Congestion Pricing for Congestion Avoidance

Peter Key   Derek McAuley   Paul Barham
Koenraad Laevens

Microsoft Research, Cambridge, UK

February 1999

Technical Report
MSR-TR-99-15

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Abstract

This paper describes the use of Congestion Pricing as a means of providing Congestion Control and Differentiated Quality of Service. The application of the proposed technique to the Internet Protocol has the advantage that it can be simply implemented using Explicit Congestion Notification. In particular: the network mechanism is independent of higher level protocols; the end systems can continue to exhibit current TCP behaviours; new multiprotocol flexibility is made available to end systems and users.

Architectural issues are discussed, including important aspects of aggregation and charging. We describe our methodology for assessing the scheme via a distributed network simulator. Initial results are presented which compare and contrast various adaptive strategies that achieve a variable range of TCP-like behaviours.

1 Introduction

The current control mechanisms of the Internet are not well suited to offering different Quality of Service (QoS) to different channels. TCP, the dominant transport control protocol, is appropriate for "best-effort" traffic, and aimed at data where lost information has to be retransmitted. However, other types of traffic, such as streamed video or audio, are sensitive to delay as well as packet loss. Such traffic is often carried over UDP, which has no generic feedback signals or error recovery.

Certain commercial IP offerings make use of the TOS field to give a two-level service offering, though this is only possible when all routers are under the provider's control. The Diff-Serv proposals to provide a small number of QoS classes can be seen as an extension of this idea. Within the Diff-Serv community, Explicit Congestion Notification (ECN) has been proposed as means of triggering active queue management disciplines to implement congestion avoidance.

Economists (e.g. [10]) have suggested using pricing as a means of avoiding congestion in the Internet, while Kelly and his co-workers [8, 5] have described a mathematical framework for fairness and resource pricing. Using some of these concepts, we propose the use of Congestion Prices to trigger ECN.

There are some differences between Congestion Pricing ECN and current ECN suggestions; our proposals are protocol independent, able to operate at the IP layer, not just from *some* Transport protocols. Also our underlying philosophy is to send signals to *all* the users[1] that are causing congestion, rather than sending signals to some subset; this permits end-systems more freedom in how they react, allowing them to determine their own QoS, knowing that they will be charged correctly. Indeed, the interplay between Users and the Network can be viewed as a problem in (distributed) game theory.

We are primarily thinking of flows that can adapt, sometimes termed "elastic" flows. However, others have suggested that it is possible to include these ideas with guaranteed services as well, by using a proxy between the user and network [4].

The outline of the paper is as follows. In the next section we give background and motivation for our ideas. We then present a proposed architecture, which includes how to deal with aggregates and suggestions for charging mechanisms. This is followed by an experimental framework for assessing congestion control by means of a distributed network simulation. We comment on some practical implementation issues, and give some qualitative results, where we compare Congestion Pricing and Adaptive Flow Control schemes with current versions of TCP and ECN proposals.

[1]For ease of exposition we use the term "users" to represent a reactive system, more likely to be a protocol stack or software agent rather than a human.

2 Background: Congestion Pricing and ECN

2.1 Explicit Congestion Notification (ECN)

Explicit Congestion Notification (ECN) sends some form of notification signal of Network Congestion to an end-user or application, with the implicit assumption that the user will react to such a signal by decreasing the offered rate. The aim is usually to encourage or mandate users to co-operate and so run the network in a "sensible" state, namely a mostly uncongested one. The basic idea has a long pedigree, and one of the earliest suggestions was the "DECbit" [7]. Frame Relay uses FECN, while the binary-feedback scheme of ABR uses Explicit Forward Congestion Indication, with the added wrinkle of virtual-source virtual-destination pairs to shorten control loops.

Floyd [2] proposed using ECN with TCP, and a recent IETF RFC [16] outlines an implementation in detail, the main goal being to enhance the existing congestion control. As this ECN proposal has to co-exist with TCP and achieve "equal fairness", of necessity the proposal requires flows to react to ECN signals in the same way as TCP reacts to lost packets, for instance halving the congestion window on receipt of such a signal.

The ECN RFC [16] proposes one bit to signify congestion (that is, a single feed back signal), a flag (the ECT bit) for signalling an ECN-capable source and a Congestion Window Reduced (CWR) flag to indicate a reduction in the Congestion Window. The underlying assumption is that ECN will be triggered by some active queue management control, similar to that used for Random Early Detection (RED) [3], but with routers marking ECN-aware flows rather than dropping packets. However, since these flows are required to react in the same manner as TCP, it is hardly surprising that experimental evidence only shows "moderate" difference between ECN and non-ECN flows [2]. We regard our work as a further development of the ECN proposal, illustrating how a different notion of fairness between flows can be achieved by allowing end-systems to react rather differently than under TCP.

Models of various sophistication [11, 14, 15] have shown that if packets (segments) are lost independently, say with probability p, then the steady-state throughput of a "normal" TCP connection in connection avoidance (CA) mode is proportional to $\frac{MSS}{RTT\sqrt{p}}$ with a constant of proportionality around 1. The square root term is caused by TCP reacting to losses by halving the congestion window, coupled with the fact that the number of lost packets will itself be proportional to p times the congestion window. One way to achieve a degree of QoS differentiation is to allow a single connection to behave *as if* it were N simultaneous TCP connections, with N times this throughput [1]. However, the dependence on the square root is not easy to defend—a more natural dependence would be proportionality to $1/p$. For TCP with ECN to achieve this would require congestion signals for a connection to be independent of its load, which conflicts with the purpose of ECN. We present a different way of achieving the goal of a $1/p$ throughput, by requiring only a small reduction of rate in reaction to a congestion signal, rather than requiring a halving of the current window. In fact, these statement holds true when p is the probability of receiving a congestion signal, rather than a lost packet probability. Indeed, there several reasons why it is much better to use a different signal to packet loss as a feedback message, such as a signal based on a congestion price.

