involvement. The provider implements a congestion control mechanism to manage congestion in its network. So, we call this pricing architecture as *Pricing over Congestion Control (POCC)*.

The big-picture of the two pricing architectures PFCC and POCC are shown in Figure 1. We will describe PFCC and POCC later in Section 4.

The second problem, congestion information, is also very hard to solve in a way that does not require a major upgrade at network routers. However, in diff-serv [7], it is possible to determine congestion information via a good ingress-egress coordination. So, this flexible environment of diff-serv motivated us to develop a pricing framework on it.

The chapter is organized as follows: In the next section, we position our work and briefly survey relevant work in the area. Section 3 introduces our contract-switching paradigm. In Section 4, we present PFCC and POCC pricing architectures motivated by the time-scale issues mentioned above. In Section 5 we describe properties of Distributed-DCC framework according to the PFCC and POCC architectures. Next in Section 6, we define a pricing scheme Edge-to-Edge Pricing (EEP) which can be implemented in the defined Distributed-DCC framework. We study optimality of EEP for different forms of user utility functions and consider effect of different parameters such as user's budget, user's elasticity. In Section 7, according to the descriptions of Distributed-DCC framework and EEP scheme, we simulate Distributed-DCC in the two architectures PFCC and POCC. With the simulation results, we compare Distributed-DCC's performance in PFCC and POCC architectures. We, then, extend the pricing formulations to an end-to-end level in Section 8. We finalize with summary and discussions in Section 9.

2 Related Work

There have been several pricing proposals, which can be classified in many ways: *static* vs. *dynamic*, *per-packet* charging vs. *per-contract* charging, and charging *a-priori* to service vs. *a-posteriori* to service.

Although there are opponents to dynamic pricing in the area (e.g. [8, 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 [4], Gupta et al.'s Priority Pricing [11], Kelly et al.'s Proportional Fair Pricing (PFP) [12], Semret et al.'s Market Pricing [13, 3], and Wang

and Schulzrinne's Resource Negotiation and Pricing (RNAP) [14, 2]. Odlyzko's Paris Metro Pricing (PMP) [15] is an example of static pricing proposal. Clark's Expected Capacity [16, 17] and Cocchi et al.'s Edge Pricing [18] allow both static and dynamic pricing. In terms of charging granularity, Smart Market, Priority Pricing, PFP and Edge Pricing employ per-packet charging, whilst RNAP and Expected Capacity do not employ per-packet charging.

Smart Market is based primarily on imposing per-packet congestion prices. Since Smart Market performs pricing on per-packet basis, it operates at the finest possible pricing granularity. This makes Smart Market capable of making ideal congestion pricing. However, Smart Market is not deployable because of its per-packet granularity (i.e. excessive overhead) and its many requirements from routers (e.g. requires all routers to be updated). In [19], we studied Smart Market and difficulties of its implementation in more detail.

While Smart Market holds one extreme in terms of granularity, Expected Capacity holds the other extreme. Expected Capacity proposes to use *long-term* contracts, which can give more clear performance expectation, for statistical capacity allocation and pricing. Prices are updated at the beginning of each long-term contract, which incorporates little dynamism to prices.

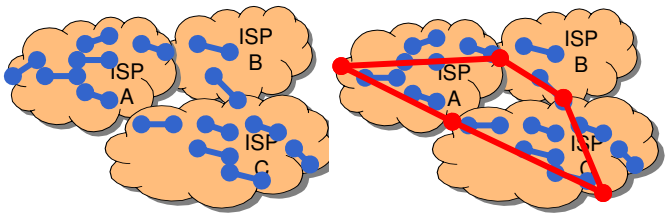
Our work, Distributed-DCC, is a middle-ground between Smart Market and Expected Capacity in terms of granularity. Distributed-DCC performs congestion pricing at *short-term* contracts, which allows more dynamism in prices while keeping pricing overhead small.

In the area, another proposal that mainly focused on implementation issues of congestion pricing on diff-serv is RNAP [14, 2]. 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 proposals that require upgrades to all routers will eventually fail in implementation phase. This is because of the fact that the Internet routers are owned by different entities who may or may not be willing to cooperate in the process of router upgrades. Our work solves this problem by requiring upgrades only at edge routers rather than at all routers.

3 Contract-Switching Paradigm

The essence of "contract-switching" is to use *contracts* as the key building block for inter-domain networking. As shown in Figure 2, this increases the inter-domain architecture flexibilities by introducing more tussle points into the protocol design. Especially, this paradigm will allow the much needed revolutions in the Internet protocol design: (i) inclusion of economic tools in the network layer functions such as inter-domain routing while the current architecture only allows basic connectivity information exchange, and (ii) management of risks involved in QoS technology investments and participation into end-to-end QoS contract offerings by allowing ISPs to potentially apply financial engineering methods.

In addition to these design opportunities, the contract-switching paradigm introduces several research challenges. As the key building block, intra-domain service



(a) Packet-switching (b) Contract-switching

Fig. 2 Packet-switching introduced many more tussle points into the Internet architecture by breaking the *end-to-end circuits* of circuit-switching into *routable datagrams*. Contract-switching introduces even more tussle points at the edge/peering points of domain boundaries by *overlay contracts*.

abstractions call for design of (i) single-domain edge-to-edge QoS contracts with performance guarantees and (ii) nonlinear pricing schemes geared towards cost recovery. Moving one level up, composition of end-to-end inter-domain contracts poses a major research problem which we formulate as a "contract routing" problem by using single-domain contracts as "contract links". Issues to be addressed include routing scalability, contract monitoring and verification as the inter-domain context involves large-size effects and crossing trust boundaries. Several economic tools can be used to remedy pricing, risk sharing, and money-back problems of a contract-switched network provider (CSNP), which can operate as an overlay reseller ISP (or an alliance of ISPs) that buys contract links and sells end-to-end QoS contracts. In addition to CSNPs, the contract-switching paradigm allows more distributed ways of composing end-to-end QoS contracts as we will detail later.

4 Single-Domain Pricing Architectures: PFCC vs. POCC

In our previous work [20], we presented a simple congestion-sensitive pricing "framework", *Dynamic Capacity Contracting (DCC)*, for a single diff-serv domain. DCC treats each edge router as a station of a service provider or a station of cooperating set of service providers. Users (i.e. individuals or other service providers) make *short-term contracts* with the stations for network service. During the contracts, the station receives congestion information about the network core at a timescale smaller than contracts. The station, then, uses that congestion information to update the service price at the beginning of each contract. Several pricing "schemes" can be implemented in that framework.

DCC models a short-term contract for a given traffic class as a tuple 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):

$$Contract = < P_v, V_{max}, T > \quad (1)$$

Figure 3 illustrates the big picture of DCC framework. Customers can only access network core by making contracts with the provider stations placed at the edge routers. The stations offer contracts (i.e. V_{max} and T) to fellow users. Access to these available contracts can be done in different ways, what we call *edge strategy*. Two

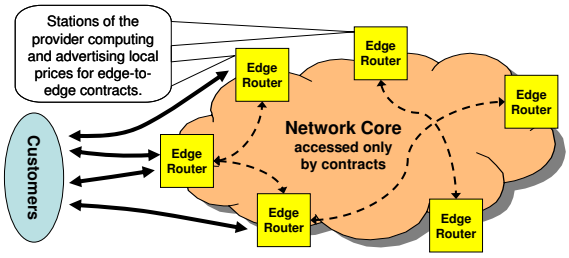


Fig. 3 DCC framework on diff-serv architecture.

basic edge strategies are "bidding" (many users bids for an available contract) or "contracting" (users negotiate P_v with the provider for an available contract).

Notice that, in DCC framework, provider stations can implement dynamic pricing schemes. In particular, they can implement congestion-based pricing schemes, if they have actual information about congestion in 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.

However, in DCC, we assumed that all the provider stations advertise the same price value for the contracts, which is very costly to implement over a wide area network. This is simply because the price value cannot be communicated to all stations at the beginning of each contract. We relax this assumption by allowing the stations to calculate the prices locally and advertise different prices than the other stations. We call this new version of DCC as *Distributed-DCC*. We introduce ways of managing the overall coordination of the stations.

