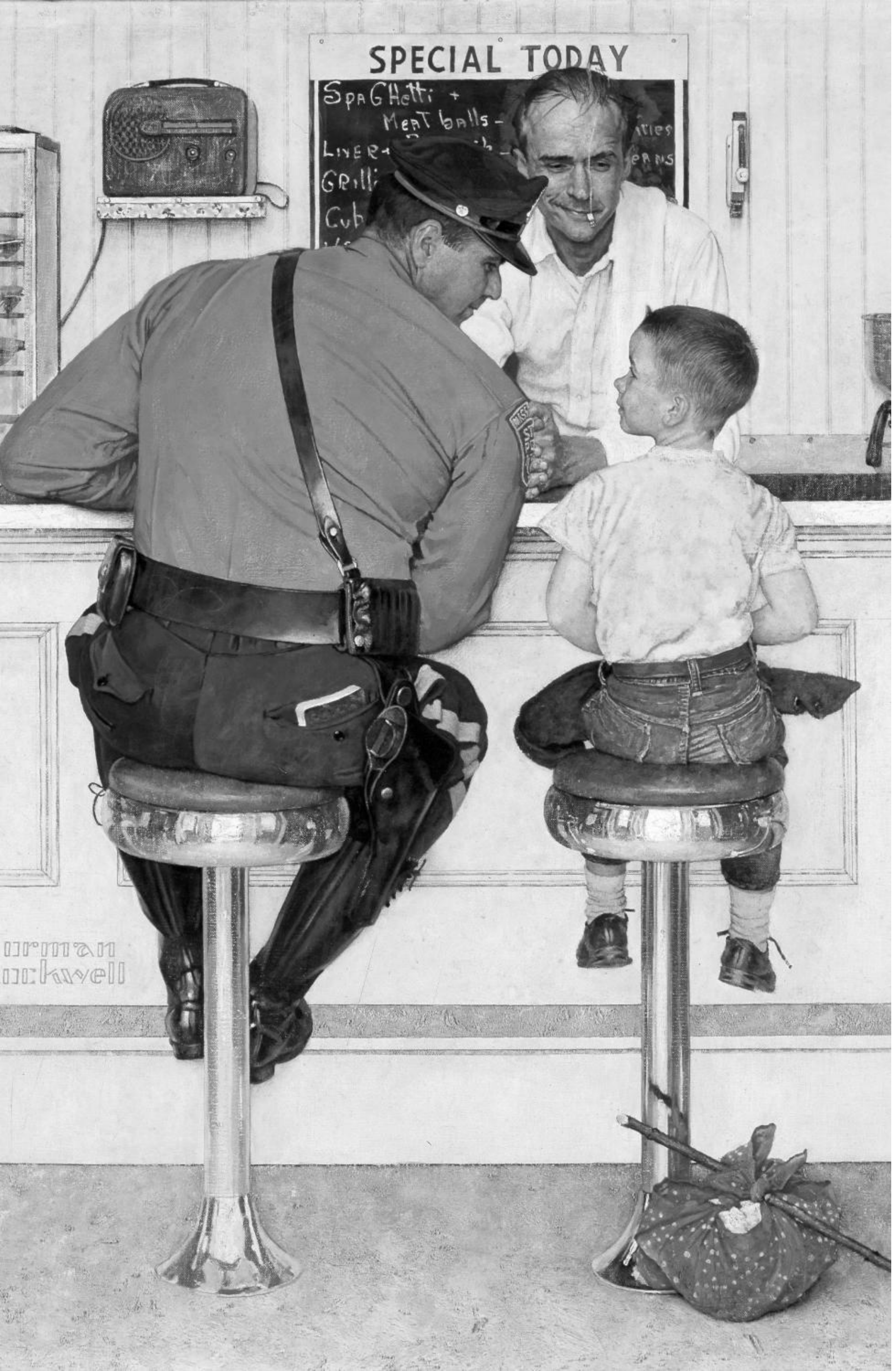


Guest Lecture @Duke

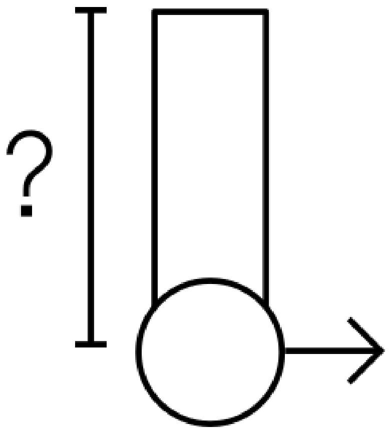
SIGCOMM 2024

Keeping an Eye on Congestion Control in the Wild with Nebby

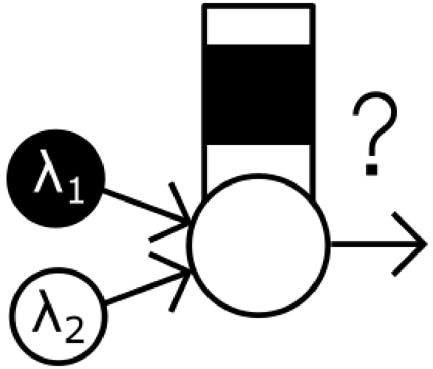
Ayush Mishra[1], Lakshay Rastogi[2], Raj Joshi[3], Ben Leong[1]

[1]National University of Singapore
[2]IIT Kanpur
[3]Harvard University

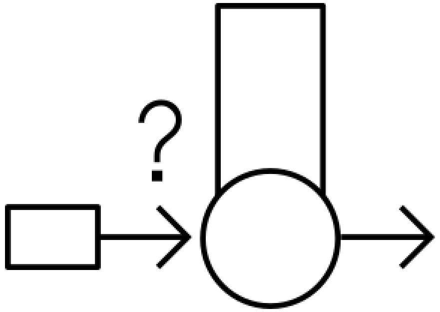
The make up of the Internet’s Congestion Control Landscape influences how we think about



Buffer Sizing



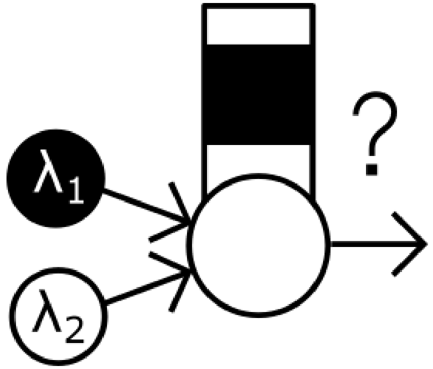
Fairness and Deployability



AQMs and In-network policing

Good CCAs can make or break the Internet

Buffer Sizing



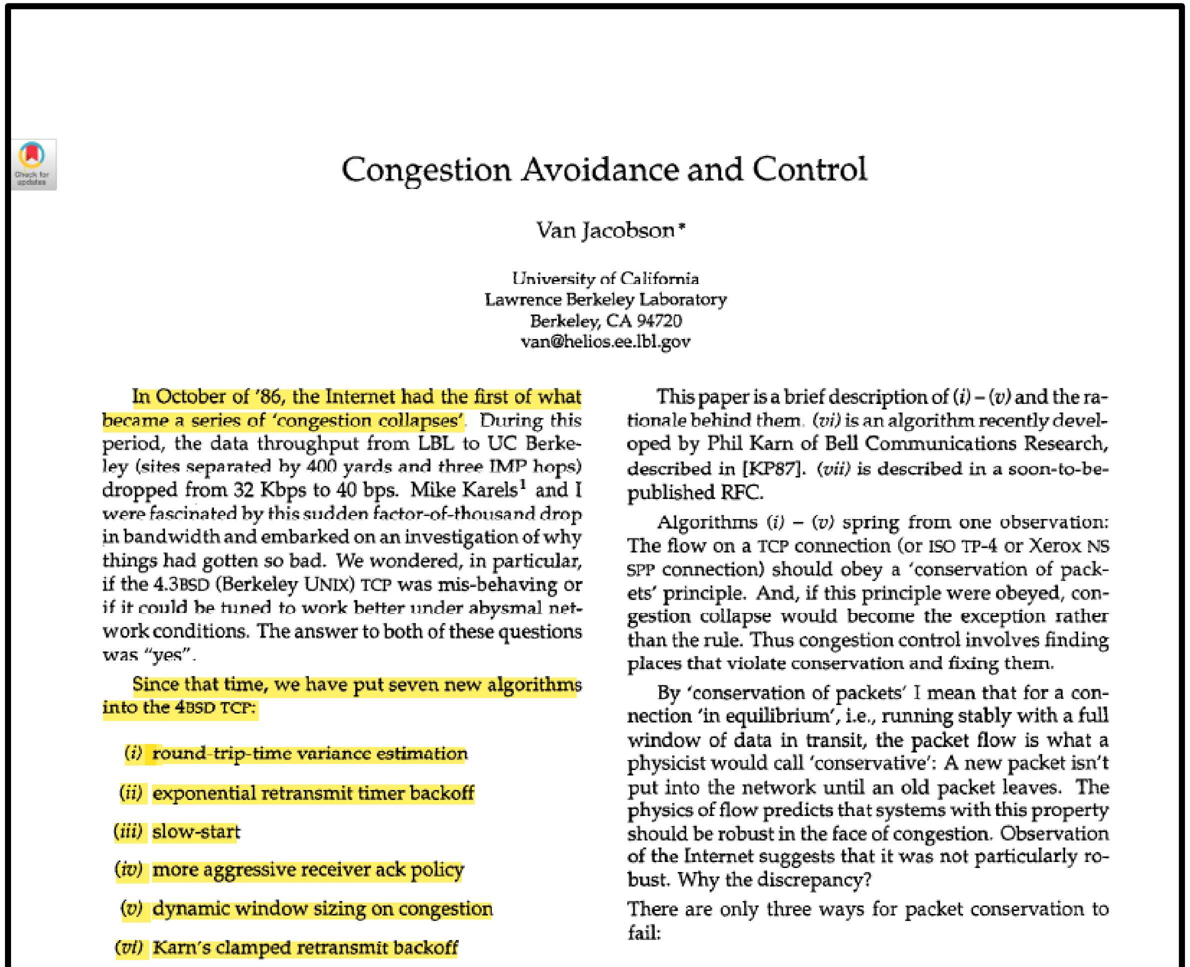
Fairness and Deployability

AQMs and In-network policing

The **Internet Congestion Collapse**

and the birth of Congestion Control

[SIGCOMM 1986]



Congestion Avoidance and Control

Van Jacobson*

University of California
Lawrence Berkeley Laboratory
Berkeley, CA 94720
van@helios.ee.lbl.gov

In October of '86, the Internet had the first of what became a series of 'congestion collapses'. During this period, the data throughput from LBL to UC Berkeley (sites separated by 400 yards and three IMP hops) dropped from 32 Kbps to 40 bps. Mike Karels[1] and I were fascinated by this sudden factor-of-thousand drop in bandwidth and embarked on an investigation of why things had gotten so bad. We wondered, in particular, if the 4.3BSD (Berkeley UNIX) TCP was mis-behaving or if it could be tuned to work better under abysmal network conditions. The answer to both of these questions was "yes".

Since that time, we have put seven new algorithms into the 4BSD TCP:

(i) round-trip-time variance estimation

(ii) exponential retransmit timer backoff

(iii) slow-start

(iv) more aggressive receiver ack policy

(v) dynamic window sizing on congestion

(vi) Karn's clamped retransmit backoff

This paper is a brief description of *(i)* – *(v)* and the rationale behind them. *(vi)* is an algorithm recently developed by Phil Karn of Bell Communications Research, described in [KP87]. *(vii)* is described in a soon-to-be-published RFC.

