An Internet-Wide Analysis of Traffic Policing

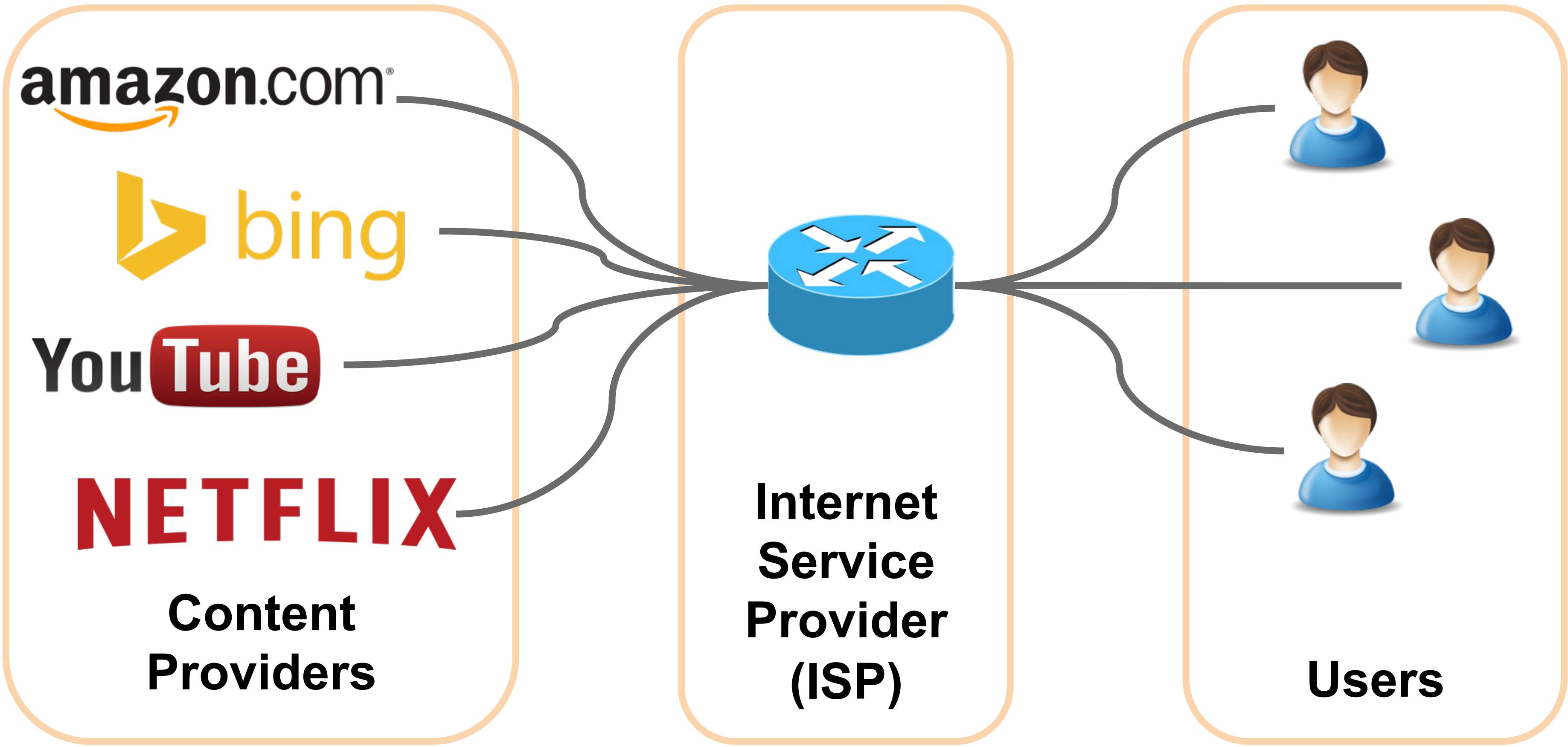
Tobias Flach, Pavlos Papageorge, Andreas Terzis, Luis Pedrosa, Yuchung Cheng, Tayeb Karim, Ethan Katz-Bassett, Ramesh Govindan

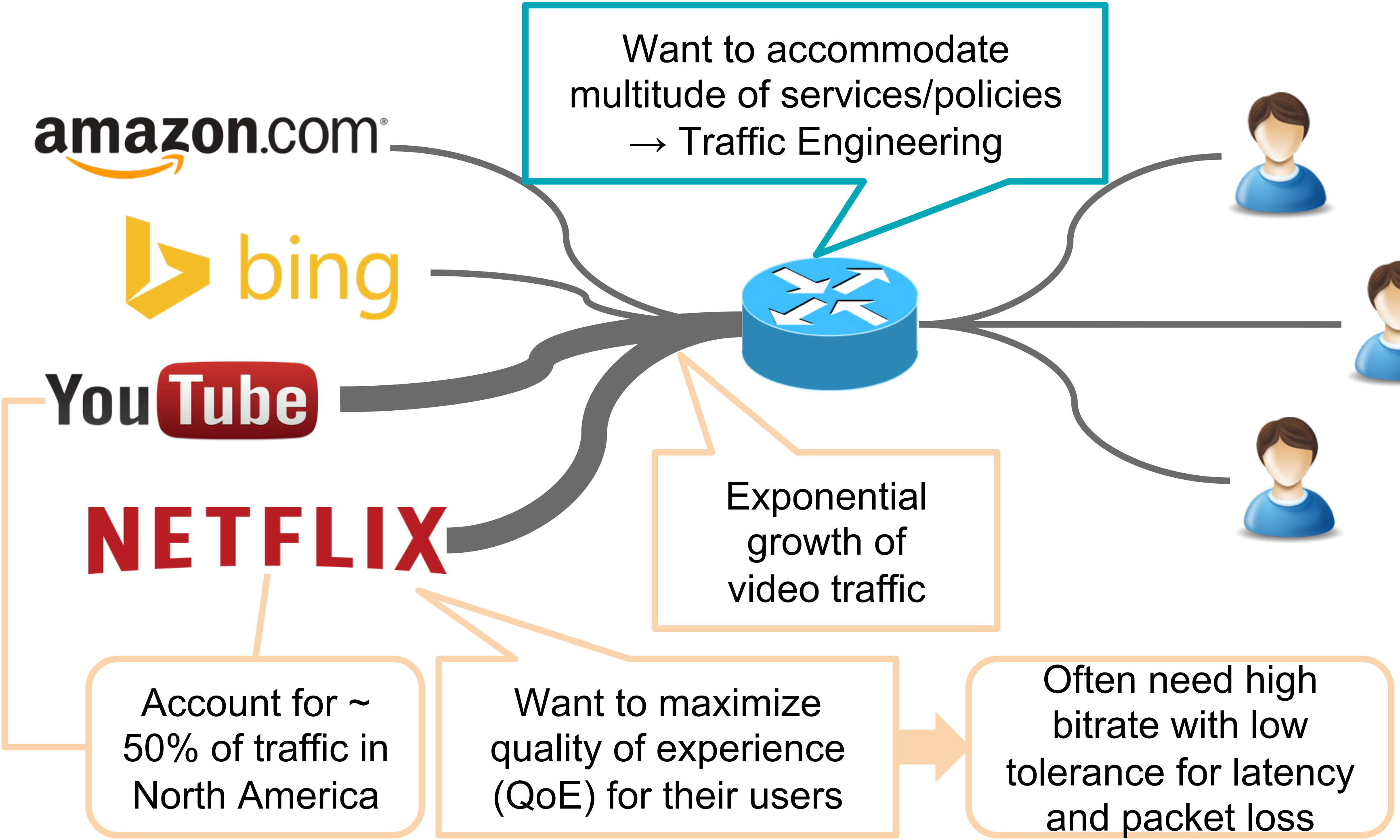
policing-paper@google.com

Slides: https://goo.gl/GVNRmX

USC University of Southern California

Google


amazon.com
bing
YouTube
NETFLIX
Content Providers
Internet Service Provider (ISP)
Users


Want to accommodate multitude of services/policies
→ Traffic Engineering
amazon.com
bing
YouTube
NETFLIX
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

