Lightweight Policing and Charging for Packet Networks

Bob Briscoe Mike Rizzo Jérôme Tassel
Kostas Damianakis
<bob.briscoe@bt.com>
BT Research, B54/130, Adastral Park, Martlesham Heath, Ipswich, IP5 3RE, England
Tel. +44 1473 645196

28 Nov 1999

Abstract

This paper suggests that a multi-service packet network might be achieved by adding classification and scheduling to routers, but not policing. Instead, a lightweight, packet-granularity charging system is presented that emulates a highly open policing function and is completely separated from the data path. A high proportion of charging operations runs on customer systems to achieve this, the proportion being configurable per-customer. Functions dispersible to customers include not only metering, accounting and billing but also per-packet or per-flow policing and admission control. Lower cost is achieved through simplicity without sacrificing commercial flexibility or security. Inter-provider charging, multicast charging and open bundling of network charges with those for higher class services are all catered for within the same, simple design. The paper is primarily architectural, referring to supporting papers for reports of early implementation experience in an Internet context.

Keywords: Charging, pricing, congestion control, quality of service, policing, active networks, Internet.

1 Introduction

The goal of a multi-service network is to simultaneously support applications with varying elasticity of requirements, ranging from those that happily cope with best efforts to the more stringent demands of interactive, real-time audio or network games. Traditionally, supporting multiple service levels requires: i) a protocol to classify the service required; ii) policing to bound the level of requests, protecting the network by denying or degrading admission in times of overdemand and iii) mechanisms to schedule resources for each class. Charging is generally relegated to a second order position in support of such resource rationing.

The ideas in this paper can be adopted at a number of levels. At one level, the paper simply offers a very cheap and highly open way of charging for multi-service packet networks by moving nearly all operations to customer machines. At the most ambitious level, charging takes over as the primary control mechanism, in the process completely removing the need for policing from a multi-service packet network. Instead the architecture of any domain can be restructured to exploit the efficiency of 'optimistic access control'. We maintain that a full multi-service packet network might be achieved with the complexity of policing and charging completely separated out except at the end-systems. This leaves the network infrastructure clear to simply classify, route, schedule and forward. If classification is based on the simple differentiated services model (diff-serv) [4], this implies no need at all for flow-related state in the network.

However, this doesn't imply the ideas will only work if adopted by all network providers. The solution is intrinsically designed to inter-work with non-usage-charging approaches allowing each provider to either evolve to it independently or choose not to. As well as being open to other approaches from peers, the architecture completely opens up the commercial interfaces of the network layer, encouraging innovative bundling of services by third parties.

We argue that the charging mechanism should match the granularity of the provision of the network service — the packet. We argue that if the service granularity is finer than the charging granularity, price signals will never be able to optimise utilisation. Also commercial flexibility becomes limited. Systems that can charge at packet granularity can be specialised to charge for arbitrary aggrega-

tions of packets (e.g. flows), but the reverse isn't so. Moreover, we have shown through implementation that very efficient packet granularity charging is possible.

We suggest using the generally spare cycles on customer machines for measurement and aggregation (with their agreement). Clearly, this move to customer systems creates a security problem. We solve this using simple, random audit at a frequency tailored to each customer. The whole proposal is very much along the same lines as 'pay and display' parking policed by traffic wardens. Although we cannot yet estimate the cost of our proposals, we can confidently predict moving charging operations to customer machines will cost less to run than 'phone charging systems, despite charging down to packet granularity rather than just per call.

However, the full benefits of a move to customer-based charging operations are dependent on another proposal in this paper: multicast dissemination of electronic tariffs and price changes to customers. This in turn has potentially far-reaching implications on the management, support and even marketing of network services. The greatest cost implication of this work could flow, not just from the dispersal of operational costs to customers, but from the potential ability to dynamically match network provision and demand using dynamic pricing.

Customer acceptance of dynamic pricing is a controversial subject [28]. However, none of the studies separate people's aversion to risk from the nuisance of regular price changes, which software could alleviate. Whatever, we show that a network might still be managed by dynamic pricing even if many customers have a preference for price stability. This is achieved by offering price stability itself at a price. We also question that preference for price stability will necessarily remain strong, given software could hide the nuisance of volatility. Further, the *threat* of dynamic pricing can also be used to encourage software to back-off before the price actually changes — unless the choice is to accept the price change. Even if the systems are only used to notify quarterly price changes in the short term, we believe it makes sense to build in the capability to announce far more volatile price changes for use in future years.

Although per-packet charging is often dismissed as impractical, per-packet policing is universally accepted, usually per-domain and sometimes per hop. We propose having *neither* in the network, moving multi-service policing to the customer as well. Instead of the network denying access when a class of service becomes over-subscribed, we propose increasing the price (or threatening to) in order to *avoid* congestion, but only as a last resort after borrowing resources from 'lower' classes of service [20]. Those customers that have most price sensitivity will then back off of their own accord.

Unlike policing of service level agreements (SLAs) or reservations, usage-charging encourages an open, 'don't-ask-just-do-it' approach, but with irresponsibility tempered by pricing. We assume, as dynamic pricing is introduced, a market will develop in software that controls adaptive applications dependent on price [36]. The mechanisms implemented can cater for both long run and short lived price changes. They use zero extra bandwidth at the time of congestion, thus avoiding a congestion avalanche. This is achieved by downloading the algorithm relating price and congestion to customer machines. Thus, when the standard indications of congestion change, the relevant price movement can be calculated locally. If congestion avoidance pricing is found acceptable, policing can be removed from the data path, giving reduced latency and increased simplicity.

We must make it clear that this architecture *allows* for dynamic pricing and the removal of policing from the network, it doesn't require or mandate either of them. The advantages of opening the network business to innovation and of a convenient way to quickly introduce new prices or pricing plans is sufficient rationale for the work.

We take an engineering approach, focussing on architecture. For brevity our reports on implementation experience are included by reference. Given the ideas are rather radical, this approach was chosen as a first step rather than modelling. We needed to understand what was required and clarify what was technically feasible. Guidance on what basis is best for network charging (what should be measured, how often prices should change, how stable they should be, etc.) is reserved for future publications. In practice, the best basis for charging will be worked out by natural selection in the market-place, but will require the advice of the research community. A major goal has been to allow ISPs the commercial freedom to use the mechanisms to differentiate themselves while still ensuring the Internet will self-manage.

Ruth provides a review of other engineering approaches [31] with Edell *et al* [16] and Brownlee [10] being notable reports of large-scale practical experience. Much of the practical work gains some validity from the relatively new field of Internet economics, comprehensively collected at the 1995 MIT Internet Economics Workshop [2]. Utility curves mapping QoS classes to willingness to pay typically provide the bridge between economic theory and engineering practice [33].

After a brief statement of our main assumptions, we present the core of the paper in Section 3 — the principles of a good charging architecture, focussing only on issues of concern to a communications audience. We consider the fundamentals of how direction and transmission mode affect the flow of value from a network, summarising the pre-requisite concepts in Briscoe [8]. We move on to consider the major building blocks of a general charging system, taking a fresh look at traditional practice in the field. We include inter-provider charging and the orthogonal issue of charging for the service the network offers to higher layers. For each aspect, we set down principles, describe systems we have invented to comply with the principles and then justify our choices. Section 4 briefly describes the results of our work so far. The closing sections highlight the main weaknesses of the approach, suggest further work and draw conclusions.