As a fundamental difference between Distributed-DCC and the well-known dynamic pricing proposals in the area (e.g. proposals by Kelly et al. [12] and Low et al. [21]) lies in the manner of price calculation.

In Distributed-DCC, the prices are calculated on an edge-to-edge basis, while traditionally it has been proposed that prices are calculated at each local link and fed back to users. In Distributed-DCC, basically, the links on a flow's route are abstracted out by edge-to-edge capacity estimation and the ingress node communicates with the corresponding egress node to observe congestion on the route. Then, the ingress node uses the estimated capacity and the observed congestion information in price calculation. However, in Low et al.'s framework, each link calculates its own price and sends it to the user, and the user pays the aggregate price. So, Distributed-DCC is better in terms of implementation requirements, while Low et al.'s framework is better in terms of optimality. Distributed-DCC trades off some optimality in order to enable implementation of dynamic pricing. Amount of lost optimality depends on the closed-loop edge-to-edge capacity estimation.

4.1 Pricing for Congestion Control (PFCC)

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

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

The major problem with PFCC is that development of user-friendly intermediate agents is heavily dependent on user opinion, and hence requires significant amount of research. A study of determining user opinions is available in [5]. In this chapter, we do not focus development of intermediate agents.

4.2 Pricing over Congestion Control (POCC)

Another way of approaching the congestion control problem by pricing is to overlay pricing on top of congestion control. This means the provider undertakes the congestion control problem by itself, and employs an underlying congestion control mechanism for its network. This way it is possible to enforce tight control on congestion at small time-scales, while maintaining human involvement into the price negotiations at large time-scales. Figure 1 illustrates the difference between POCC (with congestion control) and PFCC (without congestion control) architectures.

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

Since our main focus is to implement pricing in "diff-serv environment", we assume that the provider employs "edge-to-edge" congestion control mechanisms under the pricing protocol on top. So, in diff-serv environment, overlaying pricing on top of edge-to-edge congestion control raises two major problems:

1. *Parameter mapping:* Since the pricing protocol wants to allocate network capacity according to the user incentives (i.e. the users with greater budget should get more capacity) that changes dynamically over time, it is a required ability to set corresponding parameters of the underlying edge-to-edge congestion control mechanism such that it allocates the capacity to the user flows according to their incentives. So, this raises need for a method of mapping parameters of the pricing scheme to the parameters of the underlying congestion control mechanism. Notice that this type of mapping requires the edge-to-edge congestion control mechanism to be able to provide parameters that tune the rate being given to edge-to-edge flows.
2. *Edge queues:* The underlying edge-to-edge congestion control scheme will not always allow all the traffic admitted by the pricing protocol, which will cause queues to build up at 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 edge-to-edge congestion control scheme and when there is not.

5 Distributed-DCC Framework

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

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

Distributed-DCC framework has three major components: *Logical Pricing Server (LPS)*, *Ingress Stations*, and *Egress Stations*. Solid lined arrows in the figure represent control information being transmitted among the components. Basically, Ingress stations negotiate with customers, observe customer's traffic, and make estimations about customer's demand. Ingress stations inform corresponding Egress stations about the observations and estimations about each edge-to-edge flow.

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

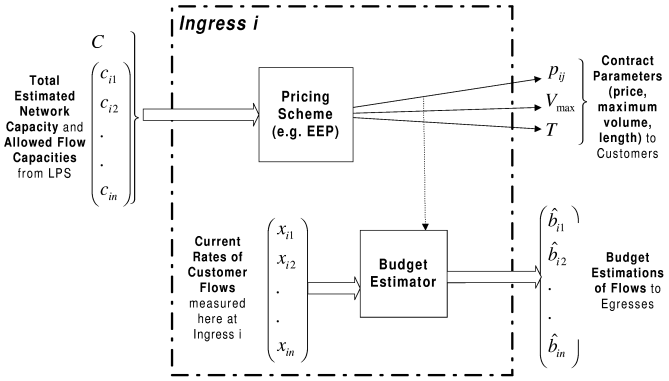


Fig. 4 Major functions of Ingress *i*.

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

5.1 Ingress Station i

Figure 4 illustrates sub-components of Ingress station *i* in the framework. Ingress *i* includes two sub-components: *Pricing Scheme* and *Budget Estimator*.

Ingress station *i* keeps a "current" price vector p_i, where p_{ij} is the price for the flow from ingress *i* to egress *j*. So, the traffic using flow *i* to *j* is charged the price p_{ij}. Pricing Scheme is the sub-component that calculates price p_{ij} for each edge-to-edge flow starting at Ingress *i*. It uses allowed flow capacities c_{ij} and other local information (such as $\hat{b}_{ij}$), in order to calculate price p_{ij}. The station, then, uses p_{ij} in negotiations with customers. We will describe a simple pricing scheme Edge-to-Edge Pricing (EEP) later in Section 6. However, it is possible to implement several other pricing schemes by using the information available at Ingress *i*. Other than EEP, we implemented another pricing scheme, Price Discovery, which is available in [22].

Also, the ingress *i* uses the total estimated network capacity *C* in calculating the V_{max} contract parameter defined in (1). Admission control techniques can be used to identify the best value for V_{max}. We use a simple method which does not put any restriction on V_{max}, i.e. $V_{max} = C * T$ where T is the contract length.

Budget Estimator is the sub-component that observes demand for each edge-to-edge flow. We implicitly assume that user's "budget" represents user's demand (i.e. willingness-to-pay). So, Budget Estimator estimates budget $\hat{b}_{ij}$ of each edge-to-edge traffic flow[1].

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

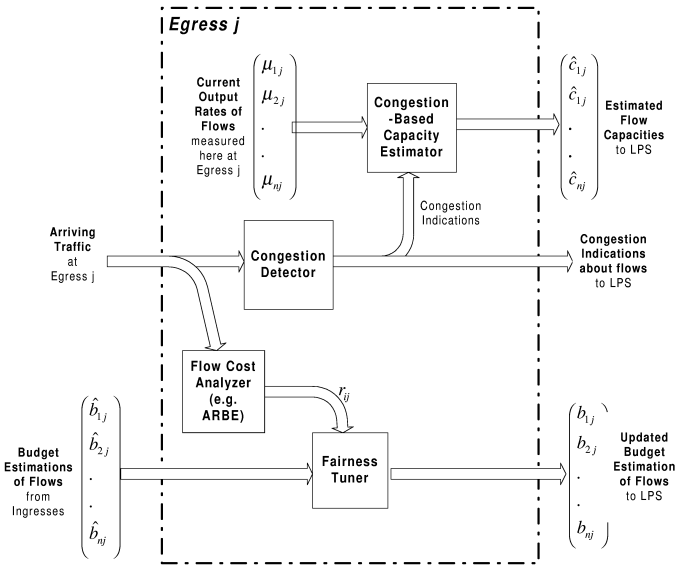


Fig. 5 Major functions of Egress j.

5.2 Egress Station j

Figure 5 illustrates sub-components of Egress Station j in the framework: *Congestion Detector, Congestion-Based Capacity Estimator, Flow Cost Analyzer*, and *Fairness Tuner*.

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

Congestion-Based Capacity Estimator estimates available capacity $\hat{c}_{ij}$ for each edge-to-edge flow exiting at Egress j. In order to calculate $\hat{c}_{ij}$, it uses congestion indications from Congestion Detector and actual output rates μ_{ij} of the flows. The crucial property of Congestion-Based Capacity Estimator is that it estimates capacity in a congestion-based manner, i.e. it decreases the capacity estimation when there is congestion indication and increases when there is no congestion indication. This makes the prices *congestion-sensitive*, since Pricing Scheme at Ingress calculates prices based on the estimated capacity.

Flow Cost Analyzer determines cost of each traffic flow (e.g. number of links traversed by the flow, number of bottlenecks traversed by the flow, amount of queuing delay caused by the flow) exiting at Egress j. Cost incurred by each flow can be several things: number of traversed links, number of traversed bottlenecks, amount of queuing delay caused. We assume that number of bottlenecks is a good representation of the cost incurred by a flow. It is possible to define edge-to-edge algorithms

that can effectively and accurately estimate the number of bottlenecks traversed by a flow [23].