2.2 Congestion Pricing

We would argue that providing different QoS to users implies some form of differential pricing. Once this is accepted, it is natural to use congestion signals that reflect in some sense the cost of congestion [8, 5, 4, 6, 9]. It then becomes possible to give the users a considerable degree of freedom, giving control to the end-systems, allowing them to determine their own QoS, knowing that they will be charged accordingly. In economic terms, congestion is an "externality", and making users pay for the effect they have on others "internalises" this externality, encouraging co-operation between users and the network.

Pricing is an emotive issue, and one we return to later in some detail, but it is worth mentioning here that prices or charges might be unrelated to money. For example, in an Intranet different users might be given the ability to generate "credits" at different rates—a model of differential service reminiscent of time-sharing and batch allowances commonplace in Mainframe computing a couple of decades ago!

We now briefly describe the framework, based on Gibbens and Kelly [5]. A rigorous mathematical treatment is given in the Appendix.

2.2.1 Theoretical basis

The basic framework has two key elements: the network supplies feedback to the users, which reflects the cost of congestion (more precisely, the shadow price of congestion); the users incur some cost associated with their actions, but otherwise can react as they please.

Assume the cost of congestion is the number of lost packets. Then, all packets contributing to a loss should be "marked" (think of an ECN mark) to reflect the shadow price of congestion. For an isolated resource, such as a buffer in a router, we would thus mark all those packets that arrive between the start of a busy period and the last lost packet in that busy period.

In a more abstract framework, if there is a load y on a resource, incurring cost at rate $C(y)$, the congestion price or shadow price is the derivative

$$p\left(y\right) = \frac{d}{dy} C\left(y\right) \quad . \tag{1}$$

If the load y is a random variable and the cost is taken to be the expected rate of loss, then the marking strategy in a single buffer as described above does return the appropriate derivative information, scaled by the load [5, 6].

Suppose a finite number of users R (indexed by r) use a finite set of resources J (indexed by j), where the 0–1 incidence matrix A_{jr} indicates whether user r uses resource j or not. Following Shenker [17], we characterise an elastic user's preference for amount of bandwidth x_r by a concave utility function $U_r(x)$. Then a Social Planner would seek to maximise the *total* net utility,

$$\text{Maximise} \quad \sum_r U_r\left(x_r\right) - \sum_j C_j\left(\sum_r A_{jr} x_r\right) \qquad \text{over} \quad x_r \geq 0; \quad r \in R. \tag{2}$$

This optimisation problem has the solution (for non-zero x_r)

$$U_r'\left(x_r\right) = \sum_{j \in r} p_j\left(y_j\right) \tag{3}$$

with a corresponding load on resource j given by

$$y_j = \sum_r A_{jr} x_r . \tag{4}$$

Hence at the (social) optimum the derivative of a user's utility function exactly matches the sum of the shadow prices of all resources along the user's route.

Usually the network has no knowledge about the users' utility functions. However, if a user is charged at a rate proportional to the amount of bandwidth x_r received, say at rate $t_r x_r$, users will seek to maximise *their* net return, that is, they will try to solve

$$\text{Maximise } U_r(x_r) - t_r x_r \qquad \text{over} \quad x_r \geq 0. \tag{5}$$

It is clear that if the prices are right, namely if

$$t_r = \sum_{j \in r} p_j(y_j) \tag{6}$$

and if each user acts to maximise his own net benefit, then they will at the same time evolve towards the system or social optimum. As the network load changes, the network will need to update its prices.

In summary, we use the feedback signal $x_r \sum_{j \in r} p_j(y_j)$, where feedback for a stream should be proportional to the stream and also proportional to the shadow prices (congestion costs) along the route.

It is interesting to note that for utility functions of the form

$$U_r(x_r) = w_r \log x_r \tag{7}$$

for some *Willingness-To-Pay* constant w_r, then at the optimum

$$w_r = x_r \sum_{j \in r} p_j(y_j) \quad . \tag{8}$$

The resulting allocation is proportionally fair [8], which is intimately related to a Nash arbitration scheme, well-known to economists, and the only scheme that satisfies certain fairness axioms [12].

In the Appendix we show how such an allocation can apply to arbitrary utility functions if w_r is treated as input to the user's control algorithm in place of true preferences.

2.3 An Example Strategy Compared to TCP

Consider the example of a user who wishes to control the rate at which he spends his "currency" w_r. A simple strategy would then be to implement the Willingness To-Pay algorithm

$$\frac{d}{dt} x_r(t) = \kappa_r \left(w_r(t) - x_r(t) \sum_{j \in r} p_j(y_j(t)) \right) \tag{9}$$

where κ_r is a constant that controls the rate of convergence, and the costs $p_j(y) = \frac{d}{dy} C_j(y)$ depend on the (time-varying) loads $y_j(t)$, by analogy with the static definition. The interpretation is that each resource j marks packets according to the price function $p_j(y_j(t))$ and marks streams in proportion to their contribution to the load. Users then react by increasing or decreasing their rates, according to whether this charge is smaller or larger than the amount they are prepared to pay.

Notice that there is no explicit dependence on RTTs in this algorithm. There can be an implicit dependence through κ_r, which might be related to the reciprocal of the RTT. Replacing w_r by w_r/RTT_r would make the steady state throughput dependent on (the reciprocal of) the RTT.

Suppose there is just a single resource, and a number of streams offered to it. For convenience, we assume that there is no buffering in the resource, which has capacity C. (The proof that this also addresses the more general case of a buffered resorce is shown in [6].) The feedback signal $p(y)$ is then the probability P_{sat} that the resource is overloaded.

