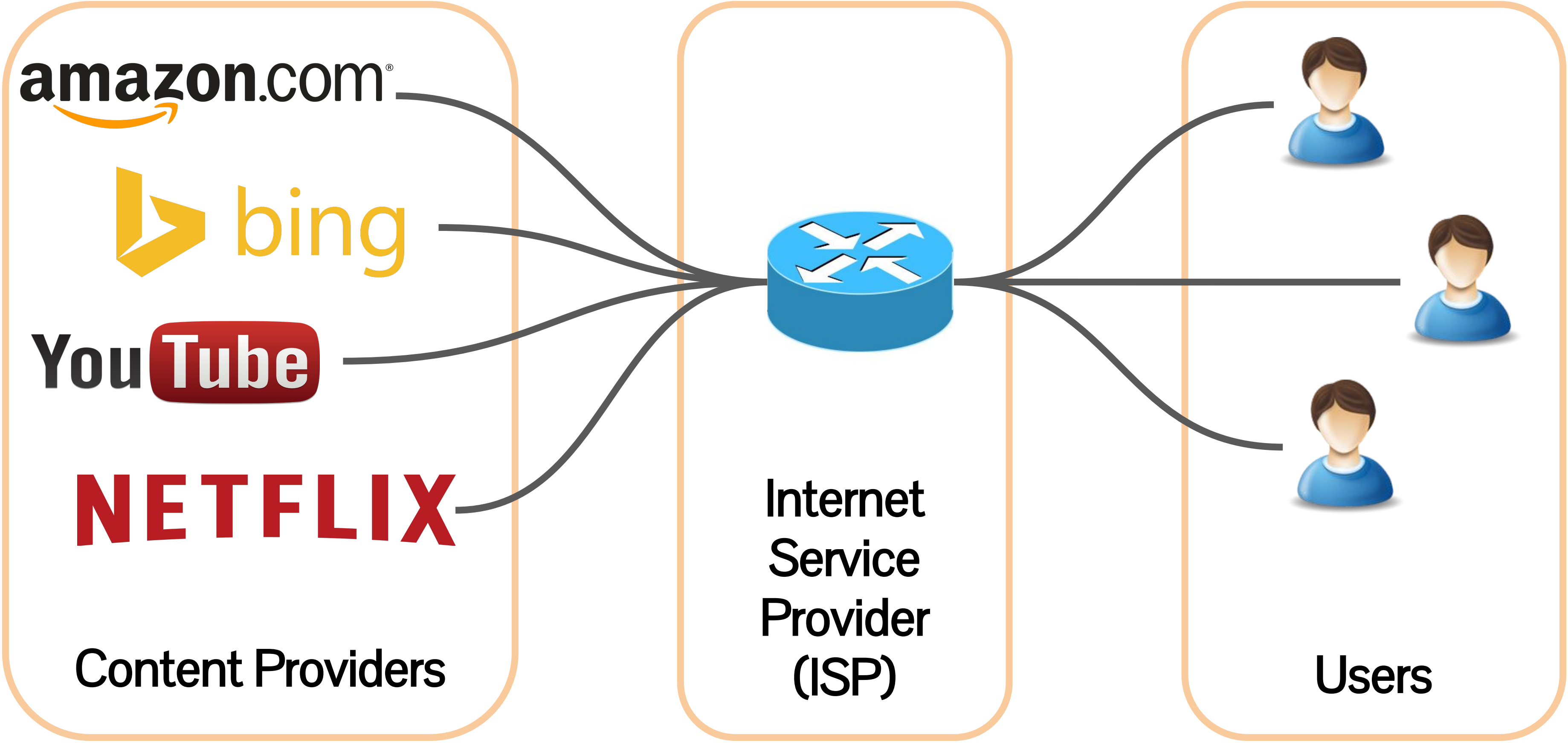
An Internet-Wide Analysis of Traffic Policing

Tobias Flach, Pavlos Papageorge, Andreas Terzis, Luis Pedrosa, Yuchung Cheng, Tayeb Karim, Ethan Katz-Bassett, Ramesh Govindan

policing-paper@google.com

USC University of Southern California

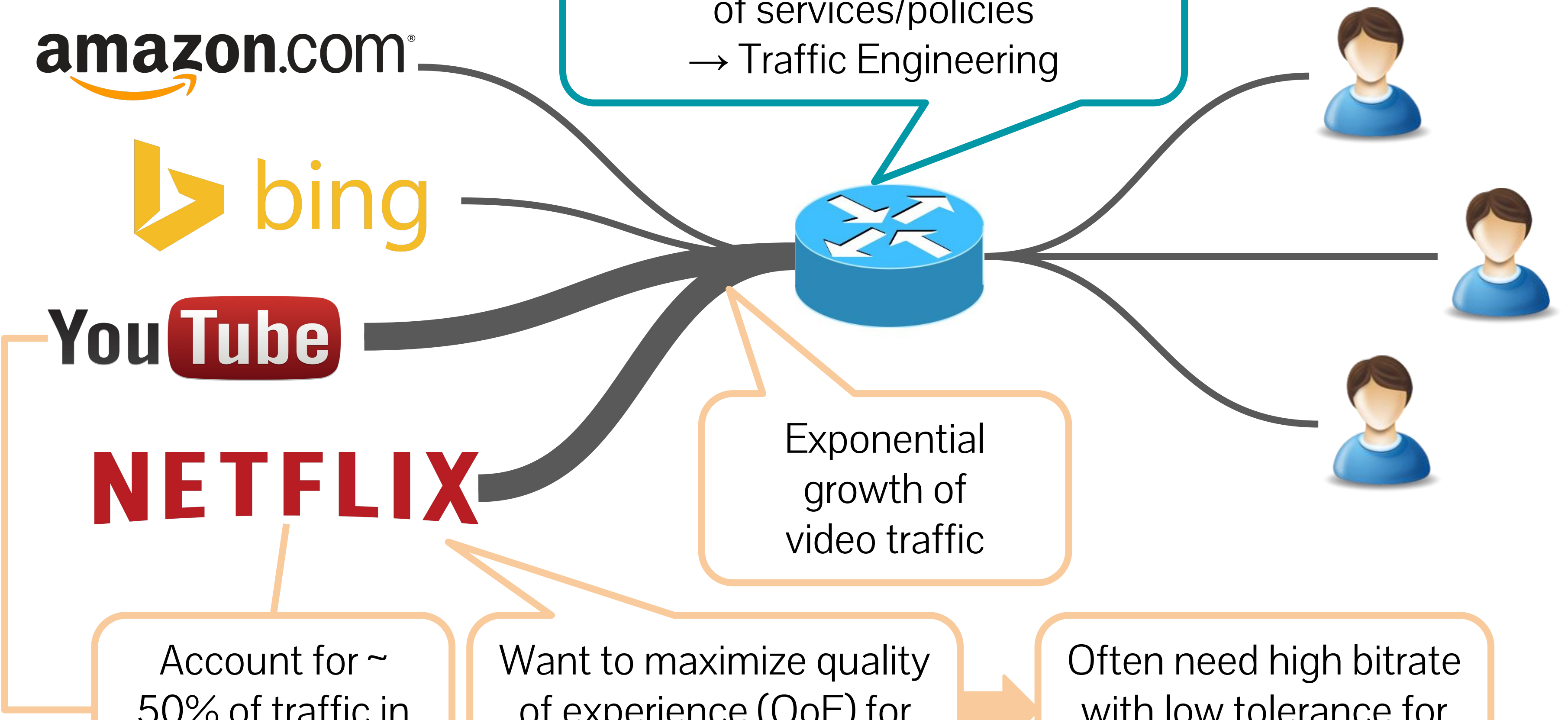
Google



Content Providers

Internet Service Provider (ISP)

Users



Want to accommodate multitude of services/policies
→ Traffic Engineering

Exponential growth of video traffic

Account for ~ 50% of traffic in North America

Want to maximize quality of experience (QoE) for their users

Often need high bitrate with low tolerance for latency and packet loss

Traffic Engineering: Policing vs. Shaping

Goal: Enforce a rate limit (maximum throughput)

Solutions:

a. **Drop** packets once the limit is reached
→ Traffic Policing

Focus of this talk

b. **Queue** packets (and send them out at the maximum rate)
→ Traffic Shaping

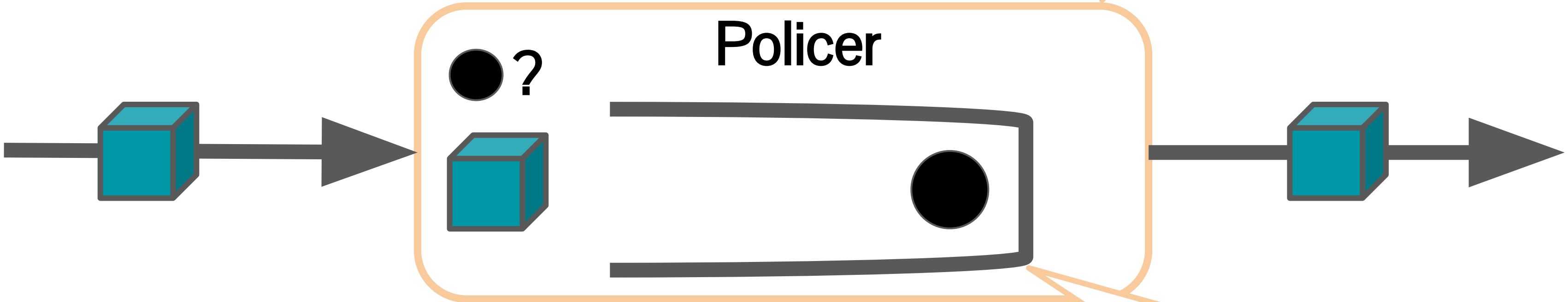
Contribution

Analyze the **prevalence** and **impact** of
traffic policing on a **global scale**,
as well as explore ways to
mitigate the **impact** of policers.