LPS, as will be described in the next section, allocates capacity to edge-to-edge flows based on their budgets. The flows with higher budgets are given more capacity than the others. So, Egress j can penalize/favor a flow by increasing/decreasing its budget $\hat{b}_{ij}$. Fairness Tuner is the component that updates $\hat{b}_{ij}$. So, Fairness Tuner penalizes or favors the flow from ingress i by updating its estimated budget value, i.e. $b_{ij} = f(\hat{b}_{ij}, \hat{r}_{ij}, < parameters >)$ where $< parameters >$ are other optional parameters that may be used for deciding how much to penalize or favor the flow. For example, if the flow ingress i is passing through more congested areas than the other flows, Fairness Tuner can penalize this flow by reducing its budget estimation $\hat{b}_{ij}$. We will describe an algorithm for Fairness Tuner later in Section 5.2.1.

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

5.2.1 Fairness Tuner

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

- *Single-bottleneck case:* The pricing protocol should charge *the same price (i.e., $/bandwidth) to the users of the same bottleneck*. In this way, among the customers using the same bottleneck, the ones who have more budget will be given more rate (i.e., bandwidth/time) than the others. The intuition behind this reasoning is that the cost of providing capacity to each customer is the same.
- *Multi-bottleneck case:* The pricing protocol should *charge more to the customers whose traffic passes through more bottlenecks* and cause more costs to the provider. So, other than proportionality to customer budgets, we also want to allocate less rate to the customers whose flows are passing through more bottlenecks than the other customers.
  For multi-bottleneck networks, two main types of fairness have been defined: max-min fairness [24] and proportional fairness [12]. In max-min fair rate allocation, all flows get equal share of the bottlenecks, while in proportional fair rate allocation flows get penalized according to the number of traversed bottlenecks. Depending on the cost structure and user's utilities, for some cases the provider may want to choose max-min or proportional rate allocation. So, we would like to have ability of tuning the pricing protocol such that fairness of its rate allocation is in the way the provider wants.

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

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

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

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

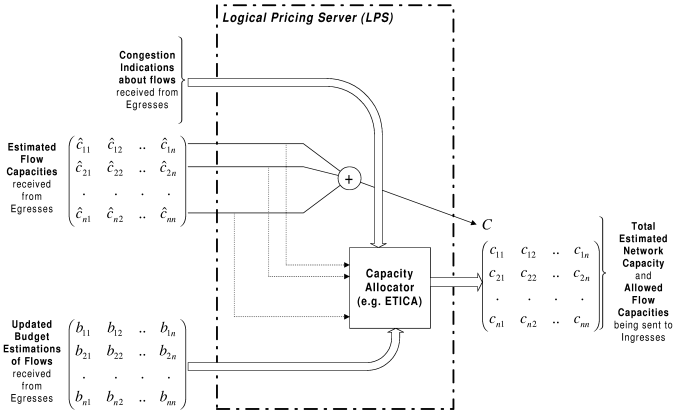


Fig. 6 Major functions of LPS.

5.3 Logical Pricing Server (LPS)

Figure 6 illustrates basic functions of LPS in the framework. LPS receives information from egresses and calculates *allowed capacity* c_{ij} for each edge-to-edge flow. The communication between LPS and the stations take place at every *LPS interval* L. There is only one major sub-component in LPS: Capacity Allocator.

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

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

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

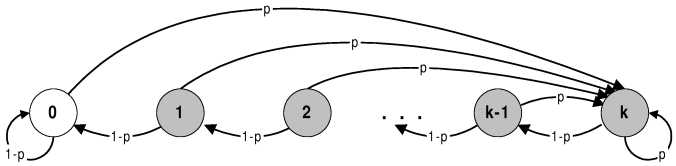


Fig. 7 States of an edge-to-edge flow in ETICA algorithm: The states $i > 0$ are "congested" states and the state $i = 0$ is the "non-congested" state, represented with gray and white colors respectively.

Firstly, note that LPS is going to implement the ETICA algorithm as a Capacity Allocator (see Figure 6). So, we will refer to LPS throughout the description of ETICA below.

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

Again at LPS, let K_{ij} determine the state of f_{ij}. If $K_{ij} > 0$, LPS determines f_{ij} as congested. If not, it determines f_{ij} as non-congested. At every LPS interval t, LPS calculates K_{ij} as follows:

$$K_{ij}(t) = \begin{cases} k, \text{ congestion in } t-1 \\ K_{ij}(t-1) - 1, \text{ no congestion in } t-1 \end{cases}$$
$$K_{ij}(0) = 0 \tag{2}$$

where k is a positive integer. Notice that k parameter defines how long a flow will stay in "congested" state after the last congestion indication. So, in other words, k defines the time-line to determine if a congestion indication is "recent" or not. According to these considerations in ETICA algorithm, Figure 7 illustrates states of an edge-to-edge flow given that probability of receiving a congestion indication in the last LPS interval is p. Gray states are the states in which the flow is "congested", and the single white state is the "non-congested" state. Observe that number of congested states (i.e. gray states) is equal to k which defines to what extent a congestion indication is "recent". [2]

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

[2] Note that instead of setting K_{ij} to k at every congestion indication, more accurate methods can be used in order to represent self-similar behavior of congestion epochs. For simplicity, we proceed with the method in (2).

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

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

6 Single-Domain Edge-to-Edge Pricing Scheme (EEP)

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

$$\hat{p}_{ij} = \frac{\hat{b}_{ij}}{c_{ij}} \tag{3}$$

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

The main idea of the EEP is to balance supply and demand by equating price to the ratio of users' budget (i.e. demand) B by available capacity C. Based on that, we used the pricing formula:

$$p = \frac{\hat{B}}{\hat{C}} \tag{4}$$

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

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

SYSTEM :

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

subject to

$$\sum_{f \in F(l)} \lambda_f \;\; \leq \;\; c_l, \quad l = 1,...,L \tag{5}$$

This problem can be divided into two separate problems by employing monetary exchange between user flows and the network provider. Following Kelly's [25] methodology we split the system problem into two:

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

$FLOW_f(p^f)$:

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

over

$$\lambda_f \geq 0 \tag{6}$$

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

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

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

subject to

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

over

$$p \geq 0 \tag{7}$$

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

$$\lambda_f(p^f) = U_f'^{-1}(p^f) \tag{8}$$

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

We model customer i's utility with the well-known function[3] [12, 24, 26, 21]

$$u_i(x) = w_i \log(x) \tag{9}$$

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

Now, we set up a vectorized notation, then solve the revenue maximization problem $NETWORK(\lambda(p^f))$. Assume the network includes n flows and m links. Let λ be row vector of the flow rates (λ_f for $f \in F$), P be column vector of the price at

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

each link (p_l for $l \in L$). Define the $n \times n$ matrix P^* in which the diagonal element P^*_{jj} is the aggregate price being advertised to flow j (i.e. $p^j = \sum_{l \in L(j)} p_l$) and all the other elements are 0. Also, let A be the $n \times m$ routing matrix in which the element A_{ij} is 1 if ith flow is passing though jth link and the element A_{ij} is 0, if not, C be the column vector of link capacities (c_l for $l \in L$). Finally, define the $n \times n$ matrix $\hat{\lambda}$ in which the diagonal element $\hat{\lambda}_{jj}$ is the rate of flow j (i.e. $\hat{\lambda}_{jj} = \lambda_j$) and all the other elements are 0.

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

$$AP = P^* e \tag{10}$$

$$\lambda = (\hat{\lambda} e)^T = e^T \hat{\lambda} \tag{11}$$

where e is the column unit vector.

We use the utility function of (9) in our analysis. By plugging (9) in (8) we obtain flow's demand function in vectorized notation:

$$\lambda(P^*) = W {P^*}^{-1} \tag{12}$$

where W is row vector of the weights w_i in flow's utility function (9).

