A path-aware rate policer: design and comparative evaluation

Arnaud Jacquet
BT Research

Alessandro Salvatori
BT Research

Bob Briscoe
BT Research, UCL

Abstract

In the current Internet, congestion control is voluntary and not responding sufficiently to congestion is becoming a growing problem. Rate policers in the literature are based on the assumption of placement at a single bottleneck and a known minimum round trip time. We aim to characterise the limitations of these policers in many practical scenarios where we believe these assumptions break down. We present the design of a policer based on a novel feedback architecture that transcends these assumptions. The new arrangement places our policer at the interface with the sender. The sender is trapped into sending packets through the policer that honestly declare the congestion and round trip time of the whole downstream path. We compare the theoretical limits of these different classes of policers.

1 Introduction

Control of Internet congestion is based on the TCP algorithm in the sender's operating system. Whether an application developer uses it is a free choice, but the large majority do, at least for data transfer applications. However, TCP's response to congestion is far too bursty for interactive media applications. And even adaptive codecs become unusable below a minimum rate.

There is strong social pressure for all applications to respond voluntarily to congestion without going faster than TCP would if it used the same path (TCP-friendliness). But the co-operative consensus that has kept Internet congestion control stable seems to be breaking down [6]. It seems that more than social pressure is needed, as usage of unresponsive voice and video services is increasing.

It is in a network operator's interest to police the congestion response of its users. Otherwise free-riders reduce available capacity for the honest remainder. But it is hard for a network operator to police whether an application is responding to congestion as TCP should, because of the distributed nature of the problem. A policer is only effective if placed at the ingress, but congestion is only visible at the egress.

Congestion is most pressing in residential and wireless access networks, because the cost of alleviating congestion is high when new infrastructure must be dispersed widely. So where flows traverse a path with an access network at either end (eg. peer to peer file-sharing or interactive media), often both ends contribute to path congestion.

The required congestion response should depend on congestion accumulated all along the path. But congestion at the egress is not easily visible to the ingress network. And the ingress cannot rely on intercepting congestion feedback, because applications trying to avoid detection can encrypt it, lie about it, or just not send it. TCP's response should also depend on round trip time. But, it is hard to inspect a mixed aggregate of packet flows and establish the round trip time of each flow, especially if the application is encrypting transport layer sequence numbers.

A growing literature addresses the rate policing problem, starting with Floyd and Fall [4] and Stabilized RED (SRED [10]), then continuing with CHOKe [12], RED with Preference Dropping (RED-PD [8]), Least Recently Used RED (LRU-RED [13]), XCHOKe [3], and Approx. Fair Dropping (AFD [11])). These approaches have become increasingly sophisticated at minimising the state required to detect cheating flows. They look for prevalence of flows in the discards from the RED active queue management (AQM) algorithm [5], which is in common use in commercial networks. But they all rely on the common assumption that they are sited at a single bottleneck and that they can use a heuristic estimate of the minimum round trip time across all flows.

The aim of a policer is to apply a sanction, which it must only do if it is sufficiently certain it has identified a misbehaving flow. We will show that, unfortunately, the assumptions of a single bottleneck and a minimum round trip time mean that mild misbehaviour cannot be detected with certainty in most practical environments where these assumptions are invalid.

Commercial policers suach as the policers provided by Sandvine [15] and Riverhead [14] ensure that no flow exceeds a maximum rate, irrespective of the condition of each path being used through the rest of the network. Some of them can also use their knowledge of local congestion on the equipment itself - in which case they relate to the category of bottleneck-policers described above.

This paper attacks the rate policing problem from a different starting point. We build on the re-feedback framework [1], which incentivises senders to declare the characteristics of the path they are using in the headers of the packets they send into the network. We wish to establish whether a policer built on this framework would be sufficiently accurate to be able to apply sanctions to flows that would be able to 'fly under the radar' of policers.

We take an analytical approach, as we are concerned with comparing the limits of intrinsically different approaches:

1. path-specific policers,
2. bottleneck policers,
3. absolute throughput policers,

We first present the generic objective a throughput policer should aim for in order to identifying misbehaving flows that are too greedy for a given rate adaptation policy. We derive from that objective a simple design for a path-specific throughput policer. We present analysis and simulation results describing the performance of the path-specific policer in comparison to other proposed approaches. Finally we present improved variants for the design to improve the performance of the policer, and extend the applicability of its mechanism.

2 Objectives and requirements

First required is a definition of what characterises a misbehaving flow. Then the objective of the policer can be set. Finally we conclude this section by describing how the relevant path characterisation information can be obtained per packet with all necessary accuracy.

For the simplicity of the argument, we will first consider the policing problem in stationary network conditions by assuming that all packets of a flow follow the same path, and experience a constant roundtrip time and a constant congestion level on that path for the duration of the flow. We will also consider that all flows are saturated in payload data, so that TCP rate adaptation is the limiting factor for their throughput.

2.1 Characteristics of misbehaving flows

Literature abounds on the characterisation of the long-term throughput of a TCP connection complying with the most common flavours of the congestion control mechanism. The most generic formula was introduced by Mathis [9]. According to it, the path-specific expected throughput (in packets per second) ν^* for a flow can be written:

$$\nu^* = \frac{1}{\kappa T \sqrt{m}} \tag{1}$$

where T is the roundtrip of the flow, m is the end-to-end congestion level it experiences, and $\kappa = \sqrt{2/3}$

The general understanding for TCP-friendliness is that a data flow is TCP-friendly if it achieves the same average long-term throughput as a TCP flow would in the same network conditions. In that context, Mathis' formula has to be read: "the expectation of the long-term throughput of a flow is in inverse proportion to the product of its round-trip and the square root of the congestion level it sees on its path".

We can define a misbehaving flow as a flow that is not TCP-friendly: a misbehaving flow achieved a long-term throughput larger than TCP flow would in the same networks. Its packet throughput ν is higher than the expected fair throughput ν^* of a TCP flow in the same conditions:

$$\nu > \nu^* = \frac{1}{\kappa T \sqrt{m}} \tag{2}$$

What characterises a misbehaving flow is not its absolute throughput, but rather its throughput relative to the expected throughput of a compliant flow in the same path conditions.