Outline

1. How Policing Works
2. Detecting the Effects of Policing in Packet Captures
3. A Global-Scale Analysis of Policing in the Internet
4. Mitigating the Impact of Policers

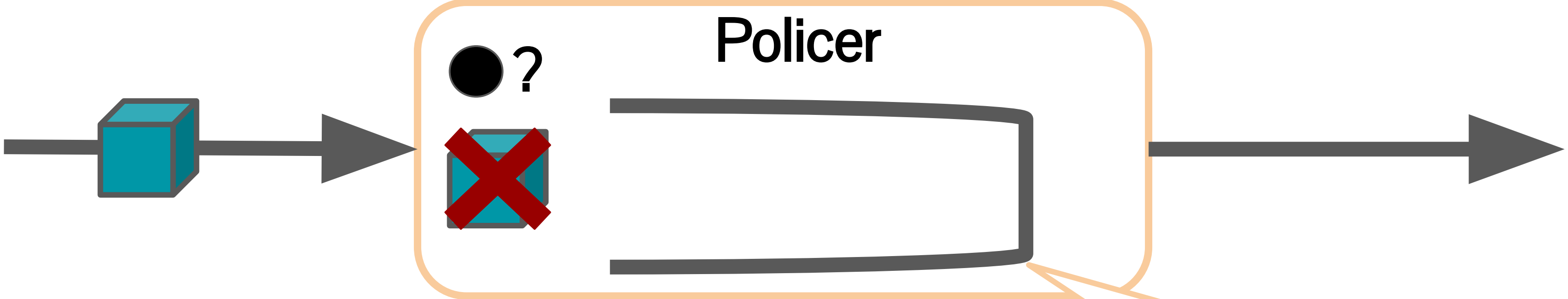
How Policing Works



Packet leaves if enough tokens are available

Tokens refreshed at predefined policing rate

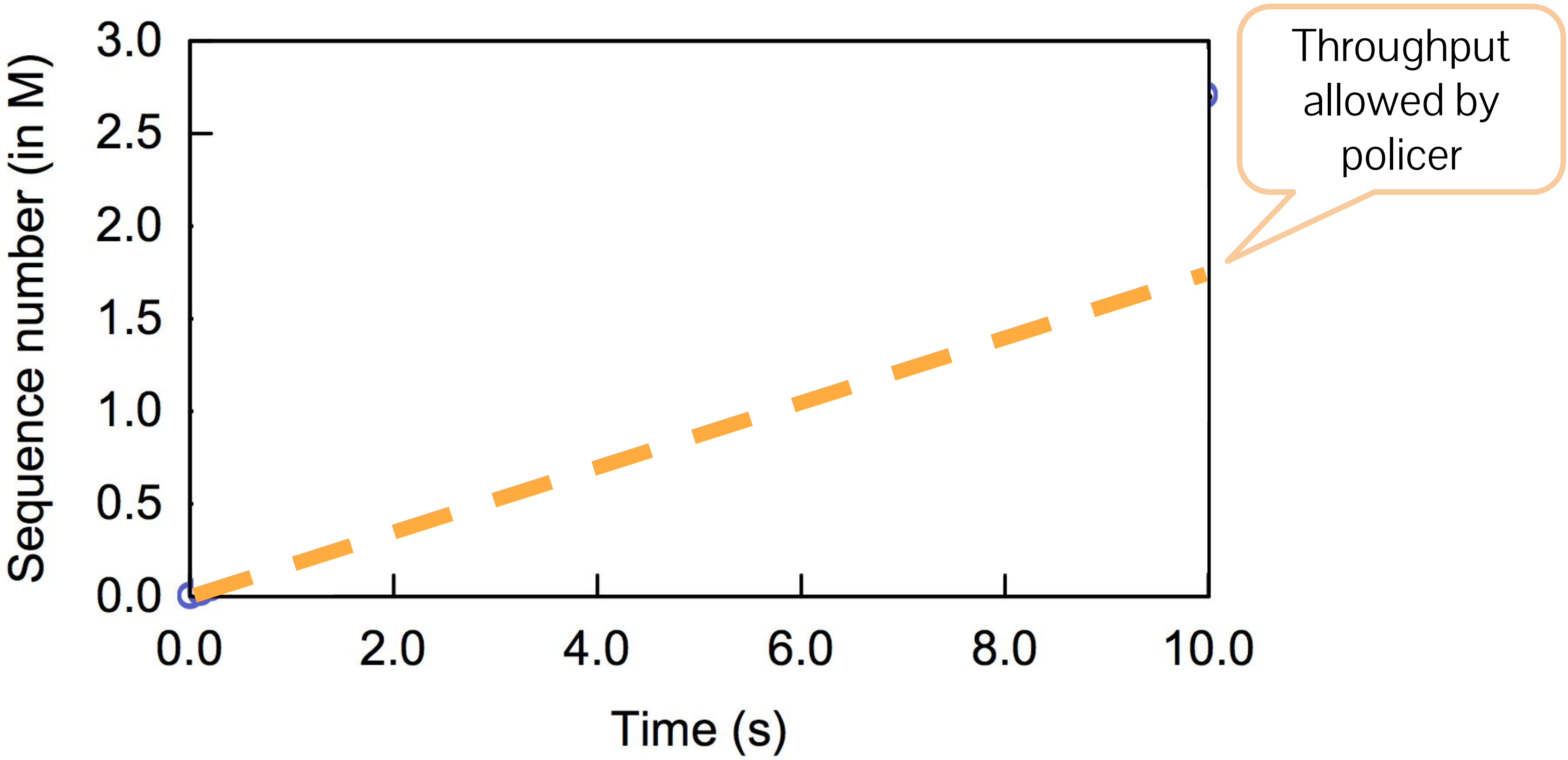
How Policing Works



Packet leaves if enough tokens are available

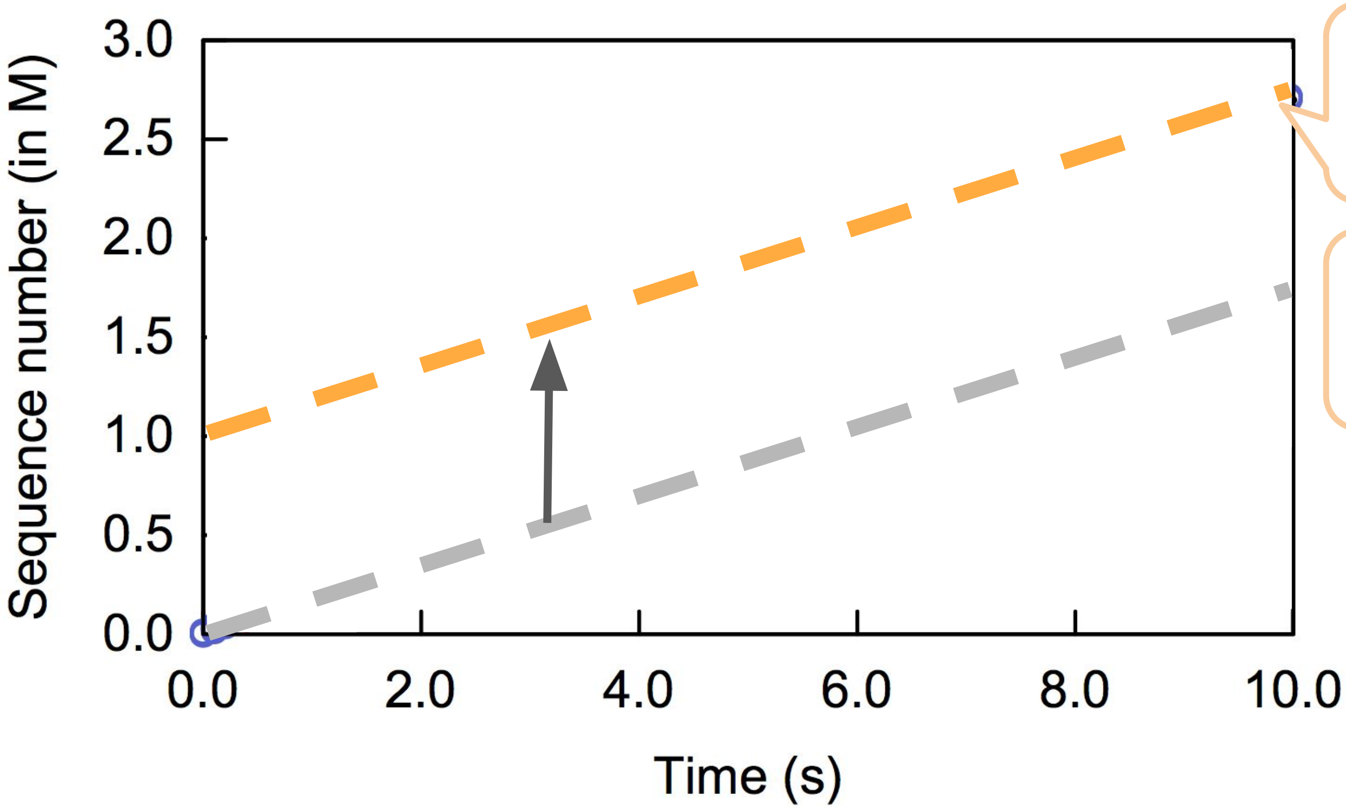
Tokens refreshed at predefined policing rate