Algorithms *(i)* – *(v)* spring from one observation: The flow on a TCP connection (or ISO TP-4 or Xerox NS SPP connection) should obey a 'conservation of packets' principle. And, if this principle were obeyed, congestion collapse would become the exception rather than the rule. Thus congestion control involves finding places that violate conservation and fixing them.

By 'conservation of packets' I mean that for a connection 'in equilibrium', i.e., running stably with a full window of data in transit, the packet flow is what a physicist would call 'conservative': A new packet isn't put into the network until an old packet leaves. The physics of flow predicts that systems with this property should be robust in the face of congestion. Observation of the Internet suggests that it was not particularly robust. Why the discrepancy?

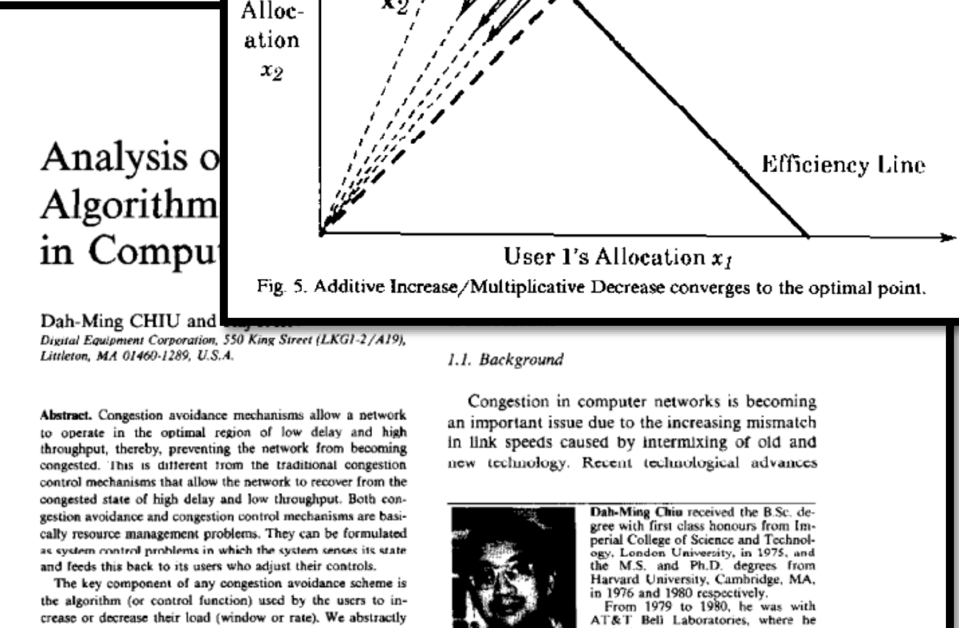
There are only three ways for packet conservation to fail:

In 1986, the Internet faced a series of **congestion collapses** – most of the Internet's bandwidth was being taken up in retransmitting lost packets!

We invented simple **Congestion Control Algorithms** in response…

These simple AIMD algorithms were **fair** and **stable**!

[Computer Networks 1989]



Analysis o
Algorithm
in Compu

Dah-Ming CHIU and

Digital Equipment Corporation, 550 King Street (LKG1-2/A19), Littleton, MA 01460-1289, U.S.A.

Abstract. Congestion avoidance mechanisms allow a network to operate in the optimal region of low delay and high throughput, thereby, preventing the network from becoming congested. This is different from the traditional congestion control mechanisms that allow the network to recover from the congested state of high delay and low throughput. Both congestion avoidance and congestion control mechanisms are basically resource management problems. They can be formulated as system control problems in which the system senses its state and feeds this back to its users who adjust their controls.

The key component of any congestion avoidance scheme is the algorithm (or control function) used by the users to increase or decrease their load (window or rate). We abstractly

1.1. Background

Congestion in computer networks is becoming an important issue due to the increasing mismatch in link speeds caused by intermixing of old and new technology. Recent technological advances

Dah-Ming Chiu received the B.Sc. degree with first class honours from Imperial College of Science and Technology, London University, in 1975, and the M.S. and Ph.D. degrees from Harvard University, Cambridge, MA, in 1976 and 1980 respectively.

From 1979 to 1980, he was with AT&T Bell Laboratories, where he

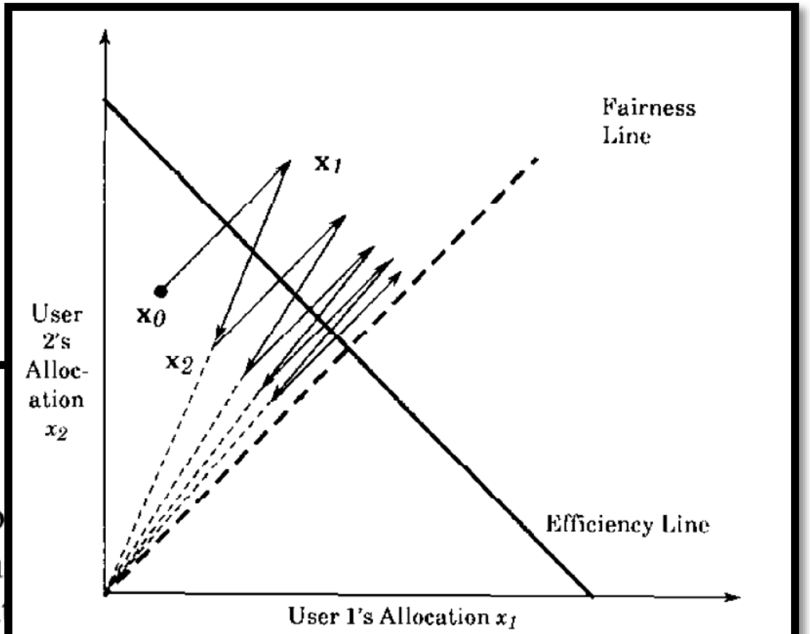


Fig. 5. Additive Increase/Multiplicative Decrease converges to the optimal point.

Since **1986** to **2015**,

we've always had some version of a loss-based congestion control algorithm dominant on the Internet

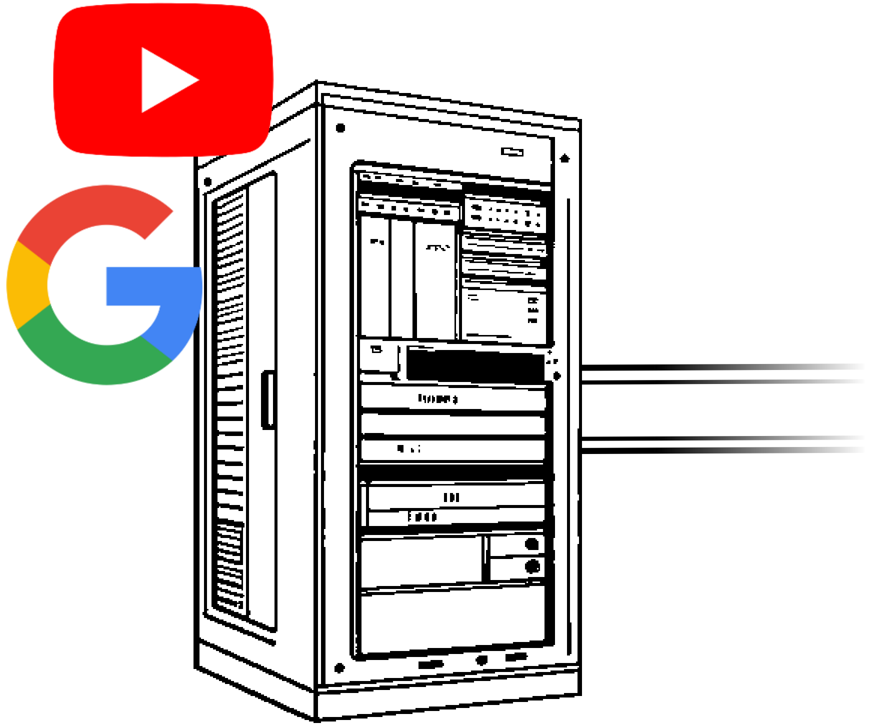
Tahoe, Reno, CUBIC…

Enjoyed the stability and predictability

We’re in the midst of a *Renaissance* in Congestion Control Research

Several recent developments risk disrupting the stable ecosystem of AIMD and MIMD congestion control algorithms we have had for three decades.

Recent developments in Internet Congestion Control

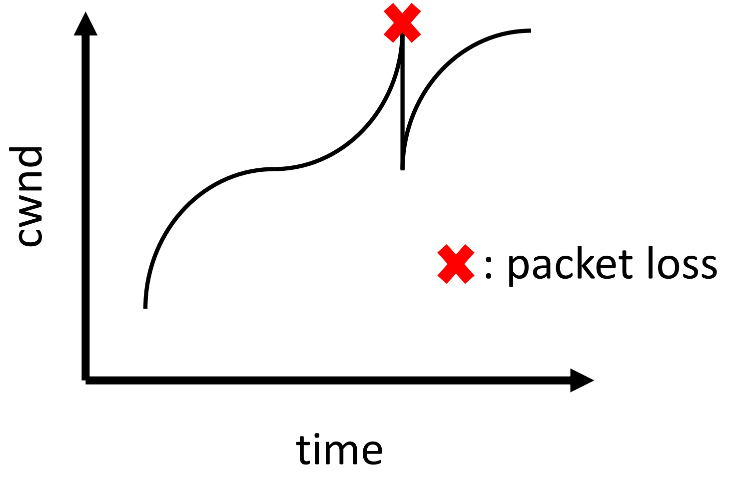


#1 **BBR**

New model-based CCA proposed by Google in 2016 abandons the traditional loss-based and window-based paradigm

Recap: CUBIC

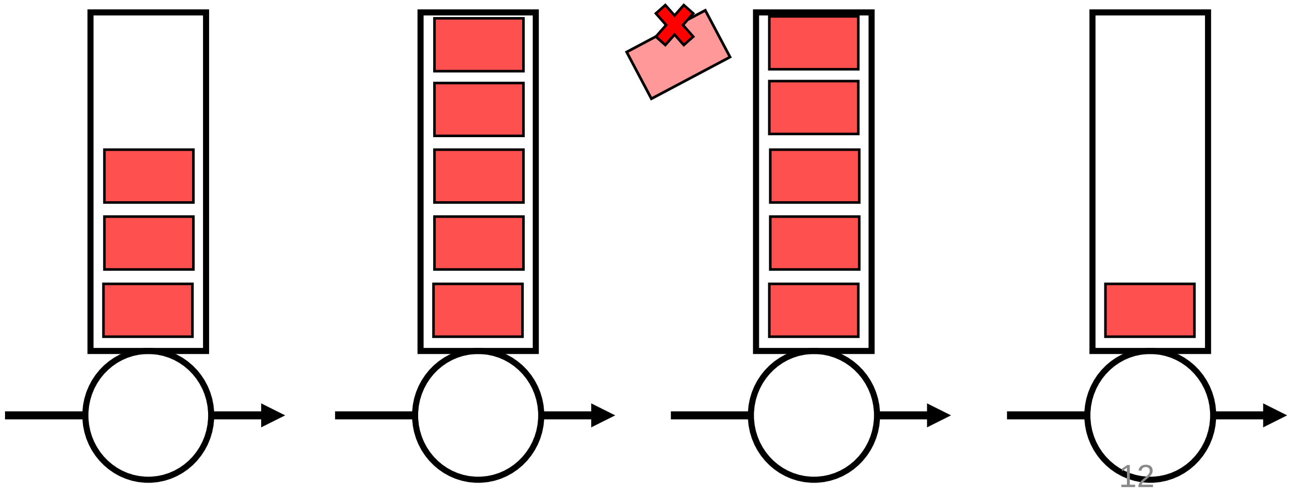
Cwnd-based congestion control algorithm



Treats packet loss as a congestion signal.