2 Assumptions & definitions

We assume network charging based on network addresses, therefore we briefly describe how ISPs should track the identity of the party liable for use of each address, dealing with scenarios where this mapping might be very volatile (e.g. address translators or multicast). Some scenarios involve an anonymous end-customer, but some other party can always be assumed to have underwritten their liability (e.g. coin-operated kiosks).

We assume that the near-universal case will be no more than one customer for each unicast network address. However, the transport and higher layers *appear* to complicate the business model by multiplexing network service to multiple customers sharing the same network address. Examples are multi-user desktop machines and Web or video hosting services. But, the subcontracted customers only own identifiers buried deep within certain packets: respectively port numbers [16] and URL fragments in our examples. So end-to-end protocols (including TCP) use addressing that doesn't key to any aspect of the network service. Therefore, apportioning network charges on this basis simply creates pressure for customers to implement new end-to-end protocols by private arrangement between themselves. This is why, pragmatically, a single party can be held responsible for all use of a network address, with any other users subcontracting this liability.

Our assumed business model for a minimalist ISP has boundaries that match the service access points above and below the network layer of the OSI stack. However, the rate and burstiness at which service is delivered *from* the network layer is typically controlled with transport layer protocols. For instance TCP includes an implicit rate control 'contract' between network and end-system [1], while the RSVP flowspec is a very explicit contract [37]. We assume routine packet drop will not be an appropriate congestion control mechanism for many higher QoS classes — those where the delay of retransmit is to be avoided.

We use 'class of service' to mean a unique combination of network service mode (unicast, multicast etc.) and quality specification (latency, instantaneous bandwidth, reliability, jitter). The quality specification can either have fixed values for each parameter, or one class of service might fix some parameters while allowing the others to take a range of values specified by the customer. This generically covers both RSVP and diff-serv.

3 Principles and architecture

3.1 Charging granularity

Most other work has started from the premise that per-packet charging is clearly impractical, suggesting instead aggregations such as quotas [5], addresses [15] or SLAs [13, 23]. In consequence, network quality of service (QoS) architectures always embed some assumption about packet aggregation whether just for charging, or also for the QoS mechanism itself. For instance, the Internet Integrated Services Architecture (ISA) [6] uses a flow as the finest grained unit of QoS provisioning. Consequently this sets the minimum granularity when charging for QoS. On the other hand the Internet differentiated services (diff-serv) architecture proposes the packet as the unit of QoS provision, but suggests a traffic conditioning agreement (TCA) based on flows as the charging granularity. The assumption that the granularity of charging will have to be greater than the granularity of use then becomes embedded in the infrastructure, e.g. diff-serv TCA policing is being standardised as if it were the *only* possible way to turn the diff-serv byte into a business proposition.

Further, if charging is on a coarser granularity than service provision, there will always be a gap between what is paid for and what is used. On the one hand, this creates an incentive for the customer to waste resources paid for but not needed (e.g. with crude robotic activity such as indiscriminate pre-fetching caches). On the other, it puts competitive pressure on the network provider to over-book capacity, thus eroding the original service guarantees. A diff-serv

TCA suffers heavily from this flaw, especially one for 'all-addresses', which institutionalises a low utilisation factor. Over-provisioning of a single class of service is merely an extreme example of the same low utilisation assumption [29, 7]. Applying this assumption indiscriminately is vulnerable to global changes in application behaviour, with a dangerous lack of any alternative control system.

Therefore we propose that the charging mechanism should match the granularity of the network service — the packet. Not only because this doesn't artificially limit utilisation, but also because it allows much fuller commercial flexibility.

3.2 The direction of value flow

Briscoe analyses the factors affecting the value senders and receivers derive from communications [8]. It generalises edge-pricing for all transmission modes: unicast, multicast and aggregation (e.g. RSVP reservations). The conclusions are summarised here, as any charging architecture must be based on the flow of value as well as cost. For those not familiar with the original edge-pricing model [34], we summarise it in two sentences first: A provider's edge-price covers its neighbours' charges plus its own costs. Each provider is free to absorb variations in neighbour prices while setting its own prices.

Briscoe shows that the common case is for value to flow from the network provider outwards to each of its customers, whichever direction traffic is flowing. This is because the large majority of transmissions are with the consent of all ends. In consequence a general architecture must cater for a provider offering each class of service in each direction at a separate price. However, a large number of cases remain where all the ends may have a very different apportionment of the value of transmitting a packet compared to the send and receive prices charged by their local networking providers. If this discrepancy is large enough and prolonged enough, it should be rectified end-to-end, not through all the networking providers on the data path.

If all edge-provider income is normalised to local prices first, inter-provider charging then becomes simple wholesale of bulk traffic distinguished only by class, without worrying about how payment for each packet or flow was apportioned at the edge. This further allows the relationship between any two network providers to collapse to a single pair of prices per class of service for each direction of transmission. The price in either direction can be further thought of as the difference between the 'half' prices that each provider offers the other. Each provider's 'half' price can be considered to be for transmission between the edge in question and their respective remote edge of the Internet. Briscoe calls this model 'split edge-pricing'.

Using the end-to-end business models proposed in Briscoe allows this architecture to be efficient and scalable. The mechanisms for pricing, accounting and payment that we discuss below could be used in other models, such as [14] or [18]. However, they lead to per flow charging in the core of the Internet and are to be avoided where possible.

3.3 Charging system topology

A usage-charging system must have two major functional parts if it is to control a system: a pricing control loop and metering to apportion charges. The question is "Where should functions execute and where should data reside? — on routers, in packets, on the provider's operational support systems or on the customer's system(s)?"

We propose the principle that **the charging system should be as separate as possible from the transmission system**. That is, there should be 'zero bits for charging', not two bit solutions or even one [27, 26]. Information in the network should only determine the behaviour of machines forwarding it. Any charges for this behaviour should be referred to indirectly, by mapping between what packet fields do and how much will be charged for doing it — in effect a contract or quotation.

Fig 1 shows different contracts for different domains mapping behaviours in the transmission infrastructure to tariffs in the charging infrastructure. Mapping between the two infrastructures via dynamic contracts allows each to evolve independently. Because each domain can offer its own contracts mapping its own traffic classes to its own tariffs this leaves each provider open to exploit its own commercial freedom. Rather than standardising the contractual meaning of special bits in packet headers, each provider is free to define through its tariffs what any pattern of bits means at any time and only for its own customers. The tariff then drives what is metered, allowing future charging schemes to be arbitrarily complex or simple.

We also propose the principle of **moving as much as possible to the customer machine**. This helps scalability, opens up customer control and improves responsiveness to price. We propose that the above contracts, or at least the tariffs they contain, should be code running on customer machines — active tariff objects. This brings the customer machine fully into the pricing control loop. All the inputs to any typical

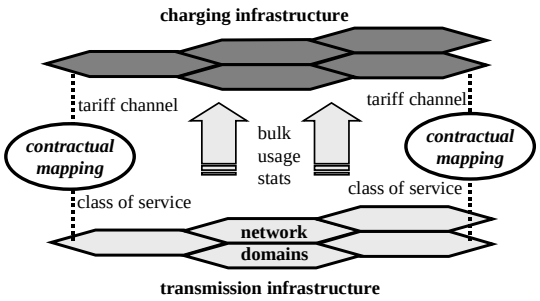

Figure 1: Zero bits for charging

tariff algorithm are already available on the customer system, or can easily be made available. This gives the customer immediate charging feedback so that her behaviour can be optimised to the price signals of the provider. Over time, we expect that the customer will suffer less and less interaction in order to accept or reject prices, as software agents similar to ours are written to control application behaviour based on customer spending policies, application preferences and these tariff algorithms.

