Traffic Policing in the Internet

Yuchung Cheng, Neal Cardwell

Google

IETF 97 maprg. Nov 2016
Policing on YouTube videos
Token bucket traffic policer

Tokens filled at 1Mbps up to the bucket size (== burst)

Packets arriving at 3Mbps

Packet forwarded if a token is available otherwise dropped
**Detection Algorithm**

1. **Find the policing rate**
   - Use measured throughput between an early and late loss as estimate

2. **Match performance to expected policing behavior**
   - Everything above the policing rate gets dropped
   - (Almost) nothing below the policing rate gets dropped
Validation 2: Live Traffic

- Observed only few policing rates in ISP deep dives
  - ISPs enforce a limited set of data plans
- Confirmed that per ISP policing rates cluster around a few values across the whole dataset
- And: Observed no consistency across flows without policing
Congestion Looks Similar to Policing!

- Packets are usually dropped when a router’s buffer is already full.
- Buffer fills → queuing delay increases.
- Use inflated latency as signal that loss is not caused by a policer.
Analysis of Traffic Policing on YouTube

- 1 week in September 2015
- 0.8B HTTP queries
- Over 28K ASes
- Servers running Linux TCP, Cubic, PRR, RACK, fq/pacing
- New algorithm to detect policed connections using packet traces

An Internet-Wide Analysis of Traffic Policing
Flach, Papageorge, Terzis, Pedrosa, Cheng, Karim, Katz-Bassett, Govindan. SIGCOMM (2016)
Policing rates are often static
Policing rate is often less than half of burst rate
Policing causes heavy losses

<table>
<thead>
<tr>
<th>Region</th>
<th>Policed segments (overall)</th>
<th>Policed (lossy conns)</th>
<th>Loss Rate (policed)</th>
<th>Loss (non-policed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Africa</td>
<td>1.3%</td>
<td>6.2%</td>
<td>27.5%</td>
<td>4.1%</td>
</tr>
<tr>
<td>Asia</td>
<td>1.3%</td>
<td>6.6%</td>
<td>24.9%</td>
<td>2.9%</td>
</tr>
<tr>
<td>Europe</td>
<td>0.7%</td>
<td>5.0%</td>
<td>20.4%</td>
<td>1.3%</td>
</tr>
<tr>
<td>N. America</td>
<td>0.2%</td>
<td>2.6%</td>
<td>22.5%</td>
<td>1.0%</td>
</tr>
<tr>
<td>S. America</td>
<td>0.7%</td>
<td>4.1%</td>
<td>22.8%</td>
<td>2.3%</td>
</tr>
</tbody>
</table>
BBR congestion control

Bottleneck Bandwidth and Round-trip propagation time

Seeks high throughput with small queues by probing BW and RTT sequentially

Explicit model of the bottleneck

Track max BW and min RTT on each ACK using windowed max-min filters

Pace near BW (+−25%) to keep tput high but queue low

On loss: reduce to current delivery rate but reprobe quickly

How BBR models policers

BBR explicitly models the presence and throughput of policers

Long-term sampling intervals (4 - 16 round trips)

  Starting and ending with packet loss (to try to measure empty token buckets)
  
  Record average throughput and packet loss rates over each interval

If two consecutive intervals with

  loss rates >= 20% && throughputs within 12.5% or 4 Kbps of each other) Then:

  Estimated policed rate is average of the rates from each interval

  Send at <= estimated policed rate for 48 round trips
BBR: policer modeling in action

Two sampling intervals with high loss rate, consistent goodput => estimate that flow is policed

Throughput allowed by policer

BBR Transmission rate matches policing rate
BBR: a policed YouTube trace (major US cellular ISP)

Initially detect policer

Periodically re-probe available rate, at an interval chosen by the congestion control
Conclusion

● YouTube analysis indicates prevalent traffic policing
  ○ Often uses deep token bucket
  ○ More common in developing regions deploys more
  ○ TCP bursts initially then suffers severe losses
  ○ Interact badly with video chunking delivery and rate adaptation

● Promising protocol changes under testing
  ○ BBR congestion control detects and models policer
  ○ RACK loss recovery to detect lost retransmit quickly
Backup Slides
Interaction with TCP Congestion Control

(1) Bucket filled → unbounded throughput
(2) Bucket empty → bursty loss
(3) Waiting for timeout
(4) Repeats from (1)
Interaction with TCP Congestion Control

Staircase pattern

High goodputs followed by heavy losses and long timeouts
Interaction with TCP Congestion Control

Staircase pattern

High goodputs followed by heavy losses and long timeouts

(1) Throughput with cwnd = 1 stays below policing rate
(2) Throughput with cwnd = 2 exceeds policing rate
(3) Repeats from (1)
Interaction with TCP Congestion Control

Staircase pattern

High goodputs followed by heavy losses and long timeouts

Doubling window pattern

Flipping between rates since connection cannot align with policing rate
Understanding Policing

Collect packet traces

Forward samples to analysis backend

Derive basic features

Handle over 30 billion packets daily

Apply policing detection heuristic

Store & query aggregate results

e.g. retransmissions, latency, HTTP chunks, ...
Validation

- **Accuracy of heuristic (lab validation)**
  - Generated test traces covering common reasons for dropped packets
    - Policing (using carrier-grade networking device that can do policing)
    - Congestion (bottleneck link with tail queuing and different AQM flavors)
    - Random loss
    - Shaping (also using third-party traces)
  - TODO: Result summary

- **Consistency of policing rates (in the wild)**
  - Validated that policing rates cluster around a few values (per AS)
  - No clustering in ASes without policing
    - And: false positives in lab did not observe clustering either
**Common Mechanisms to Enforce ISP Policies**

**Policing**
- Enforces rate by **dropping** excess packets immediately
  - Can result in high loss rates
  - Does not require memory buffer
  - No RTT inflation

**Shaping**
- Enforces rate by **queuing** excess packets
  - Only drops packets when buffer is full
  - Requires memory to buffer packets
  - Can inflate RTTs due to high queuing delay
Policing can have negative side effects for all parties

- Content providers
  - Excess load on servers forced to retransmit dropped packets
    (global average: 20% retransmissions vs. 2% when not policed)

- ISPs
  - Transport traffic across the Internet only for it to be dropped by the policer
  - Incurs avoidable transit costs

- Users
  - Can interact badly with TCP-based applications
  - We measured degraded video quality of experience (QoE) → user dissatisfaction
Analysis Pipeline

- Collect packet traces
- Forward samples to analysis backend
- Detect policing
- Cross-reference with application metrics
Packets are always dropped when crossing the “policing rate” line.
Detection Algorithm

1. Find the policing rate
   - Use measured throughput between an early and late loss as estimate

2. Match performance to expected policing behavior
   - Everything above the policing rate gets dropped
   - (Almost) nothing below the policing rate gets dropped

Progress

Policing rate

Time
Avoiding Falsely Labeling Loss as Policing

But: Traffic below policing rate should go through

But: Traffic above policing rate should be dropped
Packets are usually dropped when a router’s buffer is already full.

Buffer fills → queuing delay increases.

Use inflated latency as signal that loss is not caused by a policer.
Validation 2: Live Traffic

- Observed only few policing rates in ISP deep dives
  - ISPs enforce a limited set of data plans
- Confirmed that per ISP policing rates cluster around a few values across the whole dataset
- And: Observed no consistency across flows without policing