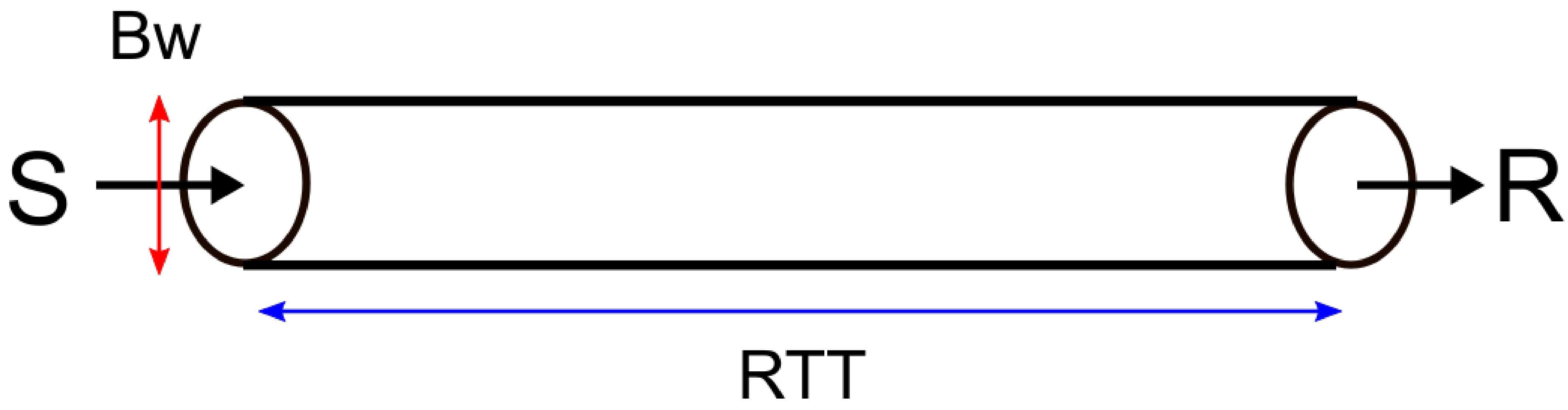
Reduces cwnd by 30% when is sees a packet loss.

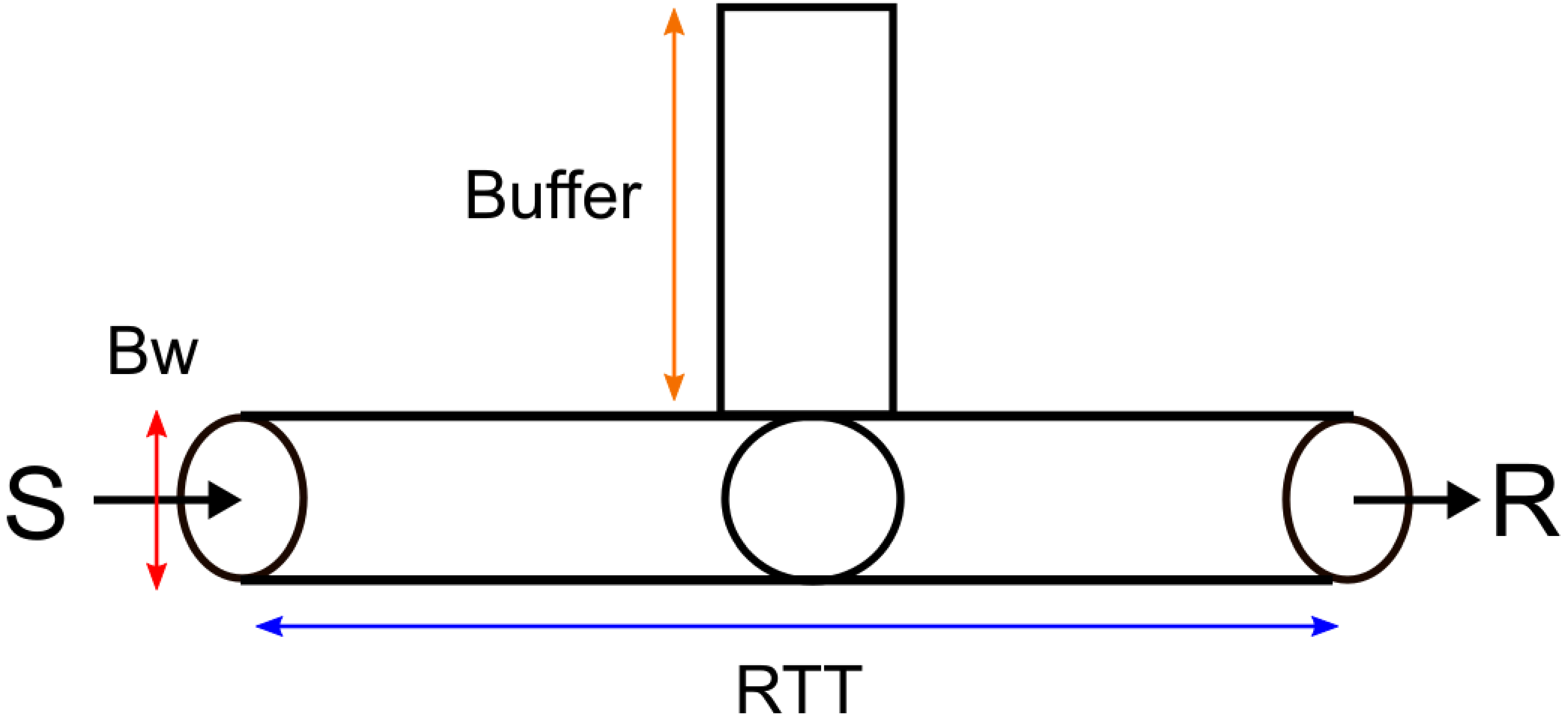
Considered a buffer filler

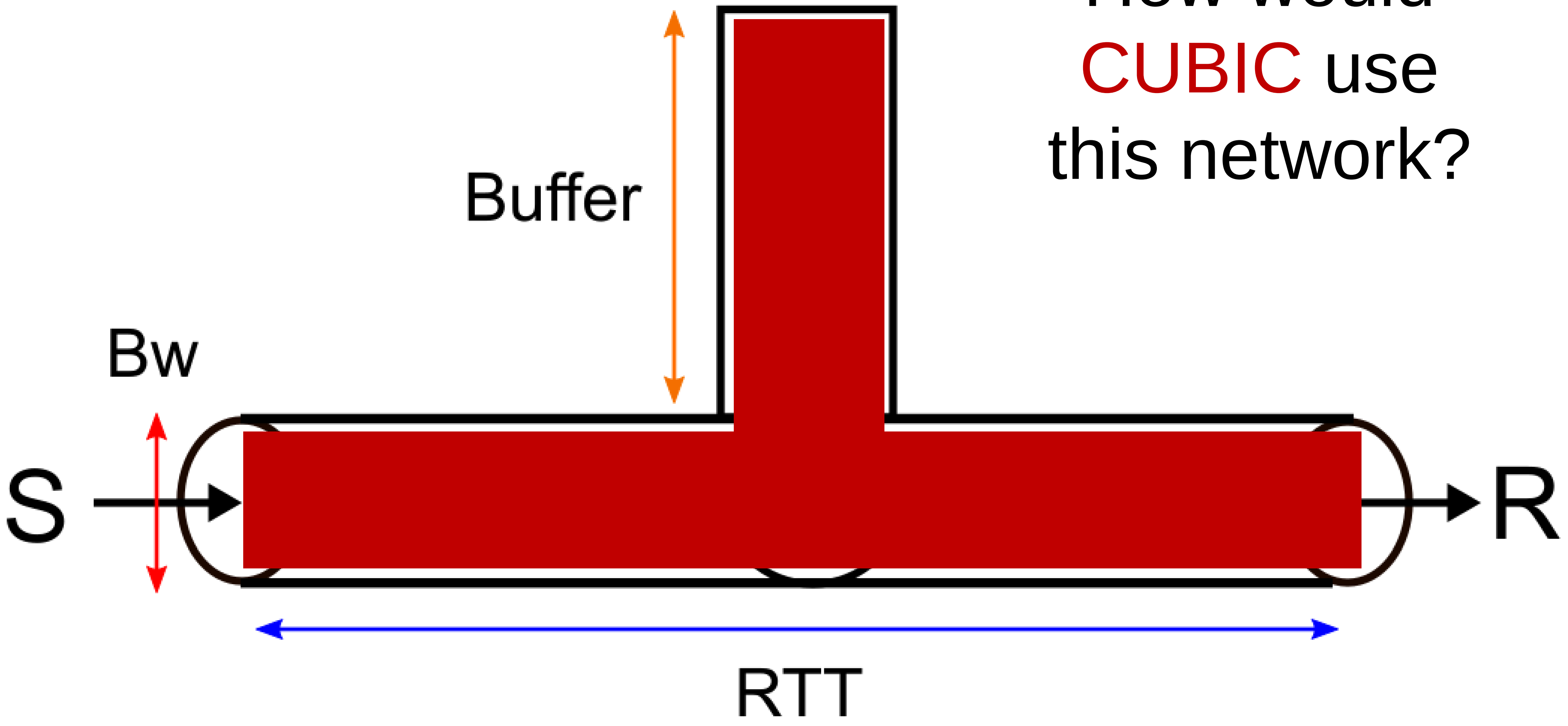


The network model

The network model

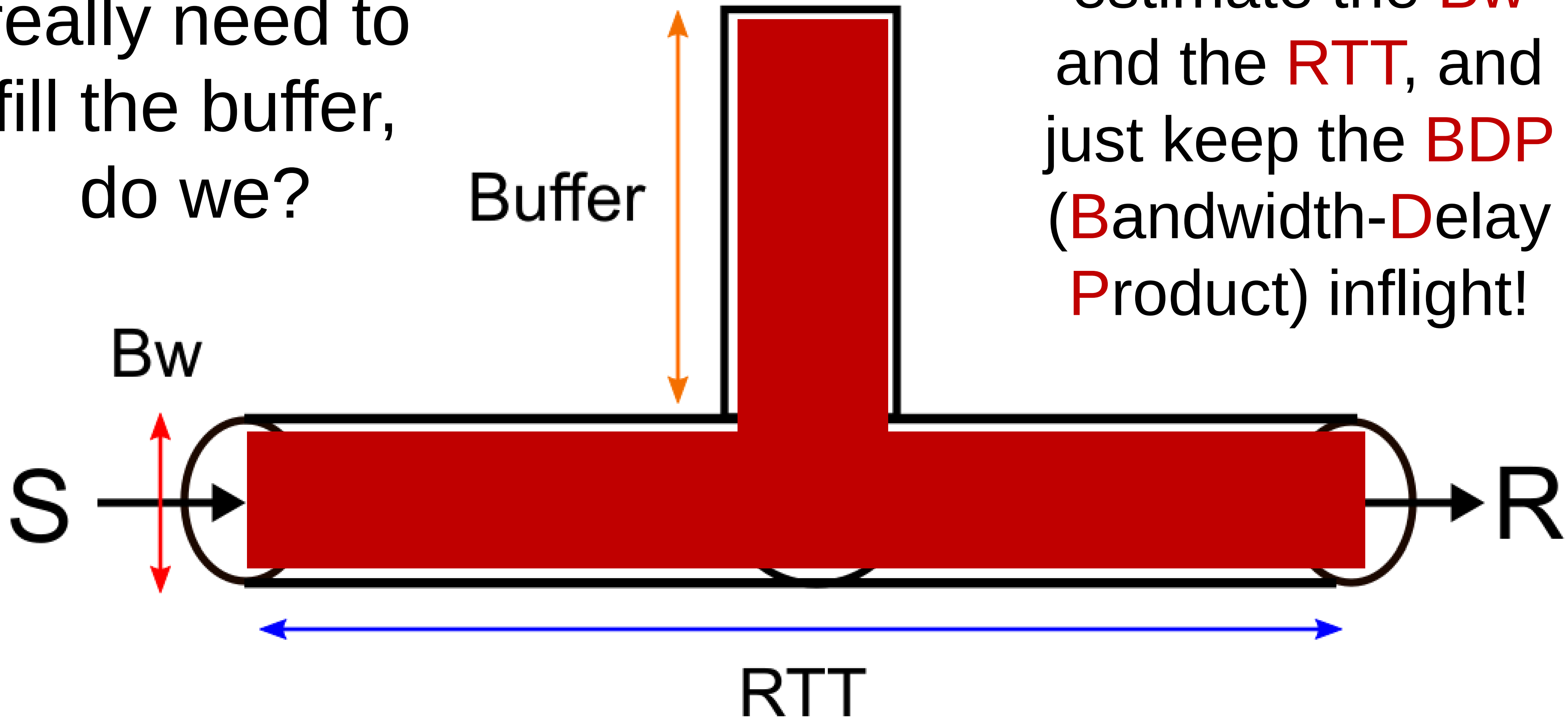



Buffer
Bw
S
R
RTT



How would CUBIC use this network?