It is therefore this relative throughput a policer should aim to monitor, which requires the knowledge, at the policing node, of the characteristics of the flow's end-to-end path. We call thie relative throughput the greediness α of the flow:

$$\alpha = \frac{\nu}{\nu^*} = \nu \kappa T \sqrt{m} \tag{3}$$

In stationary network conditions, behaving TCP flows are expected to exhibit a long-term greediness of $\alpha_{TCP} = 1$. Conversely, misbehaving flows will exceed their fair throughput and exhibit higher greediness, the more they misbehave.

2.2 Objective of a path-specific policer

The problem with absolute-throughput policers is that flows are policed according to their absolute throughput, irrespectively of the conditions of the path they follow.

This situation is illustrated in figure 1 which outlines the difference between the objective of an absolute-throughput policer (a) and a path-specific policer (b) by showing the categorisation of TCP flows in four different classes, depending on their operational point $(x, T\sqrt{m})$ as observed by the policing node.

Depending on its throughput x and the network conditions $T\sqrt{m}$ it experiences, a flow belongs to one of four fuzzy categories illustrated on the figure:

1. it has a high throughput while the network conditions are not good (high roundtrip or high congestion)
2. it has a reasonable throughput and the network conditions are favourable (low roundtrip or low congestion)
3. it has a reasonable throughput but the network conditions are awful (high roundtrip or high congestion)
4. it has a high throughput and the network conditions are very favourable (low roundtrip or low congestion)

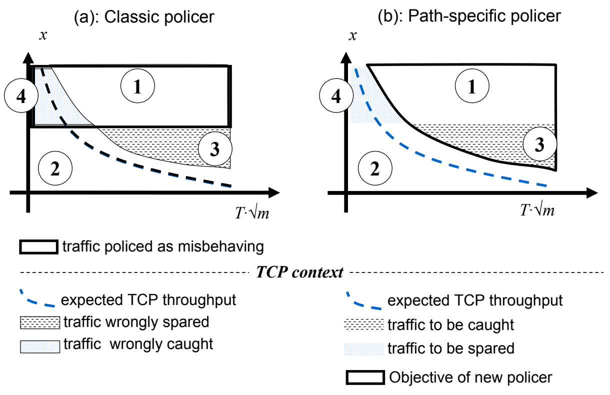

Figure 1: Comparison of the objective of a n absolute-throughput policer (a) and a path-specific policer (b)

An absolute-throughput policer defines as misbehaving flows which have a high throughput, and ignores path conditions in assessing the compliance of suspect flows. It will focus its sanction on categories -1- and -4- but will let through flows in category -3-.

In the context of TCP rate adaptation, it makes sense to sanction category -1- flows as they are not responsive to unfavourable path conditions. However it doesn't make sense to penalise flows in category -4-. Although they may have a high throughput, they only benefit from very favourable path conditions. On the other hand, flows in category -3- who may not have such a high throughput are not responsive enough to very bad network conditions, and should therefore be policed just as well as category -1- flows.

This sets the objective for a path-specific policer to identifying flows of categories -1- and -3- rather than categories -1- and -4-.

The distance between the TCP throughput equation and the region defining misbehaving flows is necessary to accommodate the dynamics of TCP rate adaptation. Although in the long-term, the average throughput is most likely to be close to the expected TCP throughput curve, the throughput of a compliant TCP flow may on occasions become significantly higher than the expected average, especially in periods when the network conditions are not stationary.

We must also stress at this point that we are not prescribing the use of policing mechanism to TCP flows. Instead we are considering a class of service being policed against a given rate adaptation policy. This way, any flow sending traffic to this class would be policed depending on the path conditions only, irrespectively of the protocol used, or declared, by the flow.

In particular this includes policing constant-bit rate (CBR) traffic. Whether CBR traffic is acceptable in a class of traffic monitored by a path-specific policer, will depend on the conditions of the network. When there is barely no congestion, a CBR flow, even a high-troughput one, could be perfectly legitimate. On the other hand when congestion increases too much, most CBR flows, apart from the lower-throughput ones, will become illegitimate.

Monitoring a class of traffic with a path-specific policer, which is sensitive to the conditions of the network will therefore give incentives for end-systems to schedule high-throughput CBR session during the the troughs of congestion rather than the peaks.

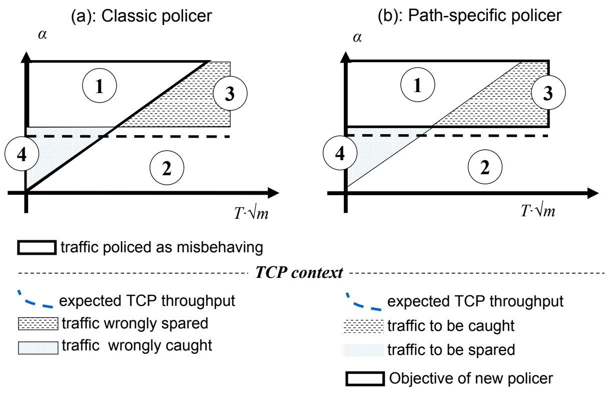

Figure 2: Transposition of the objective in the greediness domain

By having defined greediness, we can illustrate the problem more clearly in figure 2. The discriminating factor between the absolute-troughput policer and the path-specific policer is their greediness. As outline on the right, the objective of the path-specific policer, as highlighted by the rectangle on the right, aligns with detecting flows whose greediness is significantly higher than that of TCP. On the other hand the objective of the absolute-throughput policer, outlined by the triangle shows the mismatch between the detection mechanism of the absolute throughput policer and the definition of misbehaving flows.

The objective of the policer is to identify whether the flow is behaving ($\alpha \leq \alpha_{TCP}$) or misbehaving ($\alpha > \alpha_{TCP}$) and to segregate their subsequent treatment based on that information.

That decision can be made by comparing the greediness with a constant threshold α^* that is valid for all flows, whatever path conditions they experience.