Policing in Action



Throughput allowed by policer

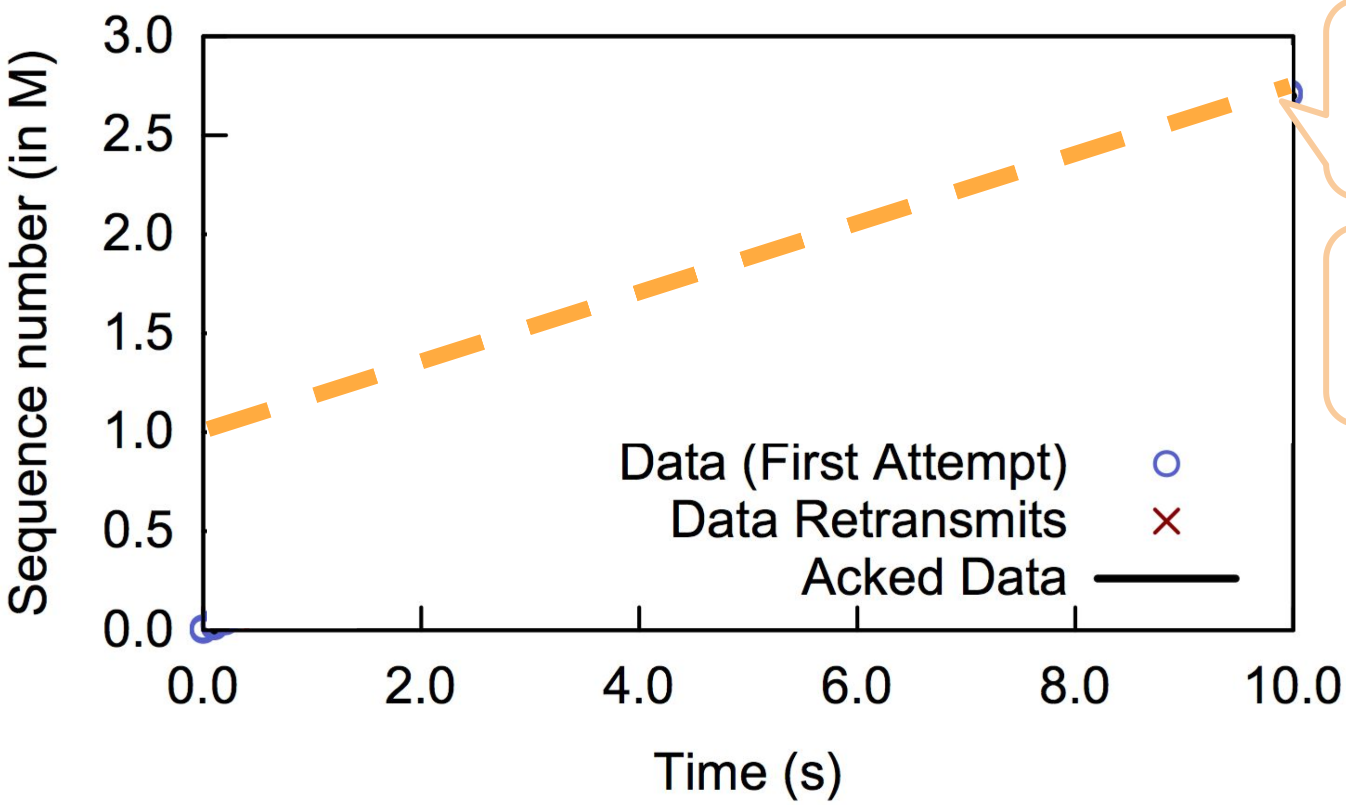
Policing in Action



Throughput allowed by policer

Plus: initial bursts from saved tokens

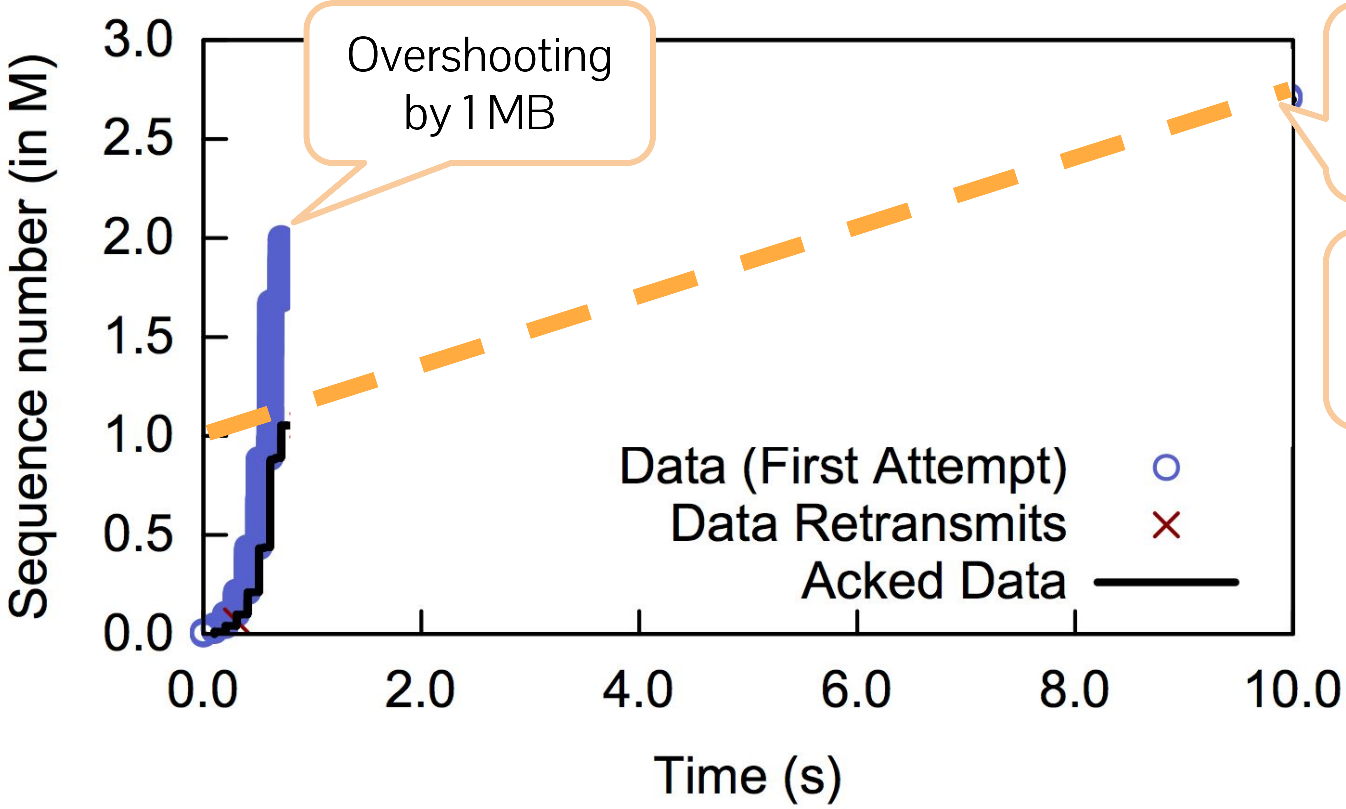
Policing in Action



Throughput allowed by policer

Plus: initial bursts from saved tokens

Policing in Action

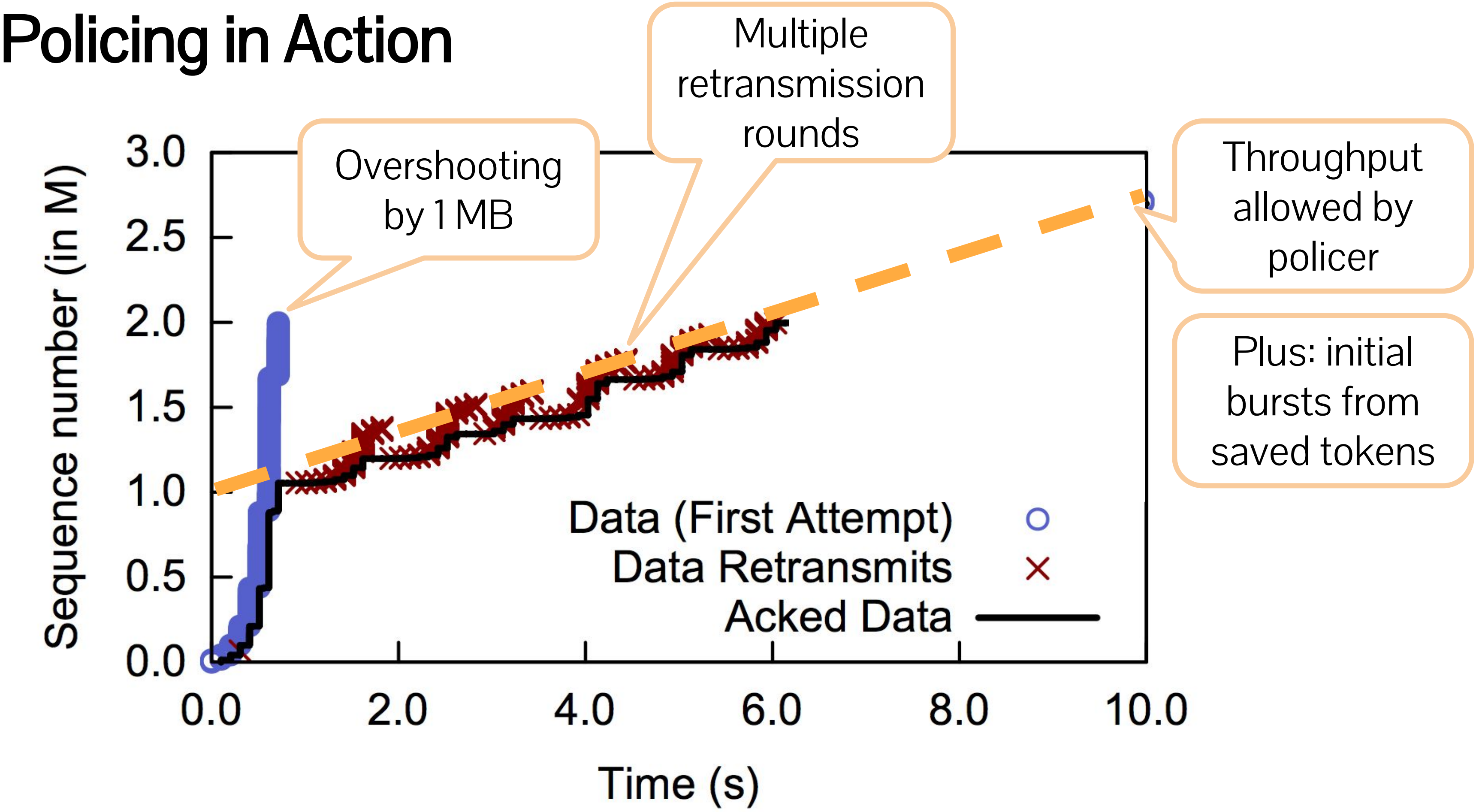


Overshooting by 1 MB

Throughput allowed by policer

Plus: initial bursts from saved tokens

Policing in Action



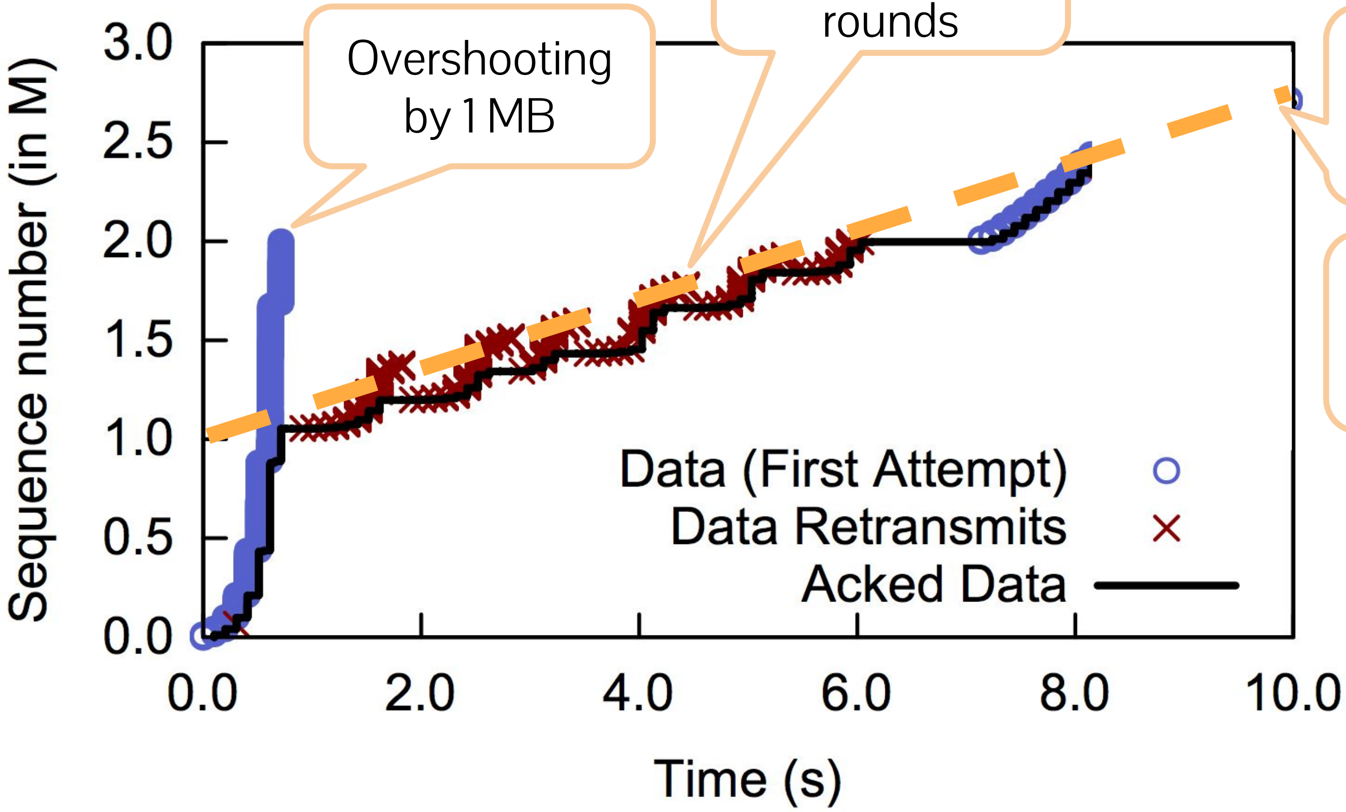
Overshooting by 1 MB

Multiple retransmission rounds

Throughput allowed by policer