We don't really need to fill the buffer, do we?



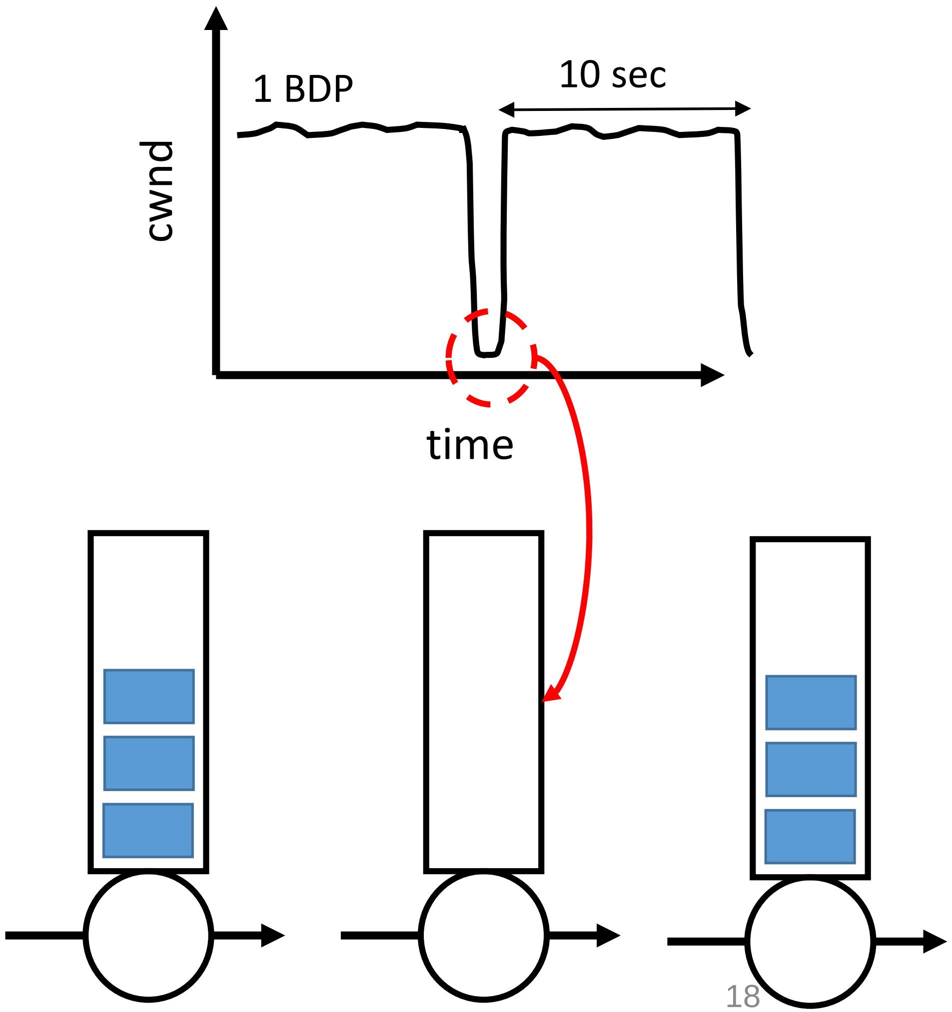
Let's try to estimate the Bw and the RTT, and just keep the BDP (Bandwidth-Delay Product) inflight!

Google's solution: BBR

Rate-based congestion control algorithm. Uses **$\mathbf{RTT_{min}}$** and **bandwidth** estimates to infer congestion.

Estimate the **bandwidth** using the receive rate

Backs off every 10 sec to measure **$\mathbf{RTT_{min}}$**



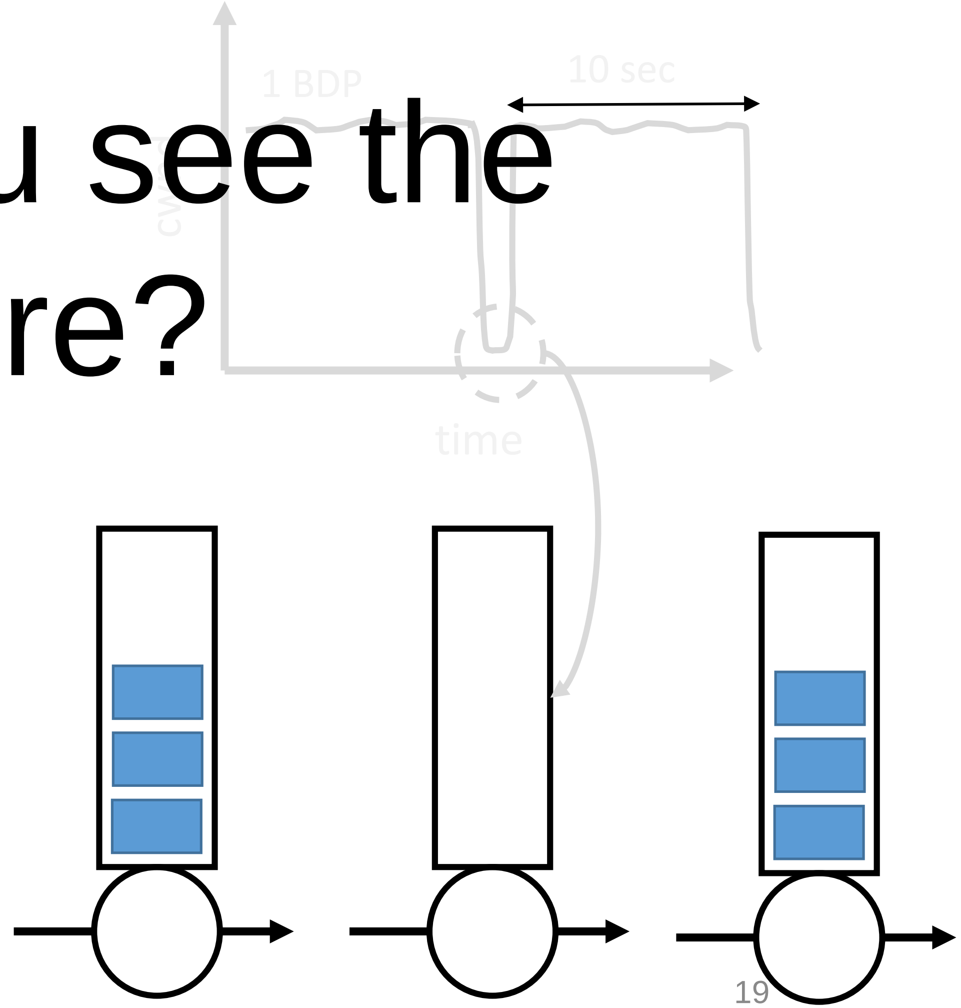
Google's solution: BBR

Rate-based congestion control algorithm. Uses **RTT$_{min}$** and **bandwidth** estimates to infer congestion.

Estimate the **bandwidth** using the receive rate

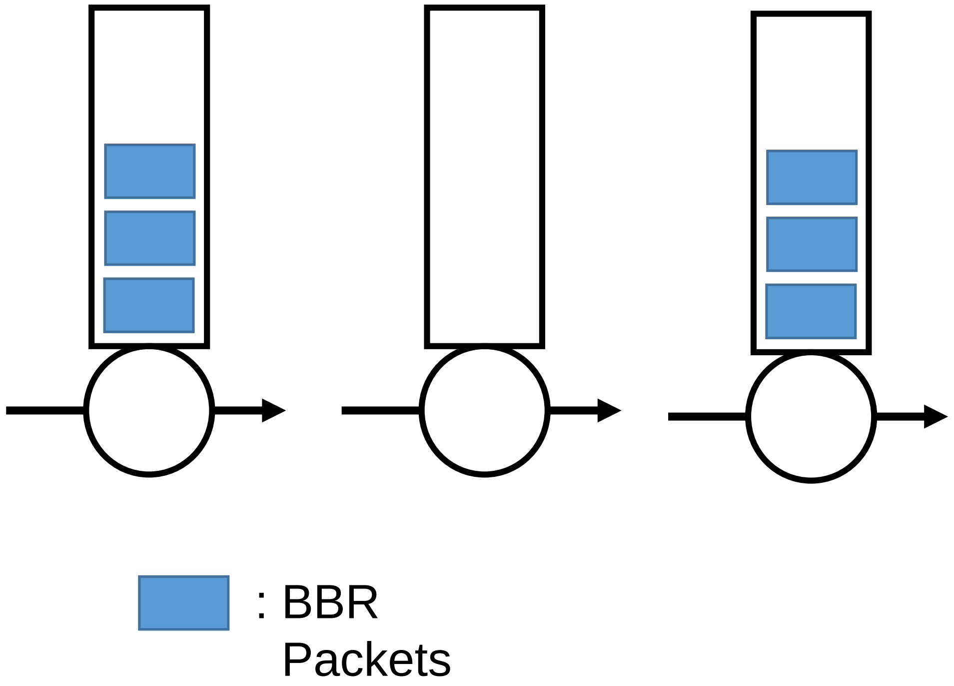
Question: Do you see the problem here?