The congestion avoidance phase of TCP can be expressed in a similar manner, with

$$\frac{d}{dt}x_r(t) = \kappa_r \left(w_r - P_{loss} x_r(t) \frac{x_r(t)}{2} \right) \quad . \tag{10}$$

The feedback signal now depends upon the loss probability P_{loss}, that is, the proportion of lost packets $\mathbb{E}[(Y-C)^+]/\mathbb{E}[Y]$.

Numerical examples of the two approaches given in [9] illustrate how the parameter w relates to how well the system is sized. The loss signal P_{loss} is typically at least an order of magnitude smaller than P_{sat}, justifying earlier remarks advocating sending many more signals back to the sources than are implied by packet loss with current TCP. Conversely, the users should react much less aggressively to a feedback signal than in TCP.

2.4 Conveying Information

The congestion pricing marking policy fits easily with the current ECN protocol proposals, although further work is needed to understand whether allowing more than one bit of information would be beneficial. It is not clear whether restricting the signal to a binary indication would have much of an impact from a control viewpoint. In a properly run network, where capacity is well matched to both demand and charges, congestion should be relatively rare, so that the probability of incurring marks at a number of resources rather than one will be small. But then there are examples in current standards where restricting a flag to a single bit has caused enormous problems, particularly with QoS.

2.4.1 Pricing and charging

It has to be emphasised that congestion prices even if translated directly into charges would only be part of the charges. In economic terms they make sense if they reflect marginal expansion costs, whereas other costs such as fixed infrastructure costs are usually recovered in other ways, such as by subscription or flat fee. However if the congestion prices are immaterial to the user (purely fictitious) then there is no incentive for the user to react.

2.4.2 Legacy issues and TCP

A class of flow control schemes can be constructed that has most of the features of TCP (for example sliding windows, RTT estimation, time-outs etc) but reacting to a Congestion Price ECN signal rather than packet loss or ECN based on a EPD or RED type strategy. The appropriate reaction can then be a small reduction rather than a halving of the current window. Standard TCP with and without SACK and ECN options can coexist along side such schemes although non-ECN TCP streams will typically incur higher charges (for a given throughput) than flows (including TCP) which react to the ECN signals.

3 Aggregates

Single channel adaptation algorithms based on reacting to signals from Congestion Price ECN have been discussed as background. Our observation is that once such a scheme is implemented, we can start to consider new options for how we might use the scheme for aggregates of traffic flows.

3.1 The Network View

We first consider an aggregate flow of packets through an IP cloud between what the IP layer considers to be two end systems. That is, a router sees a sequence of packets between an IP source/destination pair which represent multiple, possibly related, channels at the transport or even network layer. For example: web traffic composed of multiple concurrent HTTP requests; a multimedia application composed of a TCP control channel and audio/video data transfer using RTP/UDP; or the sequence of packets between the end points of a VPN tunnel.

Previously, and perhaps particularly within the ATM community, it has been suggested that the network might be made aware of such inter-channel relationships and make discard decisions based on some user signalled preference. There are three difficult issues with this approach: complexity, flexibility and feasibility. The complexity is in the implications for the network in both the control and data planes. Without resorting to active networks, the flexibility would be restricted to some predefined set of functions. The feasibility issue arises from security, since if the encryption of a VPN tunnel is doing its job, the network is not able to identify the individual streams.

We instead place the onus on the IP end-system. Consider the simple example of a set of (N) unicast channels composing an aggregate. We could treat the utility functions for these channels as independent, and react to ECN indications on a per channel basis; or we could consider the N-dimensional surface generated by the functions and react to ECN signals by traversing the surface. The arguments proceeding for single channel utility curves and Congestion Price ECN, can continue to be applied - that provided the "Utility Surface" is concave, we can still proceed to find the user and social optimum if the price is right (see the Appendix).

The main implication for the network is that the IP layer at the destination host (or VPN terminus) can return ECN events to a source in any IP packet that is going back that way, *independent* of the higher layer channel on which it arrived. It is the job of the source host IP layer to ensure that some higher layer channel reacts – a simple, effective and draconian strategy would be to implement a per destination rate control at the IP layer – indeed perhaps it becomes the default "backstop" policy. In any case the ECN function becomes one that can be solely implemented at the network layer.

3.2 The Host View

Viewing the channels between a source/destination pair as an aggregate, we can consider the wider issued of maximizing the utility of the aggregate, rather than simply maximizing the utility of each individual stream.

Using the earlier example of a multimedia application with a triplet of channels for control, audio and video, the application utility function might be maximised by reacting to a congestion event on the audio channel by reducing the load offered by the video channel.

Likewise for the VPN tunnel; the simplest solution to ECN experienced on a VPN link would be to move the congestion notification indication from the header of the encapsulating packet to that of the encapsulated packet just after decryption. This closely mirrors the behaviour we would expect today where packet loss on the VPN link is (often) packet loss to the encapsulated stream. However, as with our application example above, the option of passing the congestion event to a different channel in the encapsulated aggregate might be considered. That is, from the congestion point of view, the tunnel access point could implement a reaction policy that is dictated by some multidimensional utility function as long as the concavity constraint is maintained.

As this point we can contemplate policies based on any number of input parameters: services, port numbers, protocol types, addresses, user credentials, price constraints, historical data, time, date, etc, etc. For example, it is clear that in an academic environment, the natural candidate would be a user-based policy in which all congestion experienced by faculty should be handed off to students.

3.3 The Hard case

Could the approach be extended to the harder case of trading congestion for streams between sets of source / destination addresses? We would achieve new flexibility if we could. An operating system might implement a utility function across several applications where the peer hosts for the various applications are different, the policy based on user preferences and system policy. The richness of policies possible for the VPN tunnel then become possible for a firewall, proxy or access router.