Plus: initial bursts from saved tokens

Policing in Action



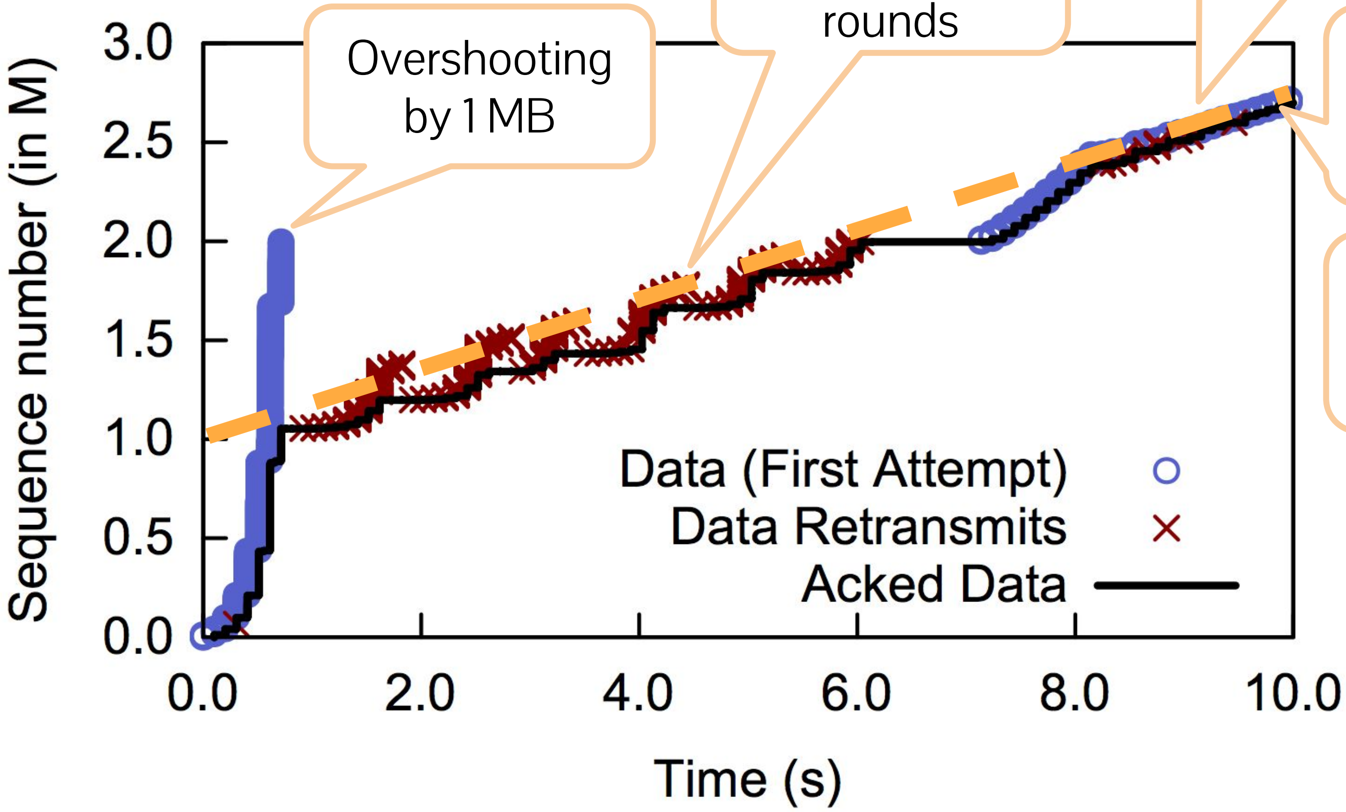
Overshooting by 1 MB

Multiple retransmission rounds

Throughput allowed by policer

Plus: initial bursts from saved tokens

Policing in Action



Overshooting by 1 MB

Multiple retransmission rounds

Transmission rate matches policing rate

Throughput allowed by policer

Plus: initial bursts from saved tokens

Policing can have negative side effects for all parties

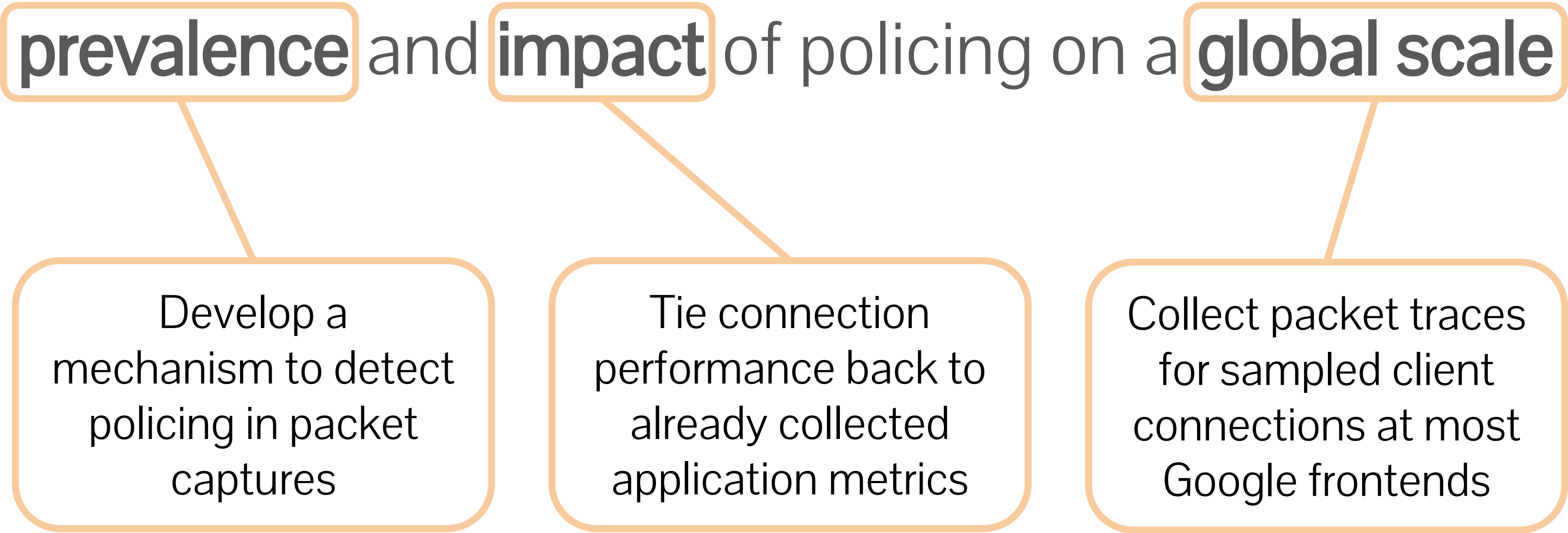
- Content providers
  - Excess load on servers forced to retransmit dropped packets