Also, we can write the utility function (9) in vectorized notation as follows:

$$U(\lambda) = W \log(\hat{\lambda}) \tag{13}$$

The revenue maximization of (7) can be re-written as follows:

$$\max_P R = \lambda AP$$

subject to

$$\lambda A \ \leq \ C^T. \tag{14}$$

So, we write the Lagrangian as follows:

$$L = \lambda AP + (C^T - \lambda A)\gamma \tag{15}$$

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

Solving (15), we derive P:

$$P = (C^T)^{-1} We \tag{16}$$

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

$$P = A^{-1} W^T C^{-1} A^T e \tag{17}$$

Notice that the result in (16) holds for a single-bottleneck (i.e. single-link) network. In non-vectorized notation, this results translates to:

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

The result in (17) holds for a multi-bottleneck network. This result means that each link's optimal price is dependent on the routes of each flow passing through that link. More specifically, the optimal price for link l is accumulation of budgets of flows passing through link l (i.e. $W^T A^T$ in the formula) divided by total capacity of the links that are traversed by the flows traversing the link l (i.e. $A^{-1}C^{-1}$ in the formula). In non-vectorized notation, price of link l can be written as:

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

Similar results can be found for non-logarithmic utility functions involving user's utility-bandwidth elasticity [27].

7 Distributed-DCC: PFCC and POCC Architectures

In order to adapt Distributed-DCC to PFCC architecture, LPS must operate at very low time-scales. In other words, LPS interval must be small enough to maintain control over congestion, since PFCC assumes no underlying congestion control mechanism. This raises practical issues to be addressed. For instance, intermediate agents between customers and Ingress stations must be implemented in order to maintain human involvement into the system.

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

To adapt Distributed-DCC framework to POCC architecture, an edge-to-edge congestion control mechanism is needed, for which we use Riviera [28] in our experiments.

Riviera takes advantage of two-way communication between ingress and egress edge routers in a diff-serv network. Ingress sends a *forward* feedback to egress in response to feedback from egress, and egress sends *backward* feedback to ingress in response to feedback from ingress. So, ingress and egress of a traffic flow keep bouncing feedback to each other. Ignoring loss of data packets, the egress of a traffic flow measures the accumulation, a, caused by the flow by using the bounced feedbacks and RTT estimations. When a for a particular flow exceeds a threshold or goes below a threshold the flow is identified as congested or not-congested respec-

tively. The ingress node gets informed about the congestion detection by backward feedbacks and uses egress' explicit rate to adjust the sending rate.

We now provide solutions defined in Section 4.2, for the case of overlaying Distributed-DCC over Riviera:

1. *Parameter mapping:* For each edge-to-edge flow, LPS 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. LPS can convey γ_{ij}s to the ingress stations, and they can multiply the increase parameter α_{ij} with γ_{ij}. Also, LPS can communicate γ_{ij}s to the egresses, and they can multiply max_thresh_{ij} and min_thresh_{ij} with γ_{ij}.
2. *Edge queues:* In Distributed-DCC, ingress stations are informed by LPS about allocated capacity c_{ij} for each edge-to-edge flow. So, one intuitive way of making sure that the user will not contract for more than c_{ij} is to subtract necessary capacity to drain the already built edge queue from c_{ij}, and then make contracts accordingly. In other words, the ingress station updates the allocated capacity c_{ij} 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_{ij} is the edge queue length for flow i to j, and T is the length of the contract.

7.1 Simulation Experiments and Results

We now present *ns-2* [29] simulation experiments for the two architectures, PFCC and POCC, on single-bottleneck and multi-bottleneck topology. Our goals are to illustrate fairness and stability properties of the two architectures with possible comparisons of two.

For PFCC and POCC, we simulate Distributed-DCC's PFCC and POCC versions which were describe in Section 7. We will simulate EEP pricing scheme at Ingress stations. The key performance metrics we will extract from our simulations are:

- Steady-state properties of PFCC and POCC architectures: queues, rate allocation
- PFCC's fairness properties: Provision of various fairness in rate allocation by changing the fairness coefficient α
- Performance of Distributed-DCC's capacity allocation algorithm ETICA in terms of adaptiveness

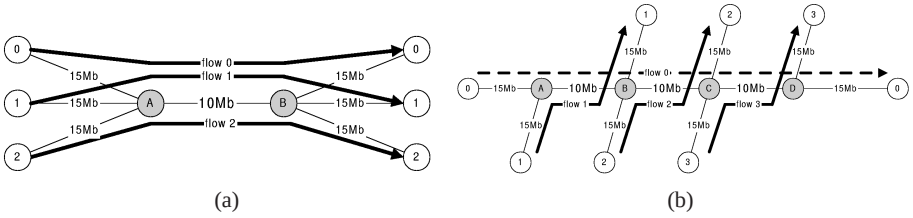


Fig. 8 (a) Single-bottleneck (b) Multi-bottleneck network for Distributed-DCC experiments.

The single-bottleneck topology has a bottleneck link, which is connected to n edge nodes at each side where n is the number of users. The multi-bottleneck topol-

ogy has $n-1$ bottleneck links, that are connected to each other serially. There are again n ingress and n egress edge nodes. Each ingress edge node is mutually connected to the beginning of a bottleneck link, and each egress node is mutually connected to the end of a bottleneck link. All bottleneck links have a capacity of 10Mb/s and all other links have 15Mb/s. Propagation delay on each link is 5ms, and users send UDP traffic with an average packet size of 1000B. To ease understanding the experiments, each user sends its traffic to a separate egress. For the multi-bottleneck topology, one user sends through all the bottlenecks (i.e. long flow) while the others cross that user's long flow. The queues at the interior nodes (i.e. nodes that stand at the tips of bottleneck links) mark the packets when their local queue size exceeds 30 packets. In the multi-bottleneck topology they increment a header field instead of just marking. Figure 8-a shows a single-bottleneck topology with $n = 3$. Figure 8-b shows multi-bottleneck topology with $n = 4$. The white nodes are edge nodes and the gray nodes are interior nodes. These figures also show the traffic flow of users on the topology. The user flow tries to maximize its total utility by contracting for b/p amount of capacity, where b is its budget and p is price. The flows's budgets are randomized according to truncated-Normal [30] distribution with a given mean value. This mean value is what we will refer to as flows's budget in our simulation experiments.

Contracting takes place at every 4s, observation interval is 0.8s, and LPS interval is 0.16s. Ingresses send budget estimations to corresponding egresses at every observation interval. LPS sends information to ingresses at every LPS interval. The parameter k is set to 25, which means a flow is determined to be non-congested at least after 25 LPS intervals equivalent to one contracting interval (see Section 5.3.1).

The parameter δ is set to 1 packet (i.e. 1000B), the initial value of $\hat{c}_{ij}$ for each flow f_{ij} is set to 0.1Mb/s, β is set to 0.95, and $\triangle r$ is set to 0.0005. Also note that, in the experiments, packet drops are not allowed in any network node. This is because we would like to see performance of the schemes in terms of assured service.

7.1.1 Experiments on Single-bottleneck Topology

We run simulation experiments for PFCC and POCC on the single-bottleneck topology, which is represented in Figure 8-a. In this experiment, there are 3 users with budgets of 30, 20, 10 respectively for users 1, 2, 3. Total simulation time is 15000s, and at the beginning only the user 1 is active in the system. After 5000s, the user 2 gets active. Again after 5000s at simulation time 10000, the user 3 gets active.

For POCC, there is an additional component in the simulation: edge queues. The edge queues mark the packets when queue size exceeds 200 packets. So, in order to manage the edge queues in this simulation experiment, we simultaneously employ both of the two techniques described before.

In terms of results, the volume given to each flow is very important. Figures 9-a and 10-a show the volumes given to each flow in PFCC and POCC respectively. We see the flows are sharing the bottleneck capacity in proportion to their budgets. In comparison to POCC, PFCC allocates volume more smoothly but with the same

proportionality to the flows. The noisy volume allocation in POCC is caused by coordination issues (i.e. parameter mapping, edge queues) investigated in Section 7.

Figures 9-b and 10-b show the price being advertised to flows in PFCC and POCC respectively. As the new users join in, the pricing scheme increases the price in order to balance supply and demand.

Figures 9-c and 10-c shows the bottleneck queue size in PFCC and POCC respectively. Notice that queue sizes make peaks transiently at the times when new users gets active. Otherwise, the queue size is controlled reasonably and the system is stable. In comparison to PFCC, POCC manages the bottleneck queue much better because of the tight control enforced by the underlying edge-to-edge congestion control algorithm Riviera.