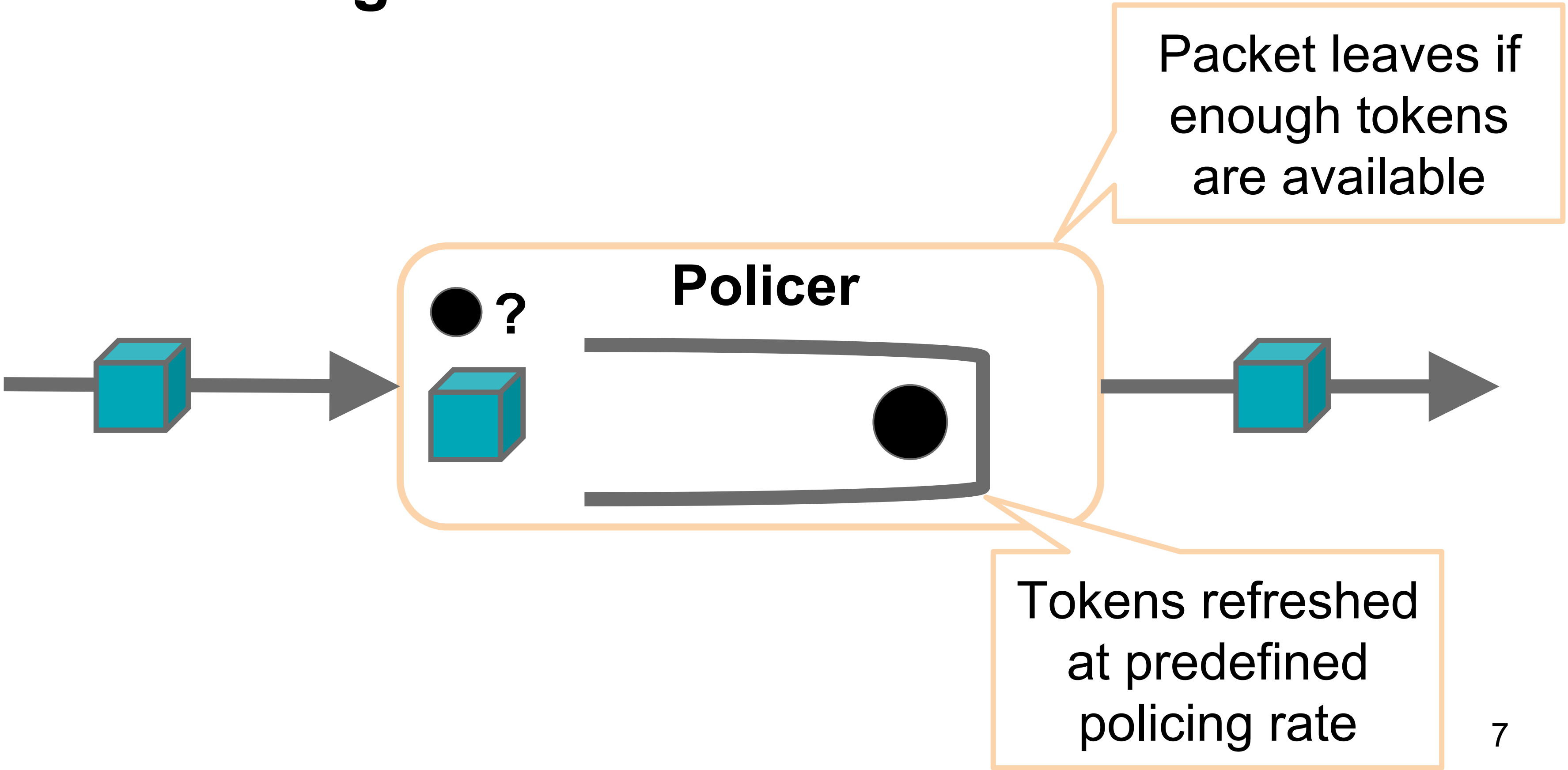
Contribution

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

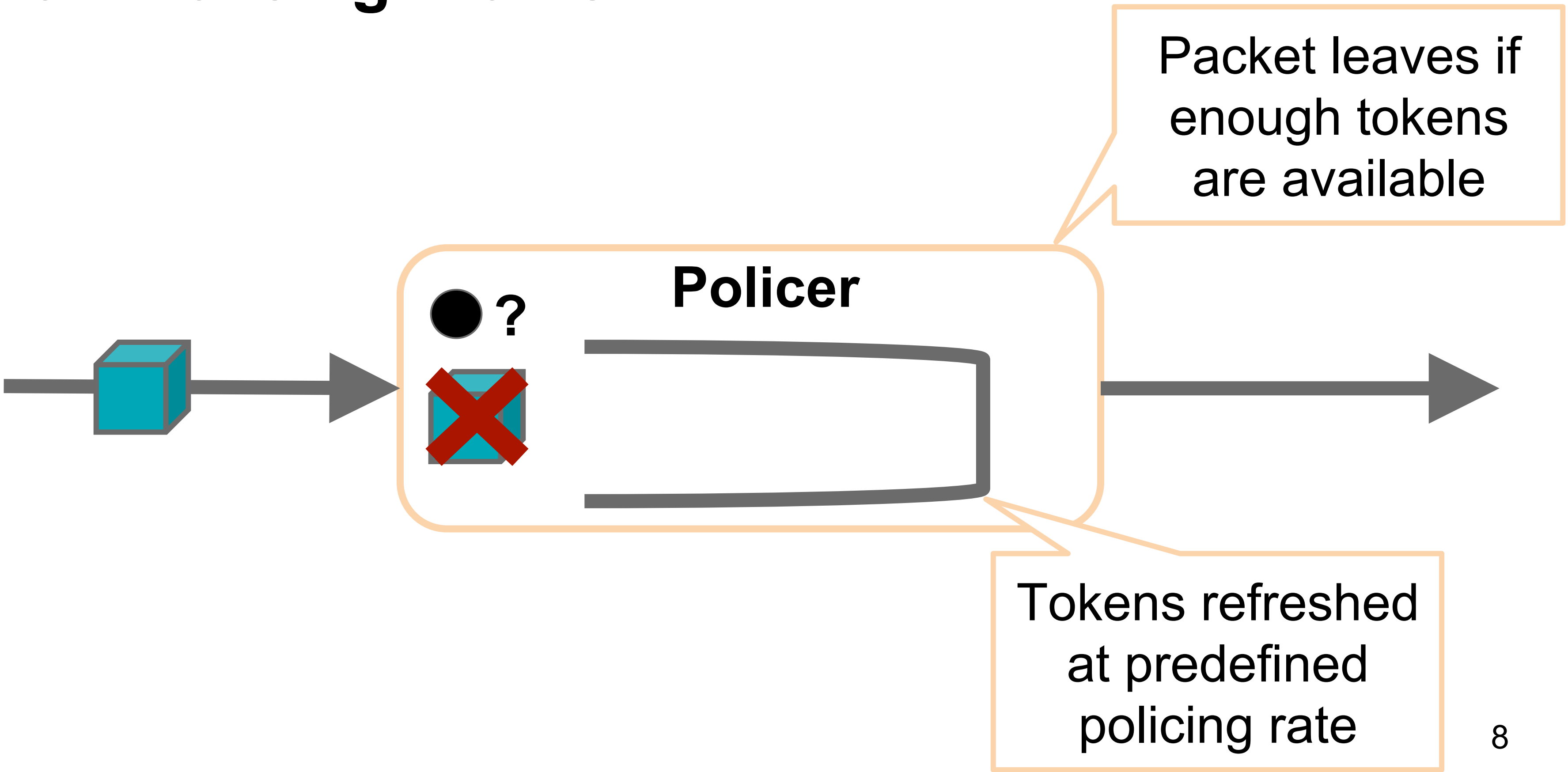
Outline

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

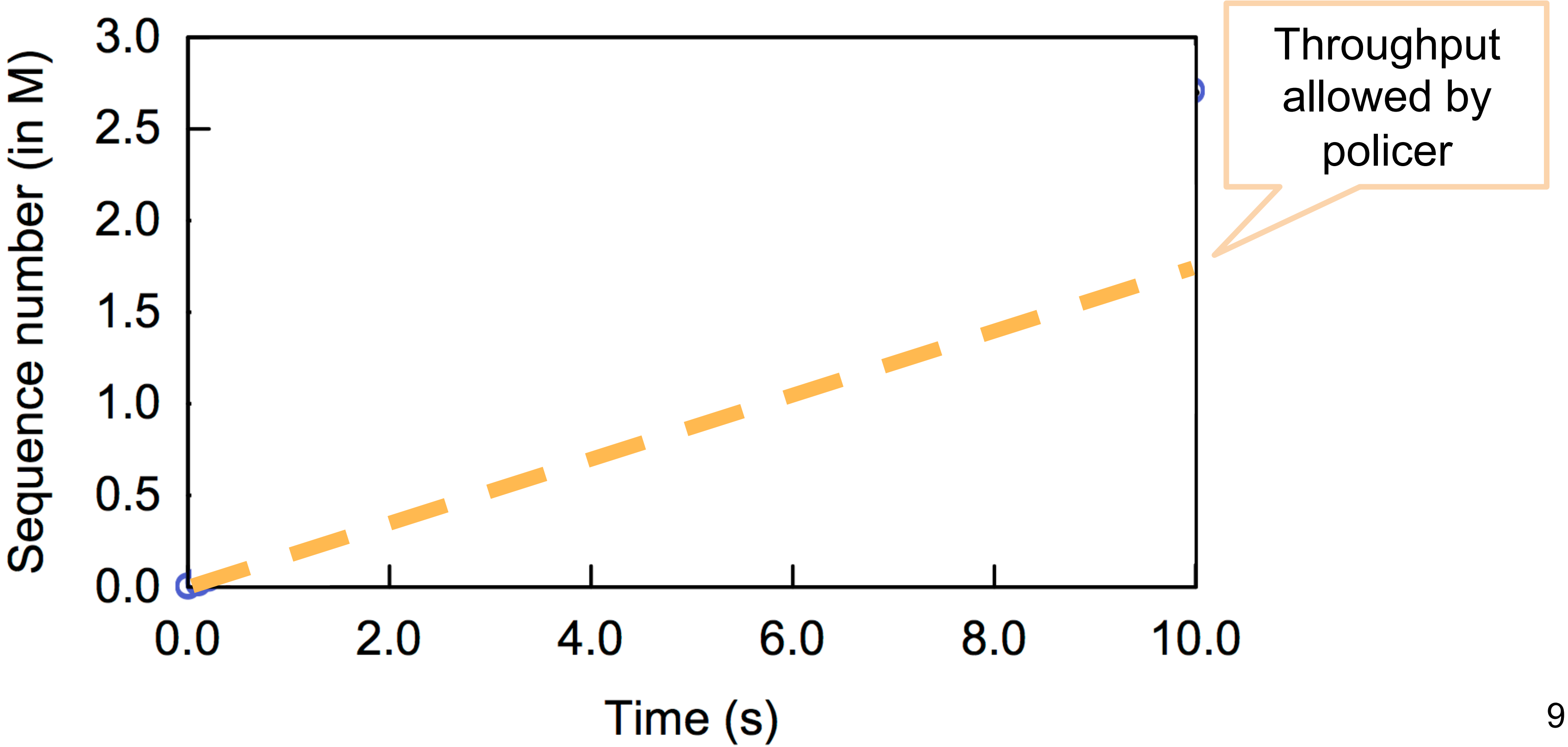
How Policing Works



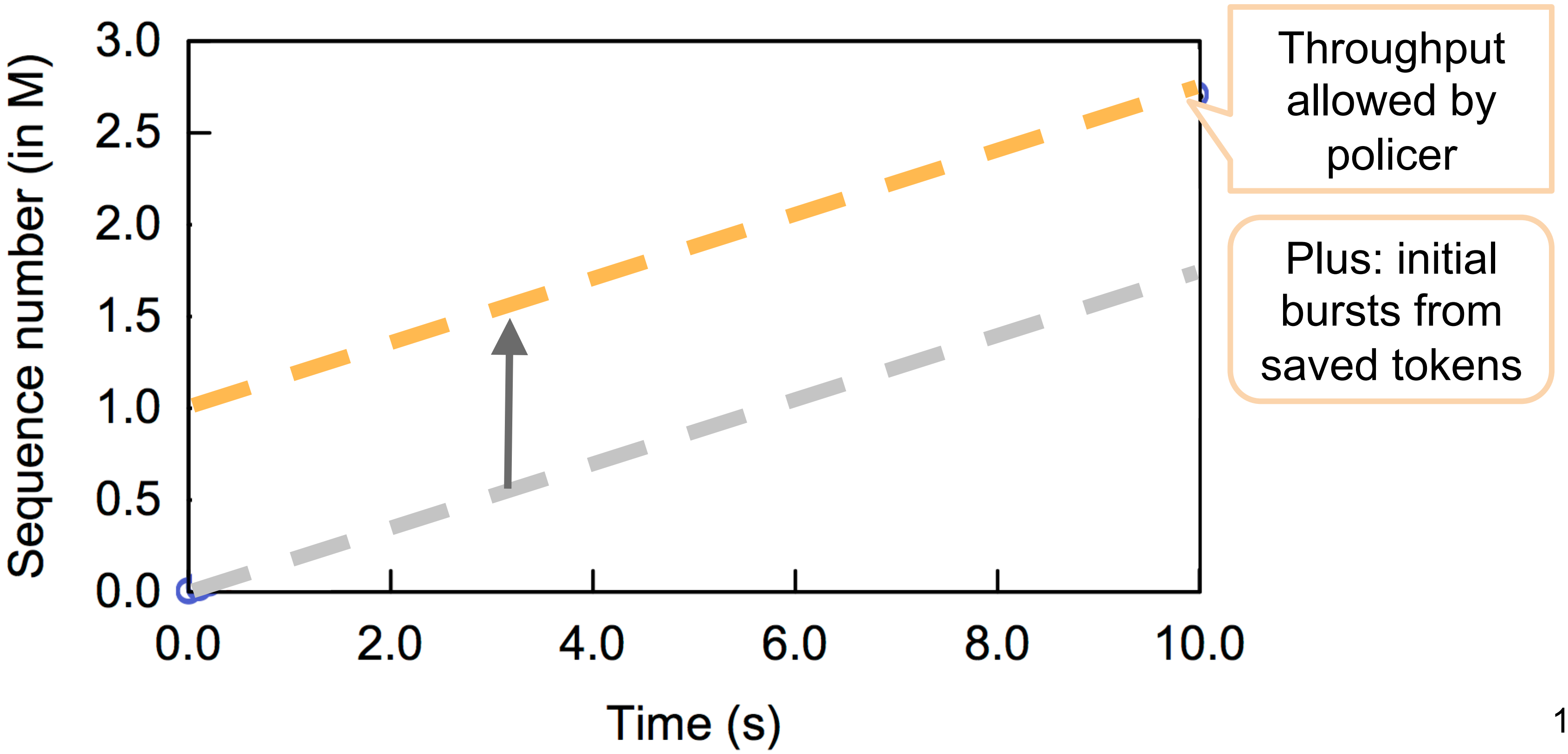
How Policing Works



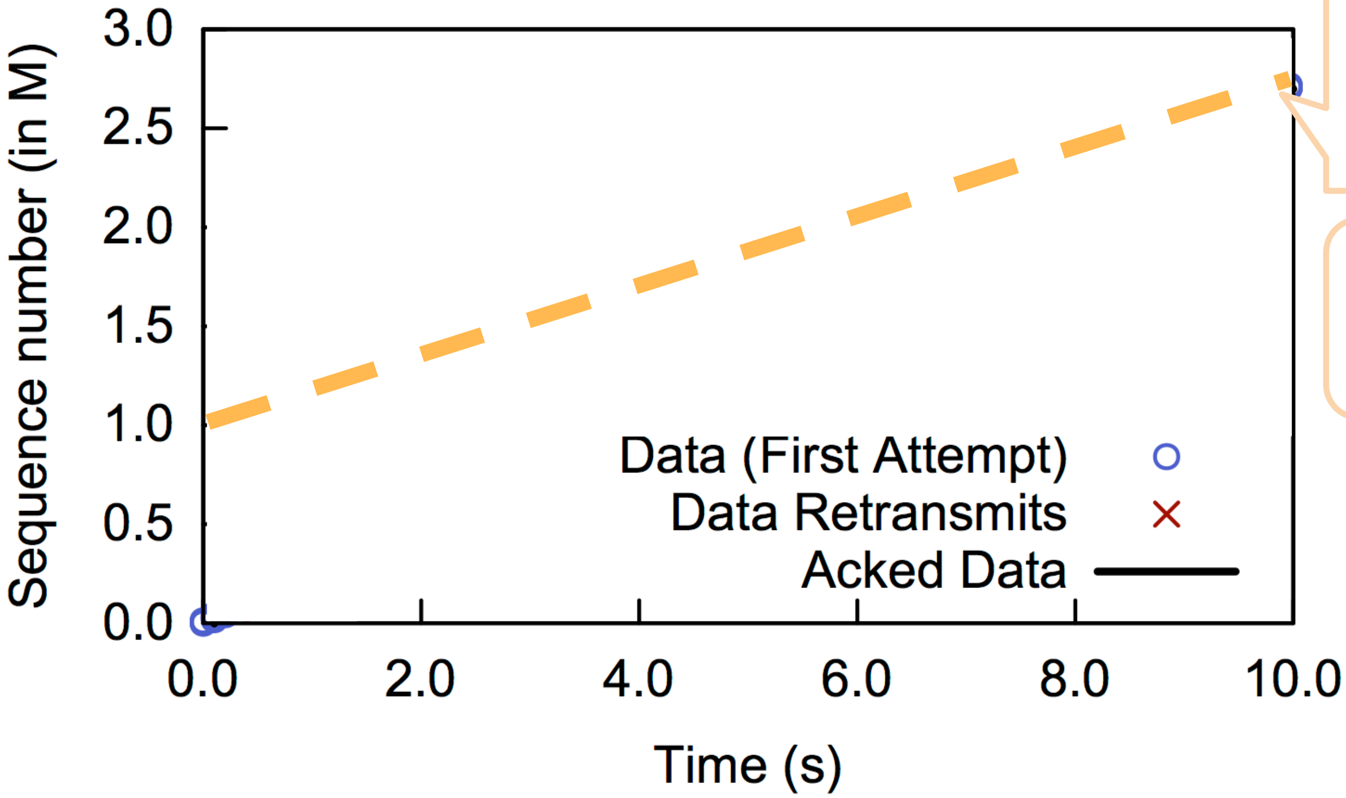
Policing in Action



Policing in Action



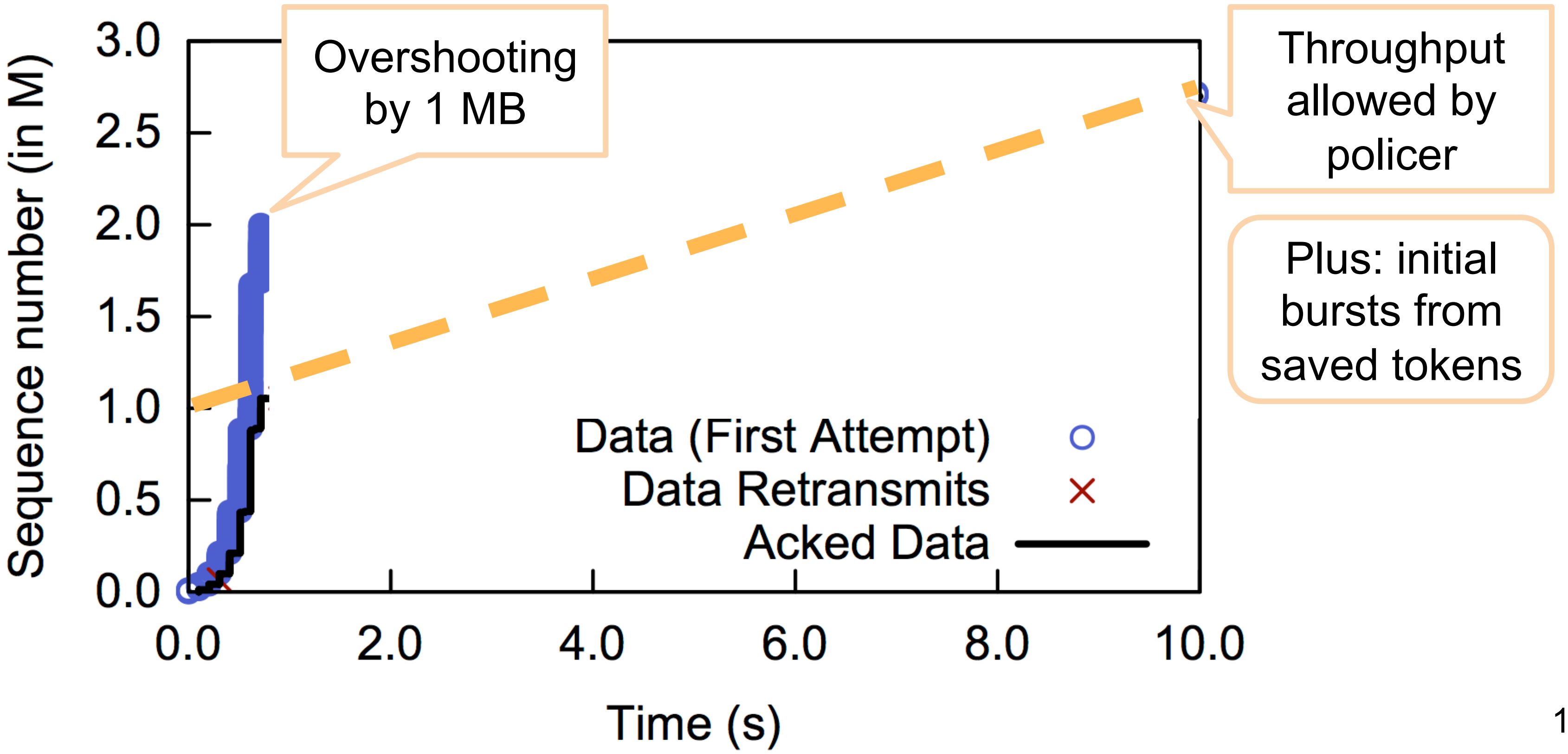
Policing in Action