    (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

Analyze the **prevalence** and **impact** of policing on a **global scale**

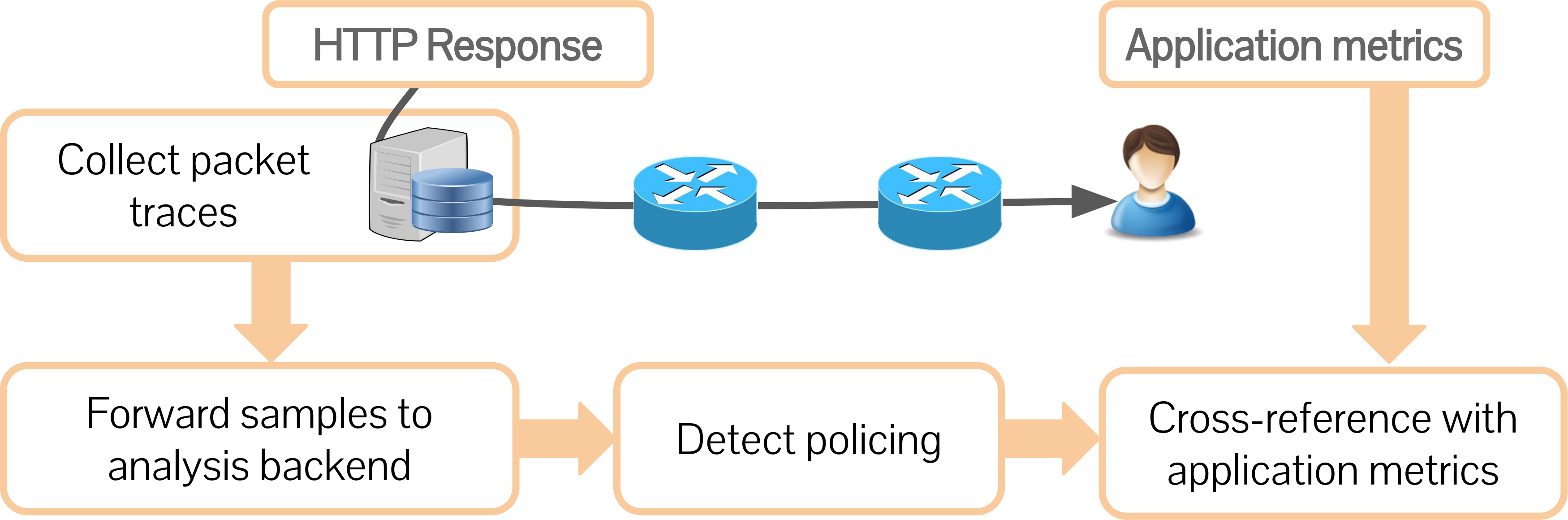


Develop a mechanism to detect policing in packet captures

Tie connection performance back to already collected application metrics

Collect packet traces for sampled client connections at most Google frontends

Analysis Pipeline



HTTP Response

Application metrics

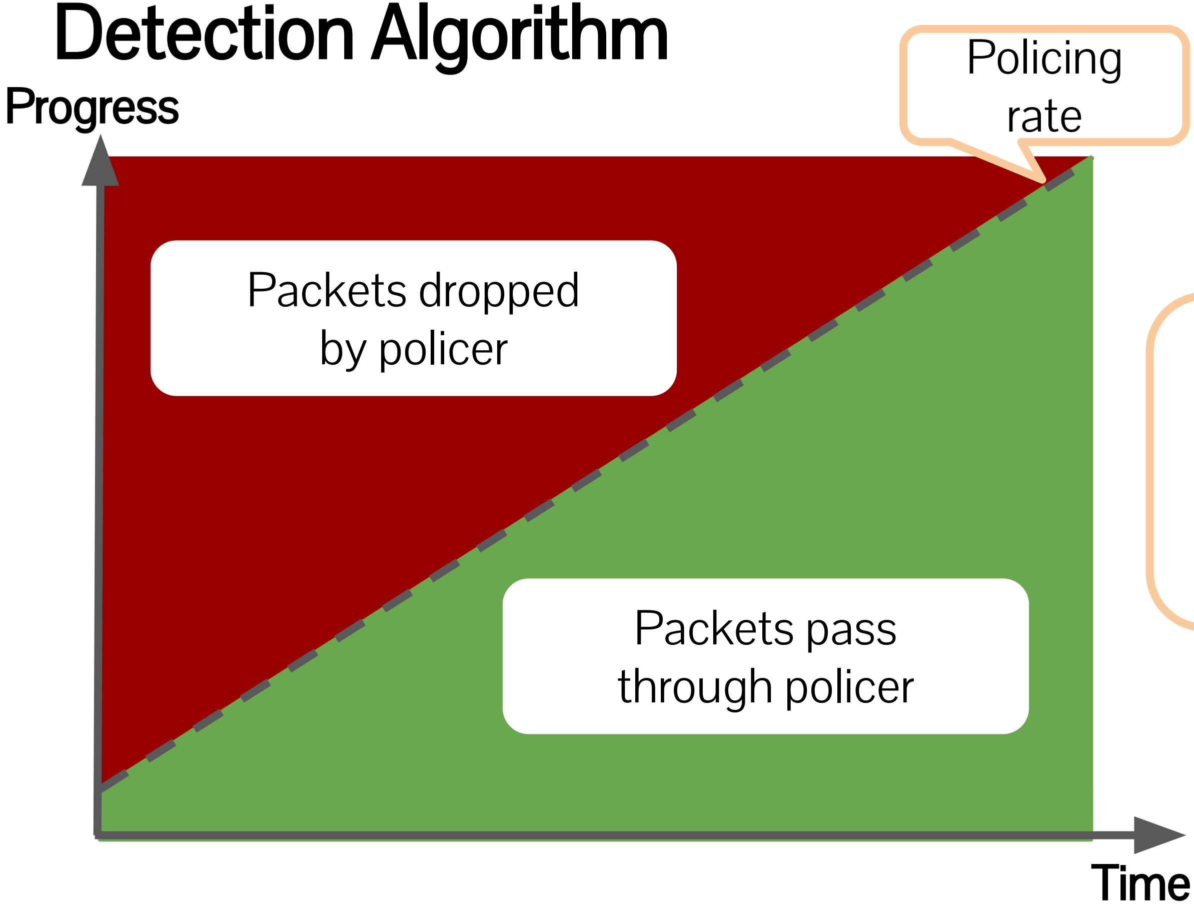
Collect packet traces

Forward samples to analysis backend

Detect policing

Cross-reference with application metrics

Detection Algorithm



Progress

Policing rate

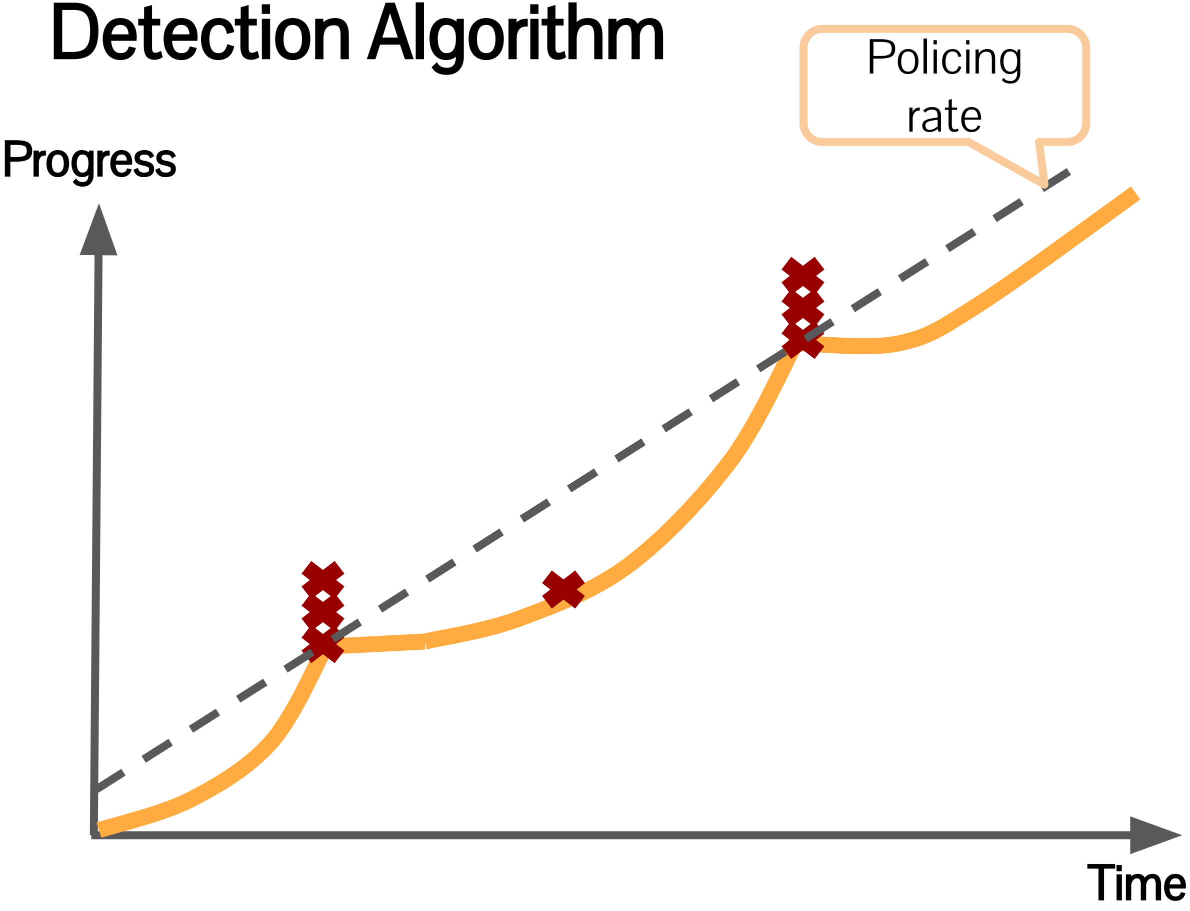
Packets dropped by policer

Packets pass through policer

Time

Packets are always dropped when crossing the “policing rate” line

Detection Algorithm



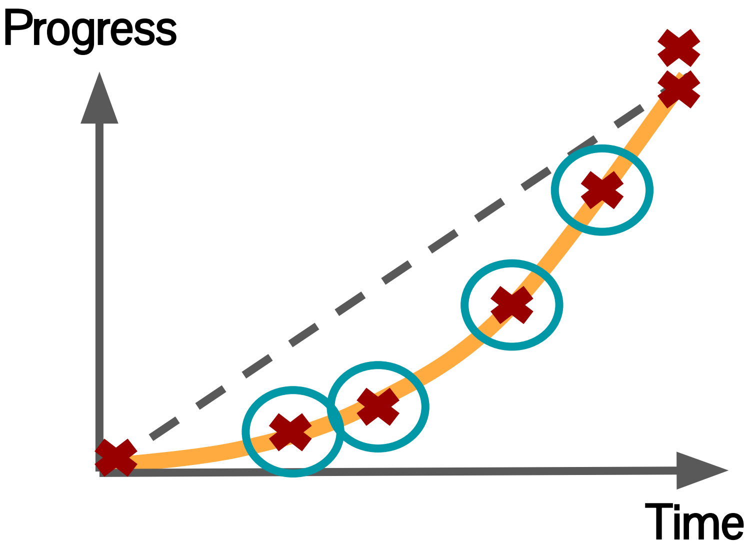
1 Find the policing rate

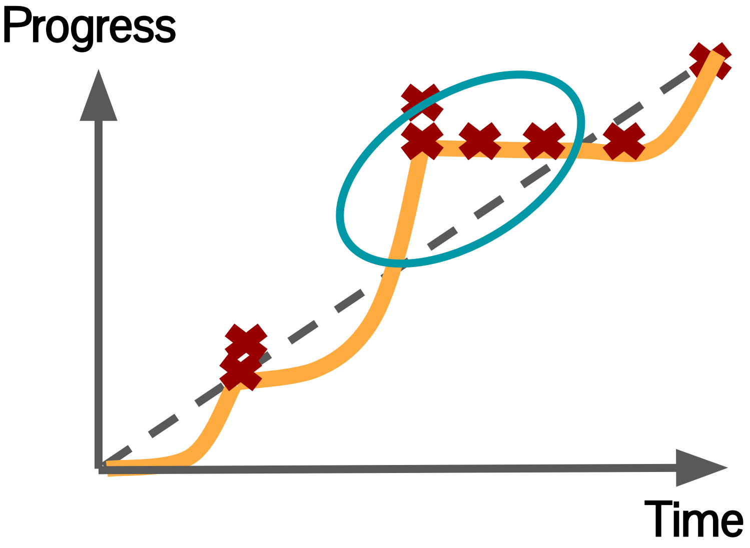
- Use measured throughput between an early and late loss as estimate

2 Match performance to expected policing behavior

- Everything above the policing rate gets dropped
- (Almost) nothing below the policing rate gets dropped

Avoiding Falsely Labeling Loss as Policing