In the context of control theory, this objective can be interpreted as Binary Hypothesis Testing. The null hypothesis H_0 is that the flow is compliant, the alternate hypothesis H_1 is that it is misbehaving.

$$\begin{aligned} H_0: &\quad \alpha \leq \alpha_{TCP} \\ H_1: &\quad \alpha > \alpha_{TCP} \end{aligned}$$

We further describe the performance requirements for the implementation of the policer by defining two performance objectives that should be applicable in a dynamic network environment:

A) for robustness: over an observation τ^*, the probability that a compliant flow be detected as misbehaving should remain smaller than ϵ_A

B) for responsiveness: over an observation τ^*, the probability that a misbehaving flow of greediness α^* be undetected as misbehaving should remain smaller than ϵ_B

In other words, if we call $D(\tau)$ the decision of the policer for flow, we can rewrite the two objectives:

$$\begin{aligned} A) &\quad Pr[D(\tau^*) = H_1 | \alpha = \alpha_{TCP}] < \epsilon_A \\ B) &\quad Pr[D(\tau^*) = H_0 | \alpha = \alpha^*] < \epsilon_B \end{aligned}$$

These relations are central to defining the performance of the policer presented in Section 4.1 on comparative analysis.

2.3 Requirements on the information framework

The objective of the policer is to detect flows of high-greediness. This relies on being able to characterise the greediness of each single flow, which requires the policer to estimate the expected fair throughput of the flow, and therefore to know the path conditions for each flow - that is its roundtrip time, and the end-to-end congestion level it experiences.

In current IP networks, very few nodes do possess the necessary information to perform the policing. The source and the destination of the flow are able to monitor both the end-to-end congestion and the roundtrip. In the network, only the last node on the path could have a chance to monitor the end-to-end congestion, and no router knows the the value of the roundtrip. However the most useful location for a rate policer would be the first node in the network, so that any excessively greedy flow is constrained upfront.

In order to resolve this problem we need to use a signalling framework that provides the end-to-end information near the source rather than near the destination. This feature is achieved by the re-feedback framework [1] where every sender has to state in every packet it sends an estimate for the downstream metrics of the path to the destination; this is effectively the end-to-end congestion level and half the roundtrip. Intermediate nodes update the stated values by removing their contribution. If network conditions are stationary, at every node in the network, this mechanism makes available the congestion and the delay on the downstream path of the flows traversing it.

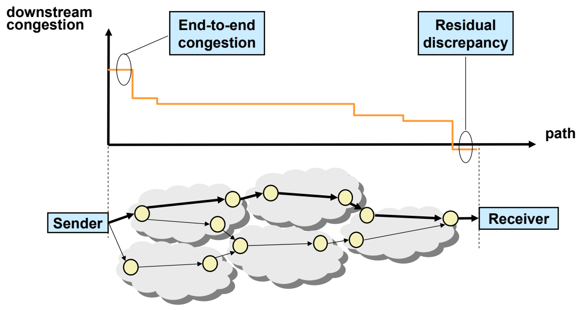

Figure 3: Re-feedback framework - sources declare their estimate of the end-to-end metric while forwarding nodes remove their contribution from that value - this ensures that the residual discrepancy at the end of the path is one and provides an estimate of the metric for the downstream path at every point along the path

At the first node along the path, downstream metrics will coincide with the values stated by the source. At the last node, if network conditions are stationary, the values of the fields will hit zero. The sender feeds back any discrepancy so the source can tune the values to which it sets the fields of the metrics, in order to track the dynamic evolution of the path conditions. Should there remain a systematic negative bias for a flow in particular, the last hop router is entitled to suspect the source of lying about the path characterisation, and take prescriptive action (e.g. dropping packets) on its traffic until the flow declares truthful information again.

The punishment at the last hop must be heavy enough to completely remove the marginal throughput gain a policed source would get by declaring path conditions more favourable than they actually are. This way, the most profitable strategy for a punished sender is again to declare truthfully the characteristics of its downstream path. Briscoe et al. describe how truthfulness enforcement can be achieved by ensuring that the average residual discrepancy of the re-feedback fields is null. In situations in which that is not the case, the information policer will start sanctioning traffic with the most incompliant residual discrepancy, until a null average is reached again.

As the re-feedback incentive framework provides adequate guarantees that path characterisation metrics stated in packet headers are truthful, it is possible for the path-specific throughput policer to rely on them to ensure that data sources conform to the rate adaptation discipline they have contracted to.

In the remainder of this document we will therefore assume that robust estimates are available per packet for the roundtrip of the flow and the end-to-end congestion level it experience on its path.

3 Basic policer design

A complete policer examines each flow during a process similar to a judicial process. Different step of the investigation have to be done in order:

1. Suspect identification
2. Suspect conviction
3. Differentiated sanction
4. Eventual redemption.

The process needs to provide guarantees both in terms of robustness and responsiveness. While guilty elements should be detected and sanctioned as quickly as possible, there should also be strong guarantees that compliant elements would not be declared as misbehaving by mistake.

In all cases, the suspect identification is required. In this paper we describe the implementation of a passive policer for which the only step implemented is the identification of suspect flows. The extension to an active policer is outlined in Section 4.2

We use a similar mechanism to a token-bucket policer for the implementation of the path-specific policer, by monitoring the cumulative discrepancy of the greediness, rather than that of the throughput. The process is described in Figure 4.

A classic token bucket ensures that a flow sends data at the throughput it contracted, on average. This is achieved by monitoring the cumulative discrepancy between the throughput of the flow and the contracted throughput. The finite depth of the bucket allows minimum burstiness but limits the credit the source can get on its throughput.

The policing agent maintains a token-bucket per flow. When a packet arrives, the token bucket corresponding to the flow is updated if one exists - otherwise a new token bucket is created.

The token bucket fills at a constant rate, proportional to the maximum acceptable greediness α_{TCP}.

Tokens are also consumed whenever a packet arrives: the relevant path characterisation information is extracted from the packet to compute the instantaneous greediness of the flow, which determines the amount of tokens to be consumed form the bucket: $\kappa T_i \sqrt{m_i}$ where T_i and m_i are the path characterisation metrics extracted from packet i.