7.1.2 Experiments on Multi-bottleneck Topology

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

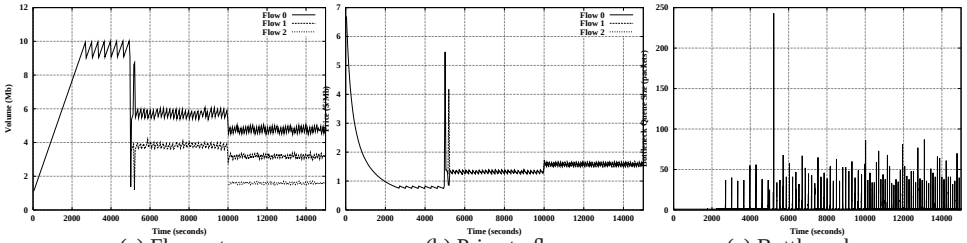
- *Property 1:* provision of various fairness in rate allocation by changing the fairness coefficient α of Distributed-DCC framework (see Section 5.2.1)
- *Property 2:* performance of Distributed-DCC's capacity allocation algorithm ET-ICA in terms of adaptiveness (see Section 5.3.1)

Since Riviera does not currently provide a set of parameters for weighted allocation on multi-bottleneck topology, we will not run any experiment for POCC on multi-bottleneck topology.

In order to illustrate Property 1, we run a series of experiments for PFCC with different α values. Remember that α is the fairness coefficient of Distributed-DCC. Higher α values imply more penalty to the flows that cause more congestion costs. We use a larger version of the topology represented in Figure 8-b. In the multi-bottleneck topology there are 10 users and 9 bottleneck links. Total simulation time is 10,000s. At the beginning, the user with the long flow is active. All the other users have traffic flows crossing the long flow. After each 1000s, one of these other users gets active. So, as the time passes the number of bottlenecks in the system increases since new users with crossing flows join in. Notice that the number of bottlenecks in the system is one less than the number of active user flows. We are interested in the volume given to the long flow, since it is the one that cause more congestion costs than the other user flows.

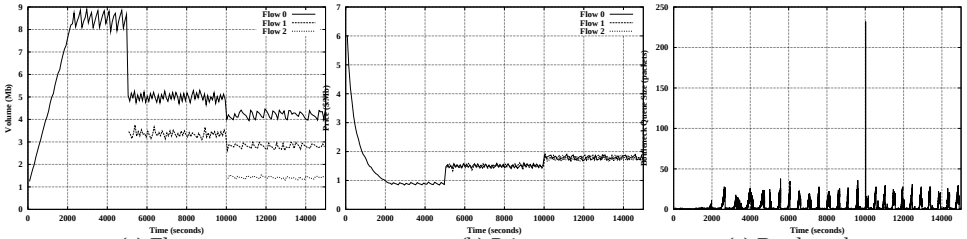
Figure 11-a shows the average volume given to the long flow versus the number of bottlenecks in the system for different values of α. As expected the long flow gets less and less capacity as α increases. When α is zero, the scheme achieves max-min fairness. As it increases the scheme gets closer to proportional fairness. Also note that the other user flows get the rest of the bottleneck capacity, and hence utilize the bottlenecks.

This variation in fairness is basically achieved by advertisement of different prices to the user flows according to the costs incurred by them. Figure 11-b shows the average price that is advertised to the long flow as the number of bottlenecks in



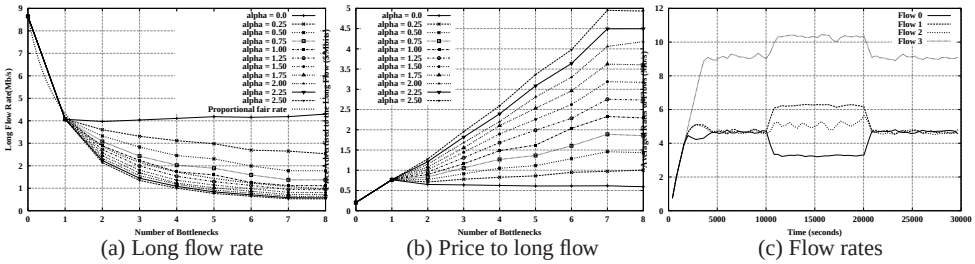
(a) Flow rates (b) Price to flows (c) Bottleneck queue

Fig. 9 Results of single-bottleneck experiment for PFCC.



(a) Flow rates (b) Prices (c) Bottleneck queue

Fig. 10 Results of single-bottleneck experiment for POCC.



(a) Long flow rate (b) Price to long flow (c) Flow rates

Fig. 11 Results of PFCC experiments on multi-bottleneck topology.

the system increases. We can see that the price advertised to the long flow increases as the number of bottlenecks increases.

Finally, to illustrate Property 2, we ran an experiment on the topology in Figure 8-b with small changes. We increased capacity of the bottleneck at node D from 10 Mb/s to 15Mb/s. There are four flows and three bottlenecks in the network as represented in Figure 8-b. Initially, all the flows have an equal budget of 10. Total simulation time is 30000s. Between times 10000 and 20000, budget of flow 1 is temporarily increased to 20. The fairness coefficient α is set to 0. All the other parameters (e.g. marking thresholds, initial values) are exactly the same as in the single-bottleneck experiments of the previous section.

Figure 11-c shows the volumes given to each flow. Until time 10000s, flows 0, 1, and 2 share the bottleneck capacities equally presenting a max-min fair allocation because α was set to 0. However, flow 3 is getting more than the others because of the extra capacity at bottleneck node D. This flexibility is achieved by the freedom given individual flows by the capacity allocation algorithm (see Section 5.3.1).

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

8 Pricing Loss-Guarantee in End-to-End Service

As stated in Section 3, in the contract-switched paradigm, it is possible to offer QoS guarantees beyond the best-effort service. In this section, we develop pricing for loss-guarantees offered along with the base bandwidth service, where the price of the additional loss-guarantee is a component of the overall price. Provision of a loss-based QoS guaranteed service is inherently risky due to uncertainties caused by competing traffic in the Internet. Future outcomes of a service may be in favor of or against the provider, i.e. the provider may or may not deliver the loss-based QoS as promised.

Uncertainty in quality of service is not unique to Internet services [31]. For example, an express delivery company may not always deliver customers' parcels intact and/or on time; and when losses or delays occur, certain remedy mechanisms, such as money back or insurance, are employed to compensate the customers. On the other hand, the provider needs to take into account such uncertainty when pricing its services. In other words, prices should be set such that the provider will be able to recuperate the possible expenses it will incur for attempting to deliver the QoS, as well as the pay-offs to customers when the promised quality of service is not delivered.

When further improving QoS deterministically gets too costly, the provider may be better off using economic tools to manage risks in QoS delivery, rather than trying to eliminate them. We use options pricing techniques to evaluate the risky nature of the loss-guaranteed service. In particular, we consider pricing from the provider's perspective, and evaluate the monetary "reward" for the favorable risks to the provider, which then becomes an added component to the base price of the contract. Pricing the risk appropriately lets the risk be fairly borne by the provider and the customer.

If all ISPs employ an options-based pricing scheme for their intra-domain loss assured services (for details, see [32]), in this section, we consider price of end-to-end loss assurance. Here risk underlying a loss guarantee refers to whether the service the *i*-th ISP delivers to a customer satisfies the loss guarantee specified in the contract or not. The price is set as the monetary "reward" the *i*-th ISP gets when it is able to meet the loss guarantee, S_i^U. The price depends on the loss behavior of a customer's traffic through the ISP's domain, which is dictated by the characteristics of the network and traffic load. Options pricing techniques are used to value the service by utilizing the ISP's preferences for different loss outcomes, captured in a state-price density (SPD) [32].

