Media Streaming, Application Informed Pacing, and Coexistence with Cloud Gaming

Grenville Armitage, Netflix

IEEE LCN 2025, Sydney, Australia
14 October 2025

First, a bit about me

My relationship with IEEE LCN started in 1993

18th Conference on
Local Computer Networks

September 19-22, 1993

Minneapolis, Minnesota

ATM & SMDS
Chair: G. Kessler

Using the Common LAN to Introduce ATM Connectivity
G.J. Armitage and K.M. Adams

Abstract.

This paper outlines a method for using LAN technologies to transport Asynchronous Transfer Mode (ATM) cells. B-ISDN service requirements are broken into three groups - high speed media, multi-media interfaces, and service control software. Compression techniques for

My relationship with IEEE LCN started in 1993

18th Conference on
Local Computer Networks

September 19-22, 1993

Minneapolis, Minnesota

ATM & SMDS

Chair: G. Kessler

Using the Common LAN to Introduce ATM Connectivity

G.J. Armitage and K.M. Adams

Abstract.

This paper outlines a method for using LAN technologies to transport Asynchronous Transfer Mode (ATM) cells. B-ISDN service requirements are broken into three groups - high speed media, multi-media interfaces, and service control software. Compression techniques for

Totally *cool*

ATM stuff 😎

My relationship with IEEE LCN started in 1993

18th Conference on
Local Computer Networks

September 19-22, 1993

Minneapolis, Minnesota

ATM & SMDS

Chair: G. Kessler

Using the Common LAN to Introduce ATM Connectivity

G.J. Armitage and K.M. Adams

Abstract.

This paper outlines a method for using LAN technologies to transport Asynchronous Transfer Mode (ATM) cells. B-ISDN service requirements are broken into three groups - high speed media, multi-media interfaces, and service control software. Compression techniques for

…followed by 17 more co-authored papers scattered unevenly across 2005 to 2025

Industry R&D (1994-2001)

Member Technical Staff
Bell Labs Research Silicon Valley, Lucent Technologies
Sep 1999 - Feb 2001 · 1 year 6 months
Palo Alto, California, USA

Product Marketing Director
Data Networking Systems business unit, Lucent Technologies
1998 - 1999 · 1 year
New Jersey, USA

Member of Technical Staff
Bell Labs Research, Lucent Technologies
1997 - 1999 · 2 years
New Jersey, USA

Senior Scientist/Research Scientist
Applied Research, Bellcore
Jul 1994 - 1997 · 3 years
Morristown, New Jersey, USA

Bellcore
Bell Communications Research

IETF

The ATM Forum

Lucent Technologies
Bell Labs Innovations

From LinkedIn (so it must be true!)

Academia (2002-2018)

Swinburne University of Technology

16 years 6 months

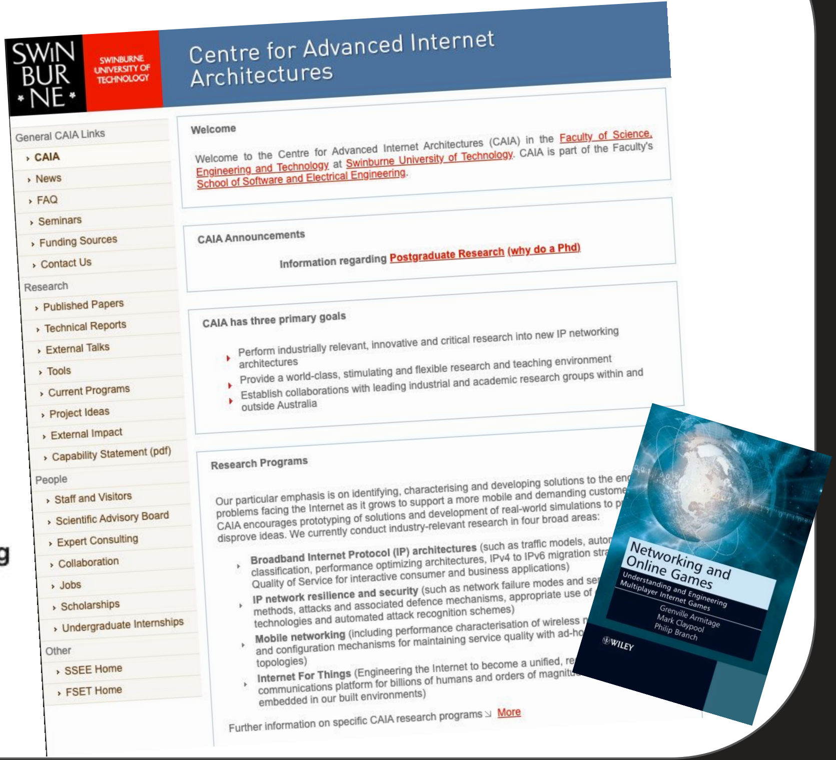
Professor (AU title), Telecommunications Engineering
Jan 2009 - Jul 2018 · 9 years 7 months
Melbourne, Australia

Head, Internet For Things (I4T) Research Group
Mar 2017 - Sep 2017 · 7 months

Director, Centre for Advanced Internet Architectures
Feb 2002 - Feb 2017 · 15 years 1 month

Head, Telecommunications Engineering Academic Group
Feb 2007 - Jan 2013 · 6 years

Associate Professor (AU title), Telecommunications Engineering
Feb 2002 - Dec 2008 · 6 years 11 months
Melbourne, Australia



Also from [LinkedIn]

And back to industry (2017 - present)

Engineering Manager, Netflix

"*I currently lead transport protocol R&D for Netflix Open Connect, [...] My team develops, refines and deploys the HTTP-based and WebRTC-based transport protocols Netflix servers use to stream content and deliver cloud gaming experiences..*"

Also from LinkedIn

Okay, new topic…

What if media streaming servers transmitted packets only as fast as needed?

Why is this even a question?

Shared networks, that's why.

Spoilers:

TCP pacing of streaming media helps coexistence with interactive apps

Paced streaming enables smoother cloud gaming experience

From: Scott Danahy

What if media streaming servers transmitted packets only as fast as needed?

Streaming content delivery is DASH-like

Server-side storage / client-side adaptation

Streaming content delivery is DASH-like

Server-side storage / client-side adaptation

Encode the content at multiple bitrates (creating a *rate ladder*)

Streaming content delivery is DASH-like

Server-side storage / client-side adaptation

Encode the content at multiple bitrates (creating a *rate ladder*)

Split each rung of rate ladder into segments ('chunks') representing 10s to 100s of video frames

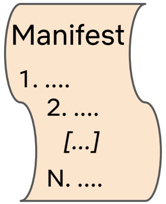
Segment 1	Segment 2	Segment 3	Segment 4	Segment 5	…	High quality
Segment 1	Segment 2	Segment 3	Segment 4	Segment 5	…	Mid quality
Segment 1	Segment 2	Segment 3	Segment 4	Segment 5	…	Low quality

Time

Streaming content delivery is DASH-like

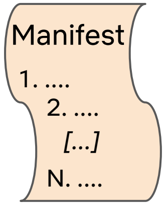
Server-side storage / client-side adaptation

Client is provided a manifest with the location of each chunk at each rung



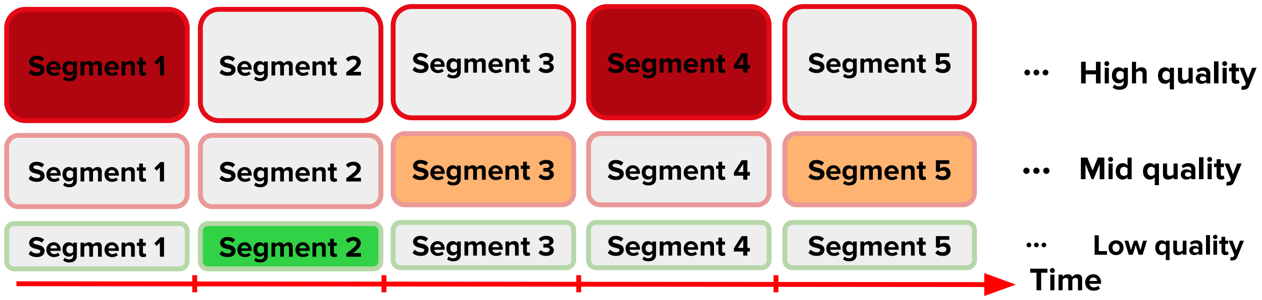
Streaming content delivery is DASH-like

Server-side storage / client-side adaptation



Client is provided a manifest with the location of each chunk at each rung

Client retrieves chunks on demand, and renders content



Streaming content delivery is DASH-like

Server-side storage / client-side adaptation

Client chooses next chunk from highest rung of rate ladder supportable by recent network conditions

Client infers network conditions from chunk delivery time

Streaming content delivery is DASH-like

Server-side storage / **client-side adaptation**

Chunk retrieval usually over TCP (or QUIC)

On-the-wire behaviour controlled by TCP (or QUIC)

Rinse, repeat

Chunk *delivery* (simplified)

<u>Client request</u>

Select next chunk,
Issue HTTP GET request to server.