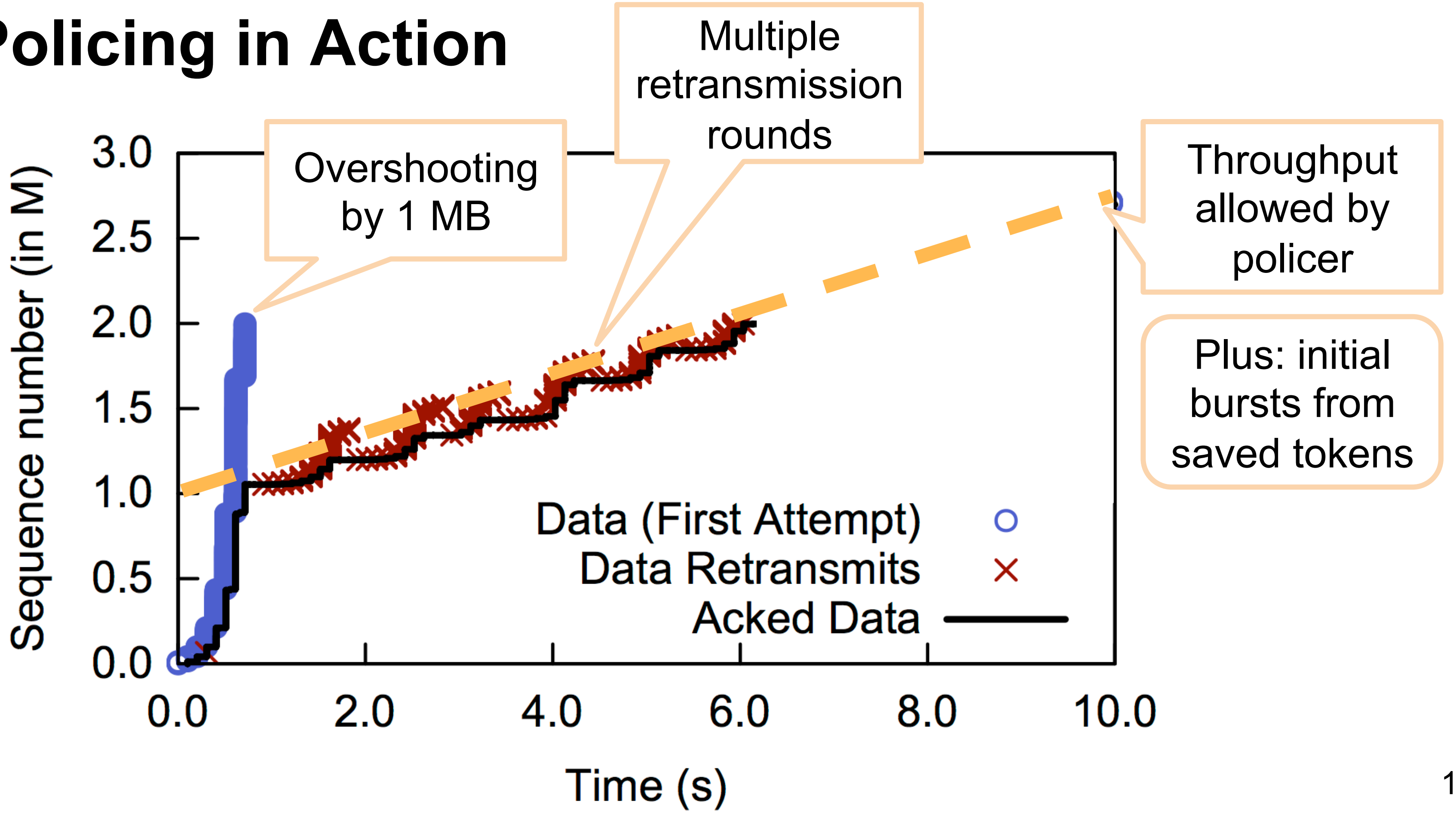
Throughput allowed by policer

Plus: initial bursts from saved tokens

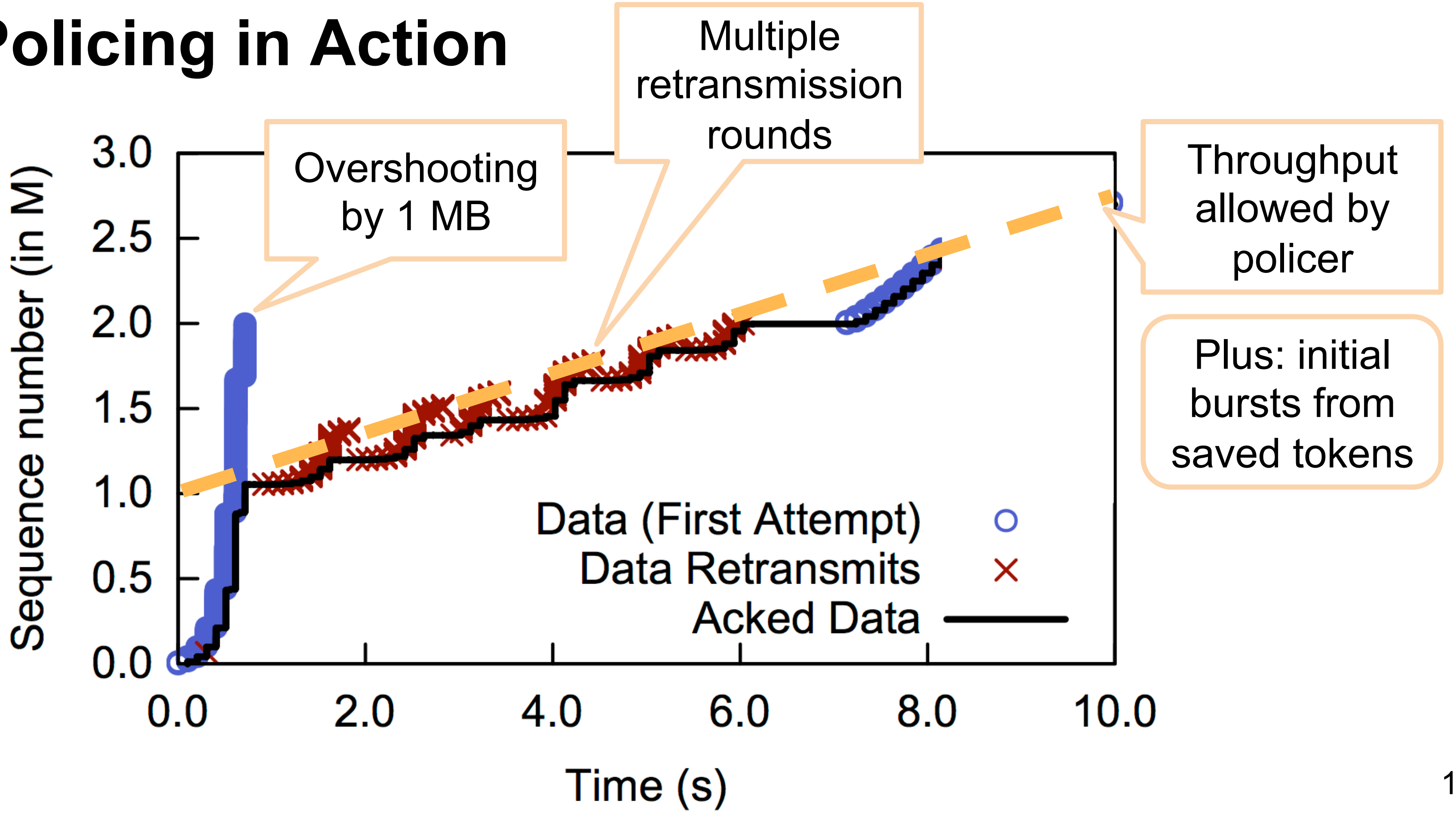
Policing in Action



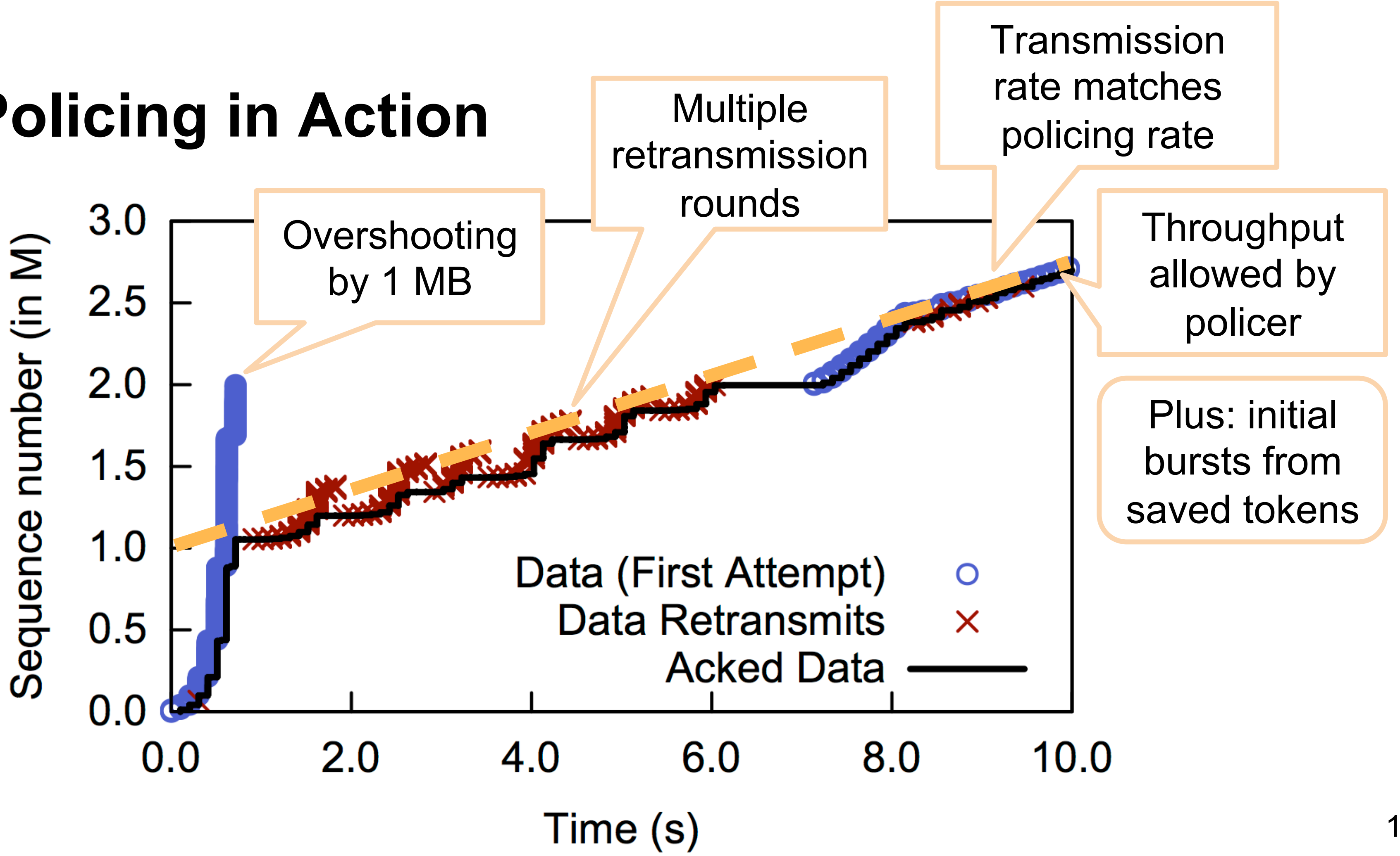
Policing in Action



Policing in Action



Policing in Action



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

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

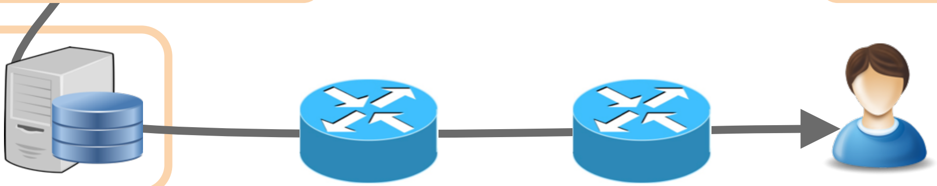
Collect packet traces for sampled client connections at most Google frontends