What we require is to classify source/destination addresses into sets that share congestion points. Such information might be gathered by access to routing information or more likely by statistical observation.

We reserve further consideration of the hard case for future work!

4 Observations

4.1 Charging

This work is presented in the language of economics. In reality the mechanism used for implementation of the charging is dependent on the degree of trust and relationship of the network users to the network operator. Furthermore, while all routers (that might experience congestion) need to implement the ECN mechanism, it is not necessary to supply each with a mechanism for billing customers.

4.1.1 Intranet Charging

An obvious solution, and one that follows the same philosophy as TCP, is to trust the end systems. An end system is provided with a rate at which it may generate tokens and the right to hold a small stock of tokens which it can match against incoming congestion events – in effect a "leaky bucket" for congestion events. Different classes of users (services or people) could be given different token rates and bucket depths.

4.1.2 Peer Charging

ISPs are starting transfer charging, a practice common place amongst Telcos. It is interesting to note that capacity charging, which relates to congestion charging, has been mooted by a certain European regulator as a means for interconnect charging. Congestion pricing on aggregates would be one way to account for the cost of traffic traversing different domains / AS.

4.1.3 End-user charging

A full implementation, where an end-user is charged, could reflect the Intranet solution where the end system is charged by the hosting ISP. Encryption suggests that Receiver Pays is a good model, which circumvents the problem of being unable to identify the user from outside. Transfer charging mechanisms are then needed to ensure that the Sender or Receiver is ultimately charged.

4.2 Guaranteed Services

How do guaranteed services fit in? One approach places the onus on brokers or agents who effectively manage the risk; buying a service from the network and incurring congestion prices whilst offering guaranteed bandwidth to users, and with attempt blocking to assure quality. Gibbens and Kelly [4] have given some numerical results, which shows how this could work for a service like telephony, with little additional inefficiency.

One type of guaranteed service might be the traditional "fixed price by access class" service in which the ISP manages the risk associated with congestion pricing to provide end-users with the simple model they have come to know, if not love.

5 Experimental Framework

5.1 The Simulator

Prior to widespread deployment, modifications and improvements to Internet protocols have often been evaluated by simulation using the VINT/LBL network simulator software [13]. This simulator allows arbitrary network topologies to be specified and provides a run-time environment where protocols can be compiled and executed as part of the simulator. Many standard protocols (e.g. most TCP variants), have been re-implemented within the `ns` framework, together with traffic generators for common workloads and a suite of analysis tools.

As has previously been discussed, it is hoped that a network supporting congestion-based charging mechanisms will enable a wide spectrum of protocols with widely divergent goals. History has repeatedly shown that it is impossible to anticipate all new applications and their demands upon the network infrastructure. How, then, should one evaluate such networks and protocols?

Conceptually, the problem may be viewed as a multi-user game [9]. From the point of view of the network provider, it is important to demonstrate the robustness and stability of the charging mechanisms in face of arbitrarily ingenious attempts to "beat the system". From a user's point of view, it should be possible to use any means at their disposal to achieve their networking aims at minimum cost. One interesting approach to evaluation of these charging ideas, therefore, is to implement and deploy such a multi-user game on the Internet, and challenge would-be protocol designers to demonstrate their cunning.

The Statistical Laboratory, Cambridge has implemented a network simulator as a Java Applet [19] which allows strategies (also written in Java) to be dynamically linked with the simulator and executed simultaneously. However, this simulator has limited richness of topology and is restricted to a slotted model of time. The necessity for all players of the game to submit their code to be linked with the simulator also has a number of practical disadvantages: the code of the strategies may simply not compile cleanly or execute properly, and players must effectively divulge their algorithms in order to compete.

The simulator we have implemented is a conventional continuous-time discrete event simulator, supporting complex network topologies and behaviours along the lines of `ns`, but which also allows a number of remote clients to participate in the simulation and compete to achieve a specified objective. Clients (players) connect to the simulator using a TCP stream and inject packets into the simulated network using a simple ASCII protocol. Charging indications are propagated through the simulated network (using an ECN-like mechanism) and may be returned to the clients along the same TCP connection.

The above architecture allows network strategies to be implemented using whatever programming language, external knowledge and computing resources are best suited to the job (anything from a user typing at a telnet prompt to a supercomputer running mathematical modelling software). It also does not preclude syndicates of players collaborating to gain an 'unfair' advantage.

At a practical level, the simulator and players execute an `NTP`-like algorithm in an attempt to maintain an approximately uniform and monotonic view of the current simulation time. The simulator periodically adjusts the rate of passage of simulation time with respect to wall-clock time such that most players and the simulator itself can 'keep up'. As with most multi-player network games, it has been carefully designed such that it is not forced to run in lock-step with the slowest player.

The simulator itself is written in C++ and allows the simulated network and various game objectives to be specified in a textual description file. The behaviour of nodes and links in the network may be selected from a precompiled set of queuing, scheduling, routing and charging classes[2]. For convenience, a simple C++ client run-time has also been implemented which hides the subtleties of the simulation time rate adjustment mechanisms from the protocol designer. This allows implementation of protocols in a more familiar environment providing the equivalent of timers, receive interrupts and a packet transmit method. A similar run-time is planned for Java.

Events from each node of the simulated network are logged to the file system for post-processing, in a format that allows us to leverage many of the VINT/LBL analysis tools, e.g. `nam`.

Several protocols have been implemented so far including CBR, random background traffic patterns and TCP variants which include the old and the new: go-back-N, SACK, ECN, and our own Willingness-To-Pay. Preliminary results from these protocols are presented below.

5.2 Implementating a Willingness to Pay Strategy

A Willingness-To-Pay strategy can be described by the differential equation (9), which summarises how the protocol should react to network feedback by adapting its rate $x(t)$. However, there are the usual practical issues that need to be resolved before it can be tested or implemented in a real network or in the network simulator.