<u>Server response</u>

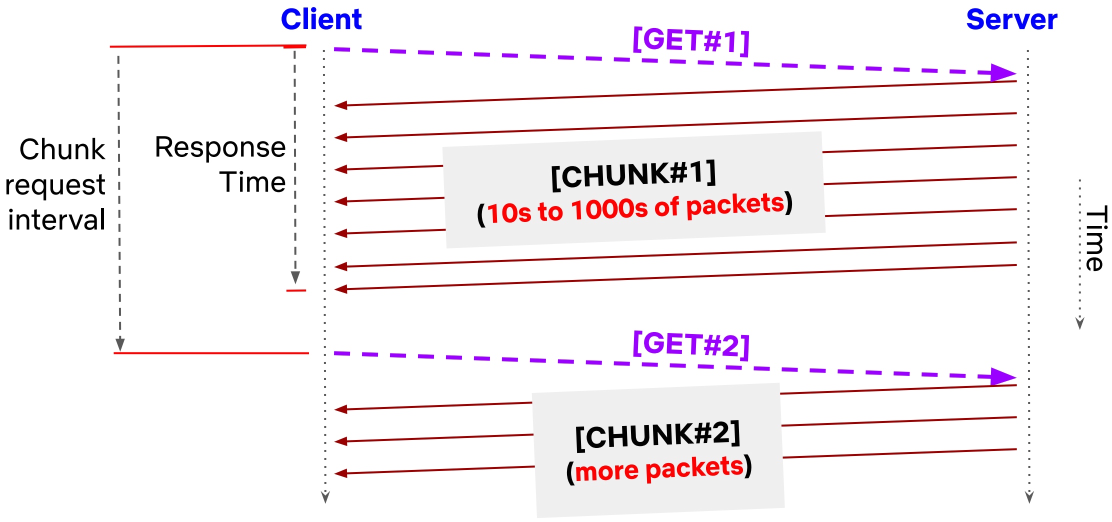
Server retrieves chunk from disk.

Send these bytes over TCP to client,

  Split into *TCP segments*,

  carried in *IP packets*.

(e.g. 2Mbyte video chunk → approx. 1500 packets)

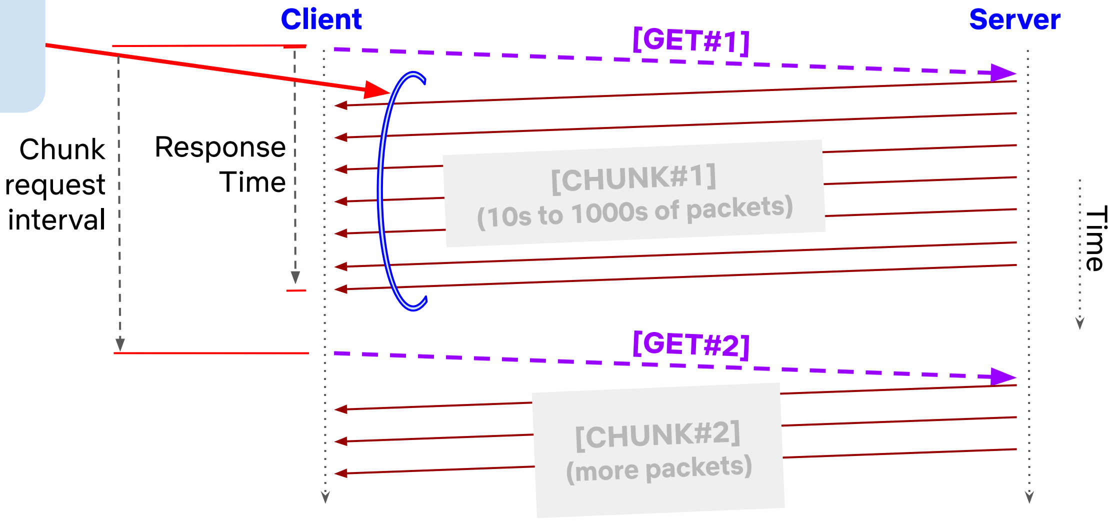


Client *experience* (simplified)

Server's transport stack controls intra-chunk packet transmission

Goal: Average response time less than a chunk's playback period.

Response time depends on path characteristics, competing traffic, and *TCP congestion/rate control algorithm*



Streaming content delivery is DASH-like

Server-side storage / **client-side adaptation**

"Rinse, repeat" is an oversimplification

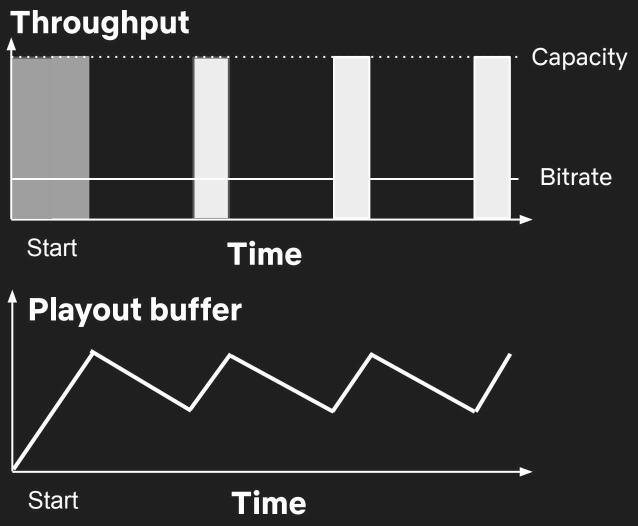
Clients queue up chunks in playout buffers
(a "*jitter buffer*" for chunks)

In steady-state: Retrieve new chunks on a regular cadence
as playout buffer drains

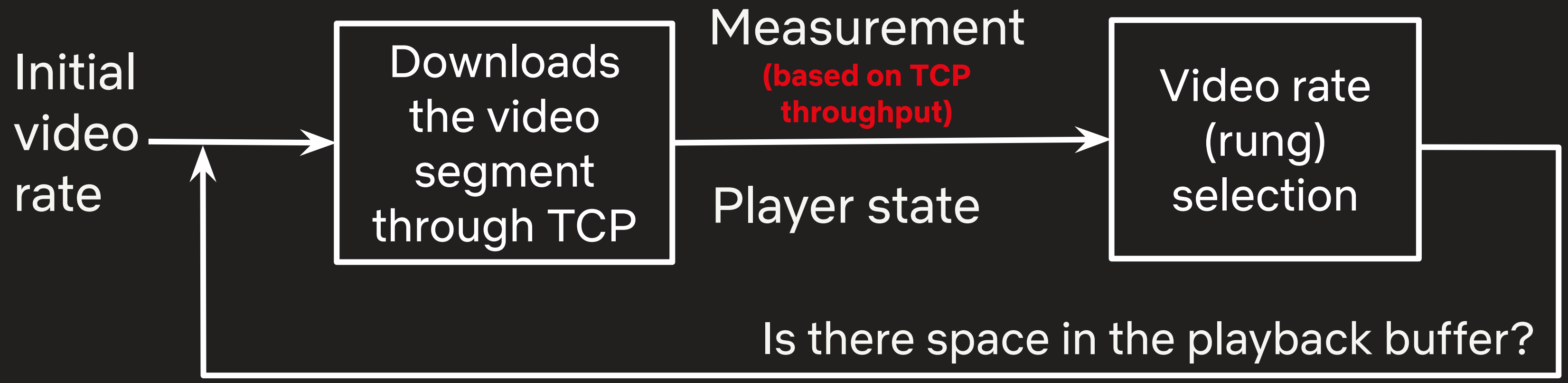
(re-)filling the playout buffer: Retrieve new chunks as fast as possible