More radically, we also propose moving metering and accounting to customer machines. The cost of meters placed at edge-routers will grow proportionally to the bandwidth that more and more end systems will throw at them. Worse, the charging systems behind the meters tend to be more disk speed limited than, say, multimedia systems, because the former demand data reliability and audit. Worse still, in a multi-service network, the requirements include metering not only volume but also service rate and burstiness. This is a particularly heavy load for routers to bear both in terms of memory and processor load, as experience with RSVP has shown. Even though very efficient token bucket mechanisms can be used that have only a tiny incremental effect per packet, the aggregate effect on a router handling many flows is still highly significant, not to mention the complexity of correctly placing and configuring the policer. For this and other reasons, an applicability statement had to be made in 1997 limiting RSVP to small-scale scope for the foreseeable future [3], a statement that applies equally to policing multi-field diff-serv classifications.

On the other hand, to measure the packet activity and flow dynamics of one host using the same host is naturally scalable. The most likely party interested in detailed usage data is the customer or local software managing resources on her behalf. The provider is generally satisfied with a statistical view. Therefore it makes sense to aggregate all the data at the customer before forwarding it to the provider. As long as either party can tailor the level of detail at any one time, it is best for the default to cater for the common case.

Thus, each customer will be able to calculate her own bill continuously and pay it on whatever schedule might have been previously agreed — self-billing. However, even though the support systems will be running remotely, the provider will want to retain the primary levers of control:

- pricing levels
- data reporting frequency and granularity
- audit of the revenue flow

The provider retains control over price levels by invoking a remote method on the tariff objects on customer machines. Below, we briefly describe how the provider can also update the whole tariff on the fly using the same mechanism. For efficiency, these tariff changes are disseminated by multicast. The customer would have to permanently join the relevant multicast groups to hear tariff changes for her class of service. The contract would need to make this clear — the contract could even include code to ensure this happened.

The provider can retain control over data reporting from the accounting system by remotely accessing the customer machine. The contract would need to make access a condition of service. Instead of the quarterly or monthly batch bills of today, these billing reports might be requested to be triggered every day, every hour or even every few minutes. The provider simply has to issue new instructions to change the frequency of reports. The same mechanisms can be used to ensure all future reports give data of a different granularity. This might be required to support a new marketing campaign, for instance. Multicast could be used unless per customer changes were required. It is also possible in our prototype for the provider to configure the reporting system back to the traditional billing mode with metering done by the provider and bills being sent to the customer. This might be necessary if there were a restricted back channel (e.g. satellite).

To audit the revenue flow, contradicting our previous statement, we propose the provider *will* run metering on edge routers, but only on a sampled basis. At any time, the provider might choose to sample a particular customer without her knowledge. Measurements are collected and aggregated to see if they match the regular reports being sent back from the customer accounting system. The sample is taken for a whole multiple of the customer's reporting periods, and in phase lock. The two are compared and if

within tolerance, the provider measurements are discarded. There might be a few customers being sampled per edge router at any one time, but the measurement load would never approach that required for full metering on the edge router. Because metering can be in parallel to forwarding, no single customer can detect that her data is being sampled (e.g. by a marginally increased end-to-end latency).

If there were a discrepancy between the customer's and provider's accounts, the provider would store both sets of data as evidence. The sampling rate might be increased on that customer. If the discrepancy persisted for future samples, eventually some penalty action would be triggered. For instance, customers might have been asked to pay a deposit and the penalty might be to take the shortfall from the deposit. For certain customers, it might be necessary to just run the system in the traditional billing mode — effectively 100% sampling.

3.4 Pricing

3.4.1 Tariff dissemination

We propose that tariffs are 'announced' over multicast channels, much as sessions are described with the session description protocol (SDP) [21] then announced on the mbone. With edge-pricing, customers only have to listen to their local provider's channels. Unlike SDP, which consists of text fields, we wish a tariff to be executable code for full flexibility. In our testbed, we serialise Java bytecode, authenticate it, add a serial number then multicast it as described in Rizzo *et al* [30]. Carle *et al* chose a text-based tariff description protocol [12], a valid initial approach, but one we deliberately avoided so as not to stifle tariff innovation. In the Results and Further Work sections we discuss how we deal with the security issues of a customer having to run arbitrary code from her provider.

Tariff announcements are regularly repeated on the soft state model. We use soft state to give timeliness to new announcements without polling and to add the reliability of repeated sends to unreliable Internet multicast. For efficiency, at least two multicast channels should be used per provider. One to disseminate the tariff classes themselves while the other announces metadata about the tariffs (e.g. the current version number, a description, minor price level fluctuations etc.). Tariffs should also be available by request-reply (e.g. Web) for newly booting machines.

Soft state reliability leads to a potential problems if the customer system doesn't hold an up to date tariff. The customer might have to risk using the service without knowing the exact price (the protocol ensures they at least know they are missing an announcement). If price changes have shown a history of being minimal, most customers would be willing to take such risks — the discrepancy would be a small proportion of a small amount. On the other hand, the possibility of loss allows her to fake non-receipt of a price rise message. To solve this, the protocol includes the activation time of the new price or tariff and a customer must make retrospective adjustments if she is out of date. However, typically, a new price might be repeatedly announced in advance of its activation time.

The system can emulate the fairly slow time-scales that providers allow today for customers to consider future price changes (a period of notice might be part of the contract). However, our general architecture must cater for a future where providers might choose to give very little notice, raising the issue of customer acceptance of dynamic pricing. Indeed, our 'tariff activation time' field can be set to null implying immediate activation on receipt. And a half round-trip time is the physical minimum delay possible in the price control loop, which our choice of multicast announcement is deliberately capable of achieving. Our solution, which we believe is novel, is to include a price for price stability itself in a tariff. Risk averse customers can choose to pay a premium for price stability (as everyone effectively does today), while those willing to play the market can take the spot price at any one time in the expectation that their average costs will be less. One simple algorithm linking a percentage premium to a period of stability can be applied to the spot tariffs of all classes of service. Disseminating this as an active tariff offers the customer a variable degree of price stability. Customers might make this decision on a long-term basis for all their usage, or just for specific sessions. An example of the latter might be a long conference where the local customer doesn't want to commit to starting without knowing the exact cost. The provider can set the price of stability so as to encourage enough customers to use low stability prices, in order to ensure instantaneous network demand can still be managed through these customers. The longer a demand trend continues, the more customers will lose the right to a period of price stability. Thus, although more research is clearly necessary, it is appears feasible that pricing can manage demand even allowing that some customers will opt for periods of price stability.

It appears that the provider has to have a certain degree of trust in the customer's idea of time (and vice versa). However, the random audit technique deters any deliberate clock shifting by customers, because they can only fool their own system, not the

provider's. The regular tariff announcements provide a convenient channel to state the provider's view of the time while the accounting reports (described later) provide a return channel where the customer can state the time they received the tariff. Thus, the round trip time is constantly being agreed between the two ends. There is scope for customers to undetectably shift their clocks to their advantage within the deviation of the round trip time, but their financial gain from such shifts would be minimal.