For example, when should updating occur? Two natural solutions are:

1. Updating at regular (small) time intervals. The interval could be a design option (e.g. every 100 or 500 ms) or dynamically adjusted as part of the strategy. This may be natural for a streaming protocol not requiring retransmissions.

2. Updating on receipt of an ACK. This is reminiscent of TCP, and indeed it is in principle straightforward to adapt the TCP sliding window schemes to such a strategy. Measures have to be taken to deal with loss of packets, which (beside requiring retransmission) could result in a feedback signal being lost for a period of time. Therefore, RTT measurements and an RTO timer (similar to those currently used in TCP) should preferably also be included in the protocol.

Another question concerns the choice of the variables κ and $x(0)$, the initial rate. Too large a κ can cause instability, whereas too small a κ results in an unresponsive source. A "good" choice will be a function of the network topology (e.g. the RTT). Enhancing the strategy to allow for an adaptive κ can allow faster convergence to the equilibrium operating point when network conditions change.

Choosing an initial rate $x(0)$ is only a problem for short-lived connections, or those which have too small a value of κ. A high value (relative to the overall network load) could result in a high cost to the user, though the starting condition should make little difference provided reasonable transfer times. A risk-averse user could start with a very small initial load, while a risk-prone user could start with w_r.

Some randomness, whether in starting values or updating intervals, has a beneficial effect in avoiding periodic behaviour.

5.3 Marking Strategies

In section 2.2.1 we outlined a strategy based on marking all packets arriving between the start of a busy period and the last lost packet in that period. However, this cannot be implemented directly; when

[2]Additional classes may easily be added at compile time.

a busy period starts we do not know whether the period will lead to a packet loss, in which case we should mark packets immediately, or whether the queue will empty first with no lost packets, meaning we should mark nothing. We will not know which of these events is happening until after some of these packets have left the queue, by which time it is too late to mark them. How to deal with this is an open issue, although on a statistical basis, it is sufficient merely to mark the right number of packets.

One approach, which has worked well in simulations, is to keep a running counter of the number of packets that should be marked, and as soon as a loss event occurs start marking from there until the right number have been marked. An alternative is to set a threshold, much like RED, but probabilistically mark packets in the queue once this threshold is exceeded. Yet another approach is to run a virtual queue with reduced rate and capacity and mark according to this.

We consider it a merit that the detailed choice of strategy is left for the router designer.

6 Results

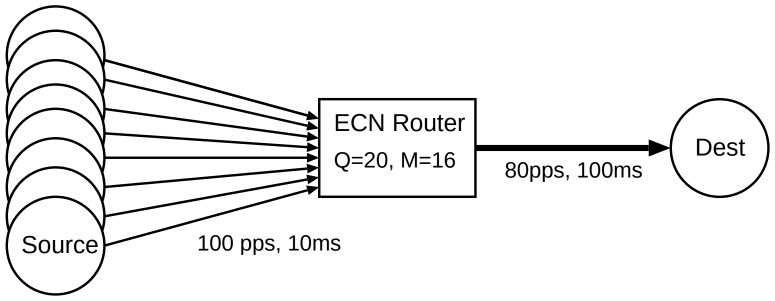


Figure 1: Network Configuration

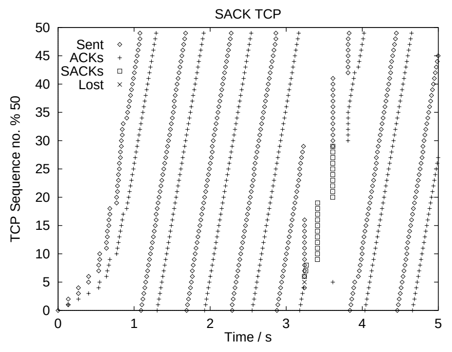
6.1 Experimental Setup

All of the experiments described below were performed using the simulated network topology shown in figure 1. The router of interest has 10 input links of 100 1.5k packets per second and 10ms latency, and a single bottleneck output link of rate 80pps and latency 100ms. All streams are terminated at the end of this link, since additional hops after the bottleneck would merely increase the RTT slightly and are thus of no significant interest.

The output queue for the bottleneck link has capacity for up to 20 packets, and is configured to mark *all* packets in the queue whenever the queue length exceeds a threshold of 80%. Packets from non-ECN flows are discarded on arrival if the queue is above this threshold.

6.2 Validation Using Standard Protocols

Two initial validation experiments were performed using a conventional TCP supporting selective acknowlegements (SACK), both with and without ECN. A single source attempts to transfer an infinite amount of data to a fast sink for 120 seconds. The first 5 seconds of the TCP sequence number plots are shown in figure 2. The proportion of packets reaching the destination, goodput, sustained bandwidth and cost incurred are also tabulated. From this initial work, the behaviour of TCP in our distributed simulation environment exhibits realistic qualitative behaviour – further work and control studies would be required before we would make any significant claims on the accuracy of the absolute quantitative results.