Chunk retrieval vs playout buffer occupancy

Not smooth (Video traffic today)



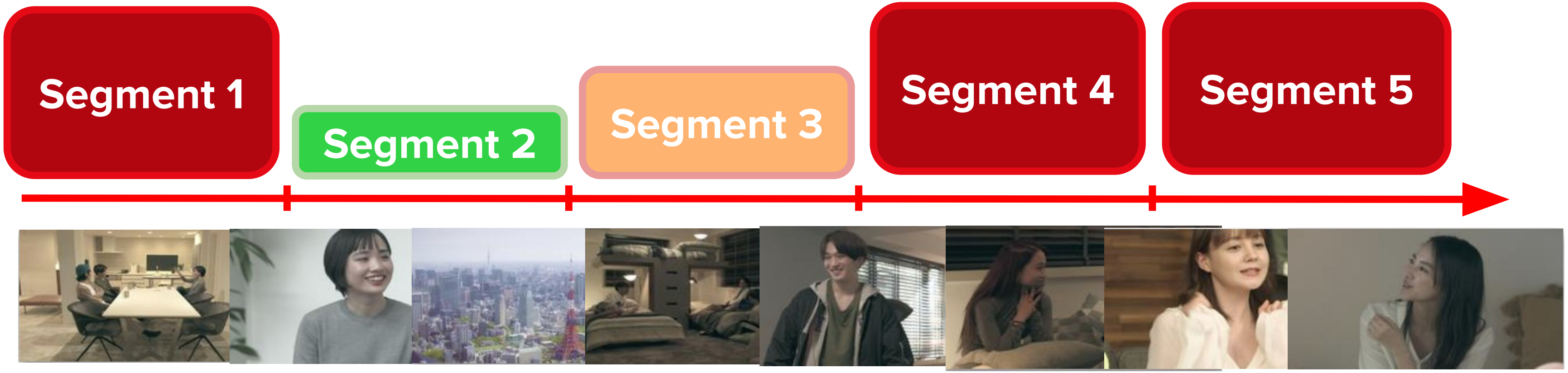
Nested control loops



Adaptive bitrate algorithm (ABR)

Each streaming service develops its own ABR to **optimize quality of experience (QoE)** based on **changing network conditions**

Resulting experience



Streaming content delivery is DASH-like

Server-side storage / **client-side adaptation**

Chunk retrieval usually over TCP (or QUIC)

On-the-wire behaviour controlled by TCP (or QUIC)

Rinse, repeat

"Fast as possible" video chunk delivery

Regular TCP is capacity seeking

→ Video chunks are transferred as fast as the network allows

→ Bursty at intra-chunk *and* inter-chunk timeframes

→ Induces RTT inflation
→ Induces packet losses

How did we get here?

TCP vs QUIC is <u>not</u> the issue

(to a first approximation)

TCP vs QUIC is <u>not</u> the issue

(to a first approximation)

Both are designed to send bytes

- **Reliably**
- **In order**
- **As fast as possible**

On-the-wire behaviour depends on your congestion (or rate) control algorithm

→ *NewReno* or *CUBIC* or *BBR* or …

(These algorithms turn up in TCP <u>and</u> QUIC stacks)

Why *capacity seeking* ?

Automagically adapt to variations in available capacity

Go **up** when possible (but not too far)

Go **down** when necessary (but not too far)

Why *capacity seeking* ?

Automagically adapt to variations in available capacity

Go **up** when possible (but not too far)

Go **down** when necessary (but not too far)

Expect no help from the network

How to do *capacity seeking* ?

Repeatedly push the network for a signal

How to do *capacity seeking* ?

Repeatedly push the network for a signal

Indirect signal: Packet losses, variations in transit delay

Direct signal: Explicit Congestion Notification (ECN)

How to do *capacity seeking* ?

Repeatedly push the network for a signal

Increase packets *in flight* over time
(and/or *rate of transmission*)

Bottleneck queue occupancy grows
→ Enough to drop packets (signal)
→ Enough to increase queuing delay (signal)

How to do *capacity seeking* ?

Repeatedly push the network for a signal

Increase packets *in flight* ... e
(and/or *rate of tr*...

Bottleneck ... pancy grows
... packets (signal)
... ease queuing delay (signal)

Backoff on signal
(rinse, repeat)

Consequences of *capacity seeking* ?

Algos differ in *how* they push,
and *what* signals they track

Consequences of *capacity seeking* ?

Algos differ in *how* they push,
and *what* signals they track

Loss-based algo <u>must fill queues</u>
(NewReno, CUBIC, ...)

Hybrid algos may seek onset-of-queuing, etc
(BBR, ...)

Consequences of *capacity seeking* ?

Algos differ in *how* the **sh,**

and *what* signa

Loss-b ill queues

UBIC, ...)

Hybrid nay seek onset-of-queuing, etc

(BBR, ...)

Over, and over, and over, and over,

and...

Consequences of *capacity seeking* ?

Regular packet loss events

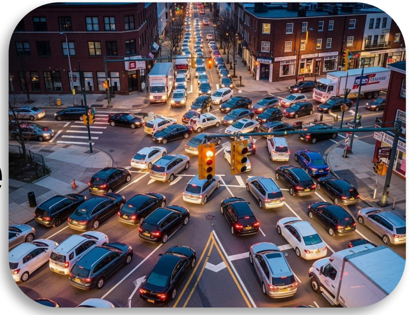
Regular latency spikes

Consequences of *capacity seeking* ?

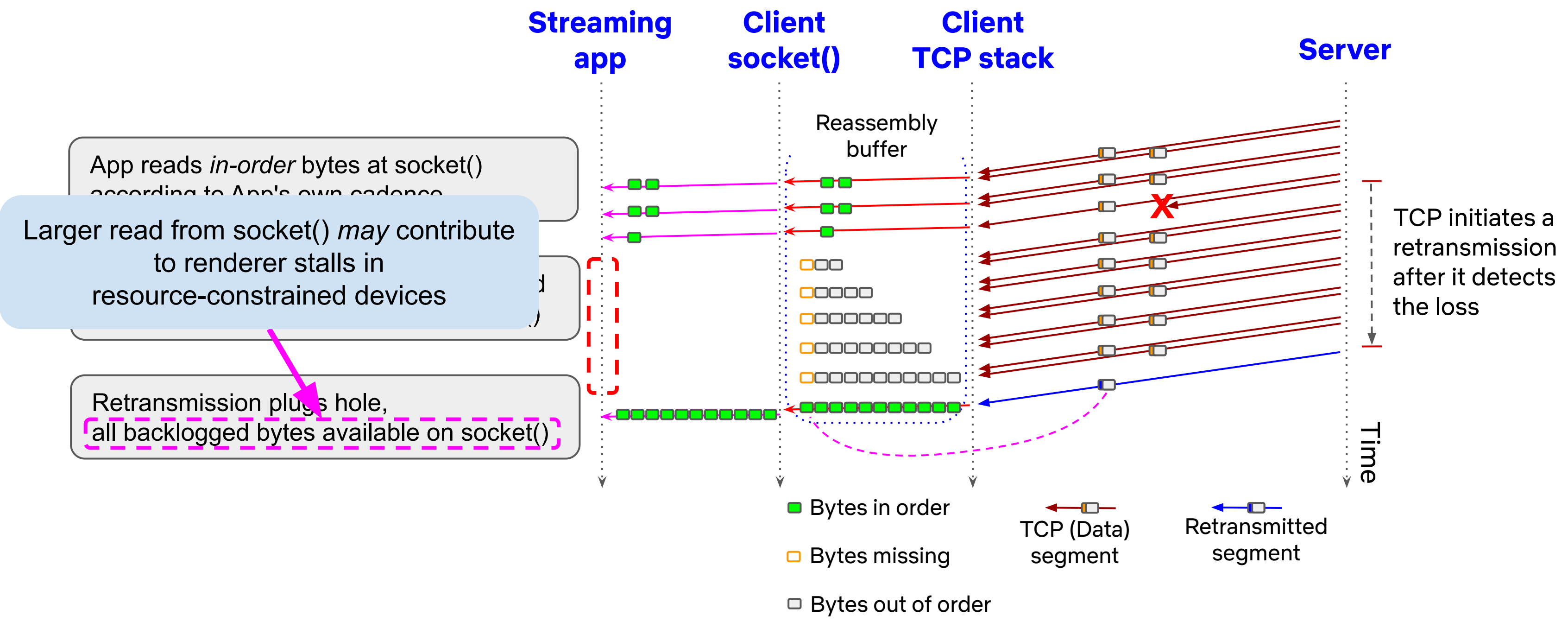
Regular packet loss events

Regular latency spikes

...hilarity for <u>all</u> traffic
sharing a given bottleneck queue



Sidebar: Packet losses and low-end clients



Network-based response: queue management

e.g. RED, PIE, FQ-CoDel, L4S, ...