If at any point the bucket empties, the flow is added to a suspect list.

So in summary, when a packet arrives, the token level is adjusted, and the packet is categorised as misbehaving depending on the resulting state of the bucket. If the update and the test are done in sequence, the policer will be stricter than if they are done in parallel (in which case the decision is taken on the ground of older information), however the last option may be preferred if the expression of the fair throughput (for instance a square root extraction in TCP's case) requires too many computational resources.

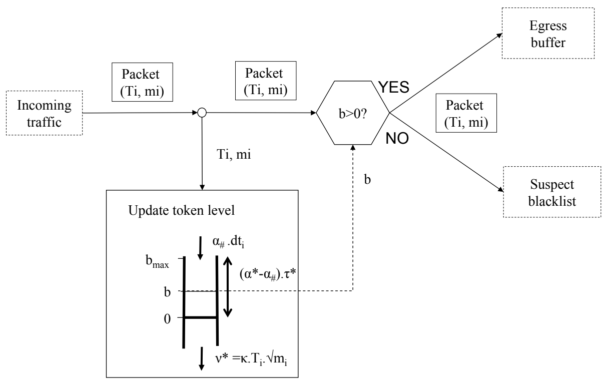

Figure 4: Implementation design of the basic path-specific policer

The resulting token level is $b_i = b_0 + (\alpha_{TCP} - \alpha_i)t_i$ where t_i and α_i are the duration and the greediness of the flow to date.

Proof - Whenever a packet arrives, the token level is adjusted so that

$$b_{i+1} = b_i + \alpha_{TCP} dt_i - \kappa T_i \sqrt{m_i}$$

where b_i is the token level after packet i is processed, dt_i is the lapse of time since the last processed packet for that flow, T_i and m_i are the path characterisation metrics associated with the packet. It follows that

$$b_i = b_0 + \alpha_{TCP} t_i - \sum \frac{1}{\nu_i^*}$$

We derive the value of the sum

$$\sum \frac{1}{\nu_i^*} = \left(\frac{i}{t_i}\right)\left(\frac{1}{i}\sum \frac{1}{\nu_i^*}\right) t_i = \nu_i \frac{1}{\nu_{est}^*(i)} t_i = \alpha_i t_i$$

by defining $\frac{1}{\nu_{est}^*(i)} = \frac{\sum 1/\nu_i^*}{i}$,

The expression of the token level follows. **- End of proof**

In the process, we have shown that the estimate of expected fair throughput performed by the token bucket is the harmonic average of the instantaneous values derived from the path characterisation metrics associated with individual packets.

Figure 5 shows the trend of the token level for different types of flows with constant greediness. On average, a compliant flow is expected to exhibit a stable token level. An idle flow will see the token bucket fill up as they are not consumed by the traffic of the flow. On the other hand, greedy flows will tend to exhibit a decreasing token level until the bucket is empty. The more greedy the flow, the more quickly it will empty the bucket and be listed as suspect.

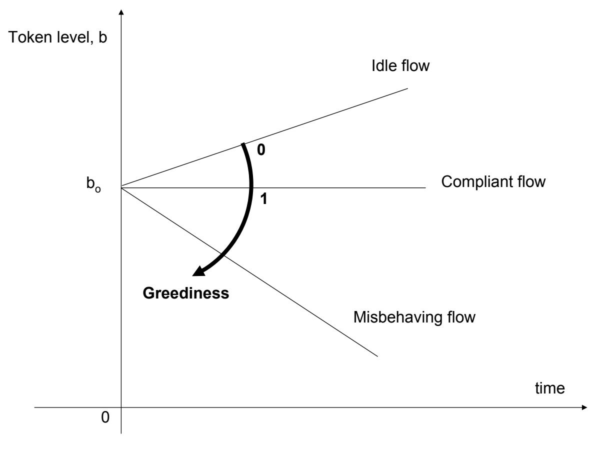

Figure 5: Trend of the evolution of the token level with time

4 Experiments

4.1 Comparative analysis

Policers can be classified in three classes:

1. absolute-throughput policers who categorise as misbehaving high-throughput flows

2. bottleneck policers, such as the penalty box approach [4] , who examine the drop history at bottlenecks and categorise as misbehaving high-throughput flows through those bottlenecks

3. path-specific policers, who categorise high-greediness flows as misbehaving, therefore taking into consideration the path conditions in order to reach a verdict

Common objective - In order to compare the performance of the three types of policers, we need to design them in order to achieve the same objective. We therefore chose the following design requirement: a misbehaving flow of constant benchmark greediness α^* should be detected within a lapse of time τ^*.

Performance criteria - We also define three performance criteria against which to compare the three policer designs. These are in order of importance:

1. for robustness, there should be as little compliant flows identified as misbehaving

2. for responsiveness, it should take as little time as possible to detect misbehaving flows more greedy than the benchmark

3. for responsiveness, it should take as little time as possible to detect misbehaving flows less greedy than the benchmark

Path-specific policer - For the path-specific policer, the objective defines the bucket depth:

$$b_0 = (\alpha^* - \alpha_{TCP})\tau^*$$

Then for any flow $b_0 = (\alpha - \alpha_{TCP})\tau$, which gives the expression of the detection time:

$$\tau = \tau^* \frac{\alpha^* - \alpha_{TCP}}{\alpha - \alpha_{TCP}} \tag{4}$$

Penalty box policer - For the penalty box, misbehaving flows are the flows that appear most in the drop history of congested routers. Therefore the objective defines the ceiling value in the drop history:

$$h_{max} = (\alpha^* \kappa T^* \sqrt{m_{local}}) m_{local} \tau^*$$

which requires to assume that there is a single bottleneck and that flows will have a typical roundtrip T^*.

Then for any flow, we have

$$h_{max} = \frac{\alpha}{\kappa T \sqrt{m}} m_{local} \tau$$

which gives the expression for the detection time:

$$\tau = \tau^* \frac{\alpha^*}{\alpha} \frac{T}{T*} \sqrt{\frac{m}{m_{local}}} \tag{5}$$

In this framework, a slow is policed not once but as many times as there are congested routers on its path. The penalty box policer on the most congested router will detect the flow in the shortest time. Therefore the outcome of all the penalty box on the path will be the same (at least in first approximation) as if there was only one congested node on the most: the most congested of them all.