3.4.2 Price control

We propose that price control should operate in two scopes: i) a base price for each class of service typically applied to a whole provider domain and varying over fairly long timescales, much as prices vary today; and ii) an optional premium on this price for congestion avoidance (see next section). How base prices are set is outside the scope of this paper. However, we expect continuing automation of today's price setting processes, with inputs based on network load monitoring, routing alternatives; resource availability in lower classes of service; new capacity costs and lead-times; the competitive position; and commercial strategy. Base prices would be disseminated electronically, as described in the previous section.

It is also possible that each tariff for each class of service for each direction could be further subdivided by various ranges of local and remote address. Thus, if there were spare capacity in just one region of a network, it might be possible to offer lower prices to customers based on the combination of their address and the remote address of any flow. These combinations would be the ones known to route through the lightly loaded region. We do not believe distance based pricing is very likely, but it is possible within the present architecture. Tariffs can include weightings to be applied to the standard price for combinations of source and destination address prefix lists.

3.4.3 Congestion avoidance pricing

Above, we suggest how to control price either for whole domains or, if necessary, for regions within a domain. However, our price dissemination mechanism is not scalable enough to control the congestion of any queue on any interface on any router in the Internet. Because data is bursty, such congestion can appear and disappear in different queues fairly rapidly. This would require a price announcement from every queue and some way of targeting these announcements just to those edge-systems transmitting through that queue at that time. Worse, we really want to charge for ignoring congestion backoff protocols, rather than charge for innocent use of a path that becomes congested. Even if price announcements could be targeted, we wish to allow each provider commercial freedom. This would include the freedom to absorb peaks and troughs in prices from neighbouring providers in order to present simpler pricing to neighbouring customers. In such scenarios, congestion prices for whole network regions would become lost in the noise.

Instead, the mechanism we suggest is to use existing congestion signalling, which effectively creates a signalling channel from any point of congestion to the end-systems using it. Various schemes exist or are proposed for the Internet, which avoid a congestion avalanche by highlighting the packets that are causing the congestion. This can either be done implicitly by dropping them (as detected by TCP and RTCP [32]) or explicitly, e.g. using the explicit congestion notification (ECN) bits [19] proposed for diffserv. All these methods require the co-operation of the receiver to notify the sender of congestion through an end-to-end protocol.

We *could* require all these congestion signalling protocols to be changed to make the marks on the packets represent a shadow price [24]. Instead, we can try to keep the protocols as they are and re-introduce the commercial freedom given by edge-pricing. We do this by including congestion signalling as one of the parameters pre-programmed to drive the active tariff. That is, by assuming a competitive market, we can allow providers to each map this shadow price to their own real market price. Thus, congestion can still be signalled in a standard way, but it can refer to charging information indirectly. This is again in compliance with the principle of keeping transmission and charging infrastructures separate. Backbone providers would charge their neighbours heavy penalties for congestion being ignored. In turn their neighbours would probably translate these into heavy penalties for end customers if they ignored congestion. However, other neighbours might choose to exercise their freedom and offer fixed prices while translating upstream price increases into denials of access to their customers rather than price changes.

As one might expect, before a router signals the need for congestion avoidance, the network should: i) attempt to re-route around the congestion; ii) borrow capacity from 'lower' classes of service; and/or iii) introduce or at least request extra capacity (possibly automatically). Further, we are not suggesting that the provider should *always* avoid congestion using price. It may be that it is decided (possibly through a rule-based system) that more revenue can

be gained by allowing the network to fill and denying service to subsequent requests. Thus, even if congestion onset is signalled, we allow the network provider to control how strongly (and even whether) congestion signals cause a price rise, by multicast 'tuning' of everyone's active tariffs. Also note that the level of overprovisioning determines how often congestion avoidance pricing is required. It could be arranged to occur as infrequently as 'equipment engaged tone' does in the PSTN.

We must also repeat that we can use the hierarchy of service classes to advantage because they can borrow capacity from each other. We can manage the load on each logical service class with price because we can always grow the size of each logical class (as long as there is sufficient best effort buffer at the bottom). It is only the best effort level that needs to have the ability to deny service — and it already does that by dropping packets. We should be clear that we are talking about congestion *avoidance* in a logical class of service. This scheme is unlikely to make sense for congestion *control* with best effort traffic. Thus we assume the onset of congestion is signalled before any packets have to be dropped. A logical service class that is approaching capacity could start penalising end-systems that didn't back off at the first sign of congestion, completely under the autonomous control of the active tariff. This would be particularly appropriate where a logical service class was designed for applications that couldn't tolerate much (or any) packet drop — the reason ECN was proposed in the first place. Using active tariffs also avoids any need for an avalanche of price rise signalling.

Kelly has already shown congestion pricing can be stable if everyone is using the same algorithm. However, making congestion control dependent on such unpredictable factors as customer sensitivity to real market prices and provider profit motive could be dangerous for total network stability. More work is required to find if there is a class of algorithms that simultaneously allow commercial freedom but still exhibit stability in reasonable scenarios.

A more practical problem is that the direction of congestion signalling, whether implicit or explicit, is typically towards the receiver. The receiver is then trusted to reflect this information back to the sender. The receiver has an interest in the sender not backing off, therefore, it may be necessary to charge the receiver if she ignores congestion, giving an incentive to reflect the network's signals correctly. To allow heterogeneity, we cannot assume the remote ISP will operate congestion charging, but we can assume it has an incentive to reduce congestion, which is all that is necessary. For instance, the remote ISP might instead, choose to run spot checks to ensure that its receivers are complying with end-to-end congestion control protocols. Finally, we propose charging the sender of the original data flow on the basis of incoming congestion signalling (from the original receiver) if it is necessary to encourage correct congestion behaviour.

For TCP and RTCP it is necessary to bury into the transport layer header of the returning packets to measure the original sender's liability for charges. If the transport layer is encrypted, this presents a nasty 'layering interaction'. However, it gives the receiver an incentive *not* to encrypt congestion feedback, which is unnecessary for security anyway. Therefore, either explicit or implicit congestion signalling can be measured, whether at the edge-router or at the end-system, and therefore form the basis of charging.

We have ensured the incentives are correct for non-co-operating sets of senders and receivers. Before any host chooses to ignore congestion back-off requirements and instead pay the price, the following steps are available to more co-operative sets of customers:

1. the end-systems should consider setting QoS requirements to a 'higher' class (if cheaper than the fine for ignoring congestion at the current class)
2. the end-systems should decide it is essential to ignore the congestion, given the fine for doing so might be quite high
3. both (all) end-systems should agree to ignore the congestion

The other side of the price control loop is just as important to understand — the customer reaction to price. We assume software components similar to those in our earlier work in Tassel *et al* [36] will be added to many types of adaptive applications, but the study of how users will configure their price behaviour is left for future work.

3.4.4 Optimistic access control

Typically, three per-packet operations are added to routing and forwarding to create a multi-service packet network: classification, policing and scheduling. Of these, classification and policing involve look-ups into tables of potentially dynamic state. These tables can become large, requiring a few levels of look-up hierarchy, each of which takes resource. Further, state introduces complexity in keeping it current, keeping it in the right place and garbage collecting the resources it uses when it is finished with. Soft state is often used to alleviate these problems.

