Policing Congestion Response in an Internetwork using Re-feedback

Bob Briscoe
BT Research & UCL
bob.briscoe@bt.com

Arnaud Jacquet
BT Research
arnaud.jacquet@bt.com

Carla Di Cairano-Gilfedder
BT Research
carla.dicairano-gilfedder@bt.com

Alessandro Salvatori
Eurécom & BT Research
sandr8@gmail.com

Andrea Soppera
BT Research
andrea.2.soppera@bt.com

Martin Koyabe
BT Research
martin.koyabe@bt.com

ABSTRACT

This paper introduces a novel feedback arrangement, termed re-feedback. It ensures metrics in data headers such as time to live and congestion notification will arrive at each relay carrying a truthful prediction of the remainder of their path. We propose mechanisms at the network edge that ensure the dominant selfish strategy of both network domains and end-points will be to set these headers honestly and to respond correctly to path congestion and delay, despite conflicting interests. Although these mechanisms influence incentives, they don't involve tampering with end-user pricing. We describe a TCP rate policer as a specific example of this new capability. We show it can be generalised to police various qualities of service. We also sketch how a limited form of re-feedback could be deployed incrementally around unmodified routers without changing IP.

Categories and Subject Descriptors

C.2.0 [**Computer-communication networks**]: Security and protection; C.2.1 [**Computer-communication networks**]: Network Architecture and Design

General Terms

Economics, Security

Keywords

Policing, congestion, QoS, characterisation, incentives

1. INTRODUCTION

The current Internet architecture trusts hosts to respond voluntarily to congestion; a feature commonly put down to the environment of mutual trust in which these algorithms emerged. Limited evidence shows that the large majority of end-points on the Internet comply with a TCP-friendly response to congestion. But if they didn't, it would be hard to force them to, given path congestion is only known at the last egress of an internetwork, but policing is most useful at the first ingress.

Without knowing what makes the current co-operative consensus stable, we may unwittingly destabilise it. At the most alarmist, if this were to lead to congestion collapse [7] there would be no obvious way back. But even now, applications that need to be unresponsive to congestion can effectively steal whatever share of bottleneck resources they want from responsive flows. Whether or not such free-riding is common, inability to prevent it increases the risk of poor returns, leading to under-investment in capacity.

In 2000, these capacity allocation and accountability problems helped to motivate a overhaul of the Internet architecture [1], but they remain unresolved. We believe their solution lies in a realignment of the feedback architecture.

Changing the Internet's feedback architecture seems to imply considerable upheaval. But, perhaps surprisingly, we believe a limited form of the new arrangement could be deployed incrementally at the transport layer, around unmodified routers using the existing fields in IP (v4 or v6). Protocol engineering isn't the focus of the present paper—an idealised numeric scheme is all that is necessary to explain the concepts. However, to satisfy doubters, we sketch our engineering and deployment ideas at the end.

Conceptually, the solution could hardly be simpler. We propose collecting path information in packet header fields as data traverses a path, just as can already be done with time to live (TTL) or congestion notification (ECN [19]). But previously, as each node added characterisation of its local hop, the header values accumulated *upstream* path knowledge. By a simple realignment, we arrange each field to characterise the remaining *downstream* path. We aim to reach a target for the metric at the destination, rather than aligning the datum at the source. For example, TTL currently always starts at the datum 255. Instead we propose it should arrive at the destination set to an agreed datum (say 16). To achieve this, each receiver will need to occasionally feed back the TTL values arriving in packets, so the sender can adjust the next attempt in order to continue to hit 16. §2 expands on this basic explanation with more precision.

We term this pattern 're-feedback', short for either receiver-aligned or re-inserted feedback, although it is actually similar to the ordinary feedback found in other disciplines (electronics, hydraulics, etc.). Once re-feedback is in place, each packet arrives at each network element carrying a view of its own downstream path, albeit a round trip ago. So full path congestion becomes visible at the first ingress, where a rate policer is most useful.

But we still don't seem to have solved the problem. It seems naïve to police traffic by trusting fields that depend on the honesty of both the sender and receiver—those with most to gain from lying. However, in §3 we explain why re-aligning feedback allows us to arrange for honesty to be everyone's dominant strategy—not only end-users, but also networks. Building on the resulting trustworthiness of path metrics, we describe how to build a rate equation policer, using TCP as a concrete example. We generalise to any rate equation, in particular Kelly's [14], showing that we can synthesise the same effect as quality of service mechanisms, but only using an ingress policer. We also describe a passive policer for inter-domain boundaries.

In §4 we sketch our incremental deployment ideas. Then we end the body of the paper (§5) with the results of simulations conducted to test whether the incentive mechanism really is responsive enough to ensure truthful congestion reporting. We wrap up with related work and conclusions.

2. RE-FEEDBACK

Characterising paths through networks requires more than one metric. We have chosen to explain how re-feedback works using two: congestion and delay (that is, unloaded delay not congestion delay). Re-feedback of just these two metrics helps solve a surprisingly large set of networking problems. But additional metrics might be useful in practice, e.g. hop count, unloaded loss rate etc. Delay re-feedback is a useful starting point because it is trivially simple to explain. Then we use congestion to highlight the similarities and differences that are encountered between metrics.

A pre-requisite for re-feedback is the *explicit* declaration of path metrics and their maintenance along the path. Setting aside protocol details for now, it will suffice to consider a multi-bit field for delay and another for congestion carried in future network layer packet headers[1]. Also equivalent fields will be necessary in the end-to-end back-channel from receiver to sender—sent frequently enough to control the most volatile metric (congestion). For instance, in future TCP acks (or RTCP receiver reports, etc.)

Network layer headers will also need a 'certain' flag, which the sender should clear at the start of a flow, when no feedback is yet available. Metric(s) carried in uncertain packets should not contribute to any bulk averaging at network equipment (e.g. see §3.2.1), but the flag is not intended to affect forwarding of the packet itself.

Fig 1 introduces our notation. Each path across the network consists of a sequence of resources, i_r; $0 \leq i_r < n_r$ indexed in the context of each path r from the sender S with resource $i_r = 0$ to resource $i_r = (n_r - 1)$ just before the receiver R. Whenever a single path context makes it obvious, we will drop the suffix r.

[1]We believe it is possible to apply re-feedback in a separate control plane, or even where control information is analogue, but for clarity we stick to one IP-based scenario.

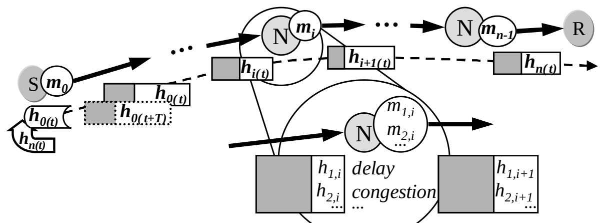

Figure 1: Notation for path characterisation metrics m and headers h.

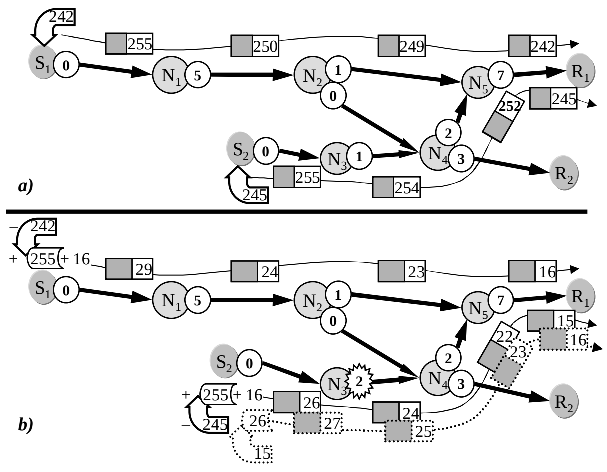

Figure 2: Network flows carrying unloaded delay in packet headers. a) With classic feedback, sources initialise headers to 255. b) With re-feedback over the same network, sources set headers so as to reach 16 at the destination.

The unloaded delay header, h_1, is carried in packets from resource to resource. Each relay N characterises its local resource's contribution to the delay—perhaps by echo tests with the downstream neighbour. It contributes to the whole path delay by combining its local contribution $m_{1,i}$ with the incoming header value, $h_{1,i}$, and forwarding the updated result, $h_{1,i+1}$ (Fig 1). The choice of combining function depends on the metric in question. As unloaded delay is additive, subtraction is an appropriate combining function (like TTL processing), $h_{1,i+1} = h_{1,i} - m_{1,i}$.