If we consider several flows with the same end-to-end congestion, and N_b bottlenecks of equal magnitude, we get in first approximation that $m \approx N_b m_{local}$, and hence:

$$\tau = \tau^* \frac{T}{T^*} \sqrt{N_b} \tag{6}$$

Absolute throughput policer - For the absolute throughput policer, misbehaving flows are those with the highest throughput. Therefore the objective defines the ceiling value in terms of absolute throughput:

$$\nu_{max} = \frac{\alpha^*}{\kappa T^* \sqrt{m^*}} \tau^*$$

which requires to assume that flows will have a typical roundtrip T^* and experience a typical congestion level m^*.

Then for any flow

$$\nu_{max} = \frac{\alpha}{\kappa T \sqrt{m}} \tau$$

which gives the expression of the detection time:

$$\tau = \tau^* \frac{\alpha^*}{\alpha} \frac{T}{T^*} \sqrt{\frac{m}{m^*}} \tag{7}$$

The performance of an absolute-throughput policer assuming a typical congestion level m^* is therefore the same as that of a penalty box experiencing the same level of local congestion: $m^* = m_{local}$. Hence, we will not detail further the performance of the volume policer.

Figure 6 shows the time it takes to classify as misbehaving flows of three different types depending on their greediness, for $\alpha^* = 4$ and $T^* = 1s$: a) typical-roundtrip single-bottleneck flows, b) typical-roundtrip flows experiencing the same end-to-end congestion level, but through four bottlenecks of equal magnitude rather than one, c) single-bottleneck flows with a roundtrip $T^*/4$.

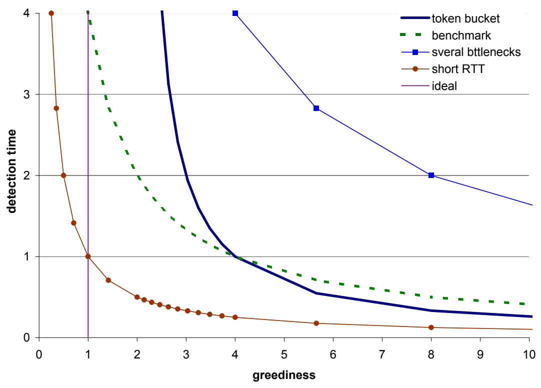

Figure 6: Comparative performance of a penalty box and a path-specific policer. When both are tuned to achieve a given objective, the penalty box becomes too harsh on flows with short roundtrips and too lax on flows crossing multiple bottlenecks

The ideal performance of a throughput policer is reflected by the vertical line for greediness $\alpha = 1$:

1. for robustness, it should take as long as possible to classify a compliant flow ($\alpha < 1$) as misbehaving,

2. for responsiveness, it should take as little time as possible to detect a misbehaving flow ($\alpha > 1$).

While the performance of the path-specific policer (plain line) is the same for the three types of flows considered, the penalty box will achieve different performance to different flows: it is quicker to classify a single-bottleneck, short-round trip flow (squares) as misbehaving than a single-bottleneck, average-round trip flow (dashes), and even more so than a multiple-bottleneck, average-round trip flow (circles).

The penalty box is harsher on short-roundtrip flows. On the good side, misbehaving short-roundtrip flows will be detected more quickly. However, this comes at a cost: short-roundtrip compliant flows risk of being classified as misbehaving much more quickly too, which is detrimental to the robustness objective.

On the other hand, flows crossing several bottleneck get a preferential treatment to single-bottleneck flows experiencing the same end-to-end congestion level.

Now let's consider the performance of the path-specific policer and the penalty box against the three performance criteria.

- Criterion 1 - robustness - The path-specific policer is completely robust for flows of constant greediness: no compliant flow can be classified as misbehaving given they will consume tokens less quickly then they are generated.

The penalty box on the other hand needs an extra mechanism to improve its robustness - although multiple-bottleneck flows are not as exposed to the threat of being wrongly classified as suspect flows. If the penalty box is to be robust for typical roundtrip single bottleneck flows, the penalty box need to flush the drop history every $T^*/\alpha^* = 4s$. Still, short-roundtrip single-bottleneck flows will remain likely to be classified as misbehaving, unless they are essentially idle.

In conclusion, the path-specific policer is more robust than the penalty box.

- Criterion 2 - responsiveness for misbehaving flows more greedy than the benchmark - When $\alpha > \alpha^*$, the path-specific policer is fairly less responsive than the penalty box for short-roundtrip flows, slightly more responsive for typical flows and significantly more responsive for multiple-bottleneck flows.

Again, the path-specific policer performs overall better than the penalty box.

- Criterion 3 - responsiveness for misbehaving flows less greedy than the benchmark - When $\alpha_{TCP} < \alpha < \alpha^*$, the path-specific policer becomes significantly less responsive than the penalty box for short-roundtrip flows, slightly less responsive for typical flows and remains fairly more responsive for multiple-bottleneck flows.

In this context, the penalty box performs overall better than the path-specific policer.

- Overall evaluation - The path-specific policer performs better in terms of robustness, and responsiveness to very greedy flows, while it is not as responsive for flow only slightly more greedy than would be fair.

Improving the robustness of the penalty box will be to the detriment of its responsiveness.

For the path-specific policer, it is actually possible to further increase the robustness and the responsiveness to very misbehaving flows by increasing the filling rate of tokens and shortening the bucket depth - see Section 5.2. The counterpart is that slightly misbehaving flows will most likely remain undetected.

4.2 Performance evaluation in non-stationary settings

In building an active policer, the most difficult step is the suspect detection that we have described in this paper in the context of a passive policer.

We are also investigating the possible options in building an active policer that will not only detect misbehaving flows, but also reliably sanction them. Options in that case range from demoting the traffic to a lower class of service, to increasing the congestion signal, to dropping the traffic altogether. The most useful sanction will reinforce the incentives framework so that the best policy for all sources is to comply with the required rate adaptation.