Proactively drop (or ECN-mark) packets

→ Probabilistic drop or mark as queue begins to grow

(Defensive "early signal" to loss-based transport algos)

Network-based response: queue management

e.g. RED, PIE, FQ-CoDel, L4S, …

Distribute flows across queues

→ Isolate *queue building* and *non-queue building* flows
→ Combine with proactive drop or mark on each queue

Network-based response: queue management

Industry deployment of
active queue management
is a work in progress

What if media streaming servers transmitted packets

$\boxed{\text{only as fast}}$ as needed?

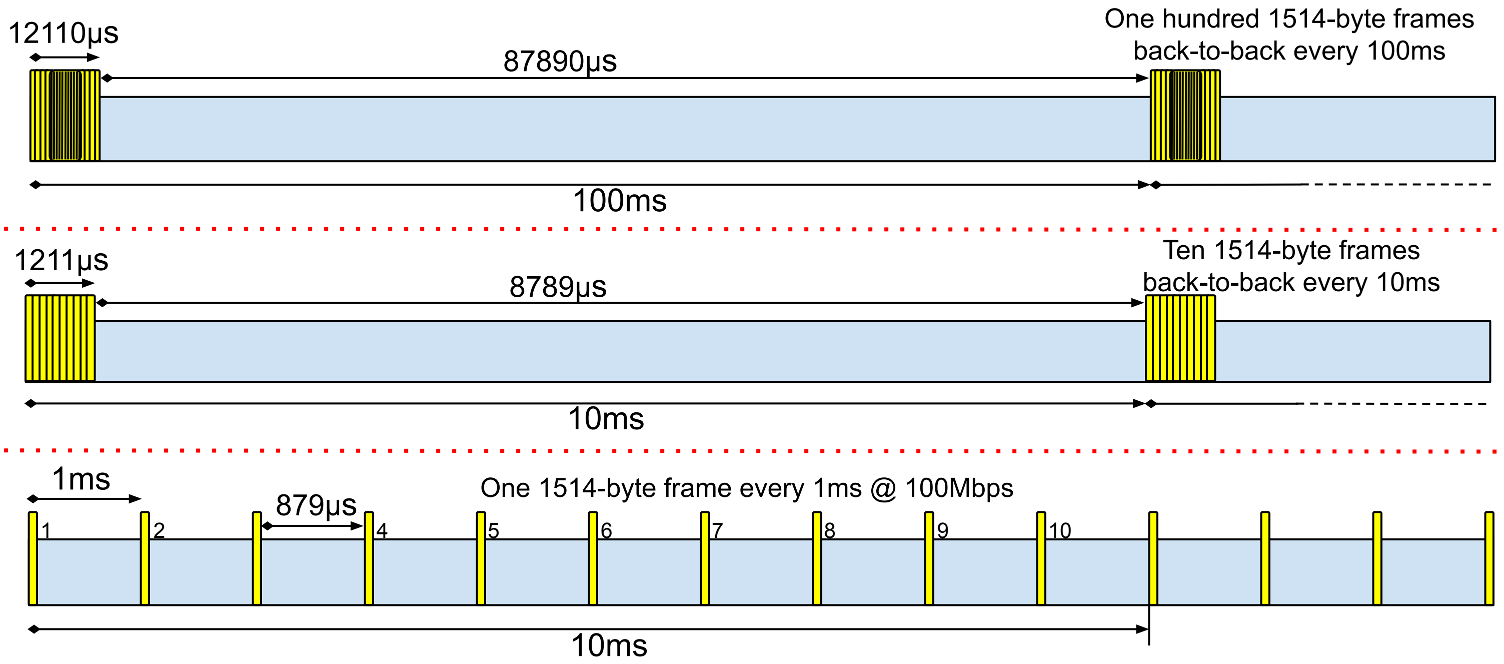
Per-packet pacing ?

Pacing: Capped rate, smooth(er) delivery

Rate limit: Upper bound the average bytes/sec

Smoothness: Spread out packets over time

Example: Three versions of 12Mbps (at Ethernet link layer measured over 1sec)



Pacing: Benefits inside the network

Pace below bottleneck speed

→ **Reduced** bottleneck queue growth

→ **Reduced** packet losses

Spreading out packets

→ **Better interleaving** with other traffic

Reduced queuing delays (pacing + VoIP)

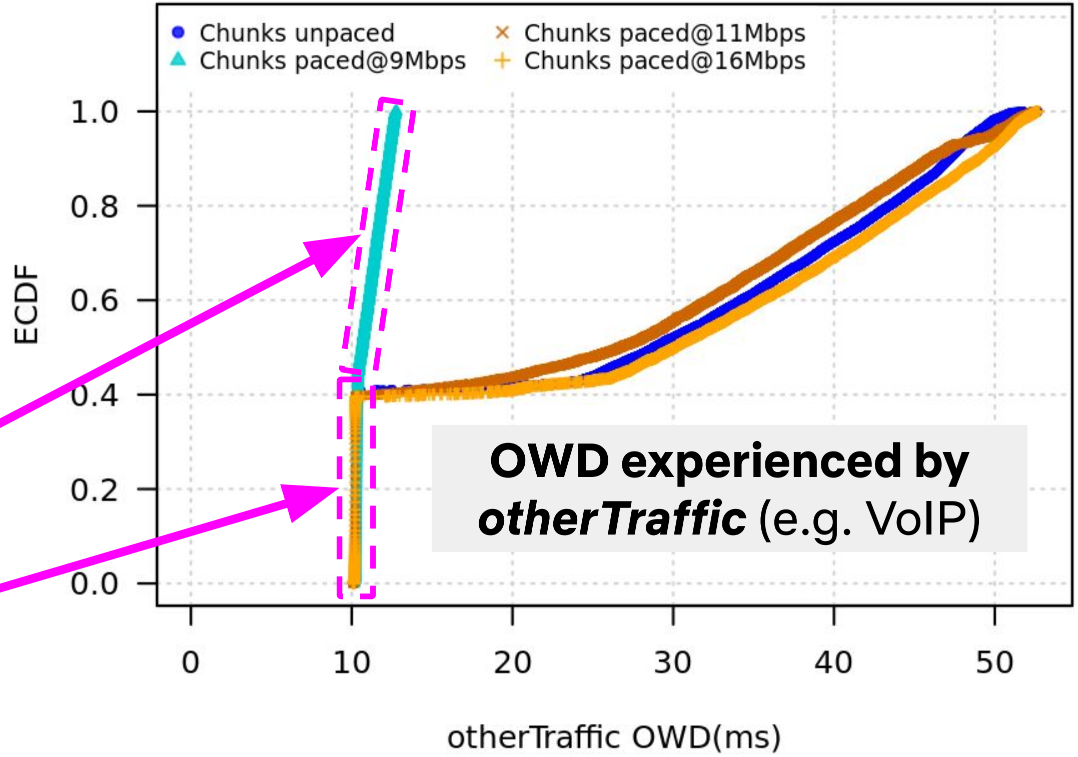
Two flows in a lab testbed:

- Chunks/TCP (paced or unpaced)
- VoIP-like/UDP @ 80kbps

Shared 10Mbps bottleneck
(20ms base RTT, 2BDP queue)

Chunks @ 9Mbps add little to *otherTraffic* OWD

No queuing delay during gaps between chunks



Reduced packet losses (pacing + competition)