Backs off every 10 sec to measure **RTT$_{min}$**



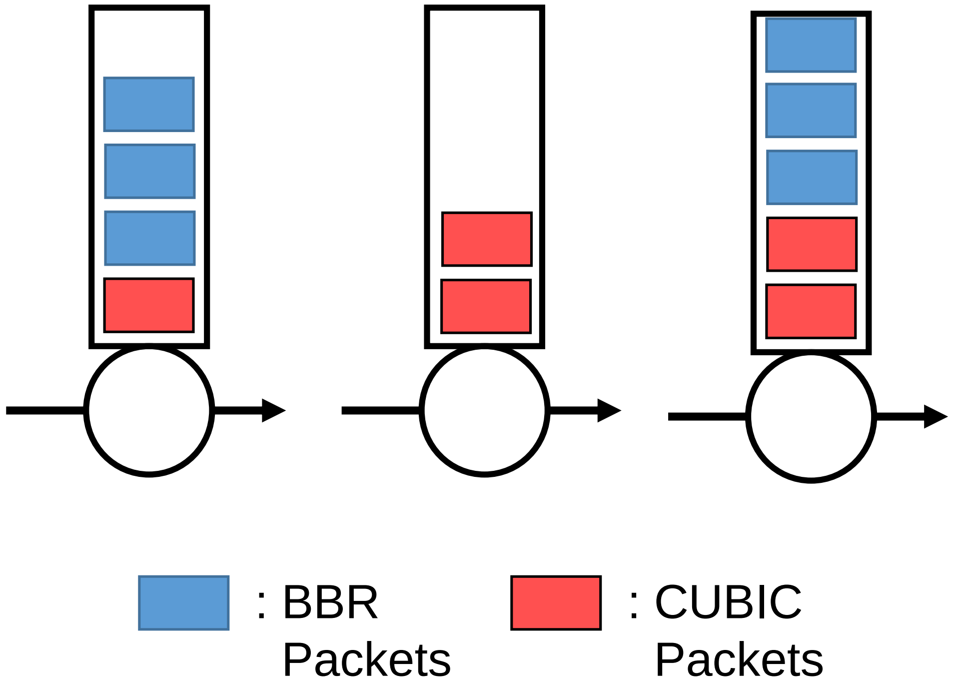
RTT_{min} overestimation

BBR wants to empty the buffer every 10 sec



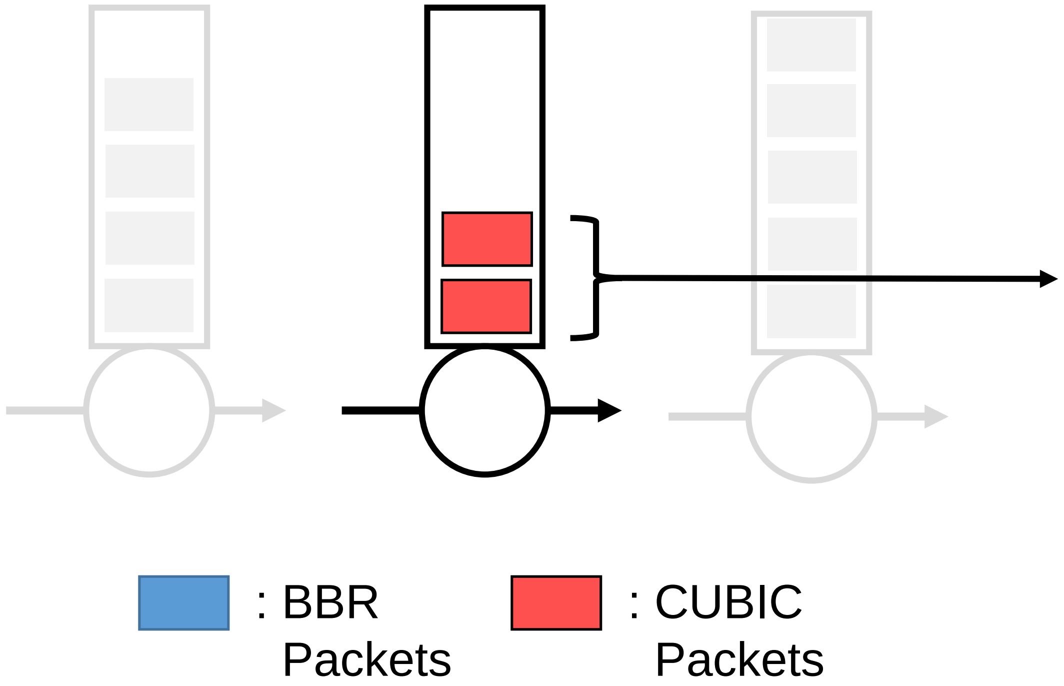
RTT_{min} overestimation

But BBR can’t empty the buffer every 10 seconds because of CUBIC’s packets!



RTT_{min} overestimation

But BBR can't empty the buffer every 10 seconds because of CUBIC's packets!

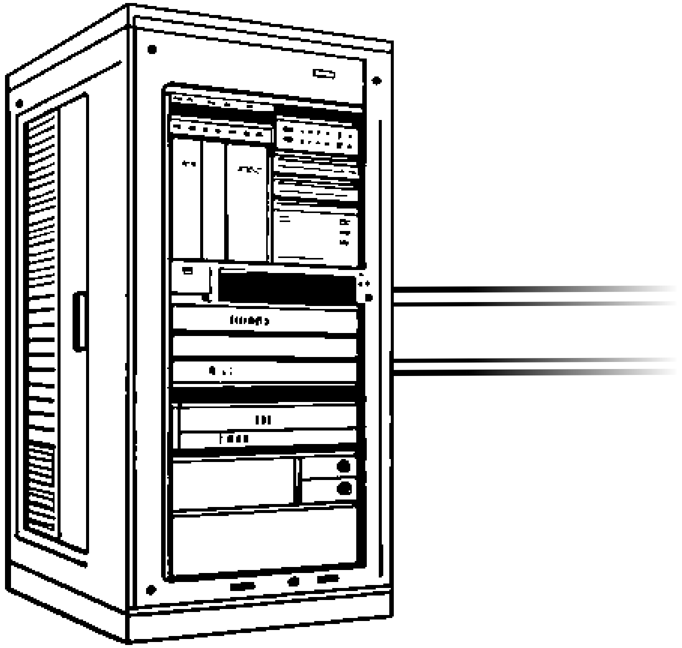


This leads to **RTT_{min} overestimation** for BBR

Bonus reason: CCA performance is contextual!

Recent developments in Internet Congestion Control

QUIC



#2 QUIC

Transport is moving to the user space, making it a lot easier to tweak and implement new CCAs!

The quick rise of QUIC

Userspace transport stack built over UDP

According to Sandvine, it contributes to

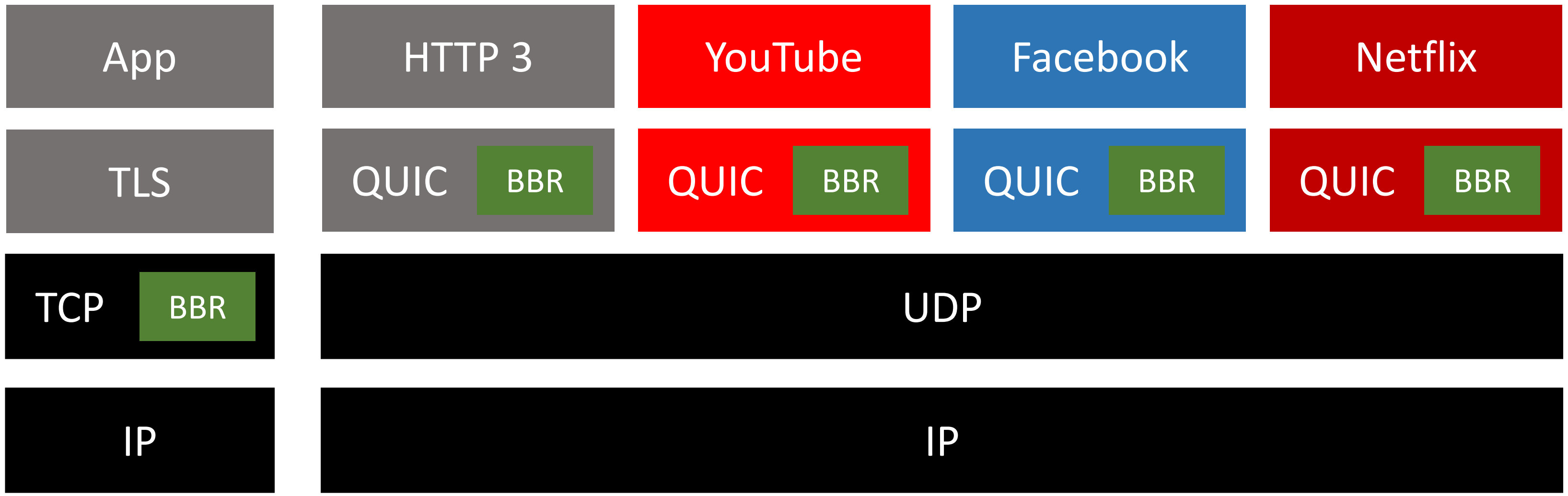
30% downstream traffic in EMEA

16% downstream traffic in North America

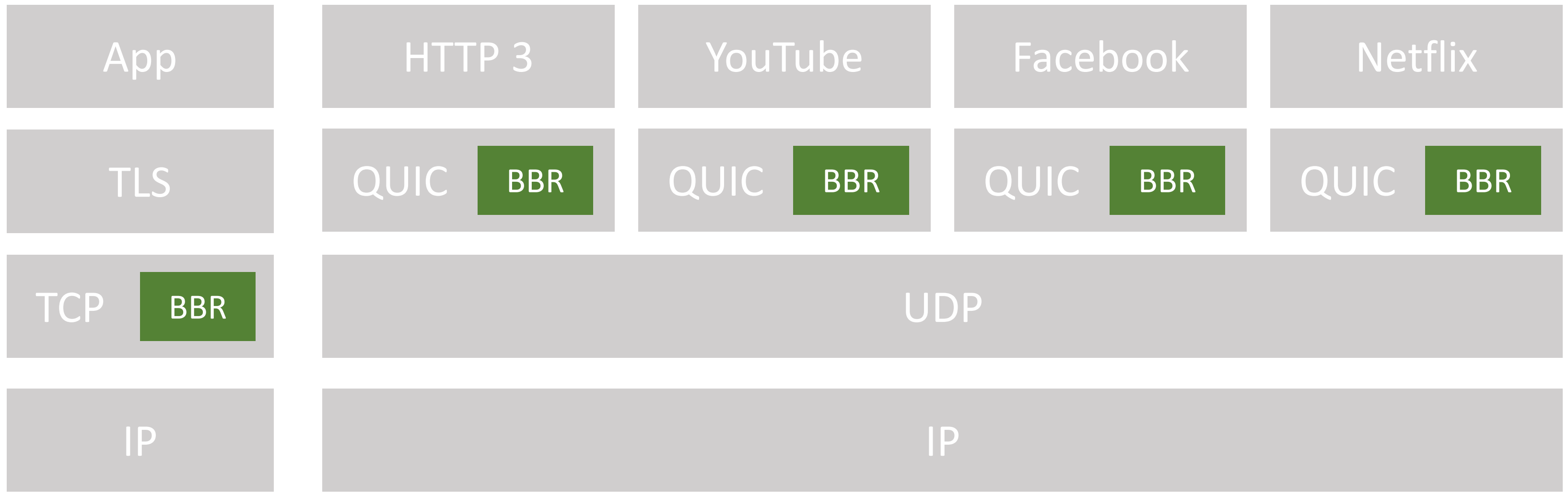
75% of Meta’s traffic

Standard with HTTP3

This **heterogeneity** risks being **increased** with the deployment of **QUIC**



This **heterogeneity** risks being **increased** with the deployment of **QUIC**



There is a **low barrier to the modification** of these re-implementations of standard CCAs.