8620	TXed	8768
98.5%	RXed	99.7%
95.3%	Good	99.7%
68.5	B/W	72.9
1241	Cost	972

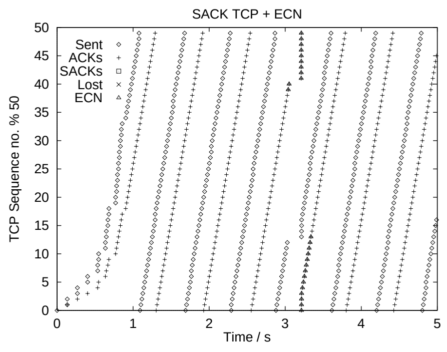


Figure 2: Behaviour of a single TCP (left) and TCP+ECN (right)

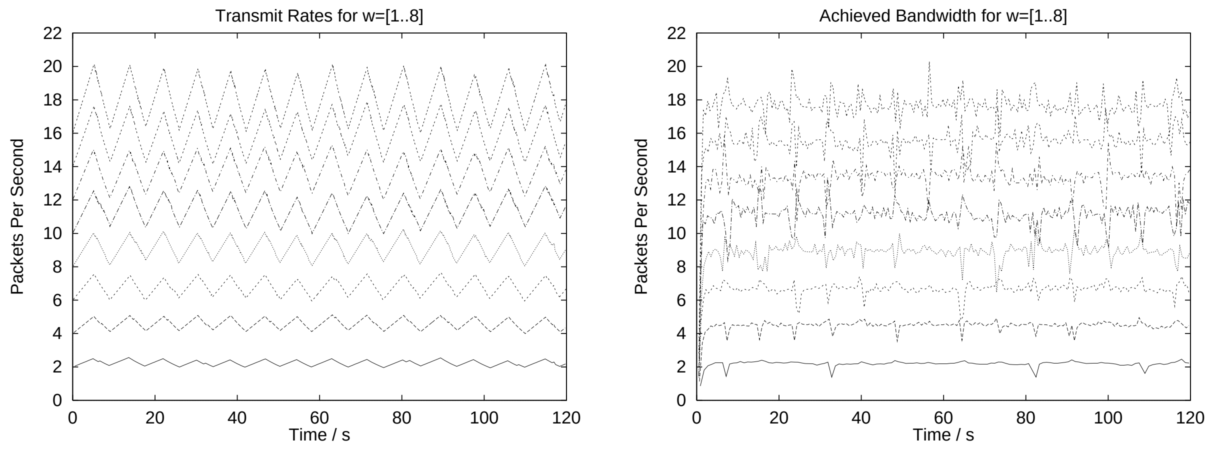


Figure 3: Transmit Behaviour of Various WTP streams.

6.3 Willingness to Pay

For the next experiments, a conventional TCP was modified to ignore the congestion window and instead transmit at a *rate* determined according to equation (9). This rate was updated for every ACK received and on every retransmit timeout. The retransmit and acknowledgement behaviour were left unmodified (i.e. selective acknowledgements, fast retransmits, etc.).

Eight such sources were configured with different values of the parameter w (i.e. "Willingness-To-Pay") and $\kappa = 0.1$. Figure 3 shows the nominal transmit rate of each stream and the achieved share of the bottleneck link bandwidth. Note that the link share is clearly proportional to w.

6.4 Comparison with TCP

The previous experiment was repeated with the addition of a single conventional TCP stream in order to compare the relative aggressiveness of WTP streams. Once again, both ECN and non-ECN TCP variants were used. The sequence number plots in figure 4 show TCP bracketed by two WTP streams with $w = 1$ and $w = 8$. It can clearly be seen that, by varying the parameter w, the WTP stream can be made either more or less aggressive than conventional TCP.

The bottleneck link bandwidths are plotted in figure 5, together with tables summarising the goodput, bandwidth and cost for each stream. As expected, the achieved bandwidth for the WTP streams is proportional to the incurred cost, and the cost incurred is $w * 120$—i.e. a rate of w marks per second.

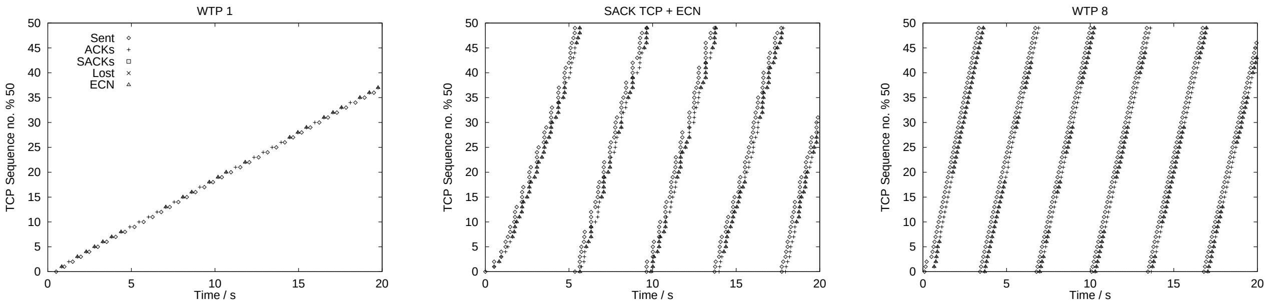
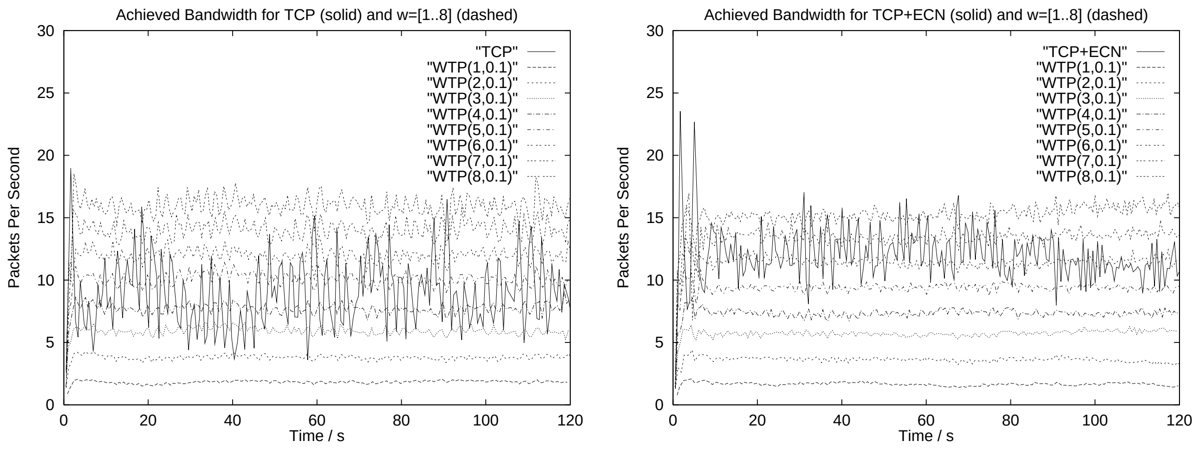