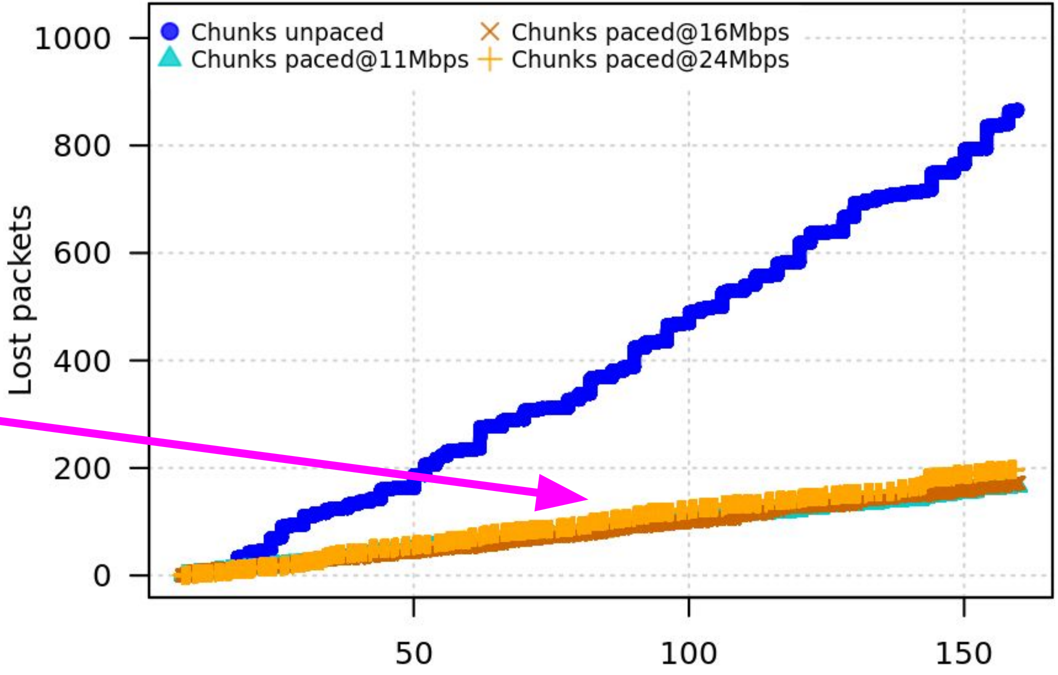
Four flows in a lab testbed:

Shared 40Mbps bottleneck

(40ms base RTT, 2BDP queue)

- Three unpaced TCP flows.
- Test TCP flow paced or unpaced.

Paced TCP flows experience lower packet loss rates



Pacing: Some background reading

Authors: M. Welzl *University of Oslo* W. Eddy *MTI Systems* V. Goel *Apple Inc.* M. Tüxen *Münster University of Applied Sciences*

Pacing in Transport Protocols

Abstract

Applications or congestion control mechanisms can produce bursty traffic which can cause unnecessary queuing and packet loss. To reduce the burstiness of traffic, the concept of evenly spacing out the traffic from a data sender over a round-trip time known as "pacing" has been used in many transport protocol implementations. This document gives an overview of pacing and how some known pacing implementations work.

https://datatracker.ietf.org/doc/draft-welzl-iccrg-pacing/

Missing piece?

Choosing the desired rate and smoothness

What if media streaming servers transmitted packets only as fast $\boxed{\text{as needed}}$?

Implies *application-layer* knowledge

Application-Informed Pacing

Sammy: smoothing video traffic to be a friendly internet neighbor

Bruce Spang
Stanford University

Shravya Kunamalla
Netflix

Renata Teixeira
Netflix

Te-Yuan Huang
Netflix

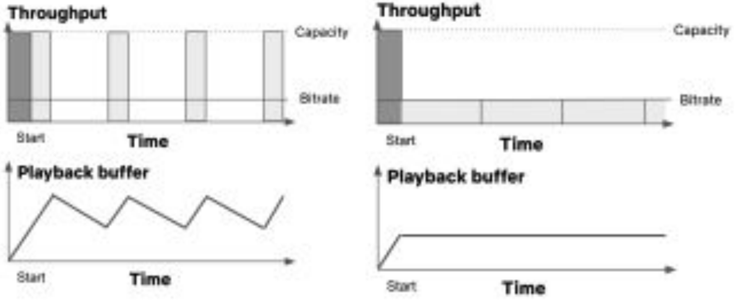
Grenville Armitage
Netflix

Ramesh Johari
Stanford University

Nick McKeown
Stanford University

ABSTRACT

On-demand streaming video traffic is managed by an adaptive bitrate (ABR) algorithm whose job is to optimize quality of experience (QoE) for a single video session. ABR algorithms leave the question of sharing network resources up to transport-layer algorithms. We observe that as the internet gets faster relative to video streaming rates, this delegation of responsibility gives video traffic a burstier on-off traffic pattern. In this paper, we show we can substantially smooth video traffic to improve its interactions with the rest of the internet, while maintaining the same or better QoE for streaming video. We smooth video traffic with two design principles:



(a) Video traffic today. (b) Smoother, same QoE.

SIGCOMM'23

Application-Informed Pacing

For each video segment (chunk)

- The client determines a *sufficient TCP delivery rate*, and
- Sends this target rate to the server with each HTTP request,

Application-Informed Pacing

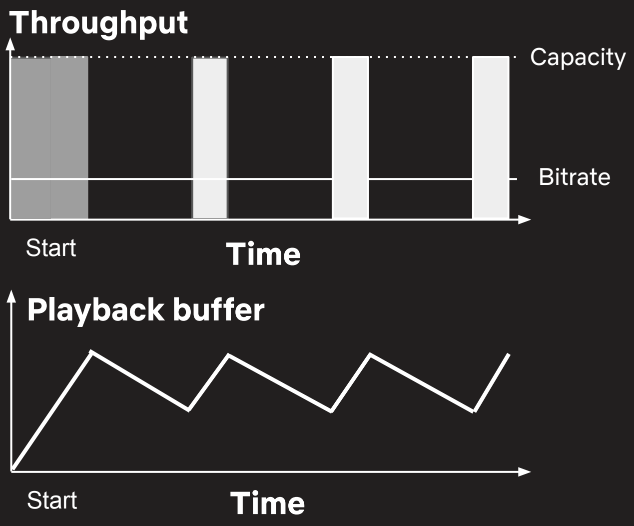
For each video segment (chunk)

- The client determines a *sufficient TCP delivery rate*, and
- Sends this target rate to the server with each HTTP request,

- Server returns the video chunk's TCP segments no faster than requested, *even if* the network allows faster delivery.

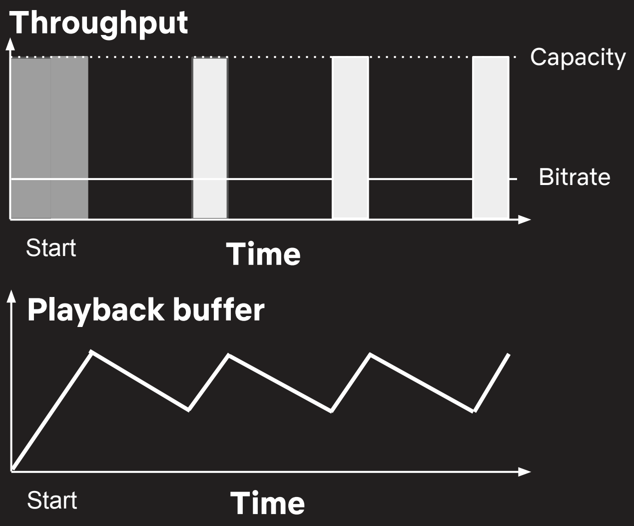
Ideal: Regular vs paced chunk delivery

Chunks delivered via Regular TCP
(Video traffic today)