Analysis Pipeline

HTTP Response

Application metrics

Collect packet traces

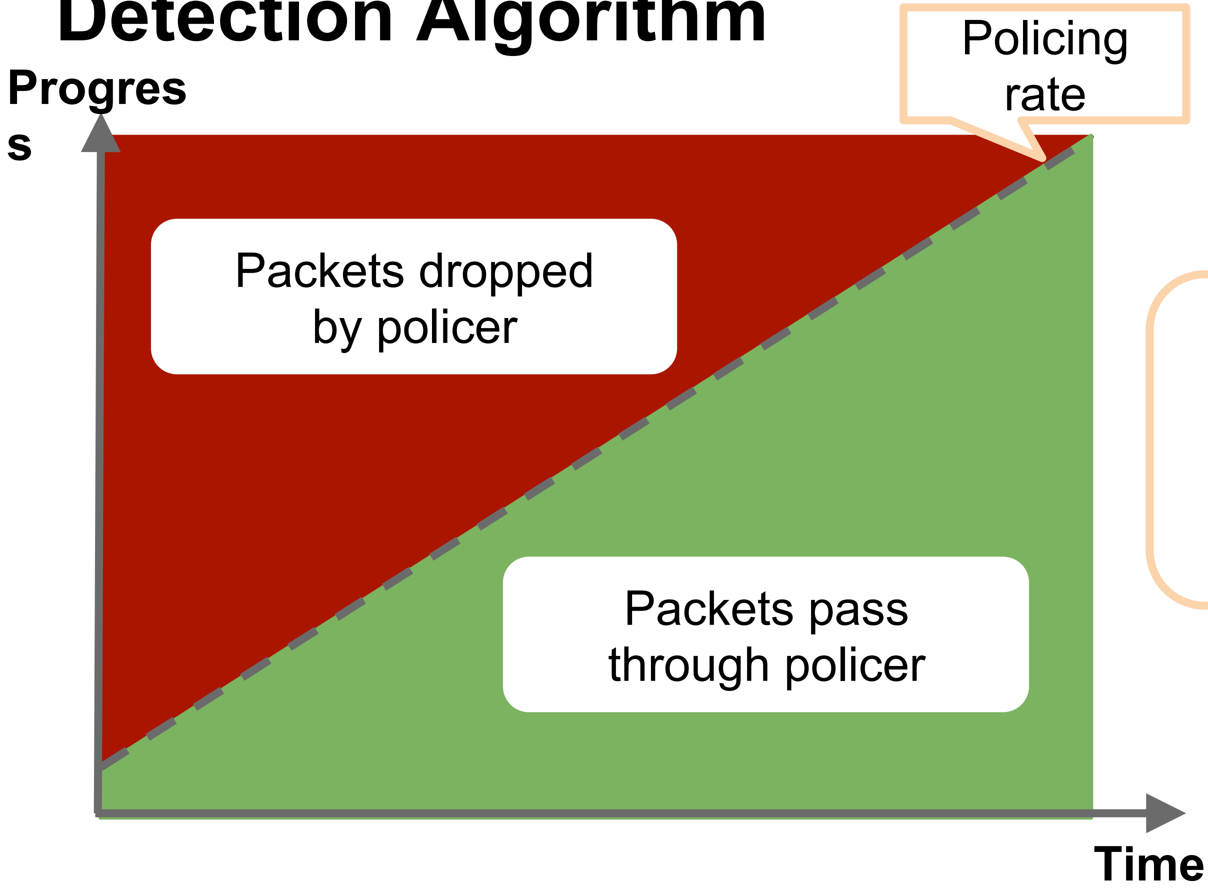


Forward samples to analysis backend

Detect policing

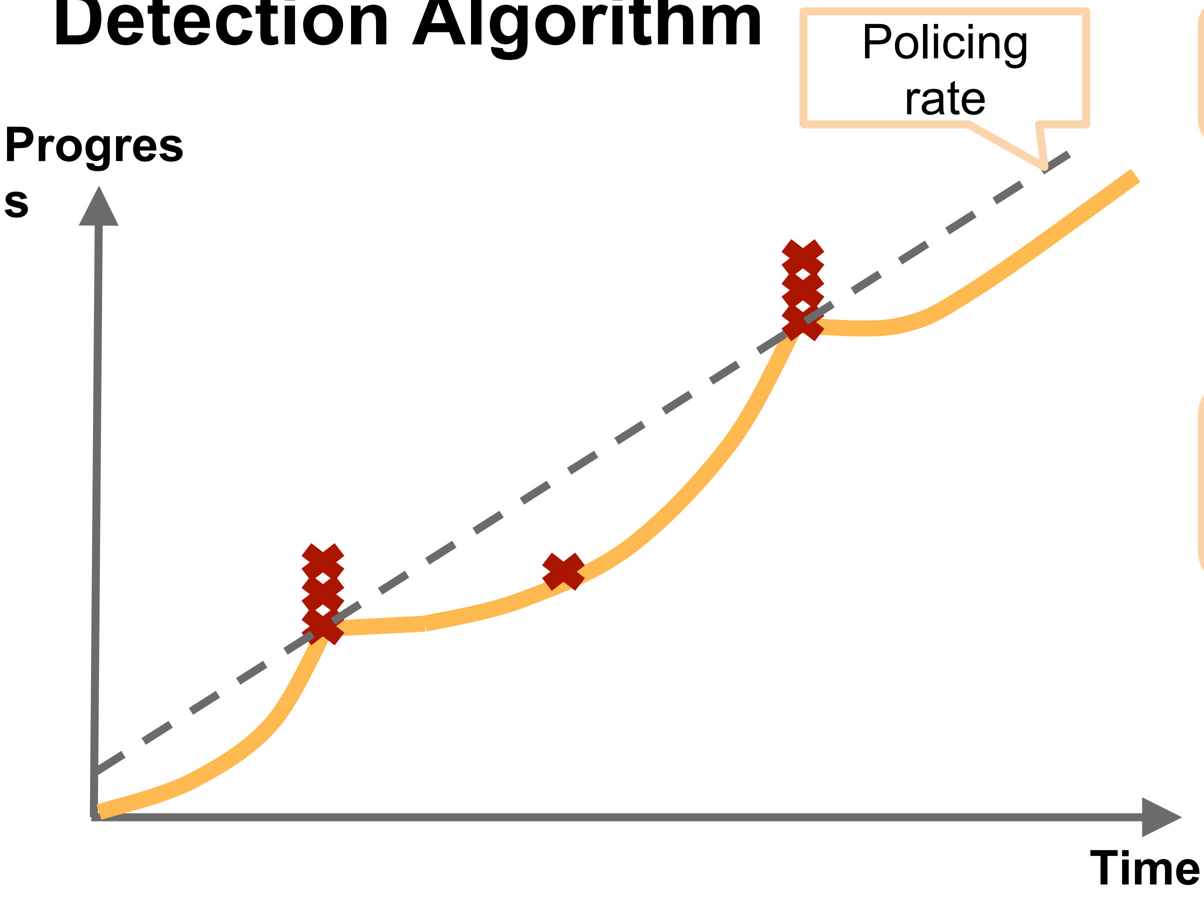
Cross-reference with application metrics

Detection Algorithm



Packets are always dropped when crossing the "policing rate" line

Detection Algorithm



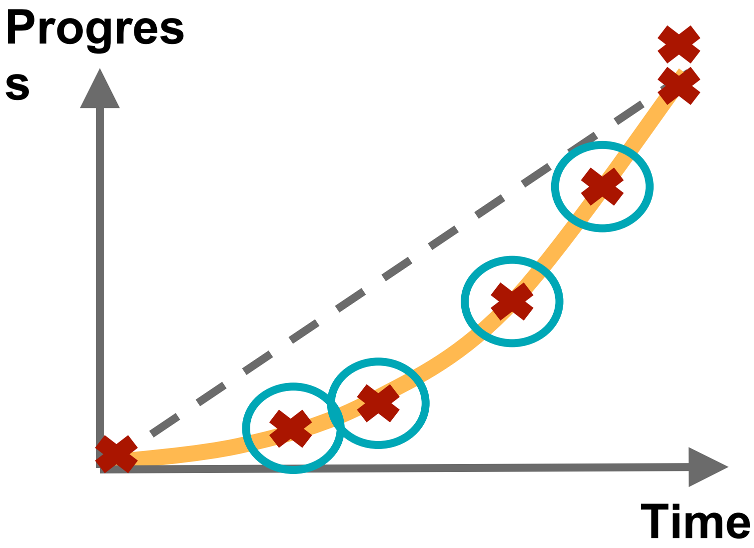
1 Find the policing rate

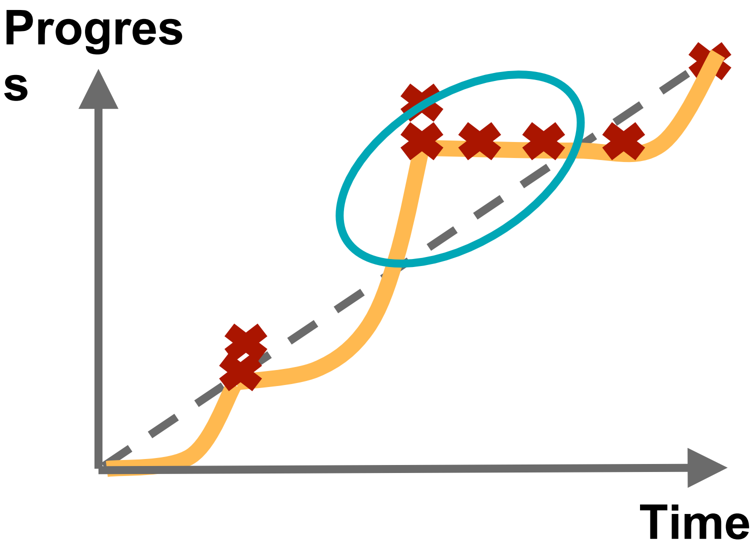
- Use measured throughput between an early and late loss as estimate