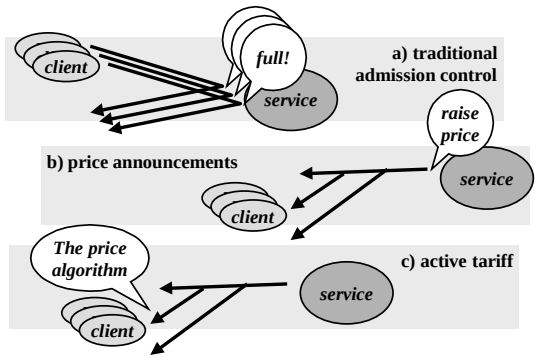

Figure 2: Admission control at source

However, as routes change, keeping route dependent state on the correct router introduces further complexity.

As far as classification is concerned, diff-serv has already taken the step of confining the number of possible classifications to a small non-dynamic table, thereby making classification and scheduling simple and independent of flow state. However, in all known multi-service network proposals, each packet cannot be forwarded until it has been checked against policing policies — 'pessimistic access control' [25]. Examples are token bucket policing against RSVP flowspecs or diff-serv SLAs [13]. The flow *cannot* continue in parallel to the policing process because the only punishment available is to hurt the packet in question. This is a particular problem in connectionless networks where any one packet cannot necessarily be related to a later packet. Novel, ultra-fast forwarding techniques will give little benefit if policing becomes the major component of latency (Amdahl's Law

In contrast, because metering is passive, we can use 'optimistic access control' where packets are allowed through without checking, but customers know they will have to answer for whatever they do. The optimistic model can only exist within some wider pessimistic context. A charging system offers just such a context. For instance, a customer may only be given an Internet account after revealing her verifiable identity, or paying a deposit. Once past this pessimistic hurdle, optimistic access to more specific parts of the service can be allowed, protected only by a punishment (respectively legal action or docking the deposit) if payment doesn't match metered activity. Metering is passive because it can proceed on the memory copy of the header, in parallel to the flow of the service. Therefore latency is always kept low in this optimistic model. Thus, the combination of an optimistic within a pessimistic model needs just a single blocking test. The effect of this single test can persist for months or even years.

Through adding a dynamic price mechanism, we can ensure that those people least willing to use a service at a certain price *deny themselves* access. The provider is then freed from having to fend off flash crowds with an avalanche of admission control signalling (Fig 2a) or the complexity of RSVP blockade state. With the tariff pre-loaded into the customer machine, we can even avoid any 'raise price' signalling at the time of congestion (Fig 2b-c). Thus, potentially, we have even moved both admission control and policing to the customer machine.

3.5 Metering, accounting and payment

We now describe a system architecture (Fig 3) for measuring usage and paying for it, confining ourselves to issues relevant to a communications audience. The left to right division is between customer and provider systems, with otherwise identical classes of objects distinguished respectively by the subscripts 'c' or 'p'. We have already described how the primary charging system can be operated by either the provider or the customer, depending on whether the system is configured for traditional or self-billing. This explains the essential symmetry of the figure. The networking service is shown flowing from provider to customer, regardless of transmission direction. The main chains of objects on either side drive a flow of data upwards:

- service usage is metered, M, where it is also aggregated under simple rules arbitrated by the meter controller, MC
- accounting, Act, consolidates results from possibly a number of meters and deals with hard storage and customer identity
- rating, Ra, is where prices are applied to the usage data to calculate charges.
- ultimately payment, Pa, of the calculated charges will be required

External control (not shown) of each side of these systems is primarily by the contract, particularly the tariff, which determines what to measure, what and how often to report, and current pricing. An agent of the customer or provider respectively arbitrates this control. Control is typically implemented by caching a policy in the relevant object, whilst listening for policy update events. Feedback to complete the control loop is shown flowing from bulk measurement on the

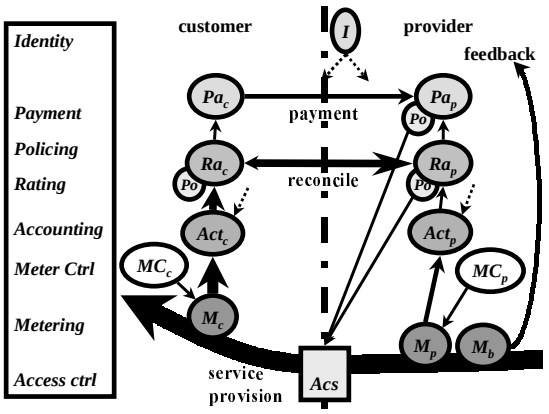

Figure 3: Metering, accounting, rating and payment

provider side ('bulk' means queue lengths etc. rather than individual flows).

We use the term reconciliation to cover the 'pay-and-display' model and traditional billing. In both cases, the aim is agreement over the level of usage. The most efficient level to reconcile customer and provider is after prices have been applied, as shown. Both sides may wish to police reconciliation, with failure to reconcile triggering some defensive action. In the case of the provider this is shown as ceasing access, but less drastic measures may precede this. Payment is also policed by the provider — it is no use agreeing on the amount owing then not checking the amount paid. Rather than expect total accuracy in reconciliation, the system would be designed to ensure discrepancies due to system problems were as likely to be positive as negative. Arbitrary packet loss from flows in either direction between the two meters clearly makes this impossible. Therefore the two meters should ideally sit on either end of the same link. If link layer error correction is in use, metering should occur after correction. Metering should be as low as possible in the stack to avoid any contention for resources within the stack due to packets with different QoS classes. Ultimately, as long as packet losses were low, both parties could write them off by widening their tolerance to reconciliation discrepancies. This introduces a security flaw, as either side can try to probe the limits of the other's tolerance. Marginally higher prices for everyone would have to cover these potential losses — similar to the cost of pilfering, spillage and spoilage in the retail trades.

Next we discuss *who* is considered liable for charges. We assert that network metering should only be concerned with apportioning use of the network service to various network addresses. The identity service, I, in Fig 3 is responsible for the mapping between network address and real world identities. This is another example of the principle of separation between network and charging. Address allocations themselves need to be 'metered' and 'usage data' sent to the accounting function to be consolidated with the network usage-data for each address (shown as dotted arrows). Examples of address allocation schemes are dynamic host configuration protocol (DCHP), network address translators (NATs), mobile cell hand-off and multicast address allocations, joins and leaves. A many to one mapping between addresses and identities must be assumed. For instance, one customer becomes liable for reception charges for any traffic to a multicast address group as soon as the join to the group from her machine is measured (in this case it is the link address in the join request that is relevant).

Finally, it is architecturally imperative that accounting and payment are treated totally separately. The process of reconciling the account with the provider proceeds in two stages: first it is agreed which usage occurred, then which usage the customer is liable to pay for. That is, the local customer's account should always include all the usage data for both sent and received traffic, whoever might pay for it. Account reconciliation can occur on a completely different schedule to payment and with any arbitrary other party.

Because the architecture we have described applies all the principles given, it will charge correctly even in the most demanding scenario. However, in inter-domain scenarios it will only be efficient by using the 'split edge-pricing' model. For instance, Briscoe works through an example scenario with inter-domain multicast and heterogeneous QoS per receiver [8]. That example also shows how different apportionments of charges between senders and receivers can be achieved for multicast, again by dealing separately with such issues — end-to-end rather than along the same path as the data.