Other packet header fields will require combining functions appropriate to the metrics they represent. The inset in Fig 1 shows packets carrying header fields for both delay and congestion being combined with the local metrics for each, as parallel, independent operations. Where the context is obvious, we drop the suffix that distinguishes between delay and congestion.

If we introduce feedback of unloaded delay, the receiver will report the header values it receives back to the sender. With classic feedback, the sender always initialises the unloaded delay header to a well-known value, say $h_0 = 255$, as shown in Fig 2a). The header will arrive at node j with a value accumulated over all the upstream resources $h_j = h_0 - \sum_{i=0}^{j-1} m_i$. We call the composition of all the local metrics m_i experienced by a packet the **path metric**

So, with classic feedback for delay, the path metric up-

	unloaded delay	congestion	Eqn
combining function at resource i, $h_{i+1} = g(h_i, m_i)$	$h_i - m_i$	$1 - (1 - h_i)(1 - m_i)$	(6)
header initialisation function at source, $h_{0(t+T)} = f(h_{0(t)}, h_{n(t)})$	$h_{0(t)} - h_{n(t)} + h_z$	$1 - \frac{(1-h_z)(1-h_{0(t)})}{1-h_{n(t)}}$	(9)
downstream path metric at resource j, $\rho_j(h_{j(t+T)})$	$h_{j(t+T)} - h_z$	$s\left(1 - \frac{1-h_z}{1-h_{j(t+T)}}\right)$	(10)

Table 1: The functions $g(\cdot)$ & $f(\cdot)$ required to implement re-feedback and $\rho(\cdot)$ to exploit it, summarising results from §2 & Appendix A, where notation is formally defined.

stream of node j is $\sum_{i=0}^{j-1} m_i = h_0 - h_j$. Node j can work this out by examining h_j in packets as they arrive, because h_0 is well-known. So the receiver (with $j = n$) can characterise the whole path delay $h_0 - h_n$. If it feeds back h_n to the sender using our notional end-to-end protocol (bent arrows in Fig 2a)), the sender can know the path delay too. So far, we have said nothing new, merely introducing notation using a familiar example.

With re-feedback the trick is simply for the sender to choose an initial header value such that, if the path metric were to remain unchanged, the header would reach a well-known value h_z at the destination—rather than *starting* from a fixed value. In our numerical example in Fig 2b) we assume the industry has standardised $h_z = 16$.

Although that is really all there is to it, we will now trace through how re-feedback works step by step to be precise about the differences:

1. For now, we will assume that the source bootstraps the very first packet of a flow with the fixed value we used with classic feedback, $h_{0(t)} = 255$. (When we need to distinguish between packets, we suffix each header value with the time t at which it was originally sent.)

2. The source has to remember the initial value it chose, as depicted by the curved boxes containing 255 at each source in Fig 2b) and containing $h_{0(t)}$ in Fig 1.

3. The packet traverses the path r, combining each local delay in turn into its header, using the combining function (subtraction) already described above.

4. The receiver feeds back the resulting delay header value $h_{n(t)}$ to the sender, which arrives a round trip T_r after the first packet was sent, depicted by the bent arrows.

5. The sender initialises the delay field in the next packet (dotted) to $h_{0(t+T_r)} = h_{0(t)} - h_{n(t)} + h_z$ as well as storing this new value in place of the last one. Each initial delay header value only depends on the previous round's initial value and the value fed back—both known locally at the source.

Now we can see that this simple shift of datum has achieved our original aim: as each packet arrives at a resource j anywhere in the network, it carries within its header h_j a prediction of its own downstream path delay, $h_j - h_z$, requiring no path state on the relay because h_z is well-known. Any packet in Fig 2b) illustrates this point, in that subtracting $h_z = 16$ from any header value predicts the sum of the remaining downstream resources on that path.

The second column of Table 1 summarises the functions to implement delay re-feedback that we have just derived. The third column gives the equivalent functions for congestion, derived in Appendix A.

As with delay, the combining function for each relay to accumulate local congestion into headers (first row) must be chosen to reflect the way congestion accumulates. In Appendix A.1 we define congestion as a probability, using axiomatic definitions[2]. So, as shown, we must use the function for combinatorial probability to combine congestion headers.

For either delay or congestion, the combining function at relays can be the same as for classic feedback, as the purpose is still to accumulate a path metric from local metrics. By avoiding arbitrary changes to the classic combining functions, re-feedback can be introduced incrementally, solely by arrangement between corresponding endpoints.

Each initialisation function (second row) ensures the header reaches h_z at the destination, given the way it accumulates along the path. Each function in the third row was derived from the previous two in order to predict the downstream path metric (DPM) from any node.

Note that neither prediction of DPM requires path state, only the state arriving in the packet itself. Further note that, for congestion, the DPM ρ_j also depends on the effective packet size s. For bit-congestible resources like links s = actual packet size. For packet-congestible resources like forwarding look-ups $s = 1$.

Fig 2 also illustrates how a change on a path affects the predictions in packets traversing it. The increase in delay at resource N_3 between Figs 2a) & b) (highlighted as a starburst) causes packets in flight upstream to underestimate their remaining downstream delay. Packets in flight downstream still correctly predict their downstream delay, but when feedback from them releases further packets, these underestimate their path delay.

With no further changes in local delays, packets in the following round (dotted) correctly predict the path again. Of course, changes in the unloaded delay at a node (e.g. due to a lower layer re-route) are rare, at least in fixed networks. However, for more volatile metrics like congestion, change is the norm. For delay, the prediction error will be $\sum_{i=0}^{n-1} (m_{i(t+T)} - m_{i(t)})$. For congestion, it is given by Eqn (11) in Appendix A.1. In both cases, the error depends on the difference between the whole path metrics.

To put these errors in context, re-feedback causes a source to suffer the same path prediction error as classic feedback—for equivalent path changes within the last round trip. So a re-feedback source transport can extract the same information, with the same timeliness and apply the same rate control algorithms with the same dynamics. For relays, it can take up to an extra half round trip before path changes reach them. But, for relays, any downstream path prediction at all is an improvement over classic feedback, which

[2]In contrast to the proposed ECN standard [19] where congestion is defined as the output of the RED algorithm—leaving no objective basis for improving RED.

path knowledge	alignment	sender	relay	rcvr
up-	sender	n/a	$\left[0, \frac{T}{2}\right]$	$\left[0, \frac{T}{2}\right]$
stream	receiver	n/a	×	×
down-	sender	$\left[\frac{T}{2}, T\right]$	×	n/a
stream	receiver	$\left[\frac{T}{2}, T\right]$	$\left[T, \frac{3T}{2}\right]$	n/a

Table 2: Comparison of sender and receiver-aligned feedback, by availability of path knowledge (x = not available; n/a = not applicable) and by range of timeliness (using symmetric delay).

offers none. And at the ingress, where policers are most appropriate, responsiveness will be similar to that of the source. Table 2 summarises the path knowledge that nodes gain or lose from re-feedback. It also quantifies the range of how long it can take for path changes to work through into correct path predictions in each case.

Previously, to achieve such knowledge at every relay would have required messages to be reverse routed hop by hop from all destinations (cf. routing messages or congestion back-pressure). Although re-feedback takes a little longer to propagate (because it travels via the source), it updates at the same *rate* as the ack rate—as often as TCP congestion control and many orders of magnitude more often than a typical routing message rate. Also, re-feedback piggy-backs on existing data, requiring no extra packet processing.

3. INCENTIVES

We aim to create an incentive environment to ensure anyone's selfish behaviour (including lying and cheating) leads to maximisation of social welfare.[3] Throughout this section we will focus primarily on characterisation of path congestion. This will stress re-feedback incentive mechanisms to the full in the face of conflict over scarce resources. Given most forms of fairness, including TCP's, also depend on round trip time, we will then outline how a path delay metric would be amenable to similar treatment.

Fig 3 sketches the incentive framework that we will describe piece by piece throughout this section. An internetwork with multiple trust boundaries is depicted. The downstream path congestion seen in a typical packet is plotted as it traverses an example path from sender S_1 to receiver R_1. They are shown using re-feedback, but we intend to show why everyone would *choose* to use it, correctly and honestly.

Two main types of self-interest can be identified:

- Users want to transmit data across the network as fast as possible, paying as little as possible for the privilege. In this respect, there is no distinction between senders and receivers, but we must be wary of potential malice by one on the other;
- Network operators want to maximise revenues from the resources they invest in. They compete amongst themselves for the custom of users.