Let l be the end-to-end data loss rate of a customer's data, $l_{i,\text{contract}}$ be the data loss rate of the customer's data in the i-th ISP's network, and $l^{\text{N}}_{j,\text{contract}}$ be the data loss rate of the customer's data in the j-th transit node, respectively. If the loss rates $l_{i,\text{contract}}$ and $l^{\text{N}}_{j,\text{contract}}$ are less than 20%, the second order terms in their product will be significant to one lower digit of accuracy, hence ignorable. If an ISP is causing a steady loss-rate of higher than 20%, it practically does not make much sense to utilize their services for end-to-end loss guarantees. As loss rates are multiplicative, and all $l_{i,\text{contract}}$'s and $l^{\text{N}}_{j,\text{contract}}$'s can safely be assume to be small, we have

$$l \approx \sum_{i \in V^{\text{N}}_{\text{contract}}} l_{i,\text{contract}} + \sum_{j \in E_{\text{contract}}} l^{\text{N}}_{j,\text{contract}},$$

where $V^{\text{N}}_{\text{contract}} \subseteq V^{\text{N}}$ are the set of ISP contracts and $E_{\text{contract}} \subseteq E$ are the set of transit nodes used in delivering a specific end-to-end contract, and V^{N}, E are all the ISPs and transit nodes in a provider's overlay. The intra-domain contract with each ISP is a guarantee that the loss rate within its domain does not exceed $S^u_i, i \in V^N_{\text{contract}}$. If the provider chooses a threshold value $S^{u,\text{N}}$ as the "assurance" for loss at transit nodes, the end-to-end loss guarantee can be approximately defined as

$$S^u \approx \sum_{i \in V^{\text{N}}_{\text{contract}}} S^u_i + S^{u,\text{N}}. \tag{18}$$

To obtain the price of a contract, a *payoff* function, Y_i, is defined that measures an ISP's performance according to the contract definition. For example, the payoff function of a sample contract, (A): defined in terms of loss-rates starting at $t = 3$pm for a duration of an hour, may be given by

$$Y_i(l_i, S^u_i) = \mathbf{1}_{\{l_i \leq S^u_i\}} |l_i - S^u_i|, \tag{19}$$

where l_i is the loss outcome of the service. Therefore, in general a payoff function, Y_i, can be created that 1) captures whether the loss outcome is within the contracted guarantee S^u_i, and 2) how much better than S^u_i the provider is able to perform. The price of a contract V_i is then determined by the expectation of the payoff under a special probability measure Q, i.e.,

$$V_i = E_Q(Y_i). \tag{20}$$

The probability measure Q captures the providers preference for loss outcomes, and is termed the State Price Density (SPD). Note that V_i and Y_i are time dependent just as l_i. Time is not explicitly indicated due to our assumption of identical time descriptions of all contracts.

For the choice of payoff function given in Eqn. (19), it can be shown that the price for a given l_i process, $V_i(S^U_i) = V_i(S^U_i; l_i)$, has the following properties, 1) $V_i(S^U_i)$ is a convex function; 2) $V_i(0) = 0$; and $V_i(S^U_i)$ is non-decreasing with S^U_i up to a certain $\overline{S^U_i}$, after which $V_i(S^U_i) = 0$. These properties are reasonable since the risk being priced is defined by the outcomes of an ISP's performance against the

contract without any account for efforts in providing the service. A lower S_i^U does not imply a greater effort on the provider's part, neither does a higher S_i^U imply less effort. A higher S_i^U, up to $\overline{S_i^U}$, however does imply that the provider gets a greater benefit of risk bearing. With an increasing threshold, greater loss outcomes will be in favor of the provider, hence the provider's performance with respect to the contract specifications becomes better, therefore gets a higher reward. It should be noted that this is only part of the picture. When the provider violates the contract, he also incurs a penalty. Clearly with an increasing S_i^U the benefits to the provider increase, but the penalty on violation must also simultaneously increase. The penalty determination, however, is beyond the scope of this chapter.

Without the need to define a specific form of an ISP's utility function for losses, we stipulate Q, that captures the ISP's preference structure for loss outcomes, by some general properties of the ISP's preferences for loss outcomes in its domain. 1) The ISP would expect that losses in its domain are rare events during a contract term. 2) If losses do happen, they will more likely take small to moderate values. 3) The ISP will not be rewarded when large losses occur. In particular, we consider ISPs with two types of preference structures, and the corresponding intra-domain prices:

1. An ISP has a strict preference for smaller losses over large losses, i.e., the loss free state is the most favorable outcome to the ISP. This results in ***performance-based prices***. Price is higher when network is at low utilization and the provider is capable of performing in better accordance with the contract.
2. We also consider an alternative preference structure where the loss outcome most desirable to the ISP is at a small but positive level. This will be the case if customers of the ISP services can tolerate certain small losses, as the ISP is possibly able to accommodate more customers by allowing small losses to an individual customer's data. This results in ***congestion-sensitive prices***. Price is higher when network is at high utilization, which discourages customers from buying loss assurances when the network is congested.

Using the assumptions described above, we develop pricing strategies for end-to-end service with loss assurances. An end-to-end loss assured service is characterized by its source and destination (*s–d* pair) location and the loss guarantee, along with other specifications of the service. Consider a contract for service between a certain *s–d* pair with an end-to-end loss guarantee S^u. In the following an *end-to-end contract* (*service*) refers to a contract thus defined, and *end-to-end pricing* refers to the pricing of the end-to-end loss assurance, unless otherwise stated. The provider can acquire and concatenate intra-domain contracts in multiple ways to provide a service. Different customers' traffic between an *s–d* pair may be routed differently within the overlay constructed from acquired contracts; on a given path, the provider may purchase from the same ISP intra-domain contracts with different loss assurances to achieve the end-to-end assurance S^u, as long as Eqn. (18) holds. The routing information and ISP contract types used, however, are invisible to customers. For a price, a customer simply expects to receive the same service from a certain contract. Therefore, at a certain time, prices are determined entirely by the specifications of

the contract – the *s–d* pair and the loss guarantee S^u. In particular, if the end-to-end services can be created in an oligopolistic competition, i.e., there are alternatives for how end-to-end services can be created, and no single provider exerts its monopoly power in creating such services, as well as there is efficiency in how such services can be created dynamically, the following is true about the price of an end-to-end contract.

Proposition 1 *Under the assumption that the market for end-to-end services is competitive and efficient, the price of an end-to-end contract is the lowest price over all possible concatenations of intra-domain services to deliver the end-to-end contract, denoted by* $V^*(S^u)$.

Assume that there are R routes to construct a loss guaranteed service between an *s–d* pair, where path r ($r = 1,...,R$) involves h_r ISPs. On path r, the provider purchases from each constituent ISP, i, a service contract with loss guarantee $S^u_{i,r}$ at a price $V_i(S^u_{i,r})$. The provider assigns a price for the risk for loss at all $h_r - 1$ transit nodes on path r, using a price function V_N. In addition, assume that the provider will assign a threshold value, $S_r^{u,\mathrm{N}}$ as an "assurance" to l_r^{N}, with $\{S^u_{i,r}, S_r^{u,\mathrm{N}}\}$ satisfying the condition of Eqn. (18). V_N depends on the loss assurance $S_r^{u,\mathrm{N}}$ and other characteristics, Θ, of the provider. To solve the end-to-end pricing problem, we first define *price of a path*.

Definition 1 (Price of a path) *The price of path* r, $V^r(S^u)$, $r = 1,..,R$, *is defined as the price of the end-to-end loss assurance if path* r *were the only path to deliver the end-to-end contract.*

The following proposition is obtained by the direct application of Proposition 1.

Proposition 2 *The price of a path* r, $V^r(S^u)$, *for an end-to-end contract with loss assurance* S^u *is determined by the least costly concatenation of intra-domain contracts on path* r, *to eliminate arbitrage between different concatenations of intra-domain contracts, i.e.,*

$$V^r(S^u) = \min_{\{S^u_{i,r}, S_r^{u,\mathrm{N}}\}} \sum_{\substack{i \\ \text{ISP } i \text{ on path } r}} V_i(S^u_{i,r}) + V_N(S_r^{u,\mathrm{N}}). \tag{21}$$

In our earlier work [33], we developed a categorization based pricing scheme for end-to-end bandwidth, where prices are determined by the hop count h and number of bottlenecks (ISPs and/or transit nodes) b on the most likely path between an *s–d* pair. We will utilize this *bh* classification for pricing end-to-end loss assurance, so that the two price components, for bandwidth and loss-guarantee, are combined consistently to form the price of the contract. In addition, only simple cycle-free paths are considered, since presence of cycles artificially inflates traffic in a network, and thus increases the provider's cost in providing the service. By applying Propositions 1 and 2, the provider's pricing problem to determine $V^*(S^u)$ is defined as,

Problem 1 (End-to-end pricing problem)

$$\min_{r \in bh\ class} \min_{\{S^u_{i,r}, S^{u,N}_r\}} \sum_{i \atop \text{ISP } i \text{ on path } r} V_i(S^u_{i,r}) + V_N(S^{u,N}_r)$$
$$s.t. \quad \sum_{i \atop \text{ISP } i \text{ on path } r} S^u_{i,r} + S^{u,\text{N}}_r = S^u \tag{22}$$
$$S^u_{i,r}, S^{u,\text{N}}_r \geq 0, \forall i, r, \textit{ ISP } i \textit{ on path } r.$$

This most general formulation of the pricing problem, where each ISP has a unique price function, can be complex to solve. This is especially true since the provider needs to solve a problem of this form for each *s–d* pair in the overlay. Estimating the price functions $V_i(S^u_i)$ of all ISPs also adds to the complexity of the pricing model. Furthermore, prices need to be recomputed when the provider changes the configuration of its overlay. Therefore, simplifying assumptions will be necessary for solving the end-to-end pricing problem. In Section 8.1, we will conduct a numerical simulation analysis of the above pricing problem.