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

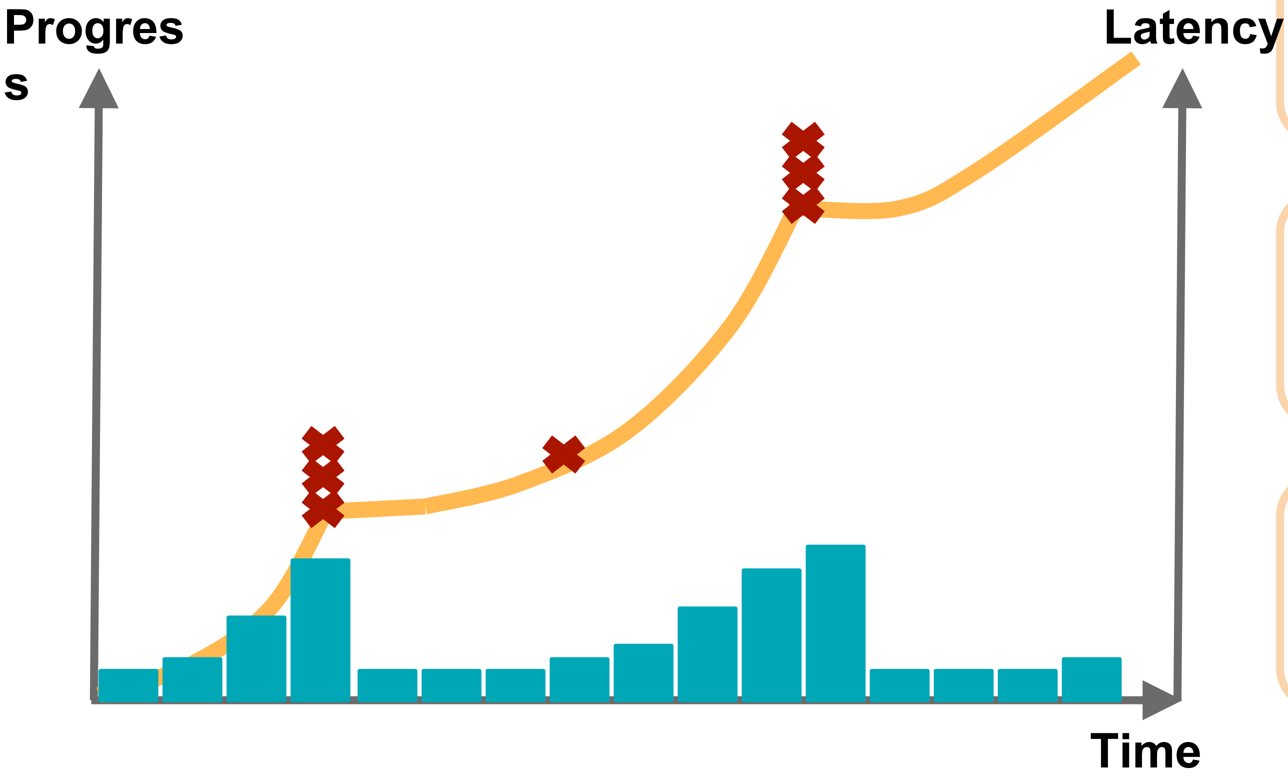




But: Traffic below policing rate should go through

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

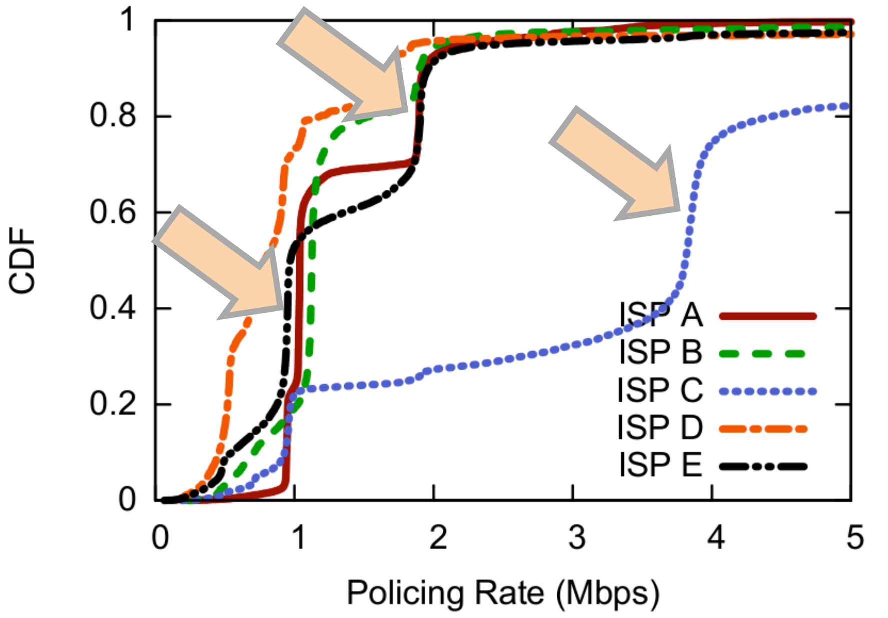
Generated test traces covering common reasons for dropped packets

- Policing (used a router with support for policing)
- Congestion
- Random loss
- Shaping

High accuracy for almost all configurations (see paper for details)

- Policing: 93%

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

#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

Lossy:
15 losses or more per segment

Up to 7% of lossy segments are policed

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

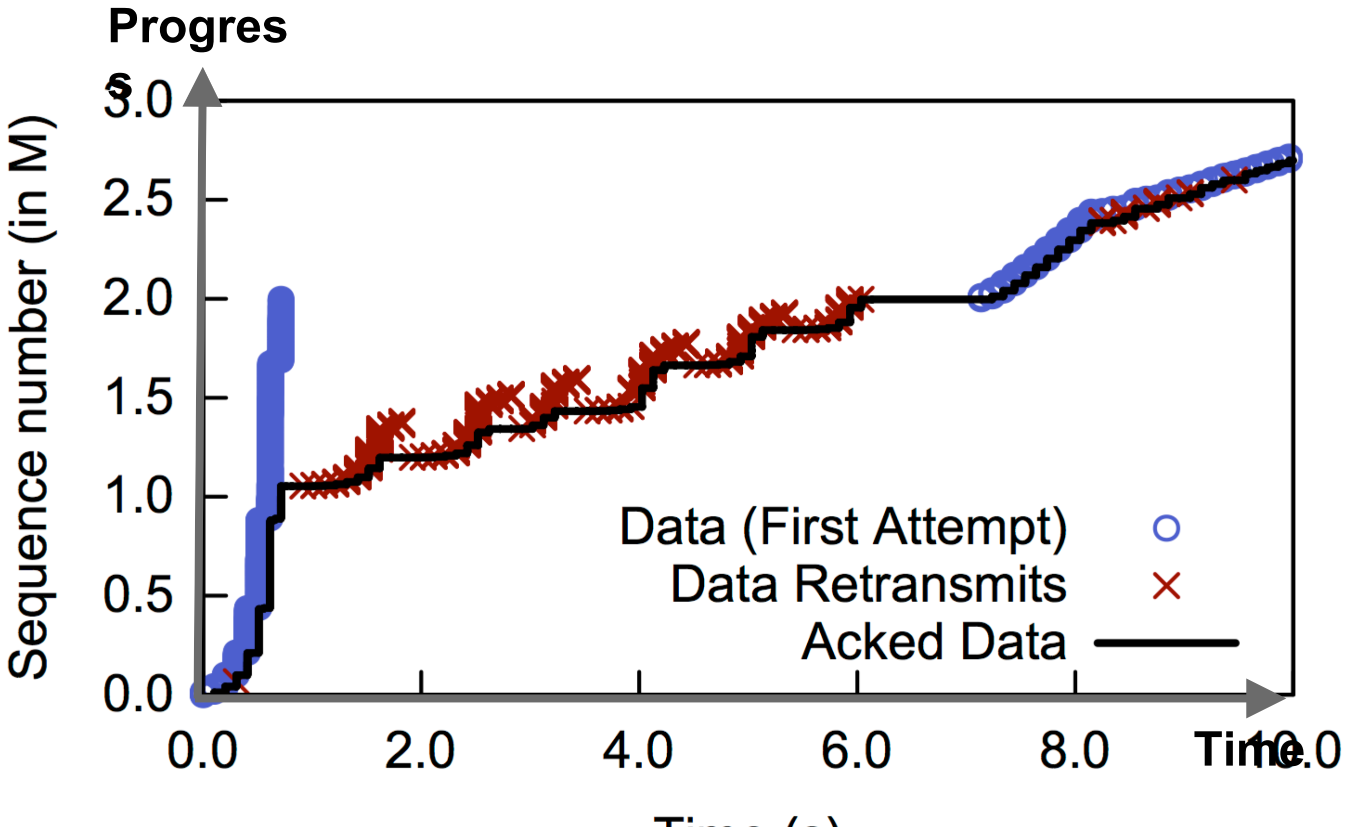
Lossy:
15 losses or more per segment

Up to 7% of lossy segments are policed

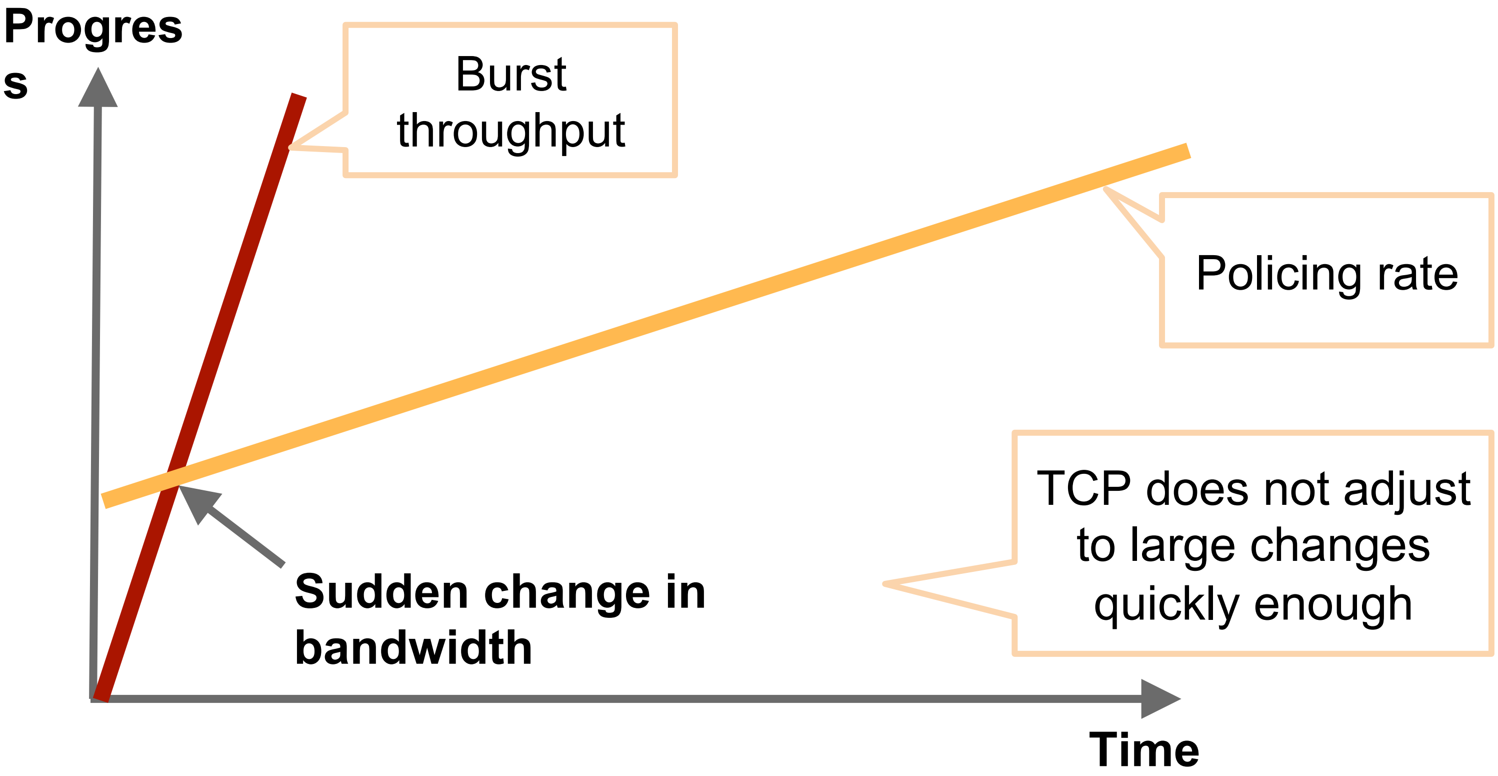
Average loss rate increases from 2% to over 20% when policed

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



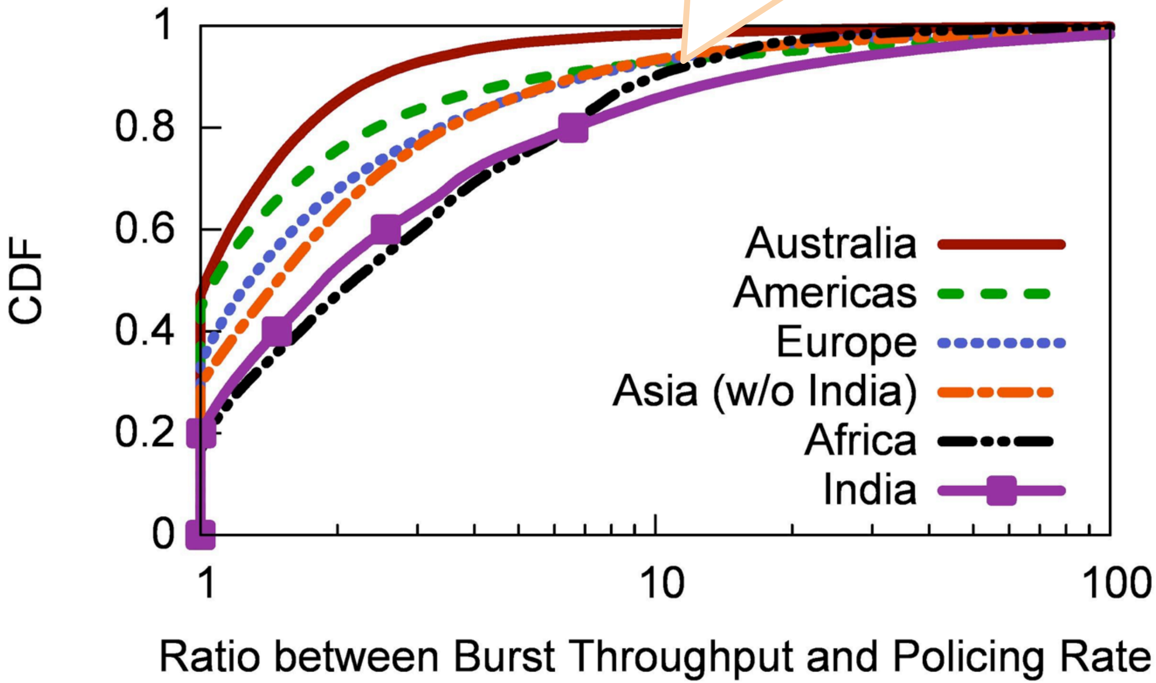
#3: Burst Throughput vs. Policing

90th percentile:
Policing rate is 10x lower than burst throughput

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



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).