Source congestion control: We want to ensure that the sender will throttle its rate as downstream congestion increases. Whatever the agreed congestion response (whether

[3]These mechanisms can lie dormant wherever co-operation is the social norm.

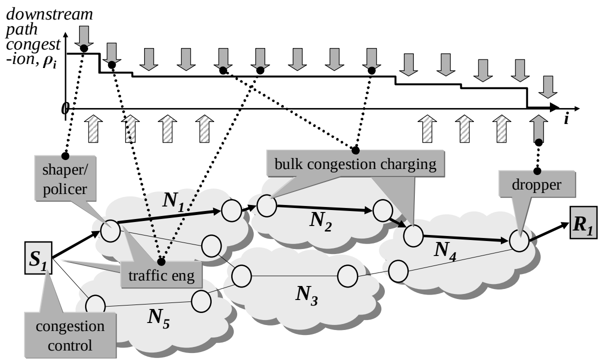

Figure 3: Incentive framework

TCP-compatible or some enhanced QoS), to some extent it will always be against the sender's interest to comply.

Edge ingress policing/shaping: But it is in all the network operators' interests to encourage fair congestion response, so that their investments are employed to satisfy the most valuable demand. N_1 is in the best position to ensure S_1's compliance and it now has a choice of mechanisms across a spectrum of customer autonomy. At one extreme, N_1 could give S_1 complete autonomy, but encourage responsible behaviour by charging for the downstream congestion in packets. Or it can shape traffic directly itself, removing all S_1's autonomy. Between the two extremes, it can police a congestion response agreed upfront with S_1 (§3.3).

Edge egress dropper: If the source has less right to a high rate the higher it declares downstream congestion, it has a clear incentive to understate downstream congestion. But, if packets are understated when they enter the internetwork, they will be negative when they leave. So, we introduce a dropper at the last network egress, which drops packets in flows that persistently declare negative downstream congestion (see §3.2).

Inter-domain traffic policing: But next we must ask, if congestion arises downstream (say in N_4), what is the ingress network's (N_1) incentive to police its customers' response? If N_1 turns a blind eye, its own customers benefit while other networks suffer. This is why all inter-domain QoS architectures (e.g. Intserv, Diffserv) police traffic each time it crosses a trust boundary. Re-feedback gives trustworthy information at each trust boundary so the congestion response can be policed in bulk.

Emulating policing with inter-domain congestion charging: However, between high-speed networks, we would rather avoid holding back traffic while it is policed. Instead, once re-feedback has arranged headers to carry downstream congestion honestly, N_2 can contract to pay N_4 in proportion to a single bulk count of the congestion metrics ρ crossing their mutual trust boundary (§3.4). Then N_2 has an incentive either to police the congestion response of its own ingress traffic from N_1 or to charge N_1 in turn on the basis of congestion counted at their mutual boundary. In this recursive way, each flow's response can be precisely incentivised, despite the mechanism not recognising flows. If N_1 turns a blind eye to its own upstream customers' congestion response, it will still have to pay its downstream neighbours.

No congestion charging to users: Bulk congestion charging at trust boundaries is passive and extremely simple, and loses none of its per-packet precision from one boundary to the next. But at any trust boundary, there is no imperative to use congestion charging. Traditional traffic policing can be used, if the complexity and cost is preferred. In particular, at the boundary with end customers (e.g. between S_1 and N_1), traffic policing will most likely be far more appropriate. Policer complexity is less of a concern at the edge of the network. And end-customers are known to be highly averse to the unpredictability of congestion charging [15]. So note well: *this paper neither advocates nor requires congestion charging for end customers and advocates but does not require inter-domain congestion charging.*

Competitive discipline of inter-domain traffic engineering: With inter-domain congestion charging, a domain seems to have a perverse incentive to fake congestion; N_2's profit depends on the difference between congestion at its ingress (its revenue) and at its egress (its cost). So overstating internal congestion seems to increase profit. However, smart border routing [10] by N_1 will bias its multipath routing towards the least cost routes, so N_2 risks losing all its revenue to competitive routes if it overstates congestion. In other words, N_2's ability to raise excess profits is limited by the price of its second most competitive route.

Closing the loop: All the above elements conspire to trap everyone between two opposing pressures (upper half of Fig 3), ensuring the downstream congestion metric arrives at the destination neither above nor below zero. So we have arrived back where we started in our argument. The ingress edge network can rely on downstream congestion declared in the packet headers presented by the sender. So it can police the sender's congestion response accordingly.

3.1 The case against classic feedback

So why can't classic congestion feedback (as used already by standard ECN) be arranged to provide similar incentives? Superficially it can. Given ECN already existed, this was the deployment path Kelly proposed for his seminal work that used self-interest to optimise social welfare across a system of networks and users [14]. The mechanism was nearly identical to volume charging; except only the volume of packets marked with congestion experienced (CE) was counted. However, below we explain why relying on classic feedback meant the incentives traced an indirect path—the long way round the feedback loop. For example, if classic feedback were used in Fig 3, N_2 would incentivise N_1 via N_4, R_1 & S_1 rather than directly.

Inability to agree what happened: In order to police its upstream neighbour's congestion response, the neighbours should be able to agree on the congestion to be responded to. Whatever the feedback regime, as packets change hands at each trust boundary, any path metrics they carry are verifiable by both neighbours. But, with a classic, sender-aligned path metric, they can only agree on the *upstream* path congestion—its offset from its well-known datum at the sender.

Inaccessible back-channel: The network needs a whole path congestion metric to control the source. Classically, whole path congestion emerges at the destination, to be fed back from receiver to sender in a back-channel. But, in any data network, back-channels need not be visible to relays, as they are essentially communications between the end-points. They may be encrypted, asymmetrically routed or simply omitted, so no network element can reliably intercept them. The congestion charging literature solves this problem by treating the sender and receiver as entities with aligned incentives. Although measuring classic ECN marking rates relative to their datum at the sender forces a 'receiver pays' model (at each trust boundary the downstream neighbour pays), at least this incentivises the receiver to refer the charges to the sender.

'Receiver pays' unacceptable: However, in connectionless datagram networks, receivers and receiving networks cannot prevent reception from malicious senders, so 'receiver pays' opens them to 'denial of funds' attacks.

End-user congestion charging unacceptable: Even if 'denial of funds' were not a problem, we know that end-users are highly averse to the unpredictability of congestion charging and anyway, we want to avoid restricting network operators to just one retail tariff. But with classic feedback, we cannot avoid having to wrap the 'receiver pays' money flow around the feedback loop, necessarily forcing end-users to be subjected to congestion charging.

To summarise so far, with classic feedback, policing congestion response *requires* congestion charging of end-users and a 'receiver pays' model. Whereas, with re-feedback, incentives can be fashioned *either* by technical policing mechanisms (more appropriate for end users) *or* by congestion charging (more appropriate inter-domain) using the safer 'sender pays' model.

Impractical traffic engineering: Finally, classic feedback makes congestion-based traffic engineering inefficient too. Network N_4 can see which of its two alternative upstream networks N_2 and N_3 are less congested. But it is N_1 that makes the routing decision. This is why current traffic engineering requires a continuous message stream from congestion monitors to the routing controller. And even then the monitors can only be trusted for *intra*-domain traffic engineering. The trustworthiness of re-feedback enables *inter*-domain traffic engineering without messaging overhead.

We now take a second pass over the incentive framework, filling in the detail more formally.

3.2 Honest congestion reporting

An honest sender will declare a certain downstream path metric (DPM) ρ_0 in packets to aim for zero at the destination after allowing for path congestion. We define cheating as the difference $\Delta\rho_{0c}$ relative to this ideal, taking overstatement as positive. To rely on the DPM packets carry, we must discourage dishonesty, whether positive or negative. If the sender declares a certain DPM, a certain rate response will be permitted, which can be policed (§3.3). For any safe congestion response, the higher the sender's declared DPM, to some degree the slower its data rate, and the lower the value it derives. So, to the right of Fig 4 we can show the sender's utility strictly decreasing with overstatement.

So senders have an incentive to understate DPM, which allows them a higher bit rate. But then DPM will turn negative before reaching the destination. If networks discard[4] negative packets, the utility to the sender of the higher bit rate will rapidly collapse, as shown to the left of the figure. Therefore honesty at $\Delta\rho_{0c} = 0$ will be the dominant sender strategy that maximises its net utility. A receiver that genuinely wants data to be sent as quickly as possible

[4]Various penalties short of discard, e.g. payload truncation, can be imposed in order to preserve the feedback loop, given a packet may be wrongly penalised.