8.1 Numerical Evaluation

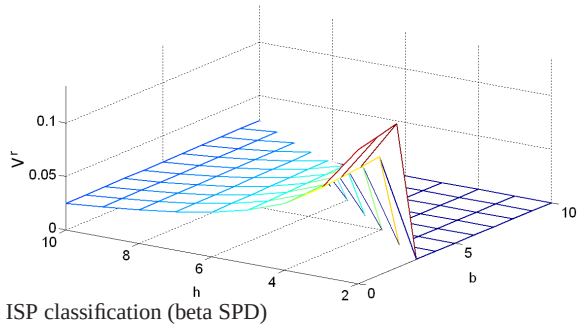


Fig. 12 V^r with ISP classification (beta SPD)

We consider the price of an end-to-end loss assurance S^u between an *s–d* pair in an overlay with $N = 10$ ISPs. We will study the solutions to the end-to-end pricing problems with loss-free and loss prone transit nodes, respectively. For simplicity, we assume that the provider and all ISPs use congestion-based pricing. Instead of a fixed overlay configuration, we consider all possible combinations of the numbers of ISPs (h_r) and of bottleneck ISPs (b_r) on a path r when $h_r \leq 10$. The price functions of underlying ISPs can be described by quadratic functions [32]: $V_0(g) = cg^2, V_1(g) = c_t cg^2, c_t > 1$. We use representative intra-domain price functions with $c = 8667$ and $c_t = 1.074$ $(t = 3$ pm$)$. c_t indicates the difference between the price functions of congested and non-congested ISPs. The price function for transit nodes is also a quadratic function $V_N(g) = c_N g^2$; c_N is varied to examine the effect of the price of transit nodes on the end-to-end price.

Solution to loss-free nodes Fig. 12 shows V^r with different combinations of h_r and b_r on the path, assuming all transit nodes are loss free. We see that V^r decreases with h_r. At the same time, V^r increases with b_r, the number of bottleneck ISPs on the path, for a given choice of h_r; this is also shown in Fig. 13 which gives the relationship between the price of a path and the number of bottleneck ISPs on the path with $h_r = 10$. It is further observed that between h_r and b_r, h_r seems to have a more significant effect on V^r; a longer path involving more ISPs is less expensive than a shorter path, regardless of how many bottleneck ISPs are encountered on the path. Therefore, in this case the end-to-end price V^* will be determined by the longest path that involves the least bottleneck ISPs. This resembles the high diversification principle of the financial portfolio theory. To obtain the end-to-end price V^*, the provider will need to:

(i) Search for the longest path between the *s*–*d* pair in a bh class,
(ii)Among such paths, search for the one involving fewest bottleneck ISPs.

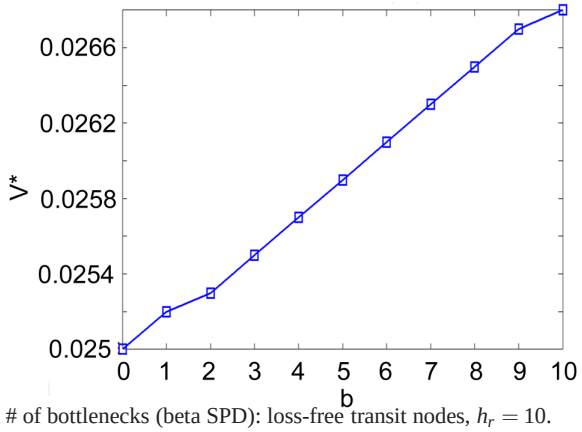


Fig. 13 V^r with # of bottlenecks (beta SPD): loss-free transit nodes, $h_r = 10$.

Solution to leaky nodes We now study the effect of introducing risks at transit nodes to the end-to-end price V^*. We assume the provider uses a congestion-based pricing for risk of loss, producing a quadratic price function $V_N(g;h) = c_{N,h}g^2$, where g, h are the loss assurance and number of ISPs on the path, respectively. Different price functions are studied by varying the coefficient $c_{N,h}$; in particularly, we study the cases when $V_N(g;h) > V_1(g)$, $V_1(g) > V_N(g;h) > V_0(g)$, and $V_N(g;h) < V_0(g)$, $\forall h$, respectively. $c_{N,h}$ also depends on h; we set $c_{N,h}$ to decrease linearly by 5% for every additional transit node involved.

Fig. 14 shows V^r with different h_r, b_r combinations, where $\lambda = 0.3$ is the constraint on the proportion of the loss assurance that can be assigned to transit nodes $S_r^{u,\mathrm{N}}$, and $V_N(g;h) < V_1(g)$ (Fig. 14 (a)), $V_N(g;h) > V_1(g)$ (Fig. 14 (b)), respectively. The scenario with $V_1(g) > V_N(g;h) > V_0(g)$ looks similar to the loss-free transit node case (Fig. 12) and is not shown here.

Comparing Fig. 14 with Fig. 12, we can see that introducing risks at transit nodes decreases the overall price levels for V^r, regardless of the relative relations between

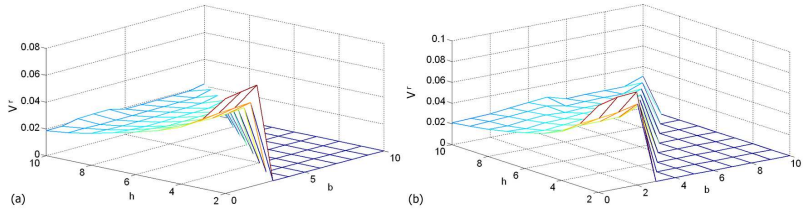


Fig. 14 V^r with leaky transit nodes (beta SPD, $\lambda = 0.3$): (a) $V_1(g) > V_0(g) > V_N(g;h)$, (b) $V_N(g;h) > V_1(g) > V_0(g)$.

the price functions, although as expected, between these two scenarios the decrease in V^r is more significant with a lower price function V_N in Fig. 14 (a). Similar to the case of loss-free transit nodes, the primary effect on V^r is of h_r; that is, the price of a longer path involving more ISPs and transit nodes is always lower. However, the relationship between V^r and the number of bottlenecks on the path b_r, becomes irregular, as is also seen in Fig. 15. Therefore, to obtain the exact end-to-end price, the provider will need to search for all the longest paths in a bh class, and choose V^* as the price of the least expensive path.

In all scenarios considered above, the longest paths involving more ISP contracts will be preferred in constructing the end-to-end loss assurance. This resembles the diversification technique to reduce risk in risk management [34]. The provider benefits from allocating risk in the end-to-end loss assurance to more ISP domains.

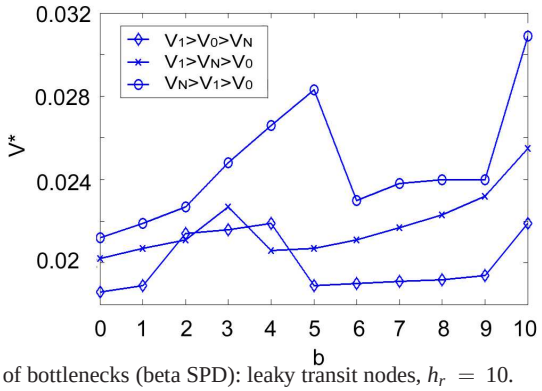


Fig. 15 V^r with # of bottlenecks (beta SPD): leaky transit nodes, $h_r = 10$.