3.6 Charging inter-operation

3.6.1 Higher layer systems

In our discussions above, and those on the direction of value, we showed how the remote end of a transmission might want to 'wholesale' some charges from the local ISP then 'retail' them back to the user at the local end. Examples are: a video on demand service that 'bundles' transmission quality charges with their videos; or the 'originator pays' model used in traditional telephony. In general, any third party not even involved in the communication might pay the local network charges. Examples are the user's employer or an agency organising a videoconference.

The simple step taken above of separating accounting from payment, combined with multicast pricing announcements, opens up the business of networking, allowing arbitrarily complex business models to be built on the foundations of the present architecture. All the relevant information is available for verification by third parties, whatever degree of trust they put in the network users for whom they are paying. For instance, prices are authenticated by the ISP. Even if the ISP is coy about allowing non-customers to join the pricing multicast, the local user can unicast the authenticated tariff object to the third party. If requested by the third party, the relevant usage data can also be authenticated and sent, either by the ISP or the local user. We are not saying these arrangements *have* to be authenticated — the third party might simply offer to cover charges up to a flat ceiling, whatever they actually were.

3.6.2 Inter-provider

So far, it may have been assumed that all scenarios only applied to edge providers and customers. However, only the previous section, on inter-operation with higher layers, is specifically about end systems. The rest of the architecture is just as applicable to an inter-provider relationship. Briscoe [8] shows how any two ISPs have a customer-provider relationship, depending only on the sign of the difference between the 'half' prices that they charge each other. We propose that all the principles described above apply equally to this relationship — the architecture is recursive. For pricing, tariffs would be announced from each provider to its neighbours. Each provider could base its prices on the costs it was receiving from its neighbours, combined with its own commercial objectives. (Even if the edge-pricing model weren't used, non-neighbouring providers could still receive these price announcements.) We should clarify that, while the architecture is recursive, it may not be appropriate to use identical *technology* inter-provider. We can use multicast and Java between relatively centralised systems managing loading and pricing for whole network domains. However, alternative, fully distributed mechanisms are the subject of continuing research.

For accounting reconciliation, each pair of neighbours could either both measure or, for efficiency, agree only one need measure — then the other need only sample. In fact this is similar to the standardised model for international carrier accounting in the PSTN [22]. Here, both parties measure everything but the payment is based on the *payer's* data. Unlike the PSTN though, as long as payments are normalised end-to-end first (as recommended earlier) the choice of factors to measure at any inter-provider boundary is local to that pair of providers. Thus, even if edge-provider to edge-customer charging is at packet granularity, charging between the same edge-provider and a peer-provider could be by average link utilisation per service class.

Recursion also applies for corporate customers, where there may be a need to apportion costs between departments. The accounting reports would first be reconciled between end-systems and departmental accounting systems. Then departments would aggregate and reconcile with corporate and provider systems. The model can even be applied recursively to multi-user machines. Each user simply runs an accounting object that reports to the machine's accounting object. Intra-machine accounting would be based on addressing above the network layer as already discussed under Assumptions. In all these recursive models, the detailed accounting within one system can either be encapsulated from the broader system or revealed if the commercial relationship requires it. If there is any shortfall between the total of all the details and the total the provider expects, that is the responsibility of the party at that level to underwrite.

This neatly brings us to the final issue to resolve in this paper: "What about failure to deliver the specified service?" With an end-to-end service, it can be very costly to apportion blame when things go awry. If the customer has paid for reliability, who should give the refund if a packet is dropped? If latency is guaranteed, who gives the refund if some network has used too much of the delay budget? Briscoe [8] proposes a pragmatic principle for these circumstances. If a customer disputes payment and their local provider accepts their case (or can't be bothered to argue) all providers on the end-to-end path share the same fate, losing the associated revenue, or at least not bothering to account for the refund between themselves. This is akin to peering between ISPs today, but need only be applied to the exceptional cases of failure. Hence this is termed 'exception peering'. Just as with peering, occasional audit would be necessary to ensure gains and losses were within acceptable tolerance of each other.

4 Early results

Other than producing the architecture summarised here, we have also implemented example prototypes of the system components described above, primarily in Java. These include the provider and customer

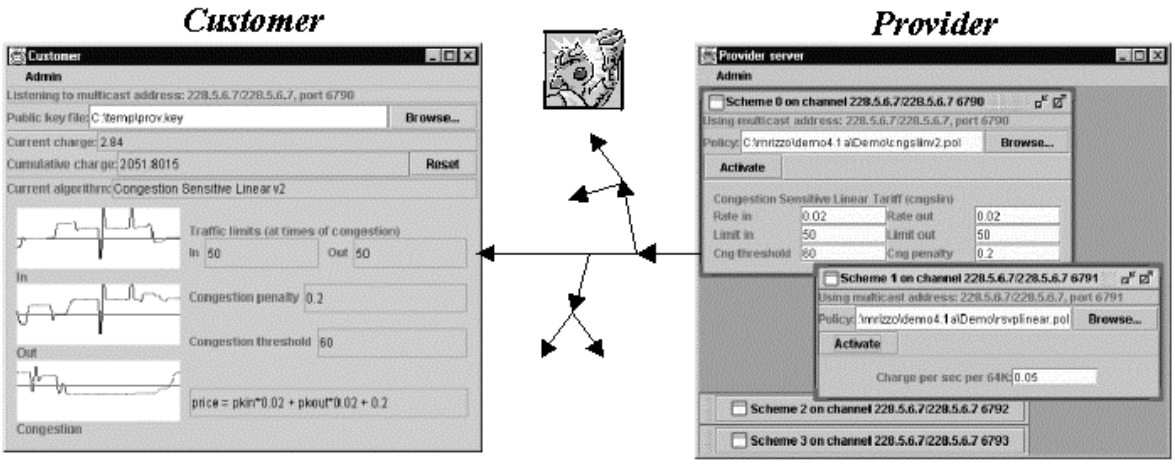

Figure 4: Demonstration of dynamic tariff dissemination and pricing

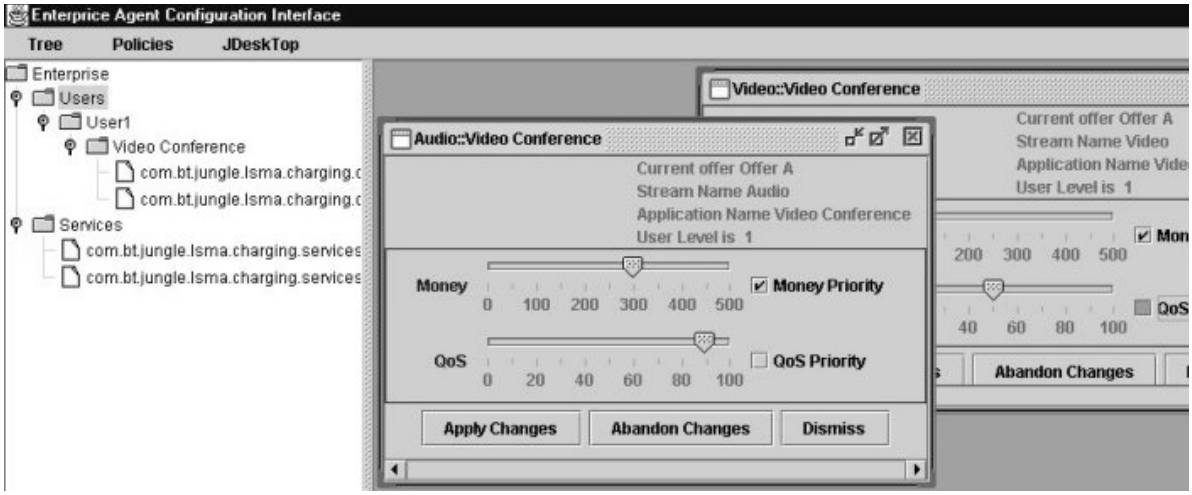

Figure 5: Policy control of price adaptation agent demo

ends of the tariff dissemination mechanism and a generic accounting system for either providers or customers with remote control capabilities. These are described in Rizzo *et al* and Tassel *et al* [30, 35].