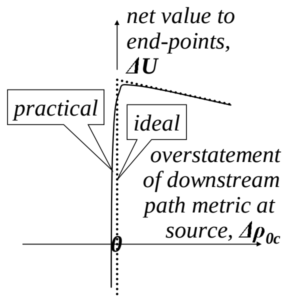
net value to end-points, ΔU
practical
ideal
overstatement of downstream path metric at source, Δρ0c
0

Figure 4: Truth telling incentives

has incentives aligned with the sender, so honest feedback also returns the maximum net gain.

In fact, the position is complicated by continuous variability of path congestion; even honest traffic will arrive at its destination spread around zero. Below we describe a dropper that makes allowances for this variability but still detects understatement of DPM. The best dropper we can currently envisage suffers some false positives and negatives, blunting the incentive to be absolutely honest (Fig 4).

3.2.1 *Adaptive dropper*

If congestion didn't vary, a malicious source understating congestion by $\Delta\rho_{0c}$ (numerically negative) would cause a proportionate understatement at the destination of $\Delta\rho_{nc}$.[5] But congestion does vary, so if the probability distribution of the DPM at the destination is $P_n(\rho_n)$ for an honest sender, it will be shifted to $P_n(\rho_n - \Delta\rho_{nc})$ for the malicious sender.

We propose a dropper[4] at the last hop before the receiver. The dropper builds a model of the prevailing pattern of cheating for all packets leaving the same interface and assumes that each new packet is characteristic of this recent history; the more recent cheating, the stricter the dropper becomes. But its strictness is further modulated by how negative ρ_n is of each packet under scrutiny.

Conceptually, the bell curve in Figure 5 shows the probability distribution of arriving packets, exponentially weighted to favour recent arrivals. We assume this will be the probability distribution of the DPM of the next packet. Superimposed on a different vertical scale is a conjectured penalty probability function, $p(\cdot)$ intended to allow through as much negative DPM as positive, but no more. This can be achieved by ensuring that the distribution remaining after applying the penalty function is symmetric about zero (the unshaded cusp curve). So for $\rho_n < 0$:

$$\big(1 - p(\cdot)\big) P_n(\rho_n - \Delta\rho_{nc}) = P_n(\rho_n + \Delta\rho_{nc}). \qquad (1)$$

Initially we choose to keep state and processing to a minimum by modeling prevailing conditions with just the exponentially weighted moving average μ and EWM variance ν. So we model the prevailing distribution $P_n(\rho_n - \Delta\rho_{nc})$ as if it were the normal distribution $N(\mu, \nu)$ reconstructed from recent traffic, whatever the actual distribution (e.g. Fig 6).

At each packet, the EWMA & EWMV are updated:

$$\mu \leftarrow \gamma\rho_n + (1-\gamma)\mu \qquad (2)$$

$$\nu \leftarrow \gamma(\rho_n - \mu)^2 + (1-\gamma)\nu. \qquad (3)$$

[5]From Eqn (7) $\Delta\rho_{nc} = \big(1 - \|_0^{n-1} m\big)\Delta\rho_{0c}$

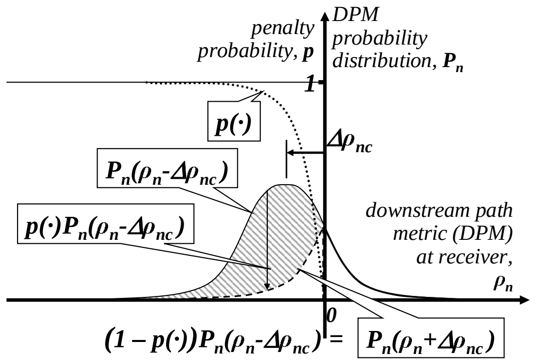

Figure 5: Penalising misbehaviour under uncertainty

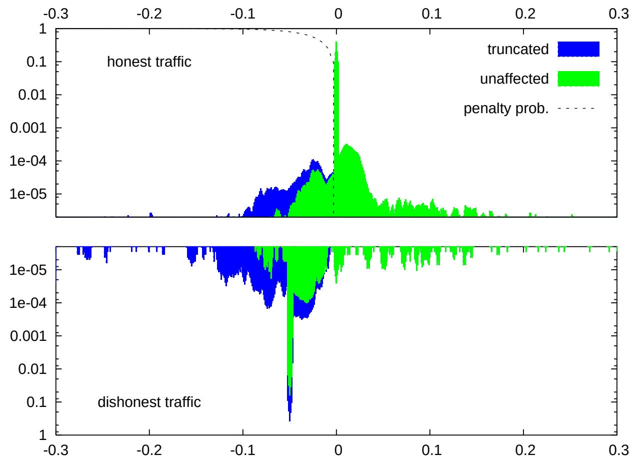

Figure 6: Typical simulated distributions of DPM at the destination from honest (top) and dishonest (bottom) sources, also showing proportion of penalised traffic (note log scale).

For attack traffic $\mu \to \Delta\rho_{nc}$, converging faster by increasing γ to weight recent values $(0 < \gamma \leq 1)$. In maintaining the EWMA, positive packets with the 'certain' flag cleared (see §2) are ignored, incentivising correct use of the flag.

Then, using the formula for a normal distribution,

$$P_n(\rho_n - \Delta\rho_{nc}) = \frac{1}{\sqrt{2\pi\nu}} e^{-\frac{(\rho_n-\mu)^2}{2\nu}}. \qquad (4)$$

we can derive the required penalty probability function to apply to each specific packet with DPM ρ_n, by re-arranging (1) and substituting from (4):

$$\begin{aligned} p(\rho_n, \mu, \nu) &= 0; & \mu \geq 0 \text{ or } \rho_n \geq 0 \\ &= 1 - \frac{P_n(\rho_n + \mu)}{P_n(\rho_n - \mu)}; & \mu < 0, \rho_n < 0 \\ &= 1 - e^{\frac{-2\rho_n\mu}{\nu}} & (5) \end{aligned}$$

As required, the penalty becomes stricter the worse the EWMA becomes, but flattens to zero discards when honest users keep the EWMA to zero.

Where a cheating flow is hidden in a large honest aggregate, it causes a slightly negative EWMA, leading to some dropping. After Floyd and Fall [7] we cache the flow identifiers of penalised packets. Once any aggregate of destination (and/or source) identifiers appears more often than would be likely by chance, a second instance of the dropper is spawned and traffic matching the identifier(s) is filtered into it. Each instance of a focused dropper maintains its own EWMA[6] and may spawn further droppers. These focused droppers should be far more sensitive than the first, also shielding honest traffic from the risk of false negatives.

Of course, if cheating negative traffic imitates identifiers used in honest traffic, both will be filtered into the same focused dropper, causing collateral damage to the honest traffic. But by definition the cheating traffic will tend to be more negative, which the above penalty function is designed to discriminate against.

Having isolated suspect identifiers, an egress edge dropper can send hints upstream. Any node can test hints because they point to traffic measurably below an objective threshold. And a node need only act on the hints if it has sufficient resources. So the hints need not be authenticated (unlike DoS filter push-back requests), avoiding vulnerability to floods of bogus authentication requests. Also, the hints can safely jump multiple domains without the need for a global key management infrastructure. So push-back of hints does not depend on the co-operation of high speed core networks, where operators are more wary of any additional processing.

Even if explicit congestion marking were universally deployed, buffers could still occasionally overflow. So irrespective of any hints, if a router must discard packets, clearly it should bias against any with negative DPM.

3.2.2 Honest delay reporting

Congestion control and traffic engineering depend on path delay as well as congestion, so we need header fields for both. The framework we built above (§3.2 & Fig 4) to incentivise honest congestion reporting relied on two properties of congestion: it should not be negative; and rising path congestion should lead to a drop in sending rate (whatever form of fairness is chosen). Delay has exactly the same properties: negative delay is physically impossible; and rising feedback delay should lead to a lower sending rate.

So, we can use a similar incentive mechanisms to that we used for congestion to ensure the sender neither overstates nor understates delay. An adaptive dropper, like the one above for the congestion field (§3.2.1), could detect and remove any negative imbalance of delay headers at the internetwork egress. And at the ingress we can use a policer like the TCP rate equation policer below (§3.3.1) that punishes sources sending faster than the TCP-fair rate, which depends inversely on both congestion and feedback delay.