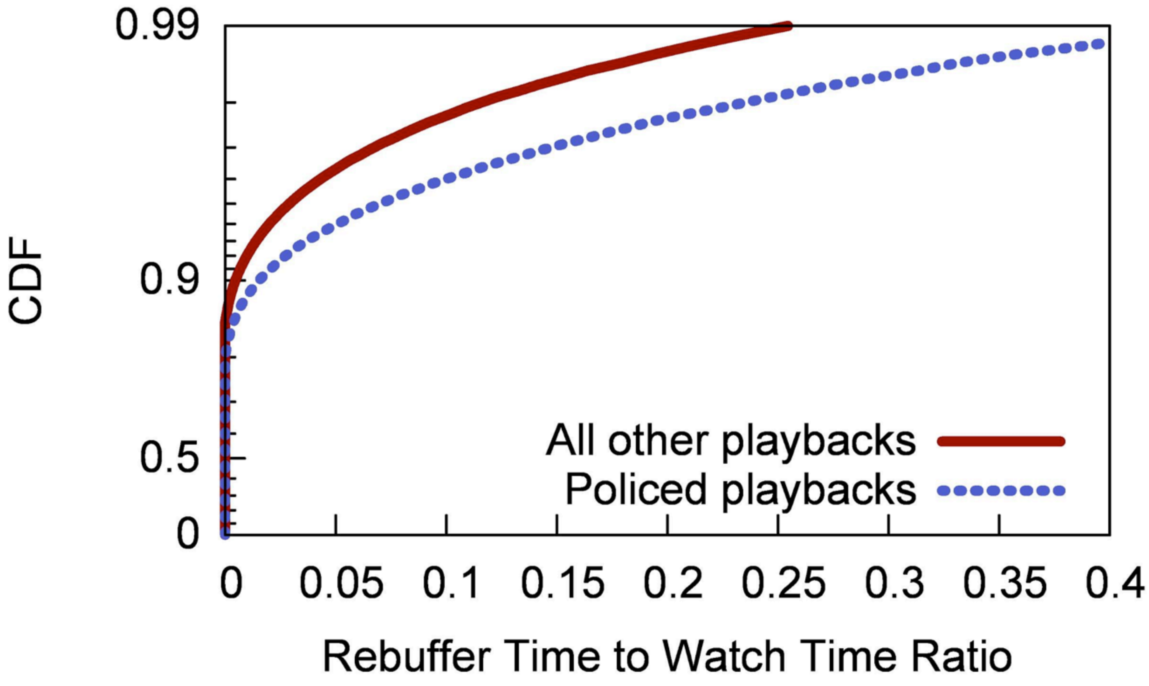
#4: Impact on Quality of Experience

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
Reduce losses during recovery	
Model policers	
Share and incorporate data plan information	

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 throughput high but queue low
- On loss: reduce to current delivery rate but reprobe quickly

[1] BBR: congestion-based congestion control.
Cardwell, Cheng, Gunn, Hassas Yeganeh, Jacobson, [ACM Queue, Oct 2016]()

How BBR Models Policers

BBR explicitly models the presence and throughput of policers

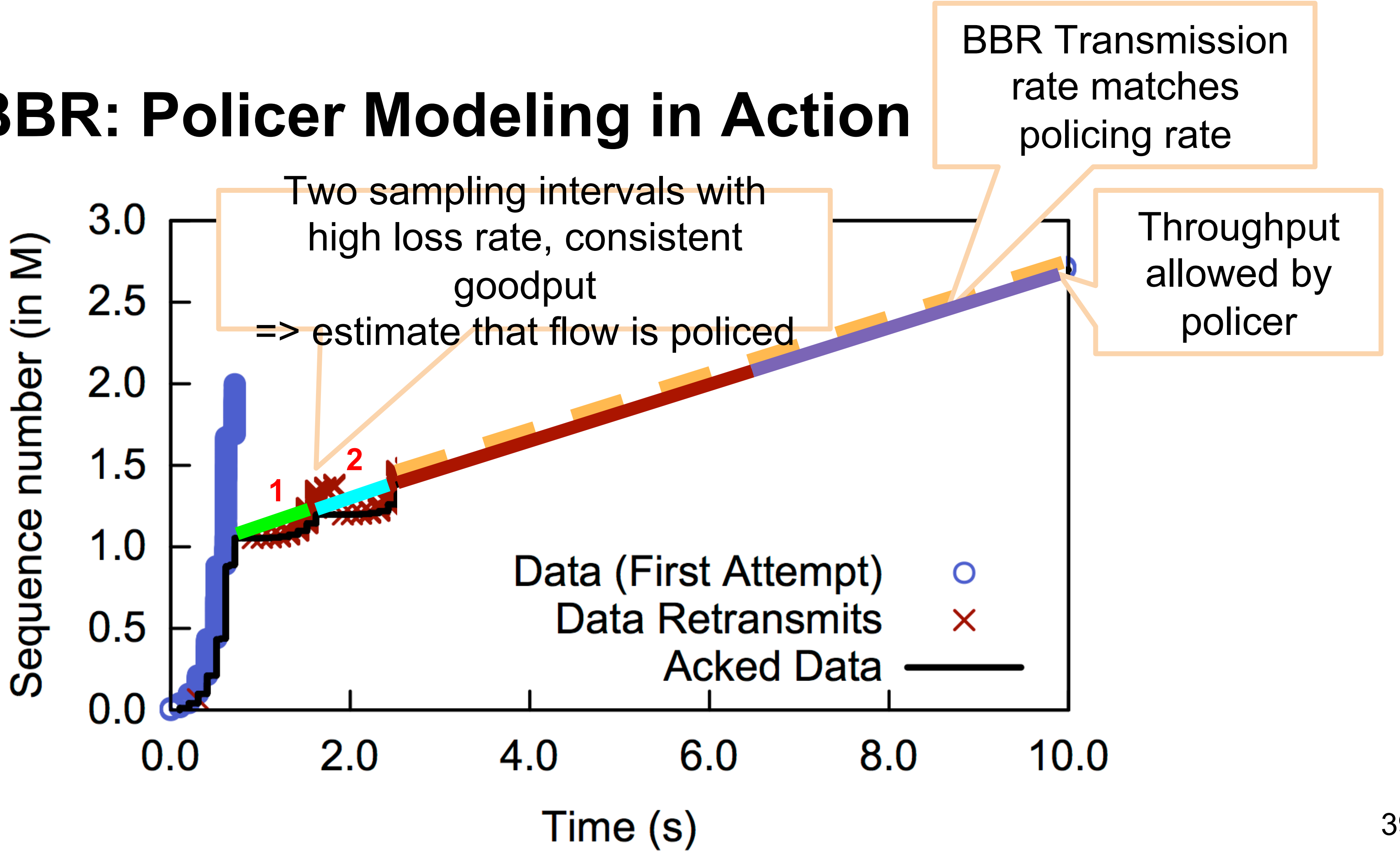
Long-term sampling intervals (4 - 16 round trips)

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

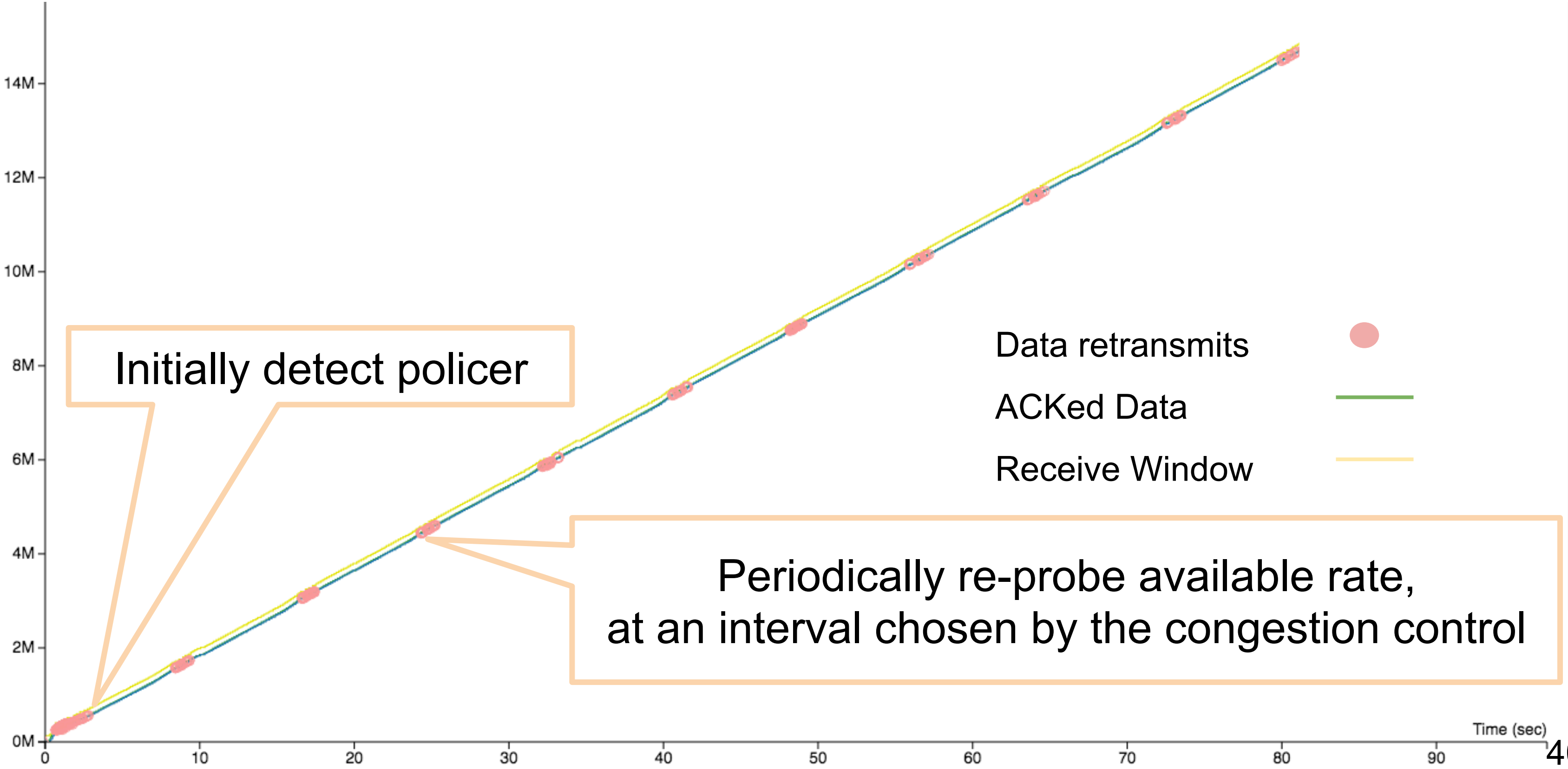
If two consecutive intervals with loss rates >= 20% and throughputs within 12.5% or 4 Kbps of each other) then:

- 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



BBR: a Policed YouTube Trace



Share and Incorporate Data Plan Information

Content providers are unaware of policer configurations

Mobile Data Plan API: Applications can request information about the user's data plan from the mobile network operator

Content providers can incorporate knowledge about a data plan to improve quality of experience

→ Example: YouTube can customize video quality / transmission rates if policing rate is known beforehand

If you would like to know more, email data-plan-api@google.com.

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

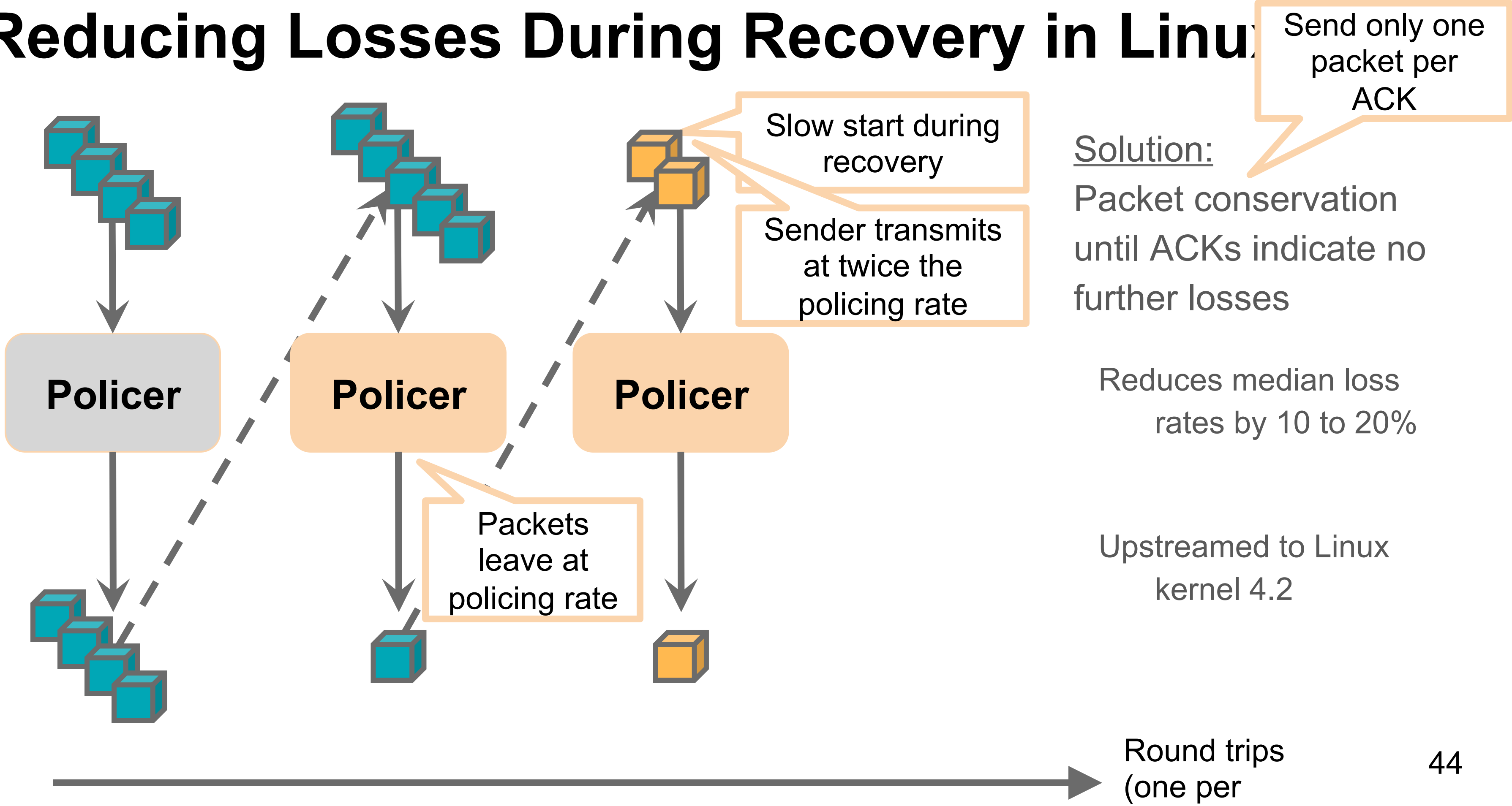
- Content providers: Rate limiting, pacing, reduce losses during recovery, model