Figure 4: Sequence no. plots of WTP(1), TCP+ECN and WTP(8)



Flow	TXed	RXed	Good	BW	Cost
TCP	1325	78.4%	74.5%	8.2	837
WTP 1	220	100.0%	100.0%	1.8	120
WTP 2	458	99.7%	99.7%	3.8	240
WTP 3	714	99.7%	99.7%	5.9	359
WTP 4	929	99.7%	99.7%	7.7	479
WTP 5	1204	99.7%	99.7%	10.0	597
WTP 6	1432	99.7%	99.5%	11.9	715
WTP 7	1678	99.7%	99.7%	14.0	835
WTP 8	1910	99.7%	99.7%	15.9	952

Flow	TXed	RXed	Good	BW	Cost
ECN	1449	99.3%	99.3%	12.0	845
WTP 1	199	100.0%	100.0%	1.7	125
WTP 2	435	99.5%	99.3%	3.6	250
WTP 3	689	99.7%	99.7%	5.7	364
WTP 4	882	99.6%	99.4%	7.3	488
WTP 5	1130	99.7%	99.7%	9.4	610
WTP 6	1377	99.5%	99.4%	11.4	731
WTP 7	1618	99.3%	97.8%	13.2	853
WTP 8	1824	99.5%	97.%4	14.8	969

Figure 5: WTP(1..8) competes against TCP (left) and TCP+ECN (right).

7 Concluding Remarks

In this paper we have proposed a mechanism for sharing bandwidth between competing streams of elastic traffic based on a simple feedback signal. These congestion signals can be simply implemented in a router, and are designed to encourage the end-systems to co-operate. We have outlined a proposed architecture, and touched on how to relate these pricing signals to real charges. Our aim is to test the robustness of this scheme via a distributed network simulator, and to use this in the context of a distributed network game to gauge the effectiveness of different user behaviour.

The results so far are encouraging: the simulator has been validated in the sense that standard TCP variants behave as predicted, and we are starting to obtain insights into new strategies. For example, simple strategies, such as Willingness to-Pay schemes do indeed give higher bandwidth to those who pay more, and moreover can co-exist with TCP. It is interesting to note that in our simulations standard TCP incurs a slightly higher charge for a given throughput; we conjecture that this effect would be more marked when there are a large number of TCP streams.

We also discussed how marks might be traded off between applications. This is an area we plan to explore by simulation, alongside more comprehensive scenarios. The software for the simulator will shortly be released, and used as the basis for a sequence of distributed experiments conducted as a network game.

Acknowledgement

It is a pleasure to acknowledge helpful discussions with Frank Kelly and Richard Gibbens. We are indebted to our intern Dave Stewart for his coding legacy.

Appendix

Outline proofs are given below. To fully complete the proof requires use of appropriate Kuhn-Tucker conditions in cases when the optimal point is on the boundary rather than at an interior point.

A.1 Convergence

Suppose that user r uses the updating equation

$$\frac{d}{dt}x_r(t) = \kappa_r \left(x_r(t)\, U_r'(x_r(t)) - x_r(t) \sum_{j \in r} p_j(y_j(t)) \right) \tag{11}$$

where $y_j(t) = \sum_{j \in r} x_r(t)$ then, provided the Utility function is strictly concave and the Cost function strictly convex, defined over $x \geq 0$, then the equilibrium is unique, given by the solution to (3) and all trajectories converge to it.

Proof. Define the function

$$\Psi(x) = \sum_r U_r(x_r) - \sum_j C_j \left(\sum_{j \in s} x_s(t) \right) \tag{12}$$

then under the assumptions on the functions U and C, Ψ is a Lyapunov function for the differential equation (11), in which case all trajectories converge to the vector x maximising this function (which will be unique for strictly concave U and strictly convex cost function, and interior to the region), see [18]. Now

$$\frac{\partial}{\partial x_r}\Psi(x) = U_r'(x_r(t)) - \sum_{j \in r} p_j(y_j(t)) \tag{13}$$

and setting these derivatives to zero gives the maximum. Differentiating with respect to t gives

$$\begin{aligned} \frac{d}{dt}\Psi(x(t)) &= \sum_r \frac{\partial}{\partial x_r}\Psi(x) \cdot \frac{d}{dt}x_r(t) && (14) \\ &= \sum_r \kappa_r x_r(t) \left(U_r'(x_r(t)) - \sum_{j \in r} p_j(y_j(t)) \right)^2 && (15) \end{aligned}$$

which is strictly positive except when x is the equilibrium value, hence Ψ is a Lyapunov function, and the result proved. This proof is adapted from [8], where discussions of speed of convergence can be found. $\square$

A.2 Mandated Controls

Note that the remarks hold true if a user is mandated to use a certain updating function, subject to certain conditions. This might be the case is a user or end-system has a restricted choice of flow-control options. For example, if the derivative of the updating function U is prescribed in equation 11, then updates are of the form

$$\frac{d}{dt}x_r(t) = \kappa_r \left(x_r(t) F_r'(x_r(t)) - x_r(t) \sum_{j \in r} p_j(y_j(t)) \right) \tag{16}$$

for some fixed concave functions F_r. Now suppose the derivative of F_r has a continuous inverse, and suppose there are some proxy variables w_r with associated functions g_r satisfying

$$\begin{aligned} F_r'(x_r) &= p_r \\ x_r &= w_r g_r(p_r) \end{aligned} \tag{17}$$

then there are vectors $w = (w_r, r \in R)$, $x = (x_r, r \in R)$, $p = (p_r, r \in R)$ such that if w_r solves the User problem, which maximises

$$U_r(w_r g_r(p_r)) - p_r w_r g_r(p_r) \quad \text{over } w_r \geq 0 \tag{18}$$

then x which evolves according the equation (16) also solves the system optimum, given as the solution which maximises equation (2) (or (12)). In this case the p_r are just the sum of the shadow prices along route r, that is $p_r = \sum_{j \in r} p_j(y_j)$.