3.3 Fair congestion response

3.3.1 TCP rate equation policer

In the fastest phase of the TCP algorithm (congestion avoidance), TCP converges to the rate $\bar{x}_{TCP} \approx ks/(T\sqrt{p})$, where $k \approx \sqrt(3/2)$ and s, T & p are respectively the packet size, round trip time and path marking (or loss) rate [17]. Re-feedback ensures that a policer at the network ingress can derive these parameters from the metrics each packet truthfully declares. It can then calculate a compliant rate against which to compare the source's actual rate.

Previous policers had to be placed at every site of possible congestion. With re-feedback, it is sufficient to place one policer at each ingress to the internetwork. Here, downstream congestion $\rho_{2,1}$ can be assumed equal to path congestion, p. The policer can approximate the round trip delay as $T \approx T_0 + 2\rho_{1,1}$, where the upstream round trip T_0 can be found by a previous echo test against each source and the downstream delay $\rho_{1,1}$ arrives in each packet[7].

If the current TTL and ECN fields in IP were used to implement re-feedback, as sketched in §4, an ingress policer would have enough information to mirror the TCP algorithm. As we explain later, binary congestion marking takes a long time to convey a congestion level—indeed, a ridiculously long time as flow rates increase over the years. So, given the architectural nature of this paper, we prefer to focus on multi-bit congestion and delay fields in future packet headers. Quantifying how quickly a policer could detect misbehaving flows using current IP, and the resulting gain from more bits in headers, is planned for future work.

Below we outline one possible policing algorithm. It requires per flow state, but this isn't necessarily a scalability problem at the edge of an internetwork, however it does lay the policer open to resource depletion attacks. In a longer version of this paper (in preparation) we describe a variant with sub-linear scaling of flow state, but our goal here is to give a clear implementation example that is concrete but avoids gratuitous distractions.

The policer requires a token bucket per flow. It empties the bucket by the size of each arriving packet and fills it at a rate equivalent to that of a TCP compliant flow experiencing the same path conditions. It calculates this by deriving p and T from the re-feedback fields as above. In other words, when a packet arrives, the policer subtracts the packet size s from the bucket and adds $ks\Delta t/(T\sqrt{p})$, where Δt is the time since the flow's previous packet.

If the bucket empties, sanctions are applied to the flow. For instance, all future packets might be discarded, or the policer could choose to take over rate control for the flow. The depth of the bucket controls the flexibility allowed for a flow to stray from its expected throughput; it is set to $\alpha\bar{x}_{TCP}\tau$, where α is the threshold greediness for a flow to be considered non-compliant over a time τ, and $\bar{x}_{TCP}$ is an EWMA of $ks/(T\sqrt{p})$. A flow with a throughput higher than $\alpha\bar{x}_{TCP}$ will be detected in a time smaller than τ.

α is chosen so that a compliant flow is most unlikely to trigger starvation of the bucket. For instance, when p=1%, the average congestion should be 12.3 packets per round-trip. The probability of getting a window larger than 42 is smaller than 0.01%. Setting α to $42/12.3 = 3.4$ and $\tau = T$ would guarantee that less than one compliant flow in ten thousand would be subjected to sanction. Increasing α and τ would reduce false positives further.

3.3.2 Edge QoS

Our interest in solving the policing problem was not solely to police a single response to congestion, such as TCP-

[6]From Eqn (5) an attacker can reduce dropping probability by increasing variance, e.g. by alternating honest & zero packets. So a focused dropper should use the EWMV of the top level dropper. We are investigating variants with varying degrees of statefulness and responsiveness.

[7]For simplicity, we choose to ignore congestion delay, because simple scaling arguments [13, §2] show that as capacity continues to grow, congestion delays will become insignificant relative to fixed propagation delays.

friendliness, although that alone is a major contribution. Once timely, truthful downstream path information is visible to ingress network operators in data packets, they can offer a spectrum of responses to incipient congestion. This is equivalent to offering different levels of QoS, perhaps ranging from a scavenger class, through best effort and premium levels of differentiated service to admission controlled bandwidth reservations (the right to zero congestion response)—all without any differential treatment on network elements beyond the first ingress (with the caveat below).

Kelly and co-workers [14] pioneered this approach, proving it optimises social welfare across a network. Further its policing architecture solves the scalability problems inherent in other QoS approaches, though this is seldom appreciated.

With traditional QoS some identification convention must distinguish which traffic the edge has decided should be given which preferential treatment as it passes to interior domains. Using flow identification (like Intserv) preserves precision, but scales badly. Using class identification (like Diffserv) loses precision at scale.

With edge QoS, instead of the edge identifying the traffic's QoS for interior routers, interior routers identify the traffic's congestion for the edge. Because traffic already carries end-point identifiers, regular packet forwarding carries congestion marking to its source cause through aggregation and deaggregation with absolute precision, with no need for a separate QoS identification convention. The only unequal treatment of different traffic identities is in the policer at the first ingress to the internetwork, where customer or flow identities have local significance.

Siris [20] has proven this approach through simulation. But deployment was confined to a radio network controller scenario where congestion feedback in the back-channel to the sender could be intercepted—a constraint that can be relaxed with re-feedback, giving general applicability.

Having sung the praises of closed-loop control, a caveat is necessary. Unusual conditions (link failure or sudden traffic shifts) can cause traffic in flight to overflow queues. So, within a round trip, each resource must still be capable of rudimentary local (open-loop) traffic class prioritisation until the closed-loop restores order.

Adaptive policer: If one user creates multiple flows, or runs flows for longer than another user (e.g. p2p file-sharing), per-*flow* approaches like TCP cannot arbitrate fairness between *users*. We can generalise to an adaptive policer based on MulTCP [6] that gives each flow an equivalent rate to w TCP flows. With the benefit of re-feedback, it can maintain a per user count of congestion sent. But, rather than levying an unpredictable charge for this congestion [14], the policer can compare the count to whatever the user chooses to pay. So a flat monthly rate would effectively buy a congestion quota. The closer the internal congestion count approached this quota, the more w would be squeezed.

3.3.3 *Flow start incentives*

At the start of each flow, a sender neither knows the state of the path to the destination nor the relative change the additional flow will cause. TCP's slow-start phase incrementally finds out both while also giving other flows time to make room for the new flow.

The re-feedback incentive framework deliberately presents a dilemma to a sender without recent path knowledge (e.g. at the first packet, or after an idle period). Sending understated DPM increases the risk of discard at the egress dropper. But sending overstated DPM increases the risk of sanction at the ingress policer as the flow rises to full rate. The strategies around this dilemma deserve a paper in their own right, so here we merely provide an outline.

We should think of TCP's exponential slow-start as dependent on an implicit evolving estimate of path congestion by the sender, starting pessimistically by assuming high path congestion. Inverting TCP's steady state rate equation gives $\rho \propto \frac{1}{x^2}$ to a first approximation. So rate doubling quarters the implicit path congestion estimate every round trip. To safely pass the policer and the dropper, the sender should be consistent, also using this implicit estimate of path congestion to set the DPM in each sent packet. If it reduces its path congestion estimate too quickly (increasing its rate accordingly), it will undershoot the true path congestion and risk being caught by the egress dropper.

So the re-feedback incentive framework encourages caution at the start of a flow in proportion to path uncertainty—reminiscent of TCP's slow start. However this claim greatly depends on how quickly our mechanisms can detect and remove non-compliant behaviour.[8]

It is well-known that repeated binary congestion feedback like ECN takes a long time to signal low congestion levels [9]. So ECN is not a good basis on which to build responsive policing mechanisms. In the years it would take to deploy the TCP modifications needed for our re-feedback extension of ECN (§4), TCP will be hitting its own scalability limits.[9] So although we believe re-ECN could start to solve policing problems fairly quickly, we must emphasise that a multi-bit congestion field will need to be considered anyway. It would provide responsive policing even if short flows dominate the future traffic mix. And at the same time, it would help fix TCP/IP for high capacity scenarios.[10]

This still leaves the problem of whether the new flow will push a currently uncongested path into congestion. If a low priority scavenger class is available, the simple solution would be to rise aggressively to full rate in that class, then pace packets for the desired class at the resulting ack rate.

3.4 Inter-domain incentive mechanisms

The overview of our incentive framework explained why bulk inter-domain congestion charging emulates policing with per-flow precision. We now describe this mechanism.