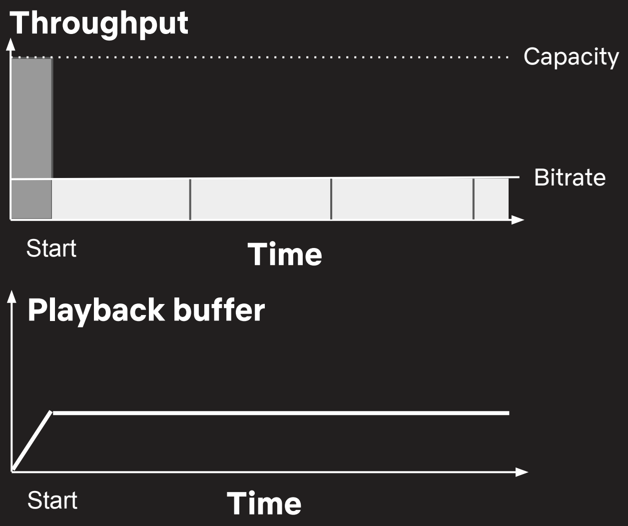


Ideal: Regular vs paced chunk delivery

Chunks delivered via Regular TCP
(Video traffic today)

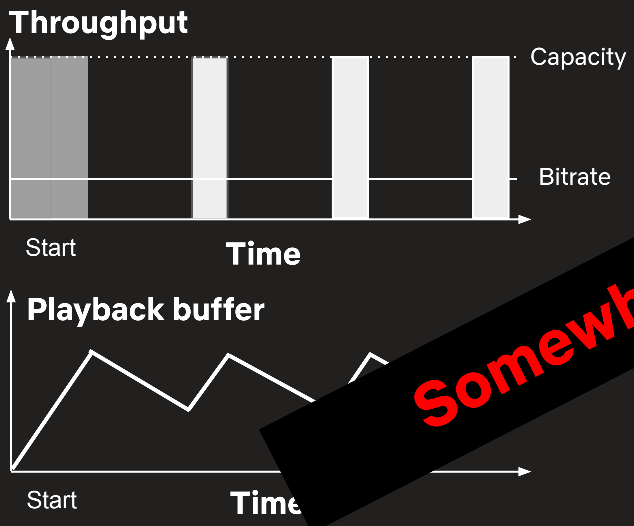


Chunks delivered via Paced TCP

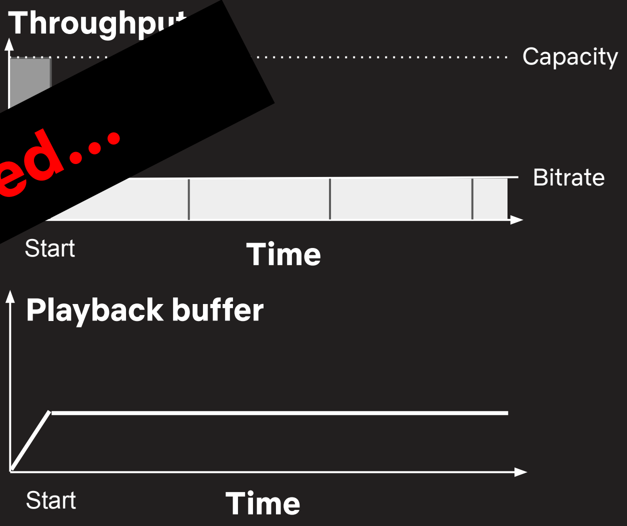


Ideal: Regular vs paced chunk delivery

Chunks delivered via Regular TCP
(Video traffic today)