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

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

Backup Slides

Reducing Losses During Recovery in Linux



Solution:
Packet conservation until ACKs indicate no further losses

Reduces median loss rates by 10 to 20%

Upstreamed to Linux kernel 4.2

Prevention of Loss During Recovery

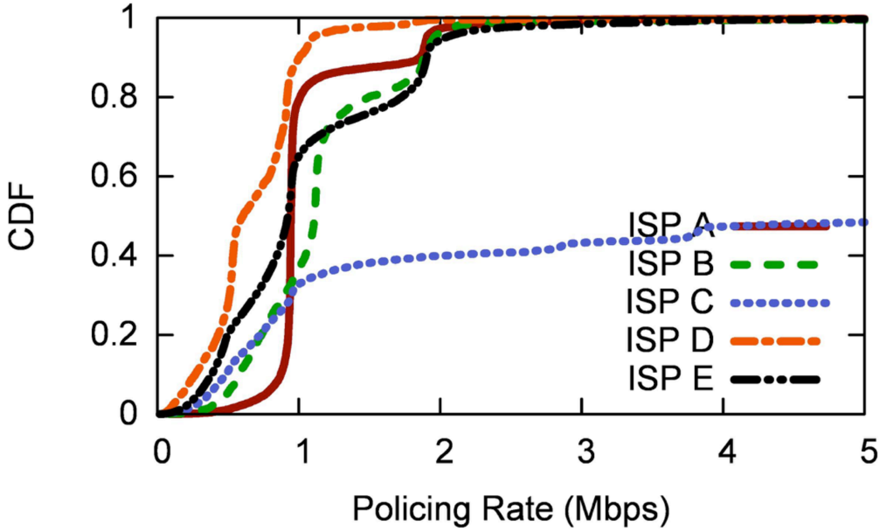
Heavy losses can occur even when transmissions have no bursts
→ steady stream of data

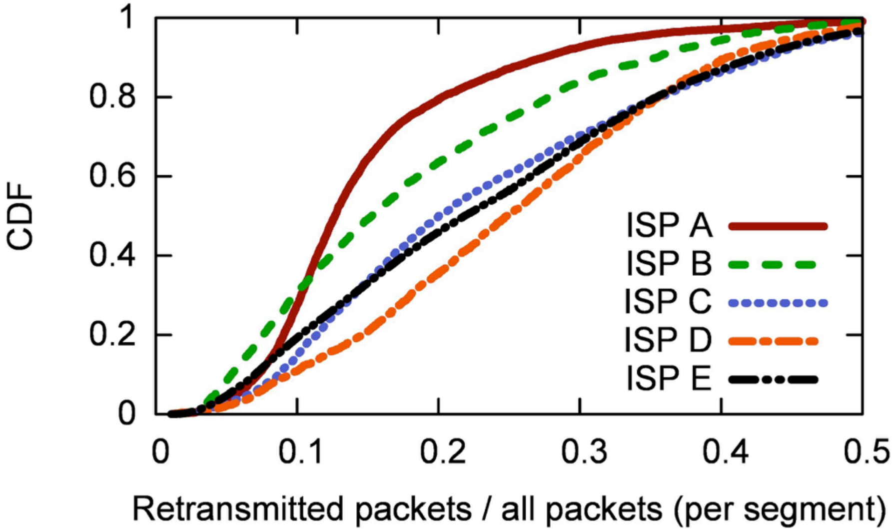
When the rate limit is reached, the policer lets packets through at the maximum allowed rate

But: Linux's recovery mechanism can trigger slow start → send two packets for every ACK
→ Sender transmits at twice the policing rate

Adjust mechanism to use packet conservation in the initial recovery round and only slow start if no retransmission is lost

ISP-specific Findings



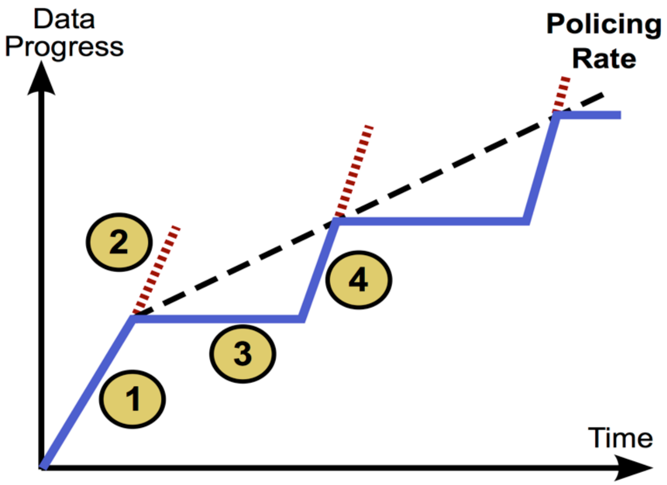


Each ISP enforces only a handful of policing rates that we (mostly) matched to advertised data plans

Median loss rates: 10 - 25%
95th percentile: 30 - 45%

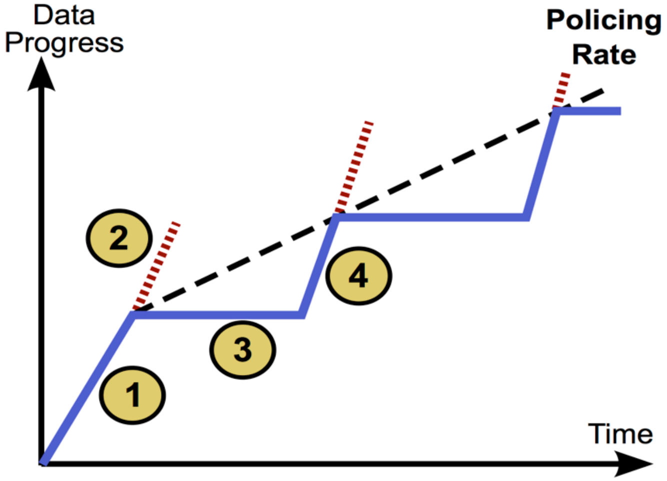
ISPs: (A) Azerbaijan, (B) US, (C/D) India, (E) Algeria

Interaction with TCP Congestion Control



(1) Bucket filled
→ unbounded throughput
(2) Bucket empty → bursty loss
(3) Waiting for timeout
(4) Repeats from (1)

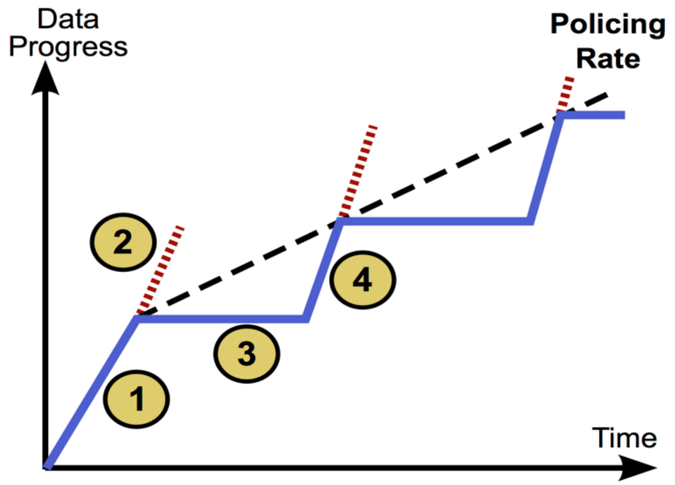
Interaction with TCP Congestion Control

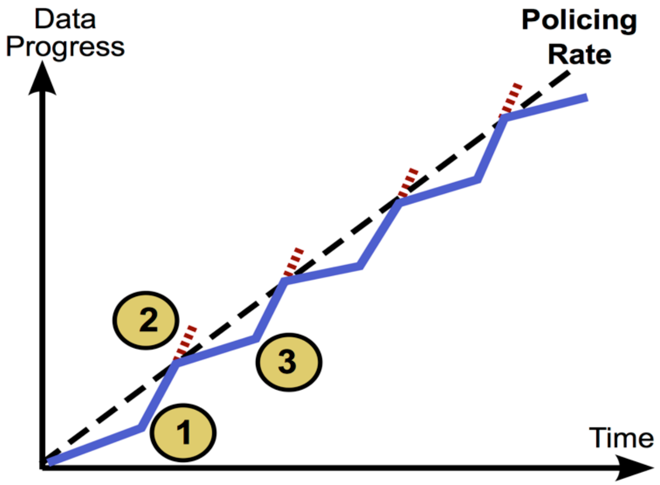


Staircase pattern

High goodputs followed by heavy losses and long timeouts

Interaction with TCP Congestion Control



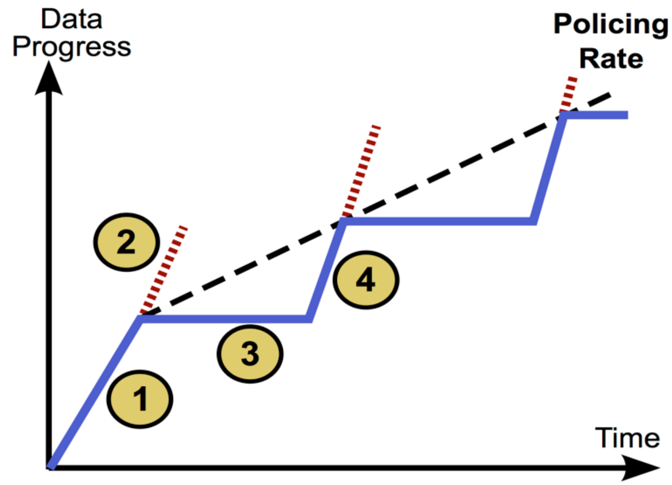


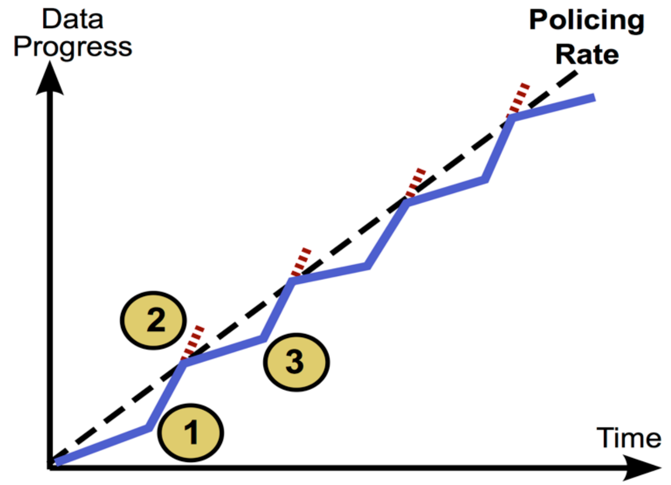
Staircase pattern

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

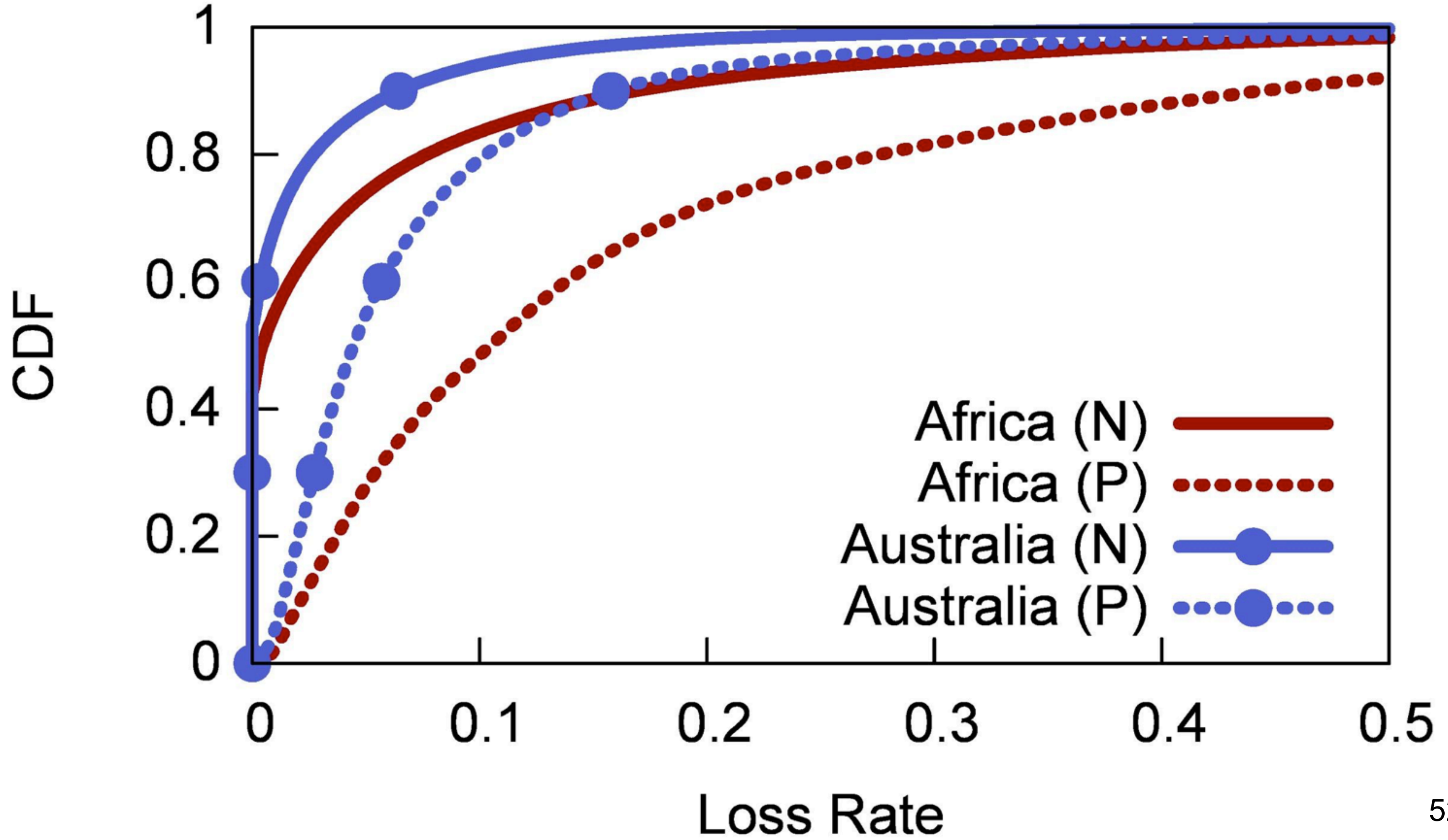
High goodputs followed by heavy losses and long timeouts