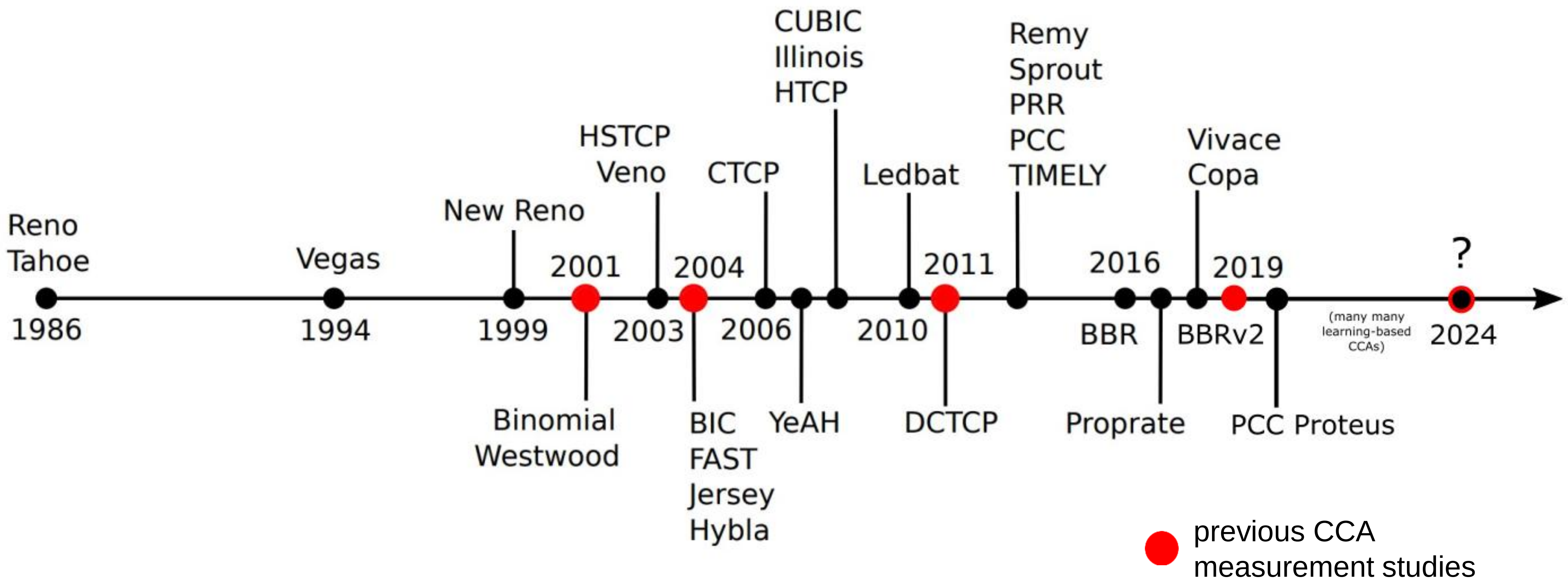
Different QUIC stacks re-implementing standard CCAs is like different fast food chains making their own version of the same old cheese burger.

It’s the same, but different

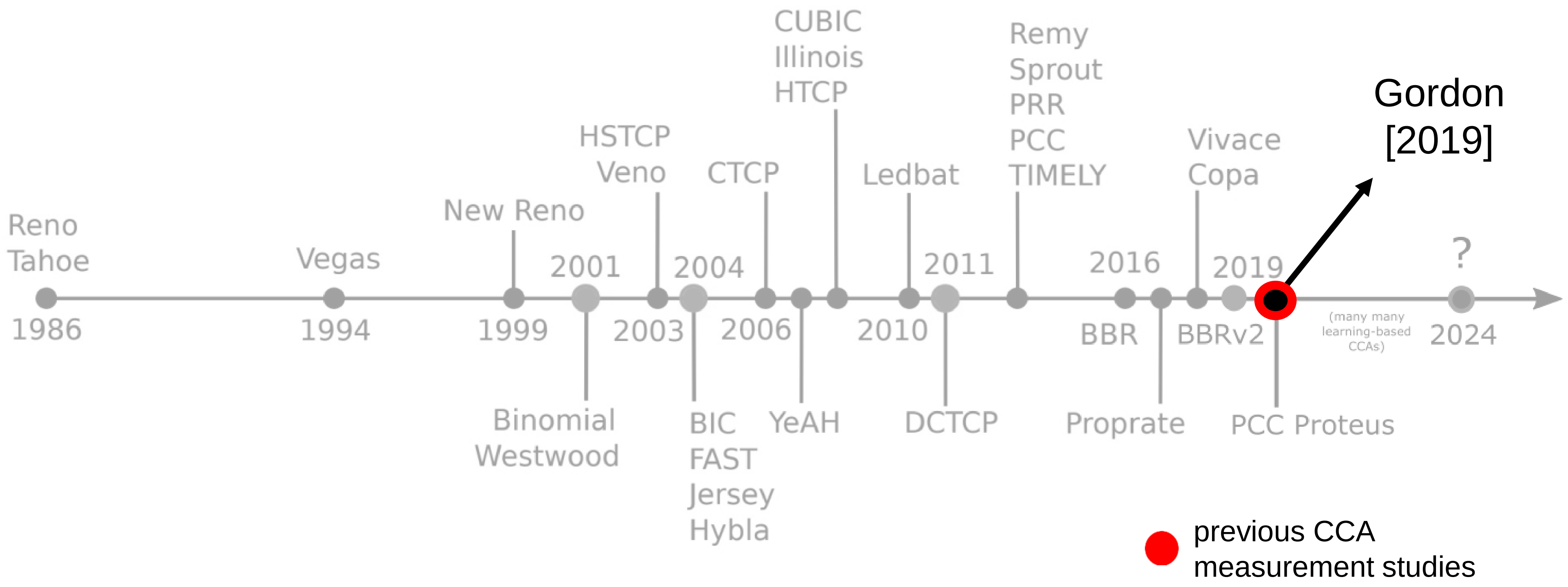
End-host Congestion Control is a unique design space where we expect users to meet their *selfish* goals without causing harm.

We also need to monitor the responsible deployment of Congestion Control Algorithms (CCAs) on the Internet.

This is not a new problem.



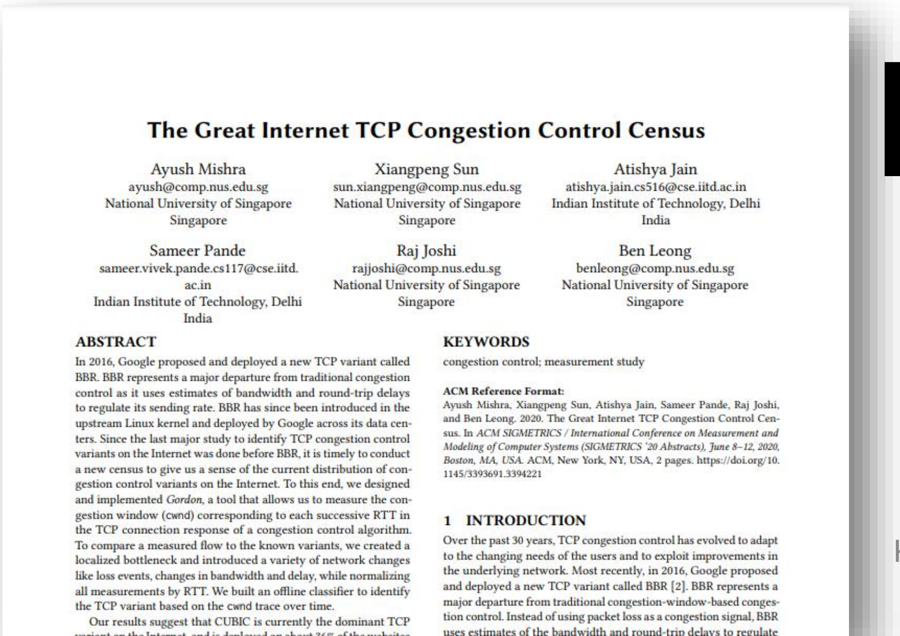
This is not a new problem.



So we decided to re-run the most recent of the measurement tools…

Gordon

[2019]



The Great Internet TCP Congestion Control Census

Ayush Mishra
ayush@comp.nus.edu.sg
National University of Singapore
Singapore

Xiangpeng Sun
sun.xiangpeng@comp.nus.edu.sg
National University of Singapore
Singapore

Atishya Jain
atishya.jain.cs516@cse.iitd.ac.in
Indian Institute of Technology, Delhi
India

Sameer Pande
sameer.vivek.pande.cs117@cse.iitd.ac.in
Indian Institute of Technology, Delhi
India

Raj Joshi
rajjoshi@comp.nus.edu.sg
National University of Singapore
Singapore

Ben Leong
benleong@comp.nus.edu.sg
National University of Singapore
Singapore

ABSTRACT

In 2016, Google proposed and deployed a new TCP variant called BBR. BBR represents a major departure from traditional congestion control as it uses estimates of bandwidth and round-trip delays to regulate its sending rate. BBR has since been introduced in the upstream Linux kernel and deployed by Google across its data centers. Since the last major study to identify TCP congestion control variants on the Internet was done before BBR, it is timely to conduct a new census to give us a sense of the current distribution of congestion control variants on the Internet. To this end, we designed and implemented *Gordon*, a tool that allows us to measure the congestion window (cwnd) corresponding to each successive RTT in the TCP connection response of a congestion control algorithm. To compare a measured flow to the known variants, we created a localized bottleneck and introduced a variety of network changes like loss events, changes in bandwidth and delay, while normalizing all measurements by RTT. We built an offline classifier to identify the TCP variant based on the cwnd trace over time.

Our results suggest that CUBIC is currently the dominant TCP

KEYWORDS

congestion control; measurement study

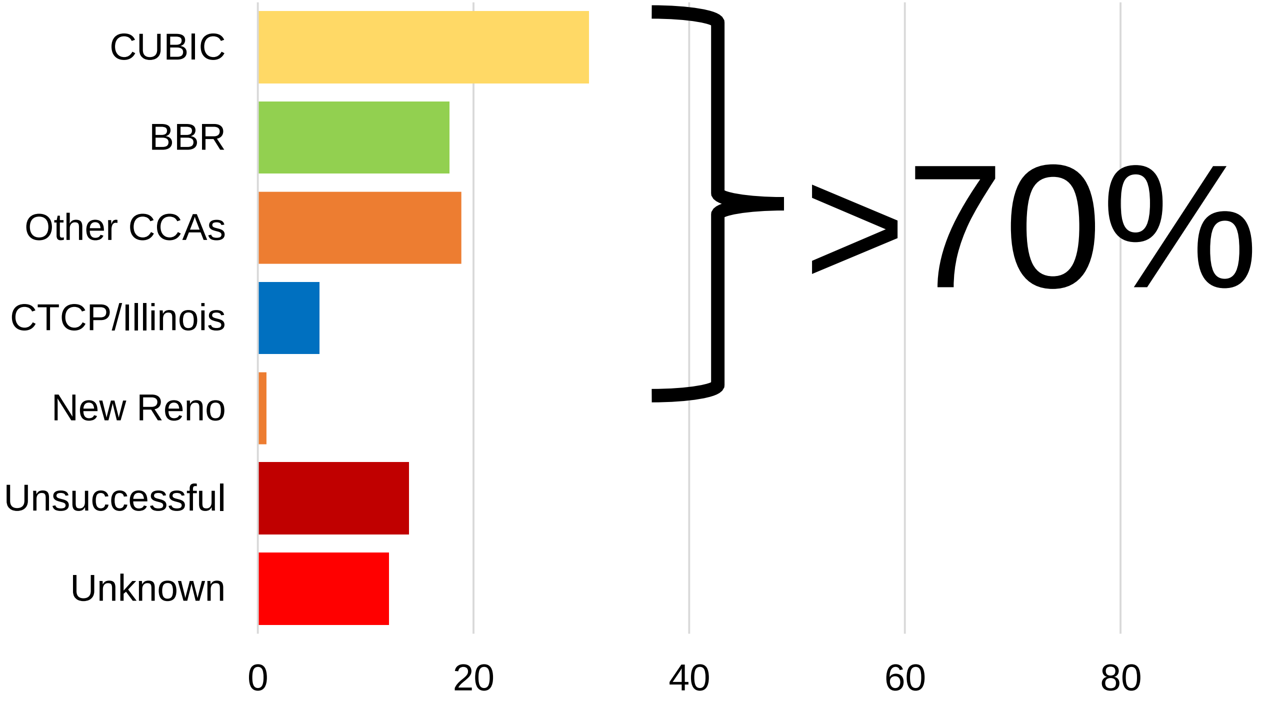
ACM Reference Format:

Ayush Mishra, Xiangpeng Sun, Atishya Jain, Sameer Pande, Raj Joshi, and Ben Leong. 2020. The Great Internet TCP Congestion Control Census. In *ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '20 Abstracts), June 8–12, 2020, Boston, MA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3393691.3394221

1 INTRODUCTION

Over the past 30 years, TCP congestion control has evolved to adapt to the changing needs of the users and to exploit improvements in the underlying network. Most recently, in 2016, Google proposed and deployed a new TCP variant called BBR [2]. BBR represents a major departure from traditional congestion-window-based congestion control. Instead of using packet loss as a congestion signal, BBR uses estimates of the bandwidth and round-trip delays to regulate

2019



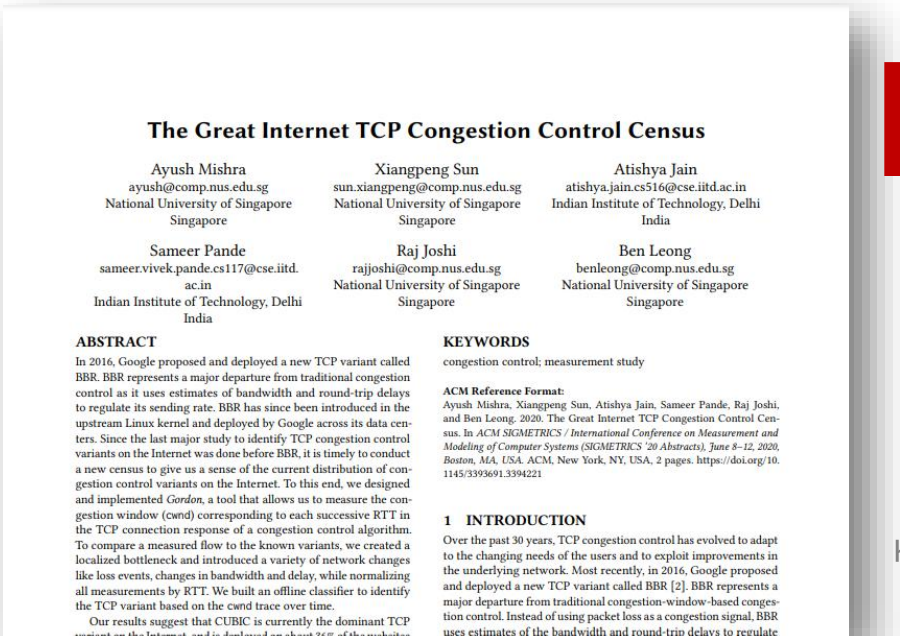
What does the Internet's current Congestion Control Landscape look like?

In any case, we decided to re-run Gordon…

but we were not successful.

Gordon

[2019]



The Great Internet TCP Congestion Control Census

Ayush Mishra
ayush@comp.nus.edu.sg
National University of Singapore
Singapore

Xiangpeng Sun
sun.xiangpeng@comp.nus.edu.sg
National University of Singapore
Singapore

Atishya Jain
atishya.jain.cs516@cse.iitd.ac.in
Indian Institute of Technology, Delhi
India

Sameer Pande
sameer.vivek.pande.cs117@cse.iitd.ac.in
Indian Institute of Technology, Delhi
India

Raj Joshi
rajjoshi@comp.nus.edu.sg
National University of Singapore
Singapore

Ben Leong
benleong@comp.nus.edu.sg
National University of Singapore
Singapore

ABSTRACT

In 2016, Google proposed and deployed a new TCP variant called BBR. BBR represents a major departure from traditional congestion control as it uses estimates of bandwidth and round-trip delays to regulate its sending rate. BBR has since been introduced in the upstream Linux kernel and deployed by Google across its data centers. Since the last major study to identify TCP congestion control variants on the Internet was done before BBR, it is timely to conduct a new census to give us a sense of the current distribution of congestion control variants on the Internet. To this end, we designed and implemented *Gordon*, a tool that allows us to measure the congestion window (cwnd) corresponding to each successive RTT in the TCP connection response of a congestion control algorithm. To compare a measured flow to the known variants, we created a localized bottleneck and introduced a variety of network changes like loss events, changes in bandwidth and delay, while normalizing all measurements by RTT. We built an offline classifier to identify the TCP variant based on the cwnd trace over time.

Our results suggest that CUBIC is currently the dominant TCP

KEYWORDS

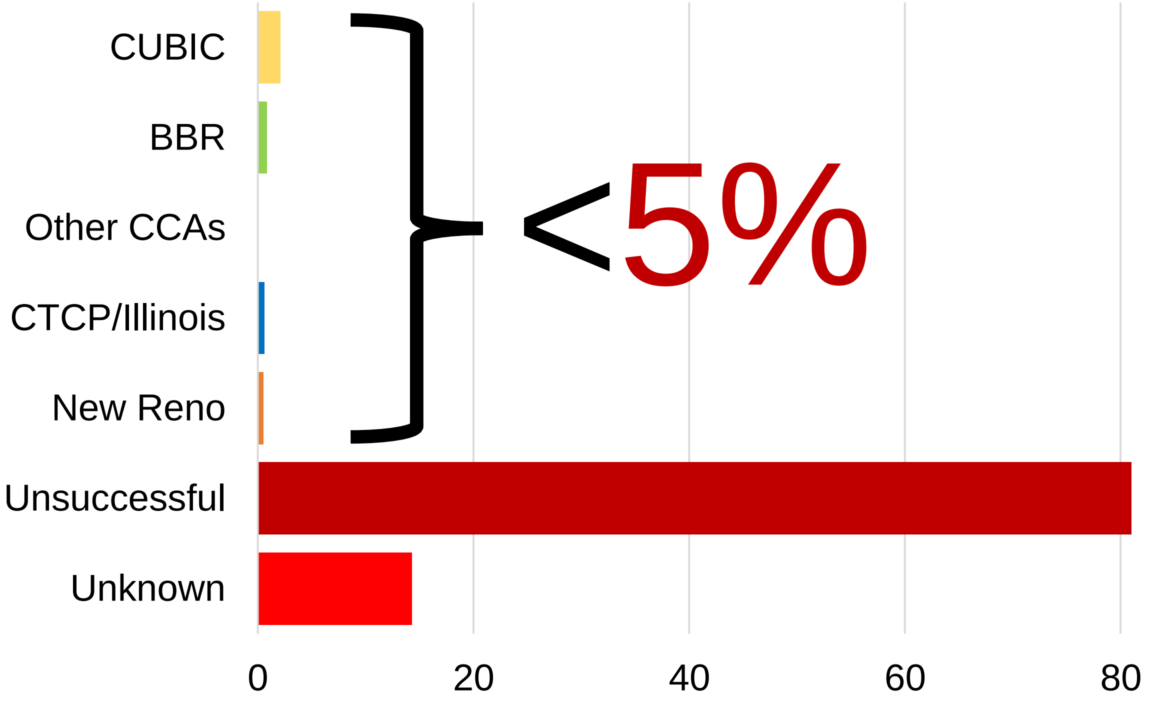
congestion control; measurement study

ACM Reference Format:
Ayush Mishra, Xiangpeng Sun, Atishya Jain, Sameer Pande, Raj Joshi, and Ben Leong. 2020. The Great Internet TCP Congestion Control Census. In *ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '20 Abstracts), June 8–12, 2020, Boston, MA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3393691.3394221

1 INTRODUCTION

Over the past 30 years, TCP congestion control has evolved to adapt to the changing needs of the users and to exploit improvements in the underlying network. Most recently, in 2016, Google proposed and deployed a new TCP variant called BBR [2]. BBR represents a major departure from traditional congestion-window-based congestion control. Instead of using packet loss as a congestion signal, BBR uses estimates of the bandwidth and round-trip delays to regulate

2023



This is has been a trend in all previous CCA classification tools.

Let’s identify CCAS

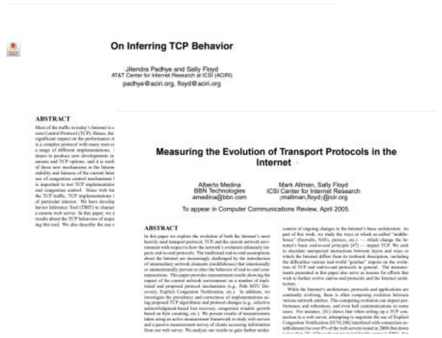
…but **TBIT** doesn’t work

TBIT

[2001,2005]

CAAI

[2011]



On Inferring TCP Behavior

Jitendra Padhye and Sally Floyd
AT&T Center for Internet Research at ICSI (ACIRI)
padhye@aciri.org, floyd@aciri.org

ABSTRACT

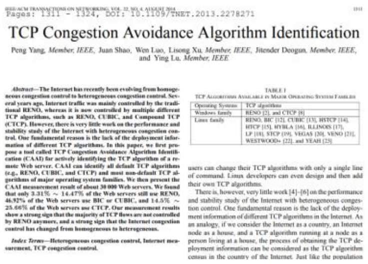
Measuring the Evolution of Transport Protocols in the Internet

Alberto Medina
BBN Technologies
amedina@bbn.com

Mark Allman, Sally Floyd
ICSI Center for Internet Research
{mallman,floyd}@icir.org

To appear in Computer Communications Review, April 2005.

ABSTRACT



Pages: 1311 - 1324, DOI: 10.1109/TNET.2013.2278271

TCP Congestion Avoidance Algorithm Identification

Peng Yang, *Member, IEEE*, Juan Shao, Wen Luo, Lisong Xu, *Member, IEEE*, Jitender Deogun, *Member, IEEE*, and Ying Lu, *Member, IEEE*

TABLE I
TCP ALGORITHMS AVAILABLE IN MAJOR OPERATING SYSTEM FAMILIES

Operating Systems	TCP algorithms
Windows family	RENO [2], and CTCP [8]
Linux family	RENO, BIC [12], CUBIC [13], HSTCP [14], HTCP [15], HYBLA [16], ILLINOIS [17], LP [18], STCP [19], VEGAS [20], VENO [21], WESTWOOD+ [22], and YEAH [23]

This is has been a trend in all previous CCA classification tools.

Let's identify CCAS

…but **TBIT** doesn't work

…but **CAAI** doesn't work

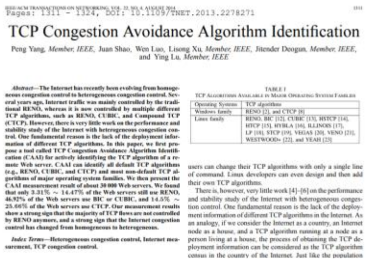
TBIT
[2001,2005]

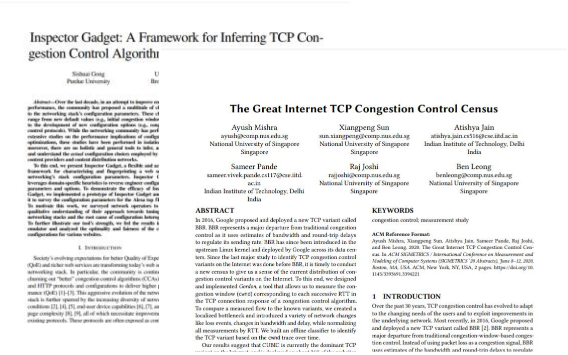
CAAI
[2011]

IG,Gordon
[2019]


On Inferring TCP Behavior

Measuring the Evolution of Transport Protocols in the Internet


TCP Congestion Avoidance Algorithm Identification


Inspector Gadget: A Framework for Inferring TCP Congestion Control Algorithms

The Great Internet TCP Congestion Control Census

This is has been a trend in all previous CCA classification tools.