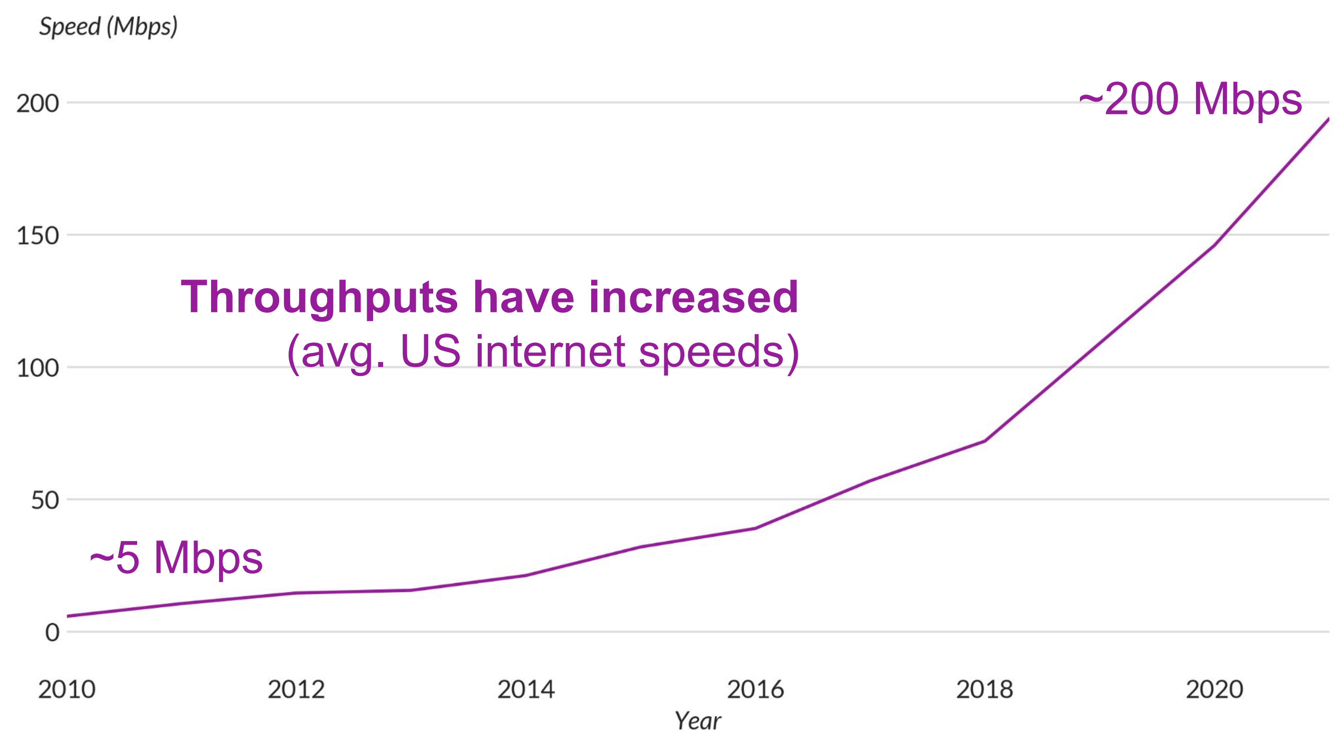


Chunks delivered via Paced TCP

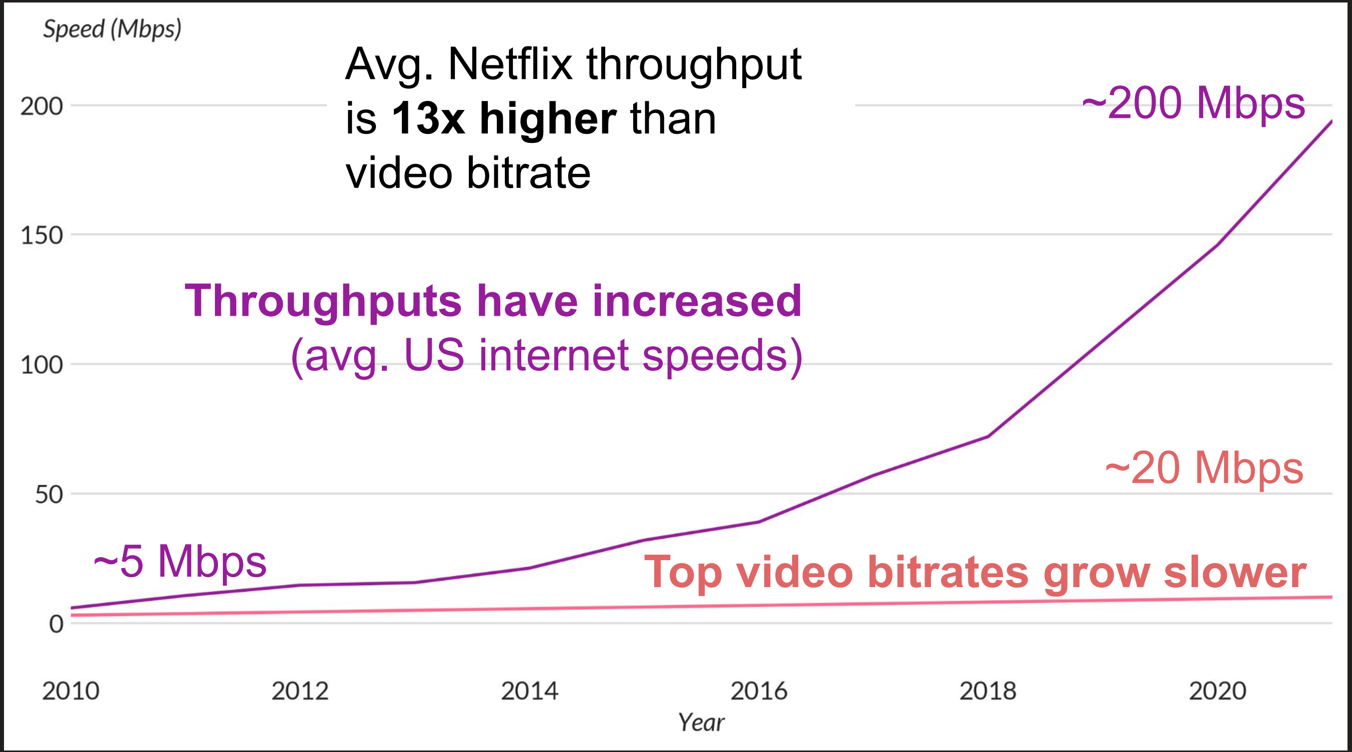


Somewhat idealized…

Video today is usually not throughput limited



Video today is usually not throughput limited



Picking the pace rate

The ideal pace *rate* is

→ above video delivery rate

→ below network bottleneck speed

Picking the pace rate

Pace too slow

→ starve client's playout buffer

→ starve client's knowledge of path capacity

→ eliminate client's ability to move up rate ladder

Pace too fast

→ smaller reduction in RTT and loss rates

Picking the pace rate

Pace too slow

→ starve client's playout buffer

→ starve client's knowledge of path capacity

→ eliminate client's ability to move up rate ladder

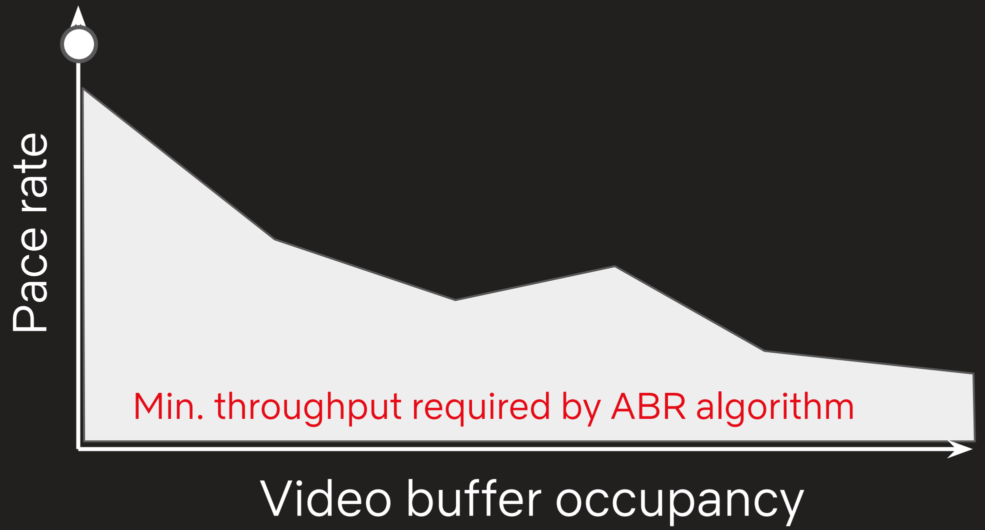
Pace too fast

→ smaller reduction in RTT and loss rates

aka, an R&D opportunity

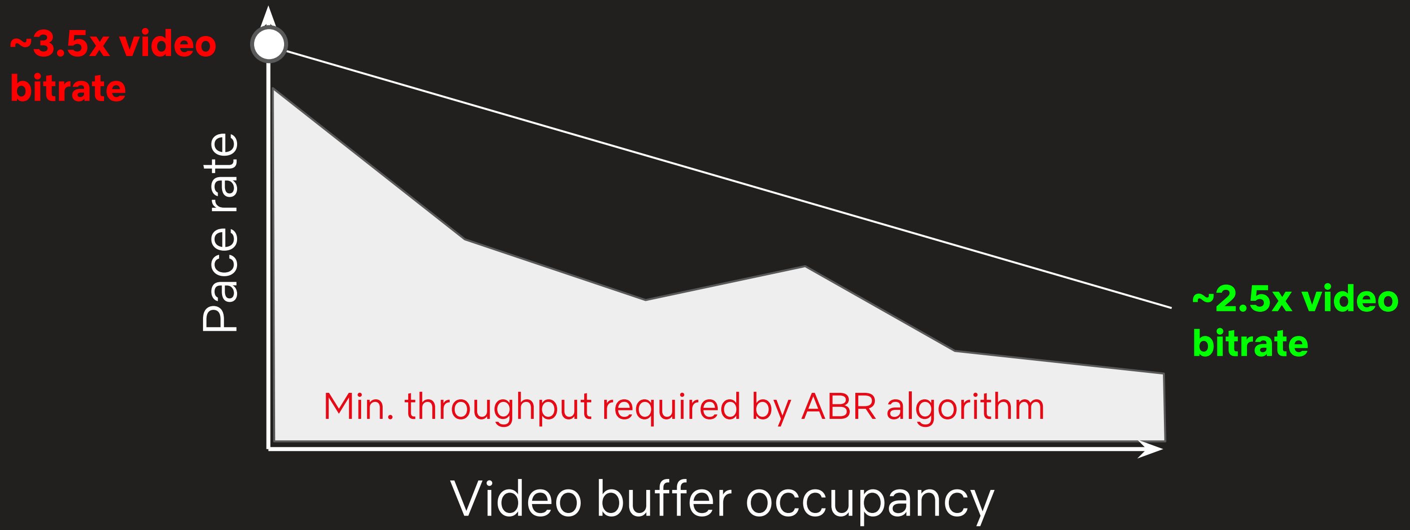
ABR picks pace rate based on QoE needs

Need enough throughput for ABR algorithm to pick highest video bitrate



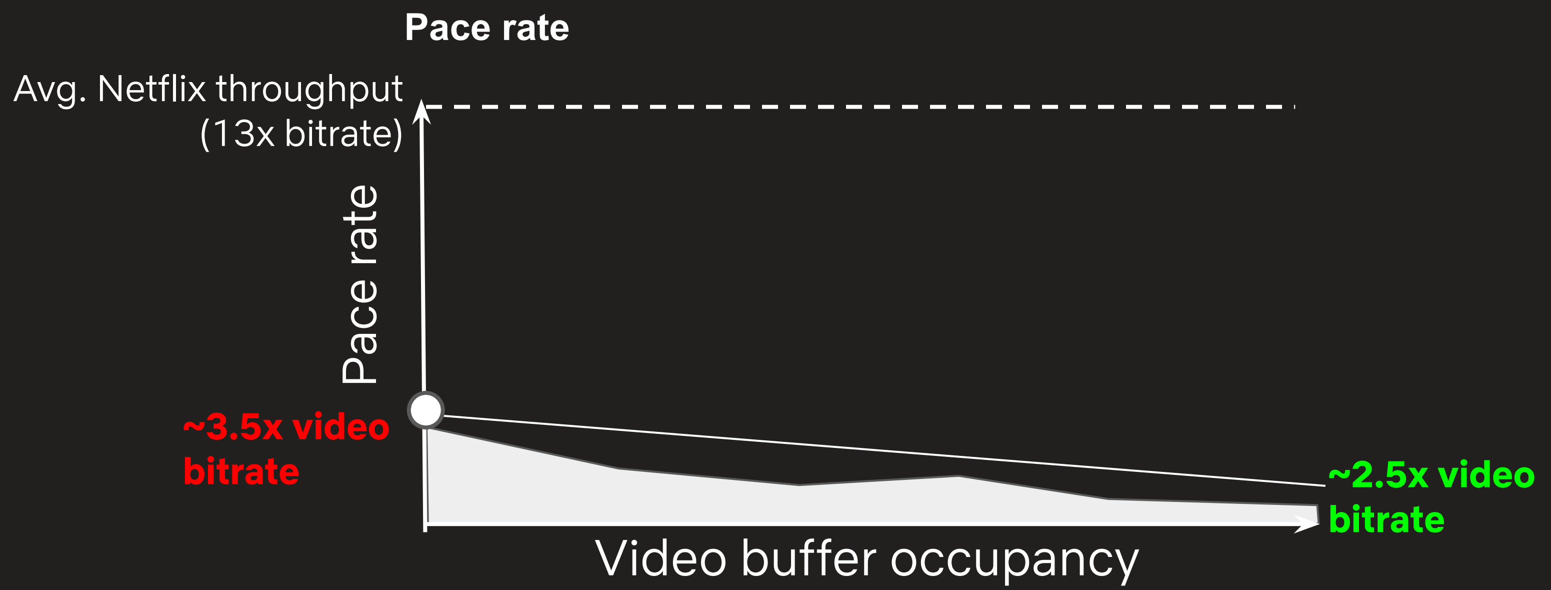
ABR picks pace rate based on QoE needs

Need enough throughput for ABR algorithm to pick highest video bitrate



ABR picks pace rate based on QoE needs

Need enough throughput for ABR algorithm to pick highest video bitrate



What have we seen?