Doubling window pattern

Flipping between rates since connection cannot align with policing rate

Prevalence

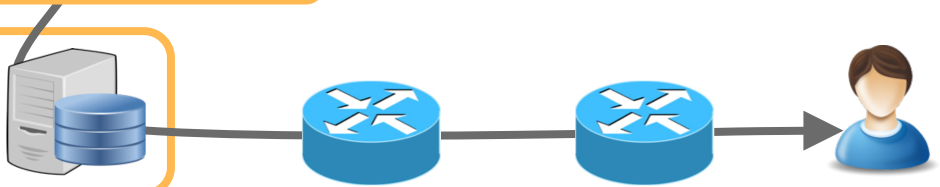
Region	Policed (among lossy)	Policed (overall)	Loss (policed)	Loss (non-policed)
Africa	6.2%	1.3%	27.5%	4.1%
Asia	6.6%	1.3%	24.9%	2.9%
Australia	2.0%	0.4%	21.0%	1.8%
Europe	5.0%	0.7%	20.4%	1.3%
North America	2.6%	0.2%	22.5%	1.0%
South America	4.1%	0.7%	22.8%	2.3%


CDF
Loss Rate
0
0.2
0.4
0.6
0.8
1
0
0.1
0.2
0.3
0.4
0.5
Africa (N)
Africa (P)
Australia (N)
Australia (P)

Understanding Policing

HTTP Response

Collect packet traces



Forward samples to analysis backend

Handles over 30 billion packets daily

Derive basic features

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

Apply policing detection heuristic

Store & query aggregate results

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)

Introduction

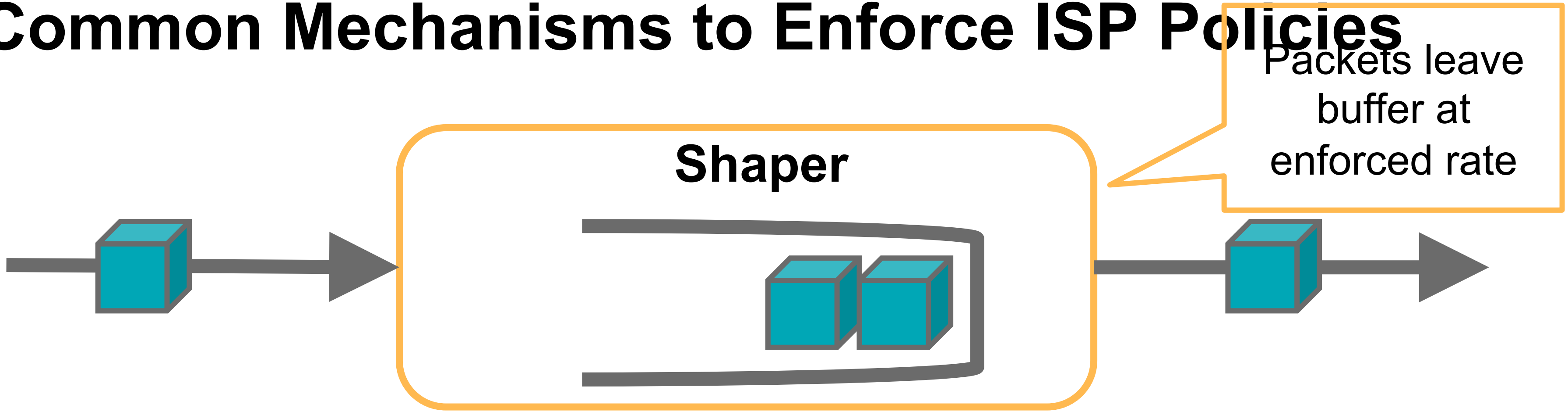
Exponential growth of video traffic

Netflix and YouTube streams account for ~ 50% of traffic in North America

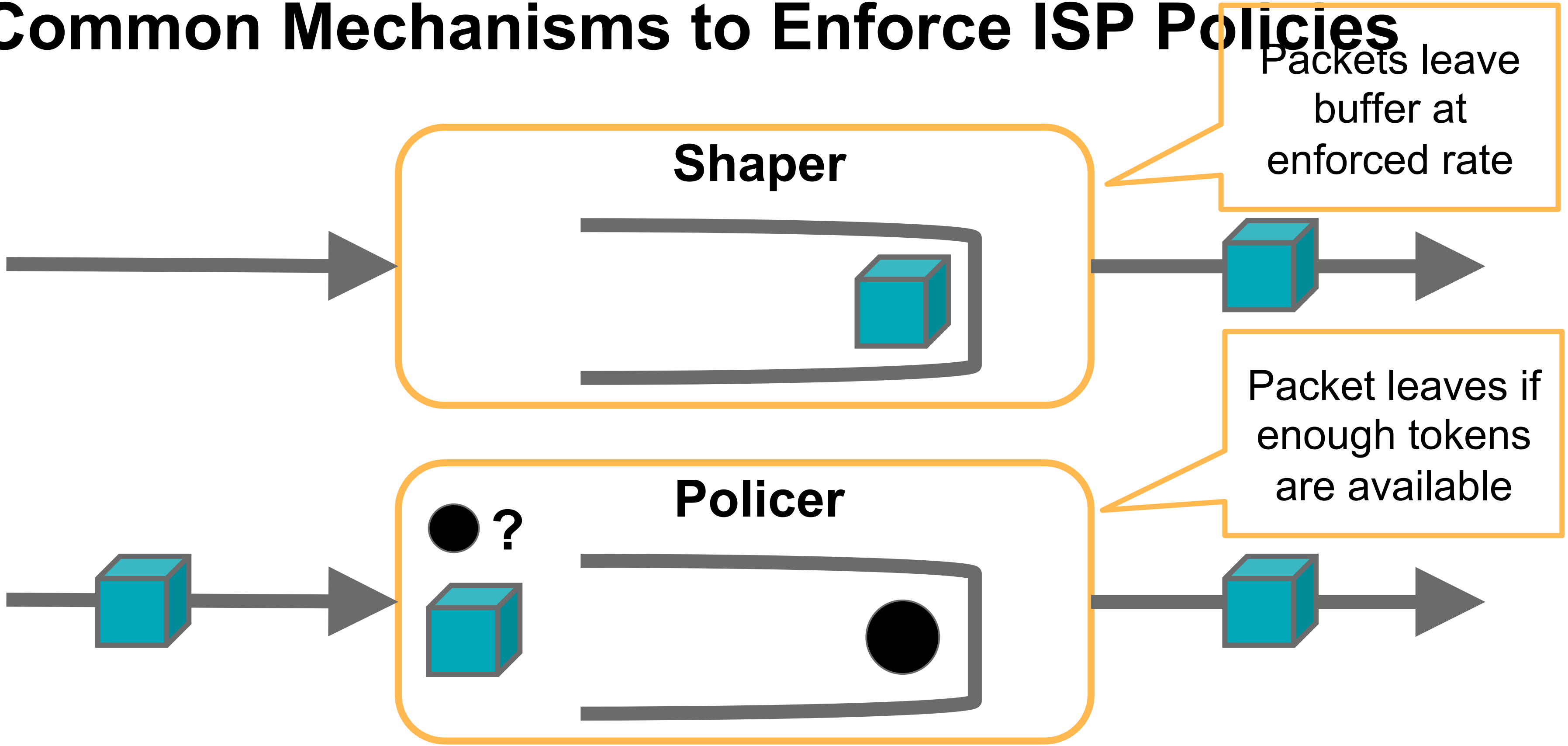
Goal for content providers: maximize quality of experience (QoE)

Competing goal for ISPs: accommodate multitude of services and policies → Traffic Engineering

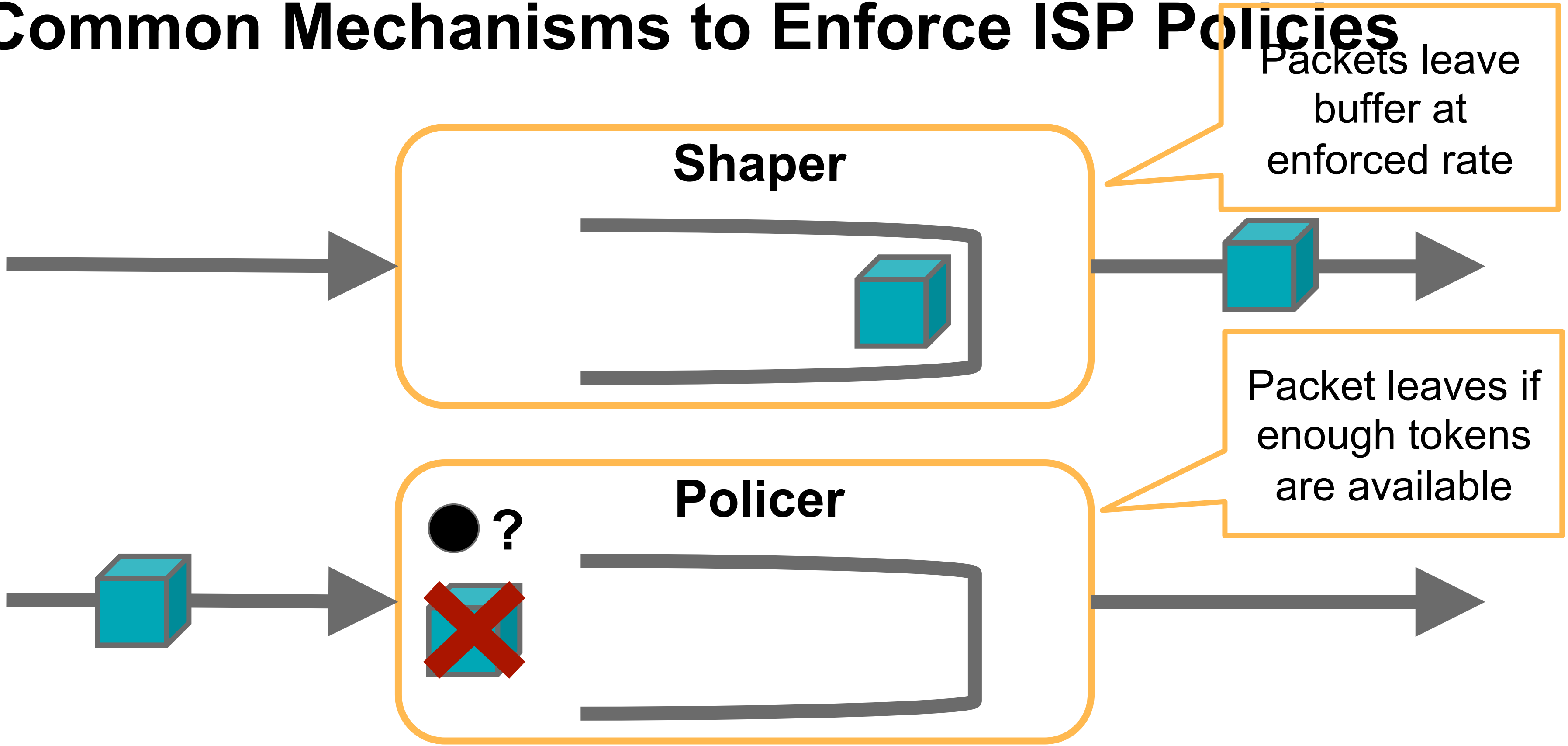
Common Mechanisms to Enforce ISP Policies



Common Mechanisms to Enforce ISP Policies



Common Mechanisms to Enforce ISP Policies



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

Understanding Policing

1. How prevalent is policing on the Internet?

   Contribution: Global-scale collection and analysis of packet traces from communication between Google frontends and users.

2. How does it impact application delivery and user quality of experience?

   Contribution: Cross-referencing of policed traces with YouTube application metrics.

3. How can content providers mitigate adverse effects of policing, and what alternatives can ISPs deploy?