To sketch a proof, in the case when the optimum lies in the interior of the region, equation (16) converges to the unique equilibrium satisfying

$$F_r'(x_r) = \sum_{j \in r} p_j(y_j); \tag{19}$$

but at the user optimum

$$U_r'(x_r) = p_r \tag{20}$$

and using (17), this is the solution to the System optimum (3).

As simple example, suppose the user has to use the willingness-to-pay algorithm described in Section 2.3, with $w_r \log x_r = F_r$, the user optimisation corresponds to a user adapting the willingness to pay parameter w_r over time according to

$$w_r = x_r U_r'(x_r) \tag{21}$$

or equivalently who is seeking to solve

$$\text{Maximise} \quad U_r\left(\frac{w_r}{p_r}\right) - w_r \quad \text{over} \quad w_r \geq 0. \tag{22}$$

A.3 Correlated feedback

Consider the case where a user has a number of streams which are related, which is equivalent to the case where "users" are not independent. This could model a multi-media connection. Now a user r may have a utility function $U_r\left(\vec{x}_r\right)$ that is a function of several variables. Provided this is a concave function, the user optimum will still converge to the system optimum, provided that downhill steps are taken, but now the user can choose to use the extra information to alter the adaptation. For instance steepest descent or gradient projection methods could be used to change the way the user reacts, which equate to trading off marks between streams, and altering rates of convergence. By way of illustration, consider two related streams, and for simplicity assume they use the same set of resources. The component feedback signals are then

$$x_i \frac{\partial U_r\left(x_i\right)}{\partial x_i} - x_i \sum_{j \in r} p_j\left(y_j\right) \quad i = 1, 2 \tag{23}$$

which can be scaled in any way to update the vector (x_1, x_2). Notice that this is equivalent to changing κ_i . Marks can also be traded amongst streams, using correlated information to increase reaction speed. For example using a proportion of stream 2 marks to reduce stream 1, provided this stops when a boundary or optimum is reached, namely (if the optimum is at an interior point) when

$$\frac{\partial U_r\left(x_i\right)}{\partial x_i} = \sum_{j \in r} p_j\left(y_j\right) \quad . \tag{24}$$

References

[1] Jon Crowcroft and Phillipe Oechslin. Differentiated end-to-end services using a weighted proportional fair share TCP. *Computer Communication Review*, 28(3):53–69, 1998.

[2] S. Floyd. TCP and explicit congestion notification. *ACM Computer Communication review*, 24(5):10–23, 1994.

[3] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993. http://www-nrg.ee.lbl.gov/floyd/red.html.

[4] R.J. Gibbens and F.P. Kelly. Distributed connection acceptance control for a connectionless network. In P.B. Key and D.G. Smith, editors, *Teletraffic Engineering in a Competitive World, Proceedings ITC16.* Elsevier, June 1999.

[5] R.J. Gibbens and F.P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 1999. http://www.statslab.cam.ac.uk/~frank/PAPERS/evol.html.

[6] R.J Gibbens and P.B. Key. A note on resource pricing and congestion control for networks with delay and loss. preprint, 1999.

[7] R. Jain and K.K. Ramakrishnan. Congestion avoidance in computer networks with a connectionless network layer: Concepts, goals and methodology. In *Proceedings IEEE Comp. Networking Symposium*, pages 134–143, April 1988.

[8] F.P. Kelly, A.K. Maulloo, and D.K.H Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[9] P. B. Key and D. R. McAuley. Differential QoS and pricing in networks: where flow control meets game theory. *IEE Proceedings Software*, 146(2), March 1999.

[10] J.K. MacKie-Mason and H.R. Varian. Pricing the Internet. In B. Kahin and J. Keller, editors, *Public Access to the Internet.* Prentice-Hall, New Jersey, 1994.

[11] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review*, 27(3), 1997.

[12] R. Mazumbdar, L.G. Mason, and C. Douglieris. Fairness in network optimal control: optimality of product forms. *IEEE Trans. Communications*, 39:775–782, 1991.

[13] NS (Network Simulator). http://www-mast.cs.berkeley.edu/ns/, 1995.

[14] T. Ott, J. Kemperman, and M. Mathis. The stationary behavior of ideal TCP congestion avoidance. Preprint, 1997.

[15] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Computer Communication Review, Proceedings Sigcomm 98*, volume 28, pages 303–315, 1998.

[16] K. Ramakrishnan and S. Floyd. A proposal to add Explicit Congestion Notificaiton (ECN) to IP. Request For Comments RFC 2481, Internet Engineering Task Force, January 1999. ftp://ftp.isi.edu/in-notes/rfc2481.txt.

[17] S. Shenker. Fundamental design issues for the future Internet. *IEEE J. Selected Area Communications*, 13:1176–1188, 1995.

[18] J.L. Williams. *Stability Theory of Dynamical Systems.* Nelson, 1970.

[19] Damon Wischik. Pricing the internet. http://www.statslab.cam.ac.uk/~djw1005/Stats/Compete/.