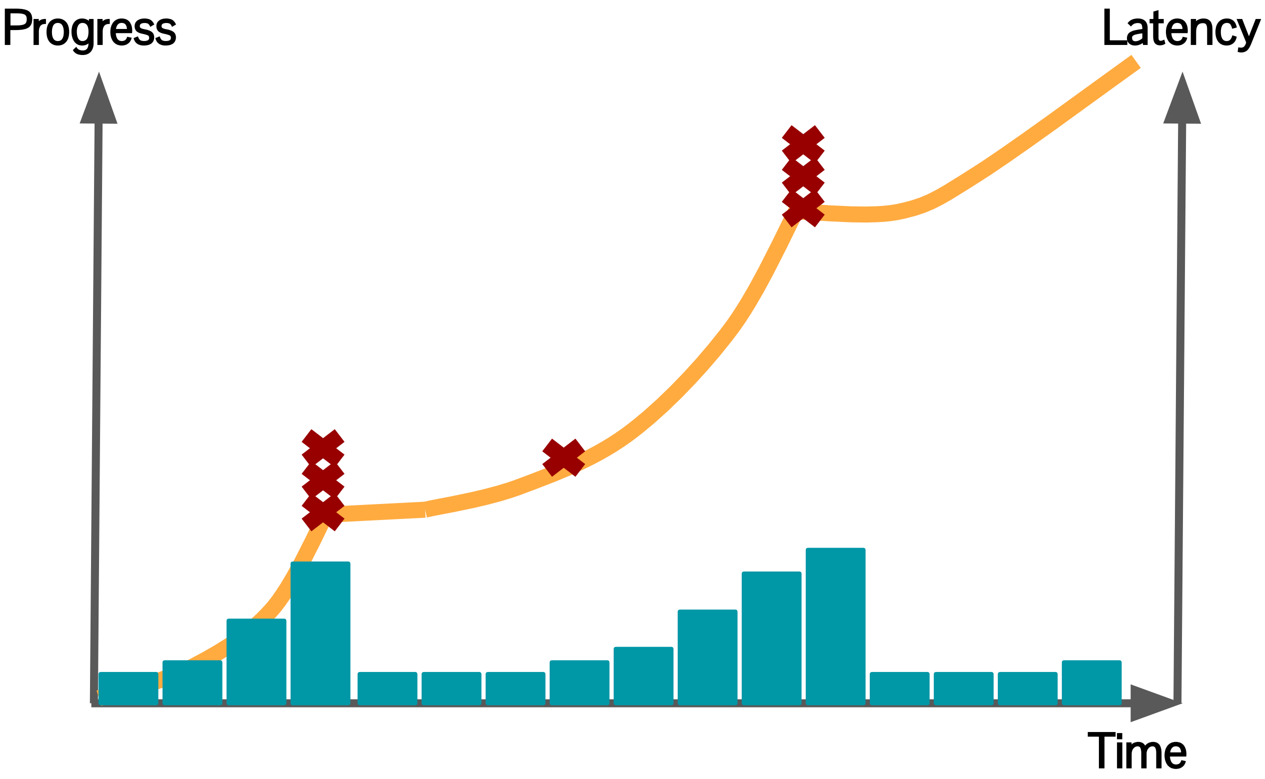




But: Traffic below policing rate should go through

But: Traffic above policing rate should be dropped

Congestion Looks Similar to Policing!



Packets are usually dropped when a router's buffer is already full

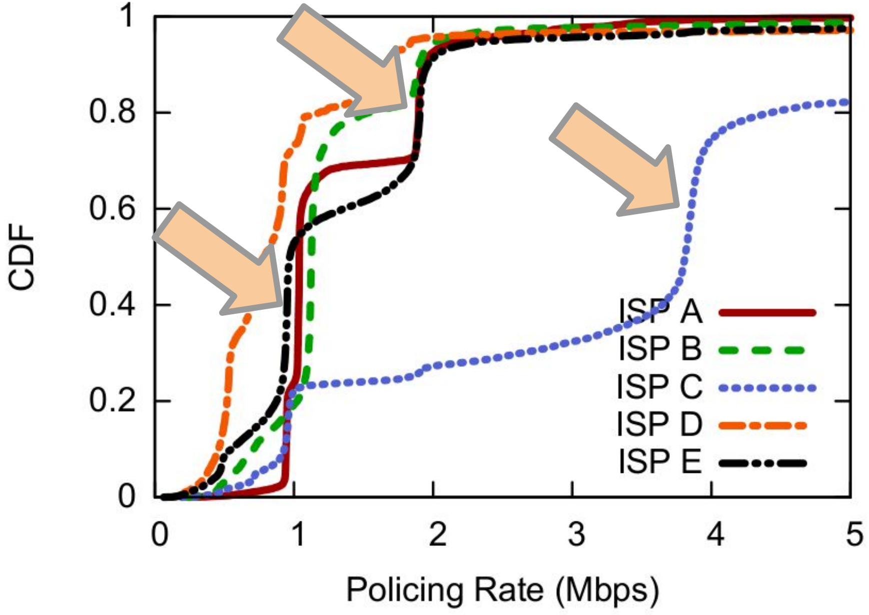
Buffer fills → queuing delay increases

Use inflated latency as signal that loss is not caused by a policer

Validation 1: Lab Setting

- Goal: Approximate the accuracy of our heuristic
- Generated test traces covering common reasons for dropped packets
  - Policing (used a router with support for policing)
  - Congestion
  - Random loss
  - Shaping
- High accuracy for almost all configurations (see paper for details)
  - Policing: 93%
  - All other reasons for loss: > 99%

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

Outline

1. How Policing Works

2. Detecting the Effects of Policing in Packet Captures

3. A Global-Scale Analysis of Policing in the Internet

4. Mitigating the Impact of Policers

Internet-Wide Analysis of Policing

- Sampled flows collected from most of Google’s CDN servers
    - 7-day sampling period (in September 2015)
    - 277 billion TCP packets
    - 270 TB of data
    - 800 million HTTP queries
    - Clients in over 28,400 ASes