Another design choice is whether a convicted flow may redeem itself or not after a subsequent time of exemplary compliance.

Probation One possible improvement is depicted in figure 7: when the token level goes below a first warning threshold, the policer could mark packets instead of dropping them. This way, the sender gets a chance to back off in response to increased congestion before its traffic gets dropped. If the source remains unresponsive, the flow's token level will go further down until packets are effectively dropped by the policer.

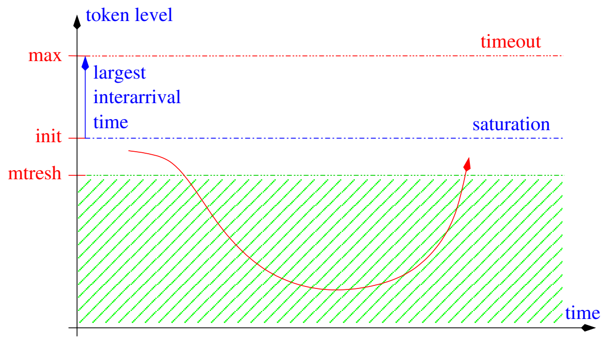

Figure 7: Policer dynamics with *ECN* warnings. Compliant flows will slow down when their token level goes below the warning threshold *mtresh*$_{-}$.

The figure also shows a timeout threshold at which point garbage collection would be operated on the token bucket. This level woulld be reached if a flow remained idle for the duration of a timeout period, from the time when the bucket became saturated. This timeout should be long enough so that a user doesn't gain from alternating heavy bursts and complete idleness.

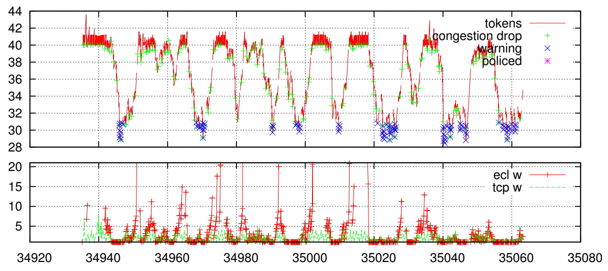

Figure 8: The token bucket for a non-greedy legacy flow.

In the topmost part of Figure 8 we plot the token level for a legacy flow that performs binary adaptation. Green ’x’ crosses show losses due to congestion, while blue ’+’ crosses show the warnings emitted when the token level falls under the warning threshold, here 30.0 tokens.

The lower part of figure 8 shows the window allowed by a multibit downstream metric and the one a TCP flow would have adopted. The values on the x-axis represent time in seconds since the start of the simulation.

The sender reacts to warnings and backs off. This means that the performance of legacy flows depend on the warning level, which may allow more or less variability in the sender’s rate.

By narrowing the difference between the maximum amount of tokens and the warning threshold , the policer would force shorter and shorter sawtooths, limiting the performances of binary rate adaptation in reaching high speeds and hence giving incentives to move to a multi-bit rate adaptation framework relying for instance on the Explicit Congestion Level, *ECL*(), provided by re-feedback.

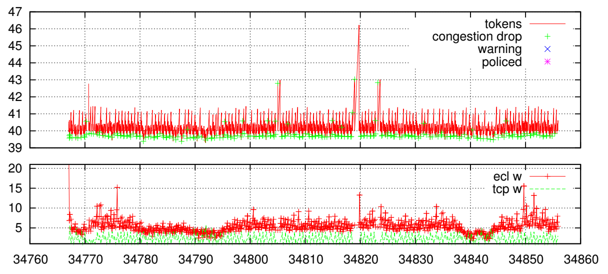

Figure 9: *ECL*-aware sender.

Figure 9 illustrate hoe well a source reacting to a multibit congestion signal can react. The reason why the sender never consumes more tokens than those that arrive is that it bounds the congestion window within the maximum allowed by the multi-bit congestion metric. A comparison of the adopted window with the one used by the legacy TCP sender also shows a much higher regularity. By narrowing the difference between the maximum amount of tokens and the warning area, the service attained by legacy senders will only get worse. This provides a potential incentive for a transition to multi-bit rate adaptation.

Firm sanction As in a token bucket, drawing tokens only when forwarding a packet should be enough, because in any case a sender would not be able to attain a higher goodput than it has a right to. But this would mean that, by pushing a higher throughput than it has a right to, it would get exactly the goodput it deserves. Hence not drawing any token from the bucket when we drop packets would give an incentive not to behave and instead leave to the policer the burden of regulating the flow’s traffic. We instead want those senders that have the necessary capability to perform rate adaptation on their own and hence we have to incentivise it. For this reason, we behave in a stricter way than a plain token bucket.

Drawing at most as many tokens are available in the token bucket does better than not drawing any token when the packet is not forwarded. Say there are 0.53 tokens and a packet arrives that requires 0.72 tokens to be forwarded. We introduce a negative saturation to 0, so that the resulting -0.19 will not allow to forward the packet and immediately after this decision the token counter will be reset to 0.0. Clearly enough, in this case the best goodput is achieved by regulating the flow’s traffic at the source, and pushing anything more than deserved will see the goodput decrease instead of stay steadily at the maximum rate possible.

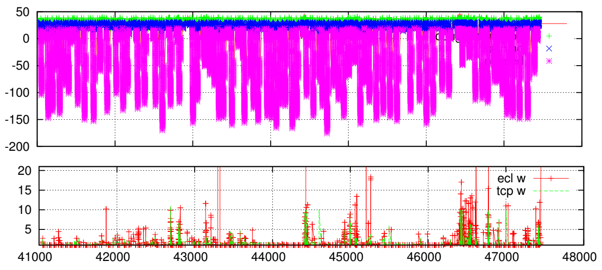

Figure 10: A heavily misbehaving flow.

Nevertheless, there are reasons that suggest that allowing the token level to go negative is more conservative. One of those reasons is the fact that otherwise packets with zero metrics would always be allowed to pass through. But, if only packets with zero metrics can reach the dropper, then the average seen by the dropper would decrease and the dropper would catch those packets.