Implementation proves that the architectural separation is clean, as there is minimal necessity for standardisation. We implement active tariffs in Java, the most complex one to date being 8kB on the wire. The receiver end includes a modified class loader to switch to new tariff objects on the fly. Remote method calls are turned into objects for dissemination then back into calls once received. The incoming object currently undergoes rudimentary checks: for a trusted code signature and to check conformance with the expected interface by introspecting the class of the object. We have also built a component for integrating local charging into any Java application. It requires only a minor edit to the relevant socket calls, being based on Tassel *et al* [36].

The whole charging system currently consumes 1.6MB of storage on the customer device, including 875kB for a stripped NeTraMeT [11], which we use for metering that complies broadly with the real-time flow measurement (RTFM) architecture [9]. We are about to start more scientific experiments, but initial indications show that the charging system consumes about 3% of CPU time on a 400MHz Pentium II for moderate traffic loads. Because the code is intended as a flexible research testbed, no attempt at streamlining has been made. This might reduce storage and system load further, particularly where single purpose Internet devices are concerned.

5 Limitations and further work

The ideas presented here are rather radical and hence prone to unpredicted weaknesses appearing as they are developed. Currently there are six areas that cause us most concern:

- Dispersal of software critical to an ISP's revenue flow into an unreliable installation environment. Although we can fall back to traditional billing, our scalability benefits rely on this fall back not being needed too often. This should be less of a problem with single purpose devices than general-purpose computers.

- Customer acceptance of foreign code installing and running itself on their systems. We have implemented rudimentary checks described above, but further work is required to allay customer security fears.

- Resource allocation purely by price could lead to bursty hogging. The laws of economics should protect us against the static effects of hogging [13] — the hogs will pay for the capacity they hog, leaving the remainder for the rest of us. However, heavily bursty traffic from relatively few price insensitive customers may cause a disproportionate need for over-capacity.

- Demand management by price could fail if deterrents like penalties or credit blacklisting suddenly became too weak, e.g. due to crisis events. Rather than remove admission control from the network, it may be advisory to leave a vestigial 'defence in depth' to be turned on in emergencies.

- Meter discrepancies between customer and provider. This becomes a problem with lossy access link technologies.

- Lack of experience of dynamic pricing behaviour of customers and providers [17]. Experiments are needed where risk aversity is distinguished from the nuisance of dynamic pricing. Also futures pricing of communications is a sparsely researched topic.

All these areas except the first are subject to further research. This paper has been presented before we have any macro scale predictions from scientific experiment, modelling or simulation, on the premise that the ideas are of interest in their own right.

6 Conclusions

A number of innovations have been proposed which will need further modelling and testing. However, it can be concluded that these rather radical proposals do offer a realistic alternative to tight, policy-based provision of multi-service networks that currently holds a near-monopoly of ideas in the industry (intserv and diffserv). This architecture is principled and simple and has a strong chance of producing a highly scalable, lightweight, high performance, secure and open system, but above all, it will be cheap to run.

A radical shift of charging functions from provider to customer systems is proposed. The provider maintains the minimum control necessary by distributing active tariff objects to all customers. Allowing tariffs to be arbitrary encourages business flexibility and innovation. The customer, on the other hand, gains full control of all other aspects of the charging system. All the provider's price signals are available locally and can be used by the customer or her software to aid decision-making. Timely feedback on the financial implications of every action are also available locally. Current networks totally rely on customers not fully utilising the services being sold. As customer software becomes increasingly sophisticated this assumption could become dangerous. Instead it is proposed that customers should be given exact price signals so they can co-operate, rather than compete, with the provider's goals. Further, the architecture deliberately focuses on the commercial interfaces to just the network layer business, so that this can be bundled in to wider collections of services in whatever innovative way is desired.

The proposed charging architecture allows as many factors as possible to be configurable. Provider control over price and reporting are the only assumptions about who might trust whom to do what. Granularity of all service charging can be as low as a single packet, but per-flow or inter-provider bulk aggregations are just as possible. However, the processing load of aggregation is distributed across customer machines. The architecture should also cater for 'clock speed' improvements from the quarterly billing and price change cycles of today to sub-second reports and price changes in the future if required. Suggesting that price stability can be offered at a price, it is shown how risk averse customers can be offered stable pricing while others can pay generally lower but more volatile prices.

It is argued that flow policing would be unnecessary in any domains where this charging architecture was in place. Instead, the customers could be relied

on to police themselves due to price back-pressure and wider deterrents against defaulting on payments. Thus, effectively, flow-based access control can also be moved to end-systems. Instead of policing every packet at every router, or at least at every border router, this offers the alternative of merely measuring every packet once at each end of its transmission. Further, measuring can be done in parallel to forwarding, whereas policing, although very lightweight, requires forwarding to be delayed until it completes. Combined with simple classification schemes like those proposed for Internet diff-serv, this would remove any need for flow related state on routers. This would also remove the complexity required to keep such state on the right routers if routes needed to change. The proposed model is termed 'optimistic access control'.

Weaknesses that give concern have been listed. Subsequent discussion about this and other equally valid approaches to the same problem will doubtless highlight further limitations. However, the authors believe it is imperative that networks should remain simple. This paper offers a multi-service packet network with the complexity of policing and charging completely separated out except at the ends. This leaves the network infrastructure clear to simply classify, route, schedule and forward.

Acknowledgements

Pete Bagnall, David Freestone, Ben Strulo, Trevor Burbridge, Jake Hill, Tony Reeder, Steve Rudkin, Sarom Ing, Ian Marshall, Alan O'Neill, Maff Holladay, Phil Flavin, James Cochrane, (BT); Jon Crowcroft (UCL)

References

[1] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. Request for comments 2581, Internet Engineering Task Force, URL: rfc2581.txt, April 1999.

[2] J. Bailey, S. Gillett, D. Gingold, B. Leida, D. Melcher, J. Reagle, J. Roh, R. Rothstein, and G. Seale, editors. *Internet Economics Workshop Notes*, Boston, MA, URL: http://www.press.umich.edu/jep/works/BailWNotes.html, March 1995. Research Program on Communications Policy, MIT.

[3] F. Baker, B. Braden, S. Bradner, A. Mankin, M. O'Dell, A. Romanow, A. Weinrib, and L. Zhang. Resource ReSerVation protocol (RSVP) — version 1 applicability statement Some guidelines on deployment. Request for comments 2208, Internet Engineering Task Force, URL: rfc2208.txt, January 1997.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. Request for comments 2475, Internet Engineering Task Force, URL: rfc2475.txt, December 1998.