- To tie TCP performance to application performance, we analyzed flows at HTTP request/response (“segment”) granularity

#1: Prevalence of Policing

Region	Policed segments (overall)
Africa	**1.3%**
Asia	**1.3%**
Australia	0.4%
Europe	0.7%
N. America	0.2%
S. America	0.7%

#1: Prevalence of Policing

Up to 7% of lossy segments are policed

Lossy:
15 losses or more
per segment

Region	Policed segments (overall)	Policed (among lossy)
Africa	**1.3%**	6.2%
Asia	**1.3%**	**6.6%**
Australia	0.4%	2.0%
Europe	0.7%	5.0%
N. America	0.2%	2.6%
S. America	0.7%	4.1%

#2: Policer-induced Losses

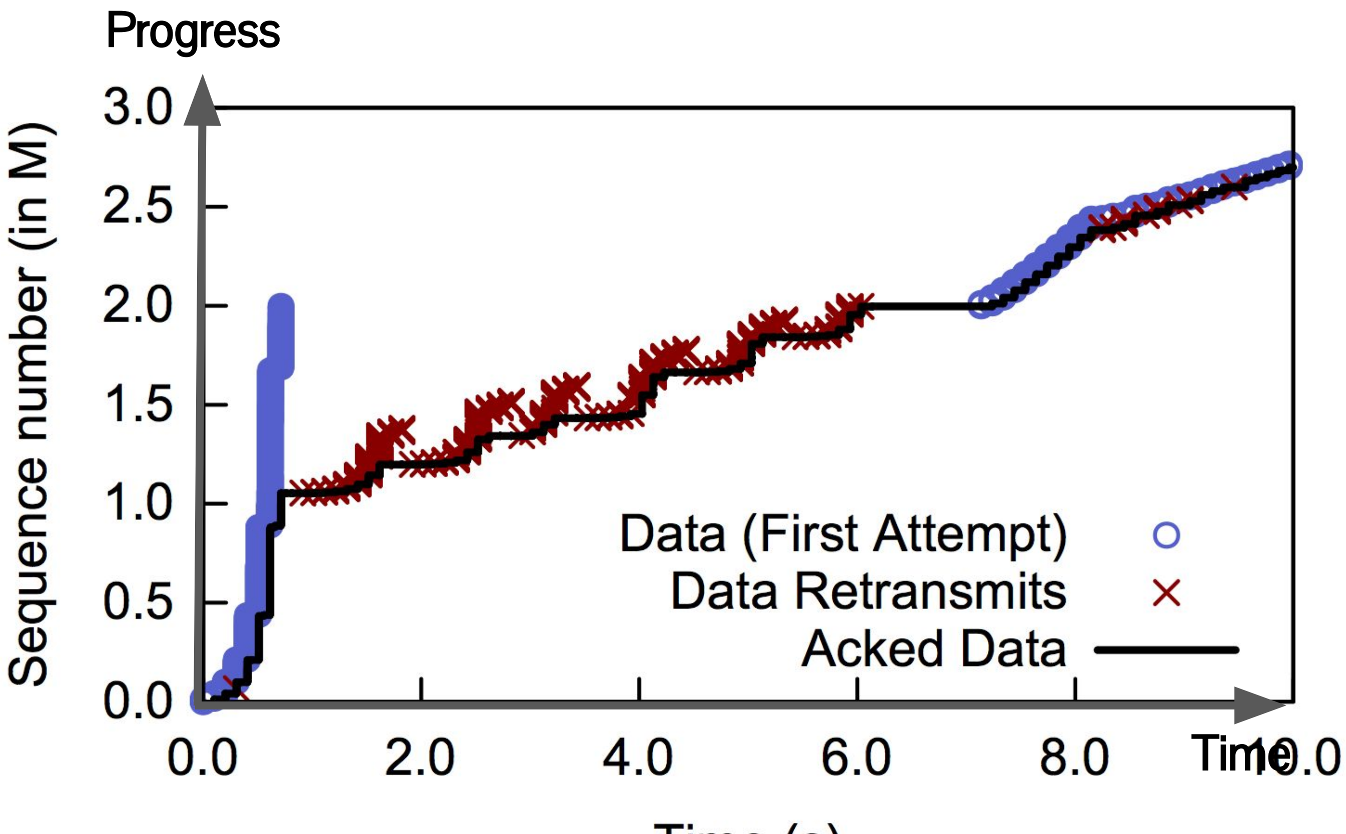
Lossy:
15 losses or more
per segment

Up to 7% of lossy segments are policed

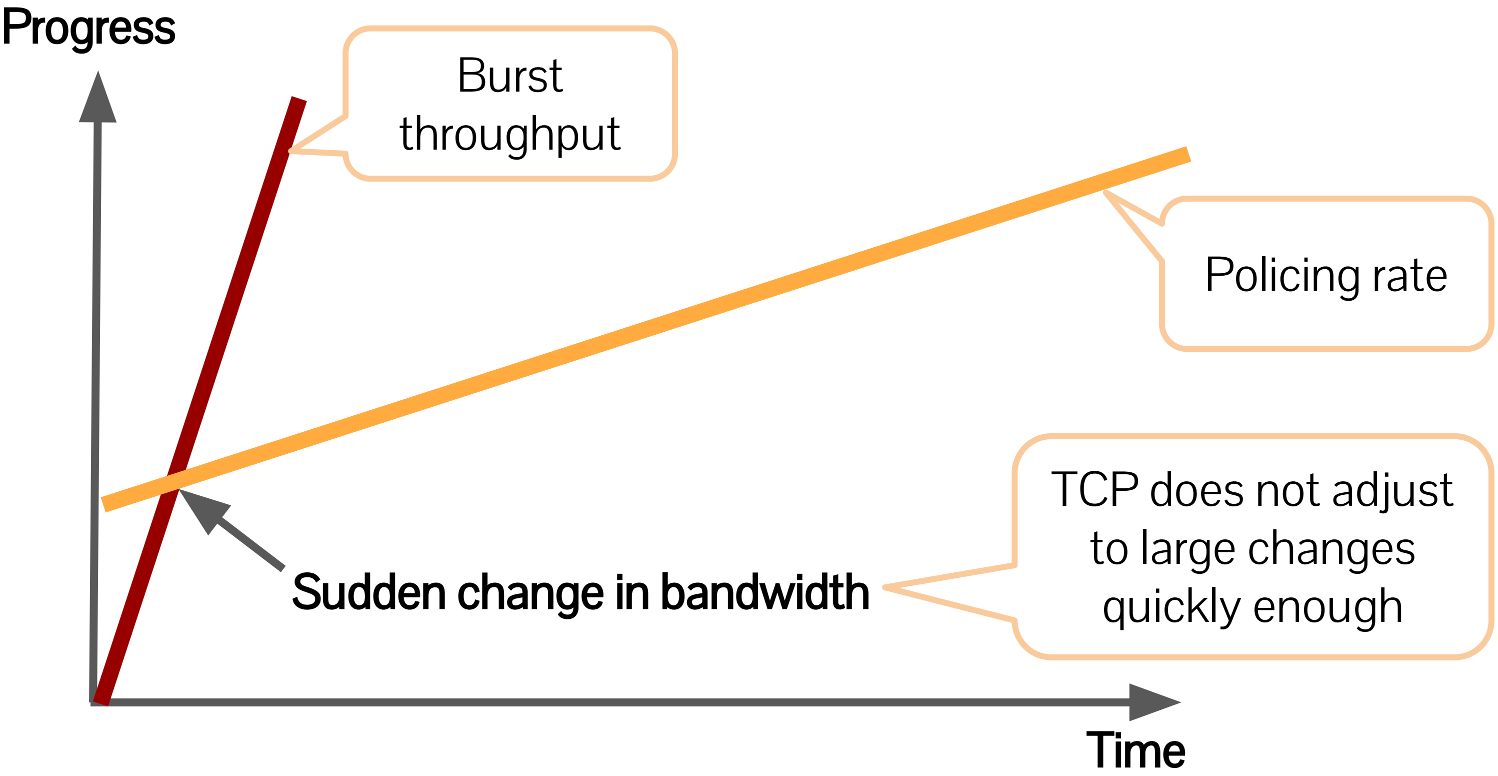
Average loss rate increases from 2% to over 20% when policed

Region	Policed segments (overall)	Policed (among lossy)	Loss (policed)	Loss (non-policed)
Africa	**1.3%**	6.2%	**27.5%**	**4.1%**
Asia	**1.3%**	**6.6%**	24.9%	2.9%
Australia	0.4%	2.0%	21.0%	1.8%
Europe	0.7%	5.0%	20.4%	1.3%
N. America	0.2%	2.6%	22.5%	1.0%
S. America	0.7%	4.1%	22.8%	2.3%