Large-scale experiments

A range of movements:

Metric	**Results (approx.)**
Instantaneous Throughput	**60-85% drop**
Retransmissions	**35-50% lower**
Round-trip times	**30-35% lower**

(Illustrative examples, actual results vary for <*reasons*> such as path capacities, pace rate selection heuristic, etc.)

Coexistence with interactive apps

Network latency
is a known issue
(…for some time)

From 1996...

Latency

and the Quest for Interactivity

Stuart Cheshire*

cheshire@cs.stanford.edu

http://www.stuartcheshire.org/

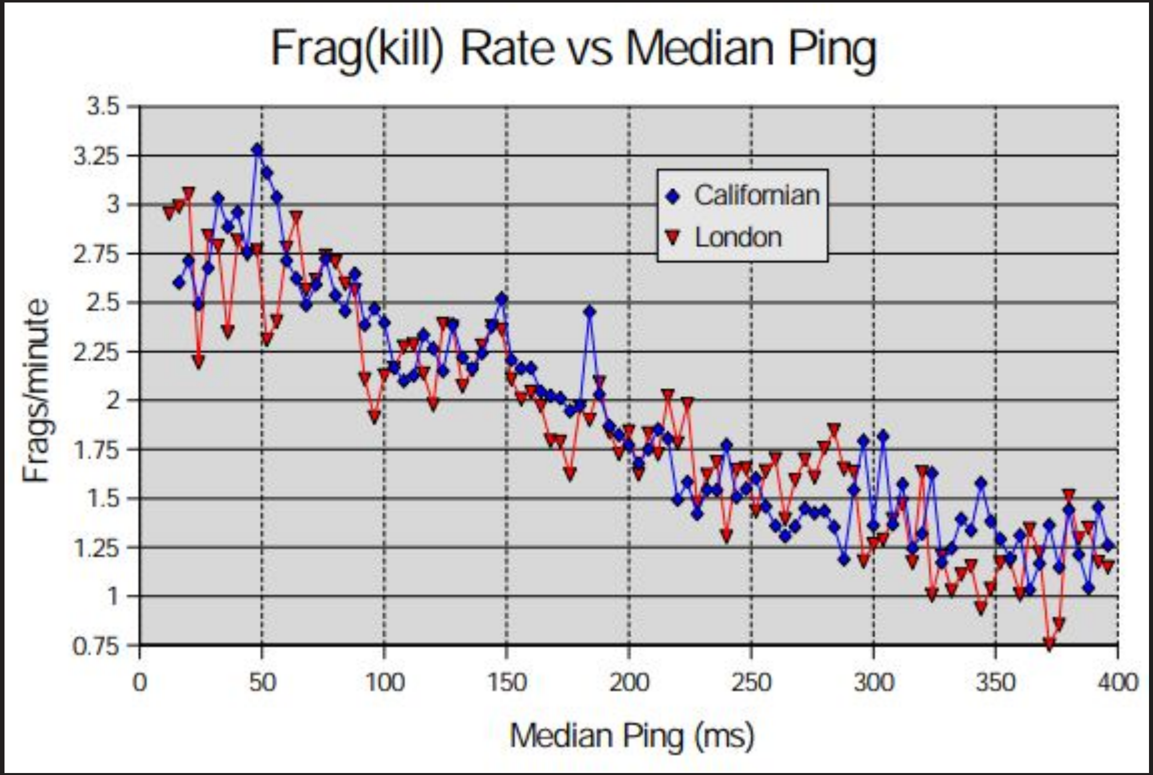
November 1996

Abstract

Poor latency, not limited throughput, is the factor that is hindering the development of a whole class of networked application software — interactive games, conferencing software, collaborative work software and all forms of interactive multi-user software.

http://www.stuartcheshire.org/index.html#LatencyQuest

(From the archives of my misspent youth)



"**[Sensitivity of Quake3 Players To Network Latency](#)**" (Poster session)

SIGCOMM Internet Measurement Workshop, November 1, [2001](#)

Cloud gaming adds
low latency video delivery
into the interactivity equation

Someone *will* try playing an online cloud game

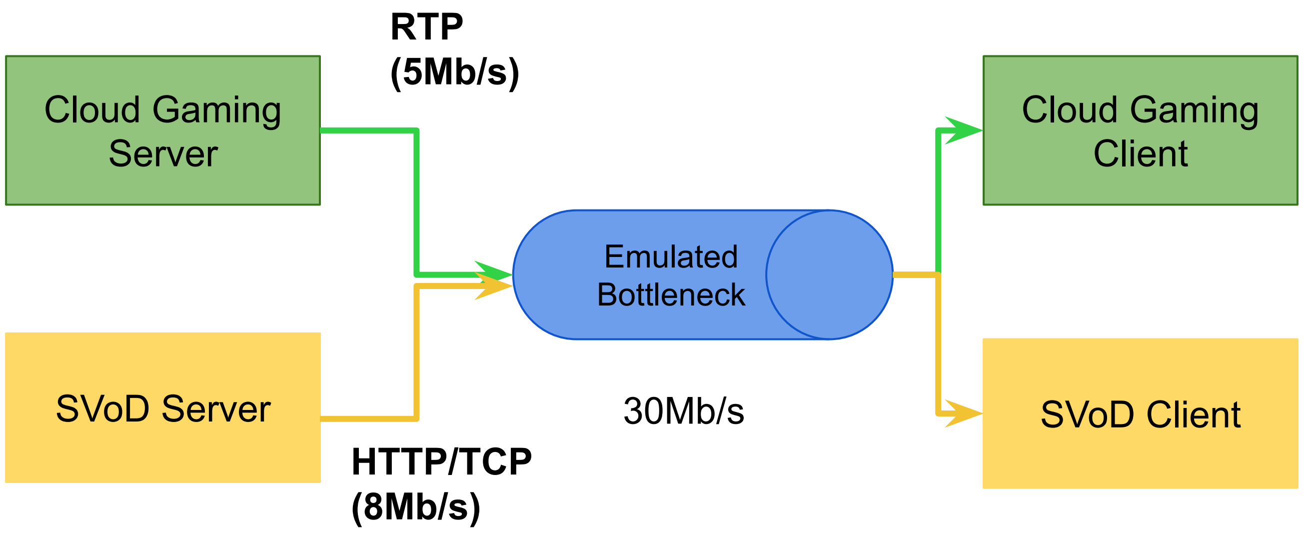
...while streaming a movie '*in the other room*', over a shared home network connection

Someone *will* try playing an online cloud game

...while streaming a movie '*in the other room*', over a shared home network connection

Application-informed pacing FTW?

Demo: Game video vs (un)paced streaming



Paced streaming enables smoother cloud gaming experience

From: Scott Danahy

Complexity, open questions, and tradeoffs

Thank you

Questions?

Acknowledgements
Many colleagues led to this point, including:
Te-Yuan (TY) Huang, Shravya Kunamalla, Bruce Spang,
Renata Teixeira, Peter Lei, Lawrence Stewart,
Yiannis Yiakoumis, Aimee Feng, Reese Enghardt,
Scott Danahy, and more...