At an inter-domain interface, only a single bulk counter (and two temporary ones) per direction is needed. The main counter merely accumulates the DPM ρ in every passing packet over an accounting period T_a (e.g. a month). At the end of the month, N_1 should pay N_2 the charge $C_a = \lambda \sum^{T_a} \rho^+$, where λ is the fixed price of congestion agreed be-

[8]Nonetheless, we can ensure detection of misbehaviour carries over from old flows to new: i) the above adaptive policer remembers misbehaving senders; ii) during a DDoS attack on a receiver, the dropper's EWMA carries over from one source to the next.

[9]The number of packets between binary congestion marks scales $O(x^2)$ with TCP's steady state rate—to sustain 10Gbps a flow would only sawtooth every 90mins between marks. While marking rate only scales $O(x)$ with Kelly's rate control algorithm, that still doesn't solve the problem, given x is currently doubling every 1.6 years or so.

[10]An extra multi-bit field in IP is proposed for the allowed congestion window in XCP [12] and for the allowed sending rate in Quick-Start [11].

tween them. To implement this with the re-feedback variant of ECN described in §4.1, the meter would simply need to increment or decrement by the size of packets marked with the ECT(0) or CE code-points respectively.

To protect receiving domains from 'denial of funds' attacks, any usage element of a charge should be 'sender pays'.[11] So $\lambda \geq 0$ and persistently negative ρ should be ignored, given negative congestion is physically impossible. To meter this, packets with positive and negative ρ should be separately accumulated in temporary counters with the two added every few seconds, only accumulating a positive sum in the main counter. Once neighbours agree that 'no-one pays' for persistent negative congestion, they are incentivised to introduce the dropper (§3.2) to remove persistent negative traffic, which no longer carries any ability to pay for further downstream congestion. 'Receiver pays' can optionally be arranged between edge operators without risk of 'denial of funds' through an end-to-end clearinghouse [2].

We should clarify that we neither require nor expect universal inter-domain congestion charging. However, because it exposes true costs, it is likely to emerge as the competitive equilibrium [2]. Current tariffs such as 95th %ile peak demand or volume charging may continue. But to compete, manual price adjustments will be needed to track the congestion price. So congestion charging is likely to predominate, given it uses a simple, passive mechanism without regard to flows, but automatically adjusts the price to give the correct upstream incentives to the precise flows that deserve them.

The main alternative to usage charging is the service level agreement, where a network contracts to keep metrics within statistical limits. Currently, proving whether delay or loss (impairment) budgets have been exceeded and by whom requires a comprehensive system of trusted echo reflectors. Re-feedback greatly simplifies these problems of SLA accountability, because it ensures downstream metrics are visible at each inter-domain border.

3.5 Distributed denial of service mitigation

A flooding attack is inherently about congestion of a resource. Because re-feedback ensures the causes of network congestion experience the cost of their own actions, it acts as a first line of defence against DDoS. As load focuses on a victim, upstream queues grow, requiring packets to be pre-loaded with a higher congestion metric. If the source does increase the initial metric, its own network's ingress policer will throttle the flow. If the source doesn't increase the initial metric, it will become negative at the congested resource, which will bias its drop against negative traffic.

Inter-domain congestion charging ensures that any network that harbours compromised 'zombie' hosts will have to pay for the congestion that their attacks cause in downstream networks. Therefore, it is incentivised to deploy our adaptive policer (§3.3.2). The adaptive policer limits hosts that persistently causes congestion to only send very slowly into congested paths. As well as protecting other networks, the extremely poor performance at any sign of congestion will incentivise the zombie's owner to clean it up.

Note, however, that delay in detecting attacks does leave re-feedback briefly vulnerable (§§3.3.3 & 5).

[11] A capacity charge made to the larger network, whatever the direction of traffic, might well complement congestion charging (or any form of usage charging).

4. PROTOCOL ENGINEERING

Although the goal of this paper is not to prove that re-feedback is immediately deployable, it does seem possible to deploy a limited form, described in detail in a paper in preparation and summarised here. Re-feedback consists of at least three protocol elements: i) a congestion field; ii) an unloaded delay field; and iii) a certain flag (§2).

It is also necessary to flag that re-feedback is in use, but the certain flag serves this purpose. There are no flags in IPv6 so an extension header would be required. In IPv4, bit 49 is possibly available, but there are many other competing claims on it. An alternative approach would be to assign new protocol ids to re-feedback transports.

4.1 Re-ECN

Two bits in the IP protocol are assigned to the ECN field [19]. The sender indicates an ECN capable transport (ECT) using either of the two code-points `10` or `01` (ECT(0) & ECT(1) resp.). Routers probabilistically set `11` if congestion is experienced (CE). The choice of two ECT code-points permitted future flexibility, optionally allowing the sender to encode a nonce [21] in the packet stream.

To re-align congestion feedback, we use this flexibility in a scheme we call re-ECN. Here we only discuss TCP/IP, not other IP transports. No changes to the IP or TCP wire protocols are required. Neither the IP handlers nor the TCP receiver need changing, only the TCP sender. We define what is effectively a virtual header field h, where $h_j = u_j - z_j$ at any node j on the path. u_j is the rate of CE and z_j the rate of ECT(0) traversing that node. As with current ECN, no packets are sent with CE set: $u_0 = 0$. And TCP feeds back to the source any CE arriving at the destination in the echo congestion experienced (ECE) field.

For re-feedback, the sender arranges the starting value h_0 of this virtual header so that it will reach a standardised datum at the destination $h_z = 0$. So, we need $z_n = u_n$, which will result if the sender sets ECT(0) on the proportion of packets $z_0 = u_n/(1 - u_n)$.[12]

To set this proportion of ECT(0), the TCP sender's ack handler should set ECT(0) in the next packet after an ECE arrives in an ack (treating drops equivalently). And it should set an extra ECT(0) every $(U - 1)$ ECEs, where U is the EWMA of the number of packets between successive ECE.

From Eqn 10 in Table 1 the virtual header h_j in packets arriving at any node j on the path is sufficient for the node to derive a prediction of downstream path congestion $\rho_j = 1 - 1/(1 - hj)$. For small $h_j \ll 1$; $\quad \rho_j \approx -h_j$. In other words, downstream congestion can be approximated simply by subtracting the rate of ECT(0) from that of CE. As we haven't changed the rate of CE marking, it still represents upstream congestion. So in 1.5 bits we have encoded downstream congestion $-h_j$, upstream congestion u_j and whole path congestion z_j.

We chose z as the rate of ECT(0) rather than ECT(1) deliberately. Existing ECN sources set ECT(0) at either 50% (the nonce) or 100% (the default). So they will appear to a re-feedback policer as very highly congested paths. When policers are first deployed their threshold greediness α can be configured permissively, allowing through both 'legacy' ECN and misbehaving re-ECN flows. Then, as the thresh-

[12] Details of what to do for high congestion beyond $u_n > 1/2$ and why it is safe are omitted for brevity.

old is set more strictly, the more legacy ECN sources will gain by upgrading to re-ECN.

We emphasise that we believe a multi-bit congestion field will eventually be needed if flow rates are to continue to rise (§3.3.3). We propose an extension header for IPv6, including a field to shadow TTL (like Quick-Start [11]). So, if the whole path of routers doesn't support the extension, the endpoints can fall back to re-ECN or drop. Transport protocols would also all have to be updated for multi-bit fields.

4.2 Re-TTL

Delay re-feedback can be achieved by overloading the TTL field, without changing IP or router TTL processing. A target value for TTL at the destination would need standardising, say $h_z = 16$ (Fig 2). If the path hop count increased by more than h_z during a routing change, it would temporarily be mistaken for a routing loop, so h_z would need to be chosen to exceed typical hop count increases. The TCP wire protocol and handlers would need modifying to feed back the destination TTL and initialise it as in Table 1.

It would be necessary to standardise the unit of TTL in terms of real time. Precision could be improved in the longer term if routers decremented TTL to represent exact propagation delay to the next router. That is, for a router to decrement TTL by, say, 1.8 time units it would alternate between 1 & 2 at a ratio of 1:4. Although this might sometimes require a seemingly dangerous null decrement, a packet in a loop would still decrement to zero after 255 time units.