Sudden Bandwidth Change Induces Heavy Loss



Sudden Bandwidth Change Induces Heavy Loss



Burst throughput

Policing rate

Sudden change in bandwidth

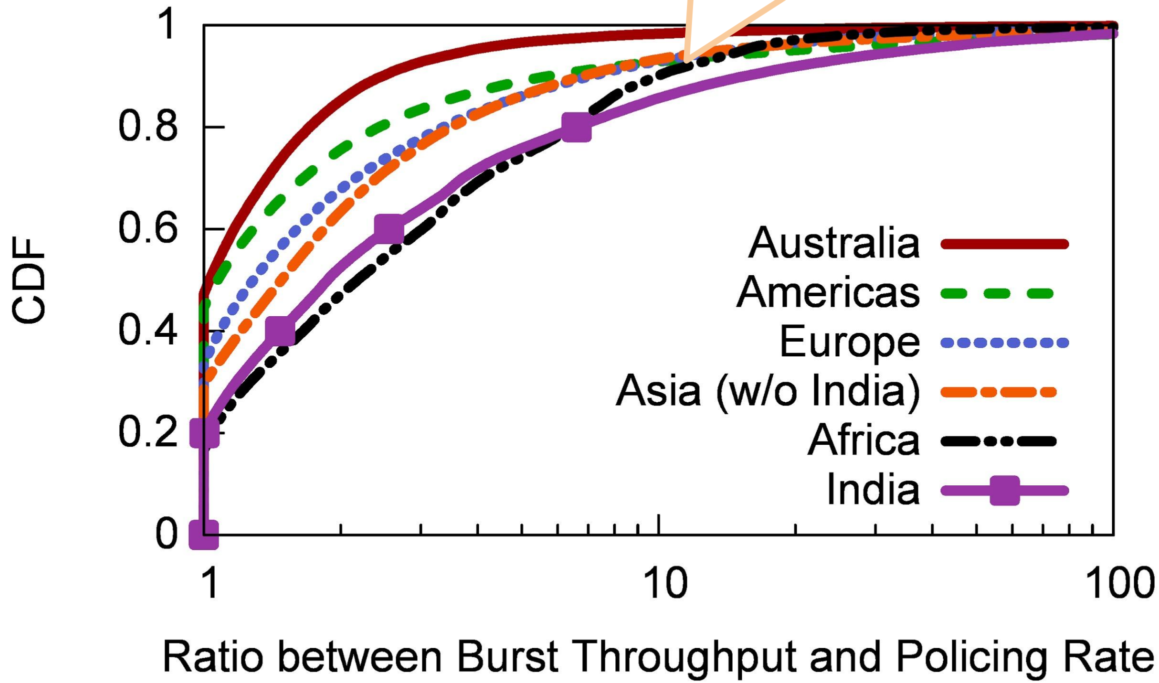
TCP does not adjust to large changes quickly enough

#3: Burst Throughput vs. Policing Rate

Up to 7% of lossy segments are policed

Average loss rate increases from 2% to over 20% when policed

Policing rate often over 50% lower than burst throughput



<ins>90th percentile:</ins>
Policing rate is 10x lower than burst throughput

Quality of Experience Metrics

Rebuffer Time:

Time that a video is paused *after playback started* due to insufficient stream data buffered

Watch Time:

Fraction of the video watched by the user

Rebuffer to Watch Time Ratio:

Goal is *zero* (no rebuffering delays after playback started).

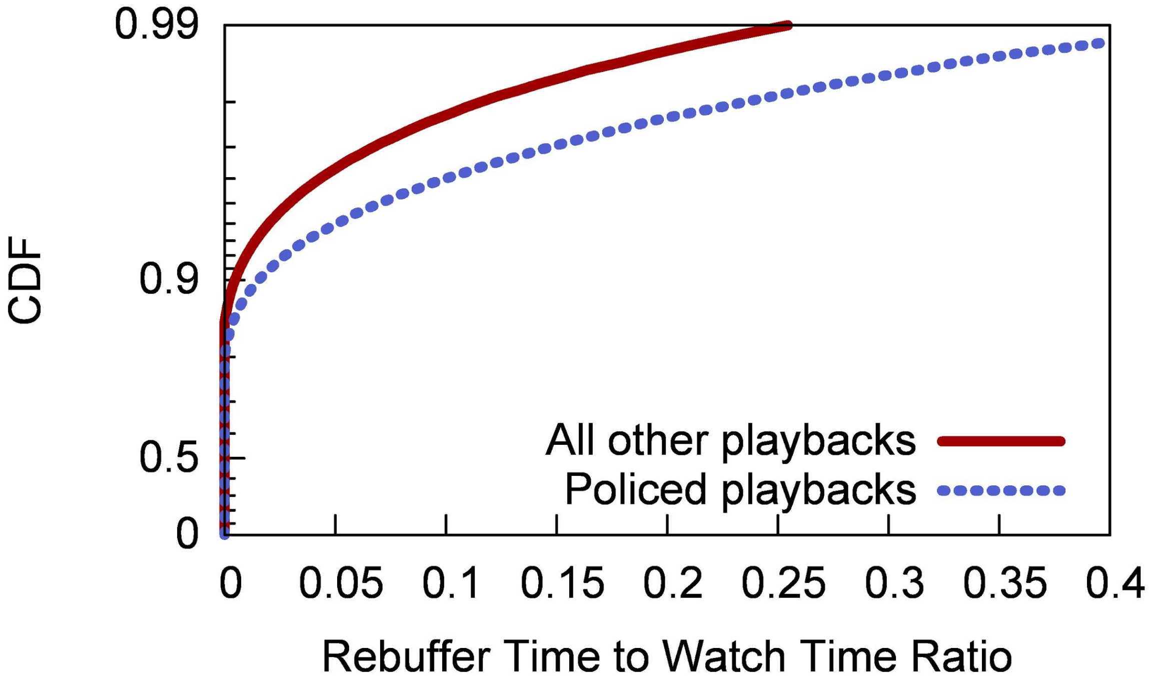
#4: Impact on Quality of Experience

Up to 7% of lossy segments are policed

Average loss rate increases from 2% to over 20% when policed

Policing rate often over 50% lower than burst throughput

In the tail, policed segments can have up to 200% higher rebuffering times



(For playbacks with the same throughput)

Mitigating Policer Impact

For content providers	For policing ISPs
No access to policers and their configurations	Access to policers and their configurations
But can control transmission patterns to minimize risk of hitting an empty token bucket	Can deploy alternative traffic management techniques

Mitigating Policer Impact

For content providers	For policing ISPs
Rate limiting	Policer optimization
Pacing	Shaping
Reducing losses during recovery in Linux	

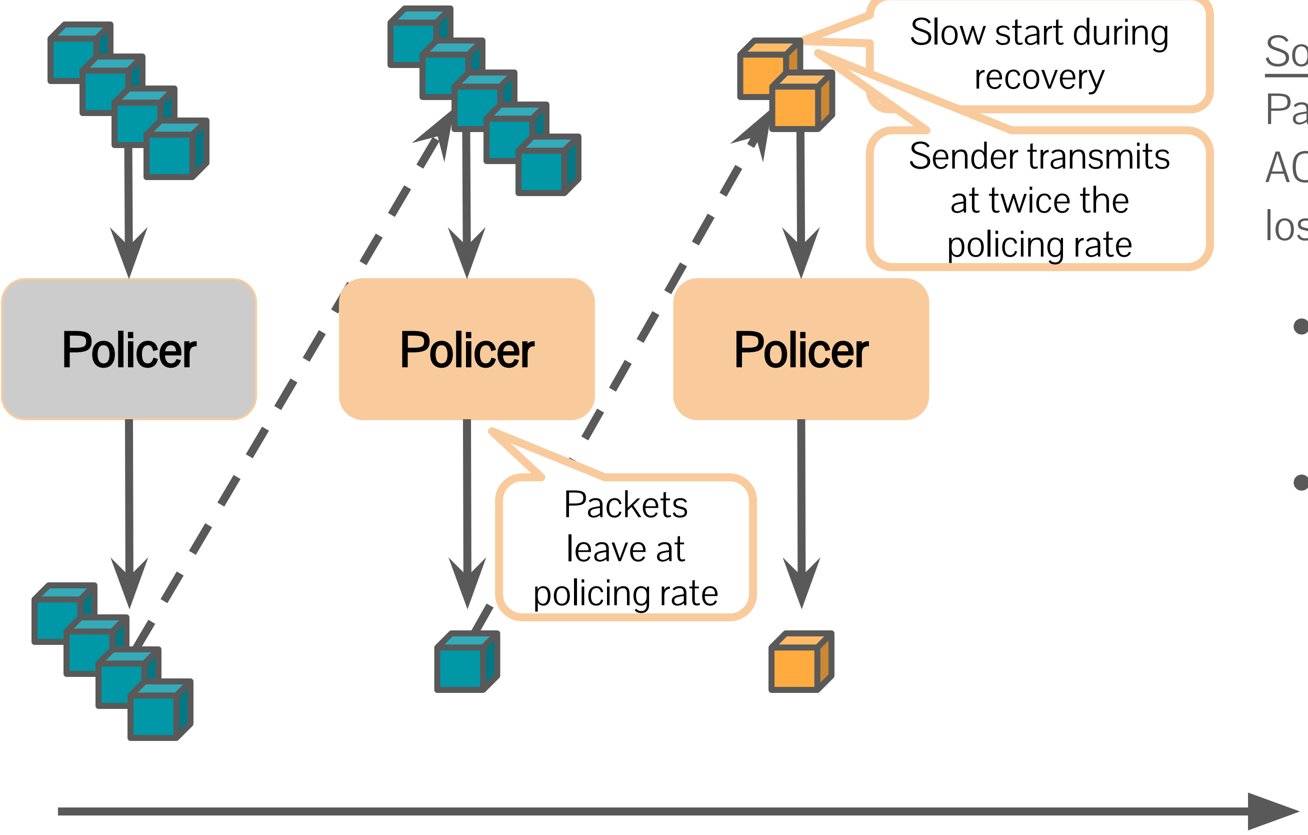
Mitigating Policer Impact

For content providers

For policing ISPs

Reducing losses during recovery in Linux

Reducing Losses During Recovery in Linux



Policer

Policer

Policer

Slow start during recovery

Sender transmits at twice the policing rate

Packets leave at policing rate

Round trips (one per column)

Send only one packet per ACK

Solution:

Packet conservation until ACKs indicate no further losses

- Reduces median loss rates by 10 to 20%
- Upstreamed to Linux kernel 4.2

Conclusion

- ISPs need ways to deal with increasing traffic demands and want to enforce plans → traffic policing is one option
- On a global scale up to 7% of lossy segments are affected by traffic policing
- Policed connections see ...
  - Much higher loss rates
  - Long recovery times when policers allow initial bursts
  - Worse video rebuffering times (QoE)
- Negative effects can be mitigated
  - Content providers: Rate limiting, pacing, prevention of loss during recovery
  - ISPs: Better policing configurations, shaping

Questions? Email us: policing-paper@google.com

Data: http://usc-nsl.github.io/policing-detection/