9 Summary

In this chapter, we describe a dynamic congestion-sensitive pricing framework that is easy to implement and yet provides great flexibility in rate allocation. The Distributed-DCC framework presented here can be used to provide short-term contracts between the user and the service provider in a single diff-serv domain, which allows the flexibility of advertising dynamic prices. We observe that Distributed-DCC can attain a wide range of fairness metrics through the effective use of an Edge-to-Edge Pricing (EEP) pricing scheme that we provide. Also, we introduce

two broad pricing architectures based on the nature of the relationship between the pricing and congestion control mechanisms: Pricing for Congestion Control (PFCC) and Pricing over Congestion Control (POCC). We show how the Distributed-DCC framework can be adapted to these two architectures, and compare the resulting approaches through simulation. Our results demonstrate that POCC is better in terms of managing congestion in network core, while PFCC achieves wider range of fairness types in rate allocation. Since Distributed-DCC is an edge-based scheme, it does not require upgrade of all routers of the network, and thus, existing tunneling techniques can be used to implement edge-to-edge closed-loop flows. An incremental deployment of Distributed-DCC is possible by initially installing two Distributed-DCC edge routers, followed by replacement of others in time.

We also describe a framework for pricing loss guarantees in end-to-end bandwidth service, constructed as an overlay of edge-to-edge services from many Distributed-DCC domains. An options-based pricing approach is developed for end-to-end loss guarantees over and above the price for basic bandwidth service. This provides a risk sharing mechanism in the end-to-end service delivery. The price of the end-to-end contract is determined by the lowest price over all valid intra-domain contract concatenations. Based on certain simplifying homogeneity assumptions about the available intra-domain contracts, numerical studies show the importance of diversification in the path chosen for end-to-end service.

Acknowledgments

This work is supported in part by National Science Foundation awards 0721600, 0721609, and 0627039. The authors wish to thank Shivkumar Kalyanaraman for his mentoring and Lingyi Zhang for excellent research assistance.

References

1. D. D. Clark, J. Wroclawski, K. R. Sollins, R. Braden, Tussle in cyberspace: Defining tomorrow's Internet, IEEE/ACM Transactions on Networking 13 (3) (2005) 462–475.
2. X. Wang, H. Schulzrinne, An integrated resource negotiation, pricing, and QoS adaptation framework for multimedia applications, IEEE Journal on Selected Areas of Communications 18 (12) (2000) 2514–2529.
3. N. Semret, R. R.-F. Liao, A. T. Campbell, A. A. Lazar, Pricing, provisioning and peering: Dynamic markets for differentiated internet services and implications for network interconnections, IEEE Journal on Selected Areas of Communications 18 (12) (2000) 2499–2513.
4. J. K. MacKie-Mason, H. R. Varian, Pricing the Internet, Kahin, Brian and Keller, James, 1993.
5. A. Bouch, M. A. Sasse, Why value is everything?: A user-centered approach to Internet quality of service and pricing, in: Proceedings of IEEE/IFIP IWQoS, 2001.
6. M. Yuksel, S. Kalyanaraman, Effect of pricing intervals on congestion-sensitivity of network service prices, Telecommunication Systems 28 (1) (2005) 79–99.
7. S. B. et. al, An architecture for Differentiated Services, IETF RFC 2475 .
8. A. M. Odlyzko, The economics of the Internet: Utility, utilization, pricing, and quality of service, Tech. rep., AT & T Labs (1998).
9. A. M. Odlyzko, Internet pricing and history of communications, Tech. rep., AT & T Labs (2000).
10. I. C. Paschalidis, J. N. Tsitsiklis, Congestion-dependent pricing of network services, IEEE/ACM Transactions on Networking 8 (2) (2000) 171–184.

11. A. Gupta, D. O. Stahl, A. B. Whinston, Priority pricing of Integrated Services networks, Eds McKnight and Bailey, MIT Press, 1997.
12. F. P. Kelly, A. K. Maulloo, D. K. H. Tan, Rate control in communication networks: Shadow prices, proportional fairness and stability, Journal of Operations Research Society 49 (1998) 237–252.
13. N. Semret, R. R.-F. Liao, A. T. Campbell, A. A. Lazar, Market pricing of differentiated Internet services, in: Proceedings of IEEE/IFIP International Workshop on Quality of Service (IWQoS), 1999, pp. 184–193.
14. X. Wang, H. Schulzrinne, Pricing network resources for adaptive applications in a Differentiated Services network, in: Proceedings of INFOCOM, 2001, pp. 943–952.
15. A. M. Odlyzko, A modest proposal for preventing Internet congestion, Tech. rep., AT & T Labs (1997).
16. D. Clark, Internet cost allocation and pricing, Eds McKnight and Bailey, MIT Press, 1997.
17. D. Clark, A model for cost allocation and pricing in the Internet, Tech. rep., MIT (1995).
18. R. Cocchi, S. Shenker, D. Estrin, L. Zhang, Pricing in computer networks: Motivation, formulation and example, IEEE/ACM Transactions on Networking 1 (1993) 614–627.
19. M. Yuksel, S. Kalyanaraman, Simulating the Smart Market pricing scheme on Differentiated Services architecture, in: Proceedings of GLOBECOM, 2001.
20. R. Singh, M. Yuksel, S. Kalyanaraman, T. Ravichandran, A comparative evaluation of Internet pricing models: Smart market and dynamic capacity contracting, in: Proceedings of Workshop on Information Technologies and Systems (WITS), 2000.
21. S. H. Low, D. E. Lapsley, Optimization flow control – I: Basic algorithm and convergence, IEEE/ACM Transactions on Networking 7 (6) (1999) 861–875.
22. G. S. Arora, M. Yuksel, S. Kalyanaraman, T. Ravichandran, A. Gupta, Price discovery at network edges, in: Proceedings of International Symposium on Performance Evaluation of Telecommunication Systems (SPECTS), 2002.
23. M. Yuksel, Architectures for congestion-sensitive pricing of network services, Ph.D. thesis (2002).
24. S. Kunniyur, R. Srikant, End-to-end congestion contro: Utility functions, random losses and ecn marks, in: Proceedings of Conference on Computer Communications (INFOCOM), 2000.
25. F. P. Kelly, Charging and rate control for elastic traffic, European Transactions on Telecommunications 8 (1997) 33–37.
26. J. Mo, J. Walrand, Fair end-to-end window-based congestion control, IEEE/ACM Transactions on Networking 8 (5) (2000) 556–567.
27. M. Yuksel, S. Kalyanaraman, Elasticity considerations for optimal pricing of networks, in: Proceedings of IEEE Symposium on Computer Communications (ISCC), 2003.
28. D. Harrison, S. Kalyanaraman, S. Ramakrishnan, Overlay bandwidth services: Basic framework and edge-to-edge closed-loop building block, Poster in SIGCOMM (2001).
29. UCB/LBLN/VINT network simulator - ns (version 2), http://www-mash.cs.berkeley.edu/ns (1997).
30. H. R. Varian, Estimating the demand for bandwidth, in: MIT/Tufts Internet Service Quality Economics Workshop, 1999.
31. B. Teitelbaum, S. Shalunov, What QoS research hasn't understood about risk, Proceedings of the ACM SIGCOMM 2003 Workshops (2003) 148–150.
32. A. Gupta, S. Kalyanaraman, L. Zhang, Pricing of risk for loss guaranteed intra-domain Internet service contracts, Computer Networks 50 (2006) 2787 – 2804.
33. A. Gupta, L. Zhang, Pricing for end-to-end assured bandwidth services, International Journal of Information Technology & Decision Making 7 (2) (2008) 361 – 389.
34. M. Crouhy, D. Galai, R. Mark, Risk Management, McGraw-Hill, New York, NY, 2001.
35. D. M. Chiu, Some observations on fairness of bandwidth sharing, Tech. Rep. TR-99-80, Sun Microsystems Labs (September 1999).
36. S. Shenker, Fundamental design issues for the future Internet, IEEE Journal on Selected Areas of Communications 13 (1995) 1176–1188.
37. H. R. Varian, Intermediate Microeconomics: A Modern Approach, W. W. Norton and Company, 1999.