To be even stricter against misbehaving flows, we chose to have an additional parameter, that amplifies the token drain rate once the token level has become negative. Let's call this parameter $stricter\_policing\_$ and suppose that it is equal to 2.0. Upon a packet arrival we drain $1/x(p, T)$ tokens. Once these have been drawn we check if the token level is negative. If the level is effectively negative, then we drain again $stricter\_policing\_$ times $1/x(p, T)$, which with the default value of 2.0 we chose for the parameter has that packet contribute three times as much as it should have done.

For this reason, once identified as misbehaving flows, packets should arrive more than three times slower than the fair rate in order to re-gain a positive amount of tokens. Plot 10 shows the very drastic effects of this measure on a highly misbehaving flow. Transferring 2MB of data takes circa 2 hours. Compare with the previous two cases, the one of the legacy sender (image 8), for which transferring the very same amount of data took 2 minutes, and the one of the smoothened rate adaptation sender (image 9), for which it took one and a half minutes.

effect on the throughput In plot 11 we show the result of an experience run at a moderate load. On the x-axis we report the sender's minimum window size in segments, those senders who have a minimum window size of 0 being the non-greedy ones and representing the 95% of all the flows. A sender that choses a minimum window of $cwnd_{min}$ segments will never adopt a congestion window lower than $cwnd_{min}$. On the y-axis, we represent the average attained throughput in bytes per second. Even with the policer active, flows with a minimum window of 1 segment have better performances than non-greedy flows. This might be reasonably due to the saturation in the representation of the downstream congestion, as a congestion metric saturation greater than -1.5 does not allow to police rate adaptation to a congestion level higher than 60%, that would correspond to a steady state average window of 1 single segment.

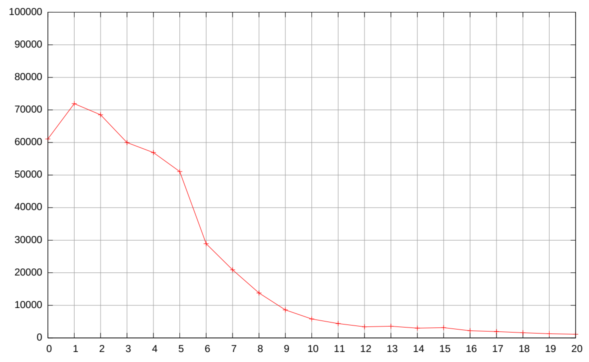

Figure 11: The average goodput with an ingress policer.

The experiment clearly shows that when a flow misbehaves, the achieved throughput starts decreasing, in such a way that the best strategy for a sender is to behave. This because by misbehaving, as the clear monotonic trend shown in the plot underlines, a sender would get a lower goodput than by behaving.

4.3 Discussion

The experiments realised on the implementation of the policer validate the performance of the mechanism in the context of a stationary network. The performance of the policing agent in a non-stationary environment is more arguable, but our experiment show that it is possible to make it work by making the policer mildly active in reinforcing the congestion signal.

Our comparison with the main other policing approaches shows that the path-specific policer we propose is more robust because it limits the number of compliant flows wrongly declared misbehaving. The evaluation of that aspect of the performance is missing for a non-stationary network, but we show in Section5.2 how it is possible to reinforce the robustness guarantee. Alternative policing proposals cannot be consolidated for robustness without a deterioration of performance in terms of responsiveness, even in a stationary network context.

The responsiveness of the path-specific policer we propose is also better than the responsiveness of other proposed approaches for traffic more greedy than the benchmark given by the design requirement. The path-specific policer is less responsive for only slightly greedy traffic but such traffic is less of a threat than more greedy traffic.

The performance improvement given by the path-specific policer must however be tempered by a number of caveats which we address in the next section: - the implementation design presented in section 3 requires state per flow which limits the scalability of the system. We present below a sampling variant on the basic design that alleviates the state requirements for short-lived compliant flows - the policing mechanism require the availability of up-to-date information on the end-to-end path of the flow, which are not available in the current context of TCP/IP standards. We outline below an incremental deployment option that enables the operation of the policer in the current Internet.

5 Extensions and further work

5.1 Policing for other rate adaptation disciplines

In today's Internet, most flows are expected to be TCP-compliant and their long-term throughput would never be expected to exceed much their expected fair throughput for too long.

In the future, other rate adaptation policies may become as common. An example of such an alternative rate adaptation defined by Kelly [7] assumes users with a constant willingness-to-pay, in a context where a fixed price may be charged for each congestion mark detected in the flow.

In that case the fair throughput of a flow would be define with respect to a constant-willingness-to-pay rate adaptation policy: $\nu^*_{Kelly} = \frac{w^*}{m}$ where w^* is a benchmark willingness-to-pay.

5.2 Optimising the performance of the policer

It is possible to improve the robustness of the path-specific policer by increasing the token filling rate. For the same design objective, a higher token fill rate will require a smaller bucket depth. The impact of the performance shown in Figure 12. The improvement in terms of robustness goes in pair with an improvement in terms of responsiveness for the detection of flows more greedy than the benchmark. The downside is that it becomes much harder to detect quickly - if at all - misbehaving flows less greedy than the benchmark.

Figure 12 shows on a logarithmic scale the detection time with respect to the greediness, for a basic policer, and a policer with ahigher-filling rate.

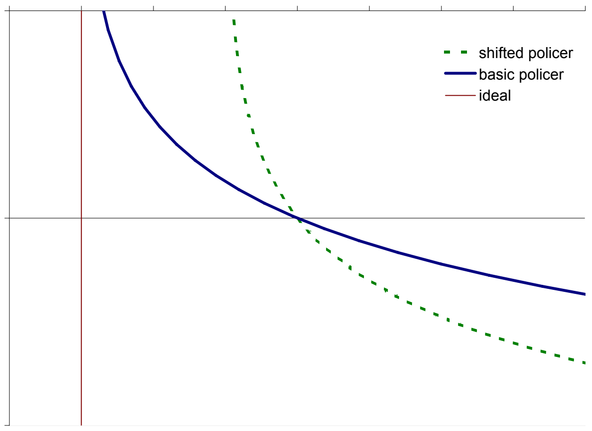

Figure 12: Increasing the token filling rate improves robustness and overall responsiveness - The responsiveness improvement is focused on flows greedier than the benchmark, while it deteriotates for less greedy flows policer

5.3 Controlling the state requirements

A drawback of the path-specific policer is that it requires state per flow. However it is possible to restrict this requirement, so that very-short lived flows don't require state, by monitoring greediness on a sampled subset of the traffic. When a packet arrives, its instantaneous expected fair throughput ν^* is computed as previously. The token level for that flow is only updated with a probability $p = \frac{1}{\nu^*} = \kappa T_i \sqrt{m_i}$, in which case one token exactly is consumed from the bucket, as illustrated in Figure 13.