[5] Roger Bohn, Hans-Werner Braun, Kimberly C. Claffy, and Stephen Wolff. Mitigating the coming internet crunch: multiple service levels via precedence. Technical report, University of California-San Diego, San Diego Supercomputer Center and NSF, URL: http://www.nlanr.net/Papers/mcic.html, November 1993.

[6] R. Braden, D. Clark, and S. Shenker. Integrated services in the Internet architecture: an overview. Request for comments 1633, Internet Engineering Task Force, URL: rfc1633.txt, June 1994.

[7] Lee Breslau and Scott Shenker. Best-effort versus reservations: A simple comparative analysis. *Proc. ACM SIGCOMM'98, Computer Communication Review*, 28(4), September 1998.

[8] Bob Briscoe. The direction of value flow in multi-service connectionless networks. In *Proc. International Conference on Telecommunicatons and E-Commerce (ICTEC'99)*, URL: http://www.labs.bt.com/projects/mware/, October 1999.

[9] N. Brownlee, C. Mills, and G. Ruth. Traffic flow measurement: Architecture. Request for comments 2063, Internet Engineering Task Force, URL: rfc2063.txt, January 1997.

[10] Nevil J. Brownlee. New Zealand experiences with network traffic charging. *ConneXions, Usage-based access charging*, 8(12), December 1994.

[11] Nevil J. Brownlee. *The NeTraMet System, Software Release Notes*. URL: http://www.auckland.ac.nz/net/NeTraMet/, December 1997.

[12] Georg Carle, Felix Hartanto, Michael Smirnow, and Tanja Zseby. Generic charging and accounting for value-added IP services. In *Proc. IEEE IFIP International Workshop on QoS (IWQoS'99) pricing workshop*, June 1999.

[13] David D. Clark. A model for cost allocation and pricing in the Internet. In *Proc. MIT Workshop on Internet Economics*, URL: http://www.press.umich.edu/jep/works/ClarkModel.html, March 1995.

[14] David D. Clark. Combining sender and receiver payments in the Internet. In G. Rosston and D. Waterman, editors, *Interconnection and the Internet*. Lawrence Erlbaum Associates, Mahwah, NJ, URL: http://diffserv.lcs.mit.edu/, October 1996.

[15] Jon Crowcroft. Pricing the Internet. In *Proc. IEE Colloquium on Charging for ATM*, pages 1/1–4, November 1996. (Ref. no. 96/222).

[16] Richard Edell, Nick McKeown, and Pravin Varaiya. Billing users and pricing for TCP. *IEEE Journal on Selected Areas in Communications, "Advances in the Fundamentals of Networking"*, September 1995.

[17] Richard Edell and Pravin Varaiya. Providing Internet access: What we learn from INDEX. Technical report 99-010W, Uni of CA-Berkeley, URL: http://www.INDEX.Berkeley.EDU/public/index.phtml, April 1999. Also submitted to IEEE Network Magazine.

[18] George Fankhauser, Burkhard Stiller, Christoph Vögtli, and Bernhard Plattner. Reservation-based charging in an integrated services network. In *Proc. 4th INFORMS Telecommunications Conference, Boca Raton, FL*, URL: ftp://ftp.tik.ee.ethz.ch/pub/people/stiller/paper/informs98.ps.gz, March 1998.

[19] Sally Floyd. TCP and explicit congestion notification. *ACM SIGCOMM Computer Communication Review*, 24(5):10–23, October 1994. (This issue of CCR incorrectly has '1995' on the cover).

[20] Sally Floyd and Van Jacobsen. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, August 1995.

[21] Mark Handley and Van Jacobsen. SDP: Session description protocol. Request for comments 2327, Internet Engineering Task Force, URL: rfc2327.txt, March 1998.

[22] New system for accounting in international telephony. Rec. D.150, ITU-T, URL: http://www.itu.ch/intset/itu-t/d150/d150.htm, October 1996.

[23] Frank P. Kelly. Charging and accounting for bursty connections. In Lee W. McKnight and Joseph P. Bailey, editors, *Internet Economics*, pages 253–278. MIT Press, URL: http://www.statslab.cam.ac.uk/~frank/charge.html, 1997.

[24] Frank P. Kelly, Aman K. Maulloo, and David K. H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49, 1998.

[25] Peter F. Linington, Zoran Milosevic, and Kerry Raymond. Policies in communities: Extending the ODP enterprise viewpoint. In *Proc. 2nd International Workshop on Enterprise Distributed Object Computing (EDOC'98), San Diego*, pages 11+, URL: http://www.cs.ukc.ac.uk/pubs/1998/740/, November 1998.

[26] Martin May, Jean-Chrysostome Bolot, Christophe Diot, and Alain Jean-Marie. 1-bit schemes for service discrimination in the Internet: Analysis and evaluation. Research report 3238, INRIA, Sophia Antipolis, France, URL: http://www.inria.fr/RRRT/RR-3238.html, August 1997.

[27] Kathleen Nichols, Van Jacobson, and Lixia Zhang. A two-bit differentiated services architecture for the Internet. Request for comments, Internet Engineering Task Force, URL: rfc2638.txt or URL: http://diffserv.lcs.mit.edu/Drafts/draft-nichols-diff-svc-arch-00.pdf, July 1999.

[28] Andrew Odlyzko. A modest proposal for preventing Internet congestion. Technical report TR 97.35.1, AT&T Research, Florham Park, New Jersey, URL: http://www.research.att.com/~trmaster/TRs/97/97.35/97.35.1.body.ps or URL: http://www.research.att.com/~amo/doc/modest.proposal.pdf, September 1997.

[29] Andrew Odlyzko. The economics of the Internet: Utility, utilization, pricing, and quality of service. *Proc. ACM SIGCOMM'98, Computer Communication Review*, 28(4), September 1998.

[30] Mike Rizzo, Bob Briscoe, Jérôme Tassel, and Kostas Damianakis. A dynamic pricing framework to support a scalable, usage-based charging model for packet-switched networks. In *Proc. Int'l W'kshp on Active Networks (IWAN'99)*, volume 1653, URL: http://www.labs.bt.com/projects/mware/, February 1999. Springer LNCS.

[31] Gregory R. Ruth. Usage accounting for the Internet. In *Proc. INET'97, Kuala Lumpur*, URL: http://www.isoc.org/isoc/whatis/conferences/inet/97/proceedings/F1/F1_1\%.HTM, 1997.

[32] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. Request for comments 1889, Internet Engineering Task Force, URL: rfc1889.txt, January 1996.

[33] Scott Shenker. Fundamental design issues for the future Internet. *IEEE Journal on Selected Areas in Communications*, 1995.

[34] Scott Shenker, David Clark, Deborah Estrin, and Shai Herzog. Pricing in computer networks: Reshaping the research agenda. *ACM SIGCOMM Computer Communication Review*, 26(2), April 1996.

[35] Jérôme Tassel, Bob Briscoe, Mike Rizzo, and Kostas Damianakis. Scaleable usage based Internet accounting. Technical report, BT, URL: http://www.labs.bt.com/projects/mware/, 1999.

[36] Jérôme Tassel, Bob Briscoe, and Alan Smith. An end to end price-based QoS control component using reflective Java. In *Proc. 4th COST 237 workshop*, URL: http://www.labs.bt.com/people/briscorj/papers.html\#QoteS, December 1997. Springer LNCS.

[37] Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: A new resource ReSerVation protocol. *IEEE Network*, September 1993.