5. DROPPER PERFORMANCE

The re-feedback incentive framework relies critically on how quickly the dropper (§3.2.1) can detect and isolate flows that are maliciously understating congestion, and how much collateral damage is suffered by honest packets. The error in an honest source's prediction of congestion for re-feedback (Eqn 11) depends on how well path congestion in one round trip correlates with congestion the next. If the correlation is weak, to avoid falsely dropping honest traffic the dropper has to heavily smooth out all the variation, making it sluggish to respond to a movement in the average due to an attack. We chose to use ns2 to run a series of simulations with highly demanding sets of flows arriving at the dropper, some having traversed up to five potential bottlenecks. Below are the highlights of the experiments. Details are in the longer version of this paper.

We implemented the multi-bit variant of congestion re-feedback carrying real numbers in TCP Reno using the initialisation and combining functions in Table 1. For the local congestion metric at each router m_i, we extracted the real value of the marking probability, p_b, used within the RED algorithm [8, §4] before its transformation into a binary mark. However, to be more demanding we still allowed TCP rate control to respond in its usual sawtooth way to binary ECN feedback and drops. We bounded headers h within $[-1, 1]$.

We implemented the dropper within the RED module, simulating packet truncation as its sanction—in order to preserve the feedback loop. We omitted flow-focused dropping as our initial aim was to assess feasibility. From Eqn 10 we approximated downstream congestion as $\rho_n \approx -h_n$, using $h_z = 0$.

Simulation model: We used a parking lot topology of 5 core nodes n_1 to n_5, connected by 10Mbps links. Queues at all core routers were RED-ECN in the direction of traffic (n_1 to n_5), and DropTail in the reverse direction with sufficiently large links to prevent ACK drops. The dropper ran on n_5. Traffic entered the network from all nodes n_1–n_4 and left it after a number of hops ranging across (1,2,3, & 5). Transmission delays between core nodes were 3ms, while edge delays defined a range of RTTs between 90–500ms, averaging $\tilde{2}$50ms. TCP flows through the dropper were grouped in three classes according to their typical RTT: low (L), medium (M), and high (U) of the order of 100, 250 & 500ms.

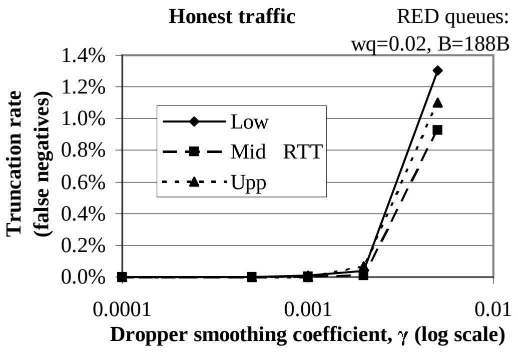

Figure 7: Effect of dropper smoothing on truncation rate for honest flows from lower, mid & upper RTT ranges (note: no focused dropper)

The traffic model consisted of 400 sources of which 110 were TCP-ECN and the rest UDP, with TCP traffic consistently $> 90\%$ of total bits. This reflected current [3] not necessarily future Internet traffic (when reduced TCP volume is expected). Packet sizes were all 1500B. We did not explicitly model HTTP but defined 100 TCP sources as FTP, uniformly varying sessions from small (20pkt) to large (1500pkt), with sources' average idle times exponentially distributed. The remaining 10 FTP sessions transferred infinite-sized files and traversed all core nodes. The UDP sources were packet trains with both ON and OFF times Pareto distributed with parameter 1.9. The resulting frequent short-lived and sporadic long-lived sessions reflected long-tailed Internet traffic. Traffic profiles were subject to random variations with RED queue utilisation varying from high 80s to low 100s percentages throughout. Traffic sources were initially generated at random uniformly between 0 and 20s; statistics collection began 30s into the 300s simulation. The (gentle) RED parameters were set to the currently recommended values relative to buffer size.

Simulation results: We used solely honest sources to find the dropper's baseline sensitivity under various conditions. Fig 7 is typical, leading us to use smoothing coefficients just below the knee of the curve for our later experiments with dishonest flows. That is $\gamma = 0.0005, 0.001$ or 0.002. Even in the last case truncation rates were only 1–7:10,000. We expected the subset of flows with below average RTT (L) to be better at predicting congestion, given it would have less time to change. In fact, they consistently suffered about 50% worse truncation rates than flows with average RTTs. Indeed, flows with average RTT were generally better at predicting the next round trip's congestion than both U and L flows either side of them. Closed loop traffic behaviour at sub-RTT timescales is a developing field,

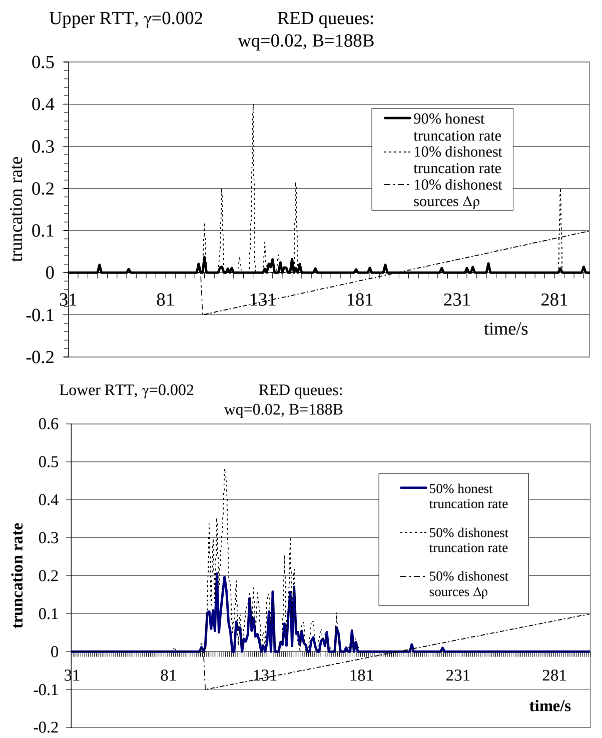

Figure 8: Truncation discrimination with a) 10% and b) 50% of sources dishonest $\Delta\rho_{0c} = -0.1 + 0.1$ ramp (note: no focused dropper).

but we are not aware of any explanation for these results.

We introduced dishonest traffic as a step underdeclaring congestion by 0.1 to see how fast a large change could be detected, then ramping up to see when a small level of dishonesty became undetectable. Fig 8a) shows how if even 10% of flows are dishonest, high truncation peaks occur that would mark out the flow for focused treatment by a focused dropper. Note how, as levels of understatement decline, the dishonesty is lost in random fluctuations. Fig 8b) shows another example where 50% of flows are dishonest, thus causing strong near-immediate discrimination.

6. RELATED WORK

Clark [5] proposed a decrementing field representing payment as a packet traversed a path, with receiver-initiated messages able to meet it in the middle to make up any shortfall. We argue that network layer fields should represent verifiable properties of the path. Then operators can choose to apply pricing to them to determine cost or value (or choose not to).

The ECN nonce [21] is an elegant mechanism for a sender to detect feedback suppression by a receiver. However, senders' interests often align more closely with their receivers' than with the welfare of all network users.

We borrowed ideas from policers in the literature [16, 7, 18] for our rate equation policer. However, without the benefit of re-feedback they don't police the correct rate for the condition of their path. They detect unusually high *absolute* rates, but only while the policer itself is congested, because they work by detecting prevalent flows in the discards from the local RED queue. These policers must sit at every potential bottleneck, whereas our policer need only be located at each ingress to the internetwork. As Floyd & Fall explain [7], the limitation of their approach is that a high sending rate might be perfectly legitimate, if the rest of the path is uncongested or the round trip time is short.

7. CONCLUSIONS & FURTHER WORK

We have argued for a re-alignment of the datum of path characterisation metrics like TTL and congestion notification. Moving the datum to the destination ensures that each packet arrives at every relay carrying a view of the remaining path to be traversed by the packet, albeit a round trip delayed. Despite overhauling the underlying feedback architecture, we have briefly described a limited form of re-feedback that could be deployed incrementally around unmodified routers without changing IP.

Once downstream information is visible, inline equipment can exercise control mechanisms that were previously impractical, such as rate policing or inter-domain traffic engineering. We describe how to police TCP's and other closed-loop rate control algorithms. Not only is it now possible to detect and remove traffic that exhibits a hostile response to congestion. It is also possible to explicitly permit applications that require such a response, perhaps given suitable payment in exchange for the enhanced quality of service.

We have introduced an incentive framework which ensures that the dominant strategy of selfish parties around the feedback loop will be to declare re-feedback honestly. It relies critically on whether malicious flows can be detected at the egress, while minimising false negatives. We have simulated an adaptive dropper to show this may indeed be feasible.