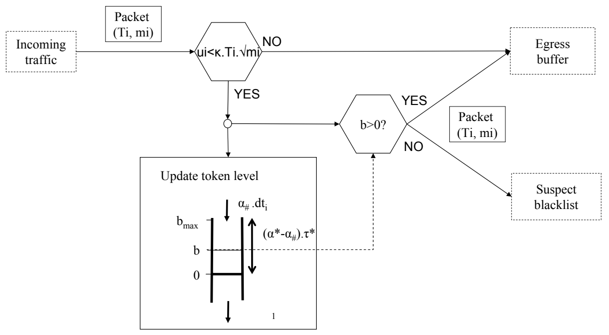

Figure 13: A sampling policer will reduce the state requirements of the policer, by avoiding to create a token bucket monitoring short-lived compliant flows

More significantly a token bucket for a new flow is not instantiated until a packet of that flow is sampled. Given the throughput of the flow ν, the sample rate will be $\nu p = \frac{\nu}{\nu^*} = \alpha$. In average, the first sample of a flow will happen when $\nu.p.\tau_s = \alpha.\tau_s = 1$.

Therefore state is unlikely to be required for short-lived compliant flows, for which $\tau_{life} > \tau_s = 1/\alpha$.

5.4 Performing path-specific policing in the current Internet

The analysis in this paper has been carried out by assuming that the congestion level and the roundtrip are readily available to the policer with all necessary accuracy. However, in today's Internet, the end-to-end congestion level is only available as a binary signal per packet, in the form of ECN. Estimating the roundtrip requires an accurate estimate of the downstream delay. While the TTL field was initially meant to be decremented by each router by a number of time units equivalent to the transmission time to the next hop, it is not used in this way.

Work is in progress to design a re-feedback framework [1] that may be incrementally deployed on the Internet [2] . We believe it is possible to take a similar policing approach in that case as the approach presented here. Further extensive experiments will be necessary to assess how the more limited information available in that case affects the performance of the policing agent.

6 Conclusion

At the moment, there is little means to assess the compliance of a flow - or an end-system in the longer term - to the rate adaptation policy they have contracted to. We have shown that proposed approaches are not completely robust as they allow for high throughput flows on short uncongested paths to be categorised as misbehaving.

In this paper, we have proposed a novel throughput policing mechanism that decides whether a flow is compliant with respect to the characterisation information of the path of the flow. We have shown that when such a path-specific policer is designed to achieve the same responsiveness objective as other policing mechanisms, it is more robust and it is more responsive to the most greedy flow.

Such a performance improvement doesn't come at no cost. The basic design requires per-flow states, but it is possible to alleviate this state requirement by sampling the information on which the policer monitors the compliance of every flow so that maintained state is focused on the most contravening flows: long-lived and very greedy.

Another complication is the need for the policer to obtain information on the end-to-end characterisation of the path. This information would be available in the context of a re-feedback framework, and there exist incremental deployment options that ensure the path-specific policer can operate with the limited information it gets from the current Internet context.

The path-specific policer we have proposed in this document is a further step to develop an accountability infrastructure for the Internet, within the re-feedack framework. Indeed, the path-specific policer allows network operators to assess that end-systems adjust their bandwidth usage to accommodate the dynamic evolution of the conditions of the path of their data, which mean they will be able to cope with a broader choice of rate adaptation disciplines to suit the needs of future applications.

References

[1] Briscoe, B., Jacquet, A., Cairano-Gilfedder, C. D., Salvatori, A., Soppera, A., and Koyabe, M. Policing congestion response in an internetwork using re-feedback. *Proc. ACM SIGCOMM'05, CCR 35*, 4 (Sept. 2005).

[2] Briscoe, B., Jacquet, A., and Salvatori, A. Re-ecn: Adding accountability for causing congestion to tcp/ip. (Work in progress).

[3] Chhabra, P., John, A., Saran, H., and Shorey, R. XCHOKe: Malicious source control for congestion avoidance at Internet gateways. In *Proc. IEEE International Conference on Network Protocols (ICNP'02)* (URL: `http://www.jungle.bt.co.uk/projects/2020comms/refs/icnp/02/276.pdf`, Nov. 2002), IEEE. (rejected).

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

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

[6] Floyd, S., and Kempf, J. IAB Concerns Regarding Congestion Control for Voice Traffic in the Internet. RFC 3714 (Informational), Mar. 2004.

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

[8] Mahajan, R., Floyd, S., and Wetheral, D. Controlling high-bandwidth flows at congested router. In *Proc. IEEE International Conference on Network Protocols (ICNP'01)* (URL: `http://citeseer.nj.nec.com/545435.html`, 2001).

[9] Mathis, M., Semke, J., and Mahdavi, J. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review 27*, 3 (1997).

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

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

[12] Pan, R., Prabhakar, B., and Psounis, K. CHOKe, A stateless active queue management scheme for approximating fair bandwidth allocation. In *Proc. IEEE Conf. on Computer Comm's (Infocom'00)* (Mar. 2000), IEEE.

[13] Reddy, S. A. L. N. LRU-RED: An active queue management scheme to contain high bandwidth flows at congested routers. In *Proc Globecomm'01* (URL: `http://citeseer.nj.nec.com/473674.html`, Nov. 2001).

[14] Riverhead. Web page URL: `http://www.riverhead.com/`, 2004+.

[15] Sandvine. Web page URL: `http://www.sandvine.com/`, 2004+.