Re-feedback allows senders a view of route costs, and networks a view of downstream congestion. By democratising access to path information, it enables a tussle over whether network control lies with end-points or the network [4].

Having laid the foundations for this idea, much detail remains to be filled in. In particular further experiment is required to fully stress both the dropper and the policer, particularly under dynamic attacks. Our other priority is to assess whether responsive policing and dropping will be possible with our binary re-ECN scheme.

Acknowledgements

Sébastien Cazalet contributed to the early ideas behind this work. The following have all contributed useful review comments: David Songhurst, Keith Briggs, Marc Wennink, Louise Burness, Phil Eardley, Alexandru Murgu, Nigel Geffen (BT), Jon Crowcroft (Cambridge Uni), Mark Handley, Stephen Hailes (UCL), Sally Floyd (ICIR) & others at ICIR & UCL.

8. REFERENCES

[1] R. Braden, D. Clark, S. Shenker, and J. Wroclawski. Developing a next-generation Internet architecture. White paper, DARPA, July 2000.

[2] B. Briscoe and S. Rudkin. Commercial models for IP quality of service interconnect. *BTTJ*, 23(2), Apr. 2005.

[3] k. claffy. The nature of the beast: Recent traffic measurements from an Internet backbone. In *Proc. INET'98*. ISOC, 1998.

[4] D. Clark, K. Sollins, J. Wroclawski, and R. Braden. Tussle in cyberspace: Defining tomorrow's Internet. *Proc. ACM SIGCOMM'02, CCR*, 32(4), Aug. 2002.

[5] D. 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, Oct. 1996.

[6] J. Crowcroft and P. Oechslin. Differentiated end to end Internet services using a weighted proportional fair sharing TCP. *CCR*, 28(3):53–69, July 1998.

[7] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, Aug. 1999.

[8] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, Aug. 1993.

[9] A. Ganesh, P. Key, and L. Massoulié. Feedback and bandwidth sharing in networks. In *Proc. 39th Annual Allerton Conference on Communication, Control and Computing*, 2001.

[10] D. K. Goldenberg, L. Qiu, H. Xie, Y. R. Yang, and Y. Zhang. Optimizing cost and performance for multihoming. *Proc. ACM SIGCOMM'04, CCR*, 34(4):79–92, Oct. 2004.

[11] A. Jain, S. Floyd, M. Allman, and P. Sarolahti. Quick-Start for TCP and IP. Internet Draft draft-ietf-tsvwg-quickstart, IETF, May 2005. (Work in progress).

[12] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. *Proc. ACM SIGCOMM'02, CCR*, 32(4):89–102, Oct. 2002.

[13] F. P. Kelly. Models for a self-managed Internet. *Philosophical Transactions of the Royal Society*, A358 (2000):2335–2348, 1999.

[14] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, 1998.

[15] A. Odlyzko. A modest proposal for preventing Internet congestion. Technical report TR 97.35.1, AT&T Research, Florham Park, New Jersey, Sept. 1997.

[16] T. J. Ott, T. V. Lakshman, and L. H. Wong. SRED: Stabilized RED. In *Proc. IEEE Conf. on Computer Comm's (Infocom'99)*, pages 1346–1355, Mar. 1999.

[17] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. *Proc. ACM SIGCOMM'98, CCR*, 28(4), 1998.

[18] R. Pan, L. Breslau, B. Prabhaker, and S. Shenker. Approximate fairness through differential dropping. *CCR*, 33(2):23–40, Apr. 2003.

[19] K. K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN) to IP. RFC 3168, IETF, Sept. 2001.

[20] V. A. Siris. Resource control for elastic traffic in CDMA networks. In *Proc. ACM Int'l Conf. on Mobile Computing and Networks (MobiCom'02)*, Sept. 2002.

[21] N. Spring, D. Wetherall, and D. Ely. Robust explicit congestion notification (ECN) signaling with nonces. RFC RFC3540, IETF, June 2003.

APPENDIX

A. RE-FEEDBACK FUNCTIONS

Below, following the notation of §2, we derive the functions required to implement re-feedback for congestion:

- the combining function on each relay, $h_{i+1} = g(h_i, m_i)$,
- the function to initialise header values $h_{0(t+T)} = f(h_{0(t)}, h_{n(t)})$
- the downstream path metric from resource j, $\|_j^{n-1} m_{(t+T)}$.

We coin the notation $\|_a^j m$ for the **path metric**, which is the composition of all the local metrics m_i experienced by a packet along the sequence of resources $\{a, \dots i, \dots j\}$ using the combining function appropriate to the metric in question.

A.1 Congestion re-feedback

Definition 1. *The congestion, m_i, caused by a packet at single resource i is the probability that the event X_i will occur if the packet in question is added to the load, given any pre-existing differential treatment of packets. Where X_i is the event that another selected packet will not be served to its requirements by resource i during its current busy period.*

So, at resource i, the contribution to congestion is $m_i = P(X_i) \in [0, 1]$, which is a function of local load.

Definition 2. *The* path *congestion, $\|_a^j m$, caused by a packet traversing a sequence of resources, is the probability that the event X will occur if the packet in question is added to the loads at each resource along its path, given any pre-existing differential treatment of packets. Where X is the event that another selected packet will not be served to its requirements by any of the sequence of resources $\{a, \dots i, \dots j\}$ during their current busy periods.*

From definition 1, the function that combines the local contribution with the incoming congestion notification field must emulate combinatorial probability resulting in an outgoing header value

$$h_{i+1} = 1 - (1 - h_i)(1 - m_i). \tag{6}$$

$\therefore$ if the header is h_a before resource a, after node $j-1$ it will be

$$h_j = 1 - (1 - h_a)\textstyle\prod_{i=a}^{j-1}(1 - m_i). \tag{7}$$

From definition 2 the path metric from resource a to $j-1$,

$$\begin{aligned} \|_a^{j-1} m = P(X) &= 1 - \textstyle\prod_{i=a}^{j-1}(1 - P(X_i)) \\ &= 1 - \textstyle\prod_{i=a}^{j-1}(1 - m_i) \\ &= 1 - \frac{1 - h_j}{1 - h_a} \end{aligned} \tag{8}$$

A source with perfect foresight would initialise a packet header to $h^*_{0(t+T)}$ in order to reach its target value at the destination

$$\begin{aligned} h_{n(t+T)} &= 1 - \left(1 - h^*_{0(t+T)}\right)\left(1 - \|_0^{n-1} m_{i(t+T)}\right) \\ &= h_z \\ \therefore h^*_{0(t+T)} &= 1 - \frac{1 - h_z}{1 - \|_0^{n-1} m_{i(t+T)}} \end{aligned}$$

A practical source will use the previous path metric as an estimator for the next and set

$$\begin{aligned} h_{0(t+T)} &= 1 - \frac{1 - h_z}{1 - \|_0^{n-1} m_{i(t)}} \\ &= 1 - \frac{(1 - h_z)(1 - h_{0(t)})}{1 - h_{n(t)}} \end{aligned} \tag{9}$$

During sudden increases in congestion, $h_n \to 1$, but if protocol fields are bounded the source will remain responsive, but understate congestion to the network, which is the safe way round.

With hindsight, the downstream path metric from resource j

$$\|{}^{*}{}_j^{n-1} m_{(t+T)} = 1 - \frac{\prod_{i=0}^{n-1}(1 - m_{i(t+T)})}{\prod_{i=0}^{j-1}(1 - m_{i(t+T)})}.$$

An efficient estimator for this metric is

$$\begin{aligned} \|_j^{n-1} m_{(t+T)} &= 1 - \frac{\prod_{i=0}^{n-1}(1 - m_{i(t)})}{\prod_{i=0}^{j-1}(1 - m_{i(t+T)})} \\ \text{From (8)} \quad &= 1 - \frac{1 - h_z}{1 - h_{0(t+T)}} \Big/ \frac{1 - h_{j(t+T)}}{1 - h_{0(t+T)}} \\ \text{From (9)} \quad &= 1 - \frac{1 - h_z}{1 - h_{j(t+T)}}. \end{aligned} \tag{10}$$

The prediction error $\|_j^{n-1} m_{(t+T)} - \|{}^{*}{}_j^{n-1} m_{(t+T)}$ is

$$\frac{\prod_{i=0}^{n-1}(1 - m_{i(t+T)}) - \prod_{i=0}^{n-1}(1 - m_{i(t)})}{\prod_{i=0}^{j-1}(1 - m_{i(t+T)})}. \tag{11}$$