Let's identify CCAS

…but **TBIT** doesn't work

…but **CAAI** doesn't work

but **IG** and **Gordon** don't work

TBIT
[2001,2005]


On Inferring TCP Behavior

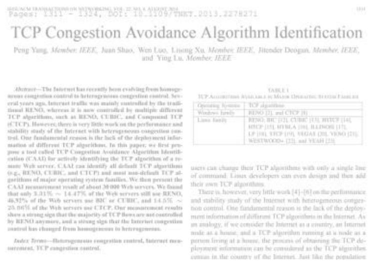
Jitendra Padhye and Sally Floyd

Measuring the Evolution of Transport Protocols in the Internet

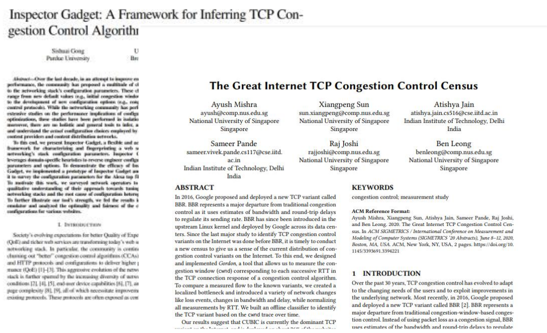
Alberto Medina, Mark Allman, Sally Floyd

To appear in Computer Communications Review, April 2005.

CAAI
[2011]


TCP Congestion Avoidance Algorithm Identification

IG,Gordon
[2019]


Inspector Gadget: A Framework for Inferring TCP Congestion Control Algorithm

Sishuai Gong, Purdue University

The Great Internet TCP Congestion Control Census

Ayush Mishra, ayush@comp.nus.edu.sg, National University of Singapore, Singapore
Xiangpeng Sun, sun.xiangpeng@comp.nus.edu.sg, National University of Singapore, Singapore
Atishya Jain, atishya.jain.cs516@cse.iitd.ac.in, Indian Institute of Technology, Delhi, India
Sameer Pande, sameer.vivek.pande.cs117@cse.iitd.ac.in, Indian Institute of Technology, Delhi, India
Raj Joshi, rajjoshi@comp.nus.edu.sg, National University of Singapore, Singapore
Ben Leong, benleong@comp.nus.edu.sg, National University of Singapore, Singapore

ABSTRACT

In 2016, Google proposed and deployed a new TCP variant called BBR. BBR represents a major departure from traditional congestion control as it uses estimates of bandwidth and round-trip delays to regulate its sending rate. BBR has since been introduced in the upstream Linux kernel and deployed by Google across its data centers. Since the last major study to identify TCP congestion control variants on the Internet was done before BBR, it is timely to conduct a new census to give us a sense of the current distribution of congestion control variants on the Internet. To this end, we designed and implemented *Gordon*, a tool that allows us to measure the congestion window (cwnd) corresponding to each successive RTT in the TCP connection response of a congestion control algorithm. To compare a measured flow to the known variants, we created a localized bottleneck and introduced a variety of network changes like loss events, changes in bandwidth and delay, while normalizing all measurements by RTT. We built an offline classifier to identify the TCP variant based on the cwnd trace over time.

Our results suggest that CUBIC is currently the dominant TCP

KEYWORDS

congestion control; measurement study

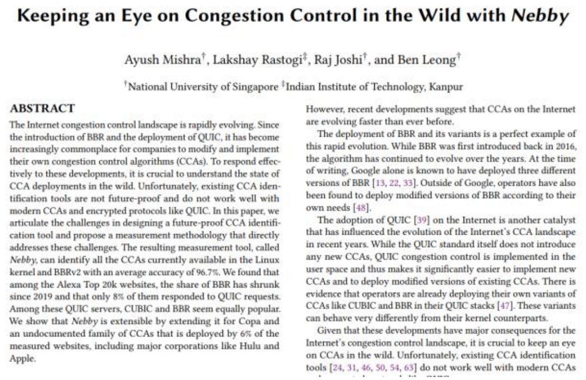
ACM Reference Format:

Ayush Mishra, Xiangpeng Sun, Atishya Jain, Sameer Pande, Raj Joshi, and Ben Leong. 2020. The Great Internet TCP Congestion Control Census. In *ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '20 Abstracts), June 8–12, 2020, Boston, MA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3393691.3394221

1 INTRODUCTION

Over the past 30 years, TCP congestion control has evolved to adapt to the changing needs of the users and to exploit improvements in the underlying network. Most recently, in 2016, Google proposed and deployed a new TCP variant called BBR [2]. BBR represents a major departure from traditional congestion-window-based congestion control. Instead of using packet loss as a congestion signal, BBR uses estimates of the bandwidth and round-trip delays to regulate

Nebby
[2024]


Keeping an Eye on Congestion Control in the Wild with *Nebby*

Ayush Mishra†, Lakshay Rastogi‡, Raj Joshi†, and Ben Leong†

†National University of Singapore ‡Indian Institute of Technology, Kanpur

ABSTRACT

The Internet congestion control landscape is rapidly evolving. Since the introduction of BBR and the deployment of QUIC, it has become increasingly commonplace for companies to modify and implement their own congestion control algorithms (CCAs). To respond effectively to these developments, it is crucial to understand the state of CCA deployments in the wild. Unfortunately, existing CCA identification tools are not future-proof and do not work well with modern CCAs and encrypted protocols like QUIC. In this paper, we articulate the challenges in designing a future-proof CCA identification tool and propose a measurement methodology that directly addresses these challenges. The resulting measurement tool, called *Nebby*, can identify all the CCAs currently available in the Linux kernel and BBRv2 with an average accuracy of 96.7%. We found that among the Alexa Top 20k websites, the share of BBR has shrunk since 2019 and that only 8% of them responded to QUIC requests. Among these QUIC servers, CUBIC and BBR seem equally popular. We show that *Nebby* is extensible by extending it for Copa and an undocumented family of CCAs that is deployed by 6% of the measured websites, including major corporations like Hulu and Apple.

However, recent developments suggest that CCAs on the Internet are evolving faster than ever before.

The deployment of BBR and its variants is a perfect example of this rapid evolution. While BBR was first introduced back in 2016, the algorithm has continued to evolve over the years. At the time of writing, Google alone is known to have deployed three different versions of BBR [13, 22, 33]. Outside of Google, operators have also been found to deploy modified versions of BBR according to their own needs [48].

The adoption of QUIC [39] on the Internet is another catalyst that has influenced the evolution of the Internet's CCA landscape in recent years. While the QUIC standard itself does not introduce any new CCAs, QUIC congestion control is implemented in the user space and thus makes it significantly easier to implement new CCAs and to deploy modified versions of existing CCAs. There is evidence that operators are already deploying their own variants of CCAs like CUBIC and BBR in their QUIC stacks [47]. These variants can behave very differently from their kernel counterparts.

Given that these developments have major consequences for the Internet's congestion control landscape, it is crucial to keep an eye on CCAs in the wild. Unfortunately, existing CCA identification tools [24, 31, 46, 50, 54, 63] do not work well with modern CCAs

Crucial design goal: let's also be ✨ Future-proof ✨

Let's identify CCAS

…but **TBIT** doesn't work

…but **CAAI** doesn't work

but **IG** and **Gordon** don't work

TBIT
[2001,2005]

CAAI
[2011]

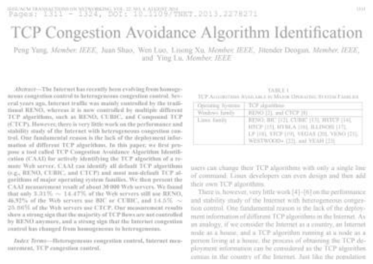
IG,Gordon
[2019]

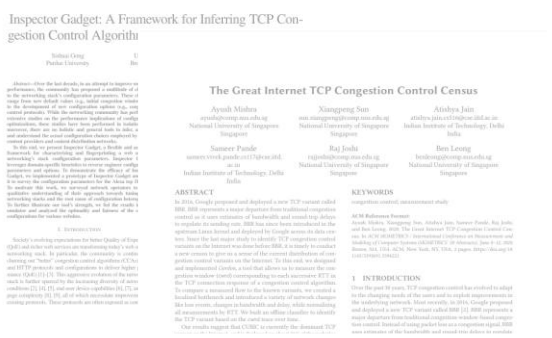
Nebby
[2024]


On Inferring TCP Behavior

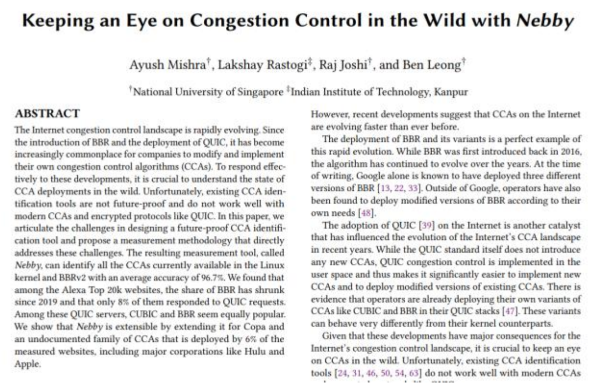
Jitendra Padhye and Sally Floyd

Measuring the Evolution of Transport Protocols in the Internet


TCP Congestion Avoidance Algorithm Identification


Inspector Gadget: A Framework for Inferring TCP Congestion Control Algorithms

The Great Internet TCP Congestion Control Census


Keeping an Eye on Congestion Control in the Wild with *Nebby*

Ayush Mishra[†], Lakshay Rastogi[‡], Raj Joshi[†], and Ben Leong[†]

[†]National University of Singapore [‡]Indian Institute of Technology, Kanpur

ABSTRACT

The Internet congestion control landscape is rapidly evolving. Since the introduction of BBR and the deployment of QUIC, it has become increasingly commonplace for companies to modify and implement their own congestion control algorithms (CCAs). To respond effectively to these developments, it is crucial to understand the state of CCA deployments in the wild. Unfortunately, existing CCA identification tools are not future-proof and do not work well with modern CCAs and encrypted protocols like QUIC. In this paper, we articulate the challenges in designing a future-proof CCA identification tool and propose a measurement methodology that directly addresses these challenges. The resulting measurement tool, called *Nebby*, can identify all the CCAs currently available in the Linux kernel and BBRv2 with an average accuracy of 96.7%. We found that among the Alexa Top 20k websites, the share of BBR has shrunk since 2019 and that only 8% of them responded to QUIC requests. Among these QUIC servers, CUBIC and BBR seem equally popular. We show that *Nebby* is extensible by extending it for Copa and an undocumented family of CCAs that is deployed by 6% of the measured websites, including major corporations like Hulu and Apple.

However, recent developments suggest that CCAs on the Internet are evolving faster than ever before.

The deployment of BBR and its variants is a perfect example of this rapid evolution. While BBR was first introduced back in 2016, the algorithm has continued to evolve over the years. At the time of writing, Google alone is known to have deployed three different versions of BBR [13, 22, 33]. Outside of Google, operators have also been found to deploy modified versions of BBR according to their own needs [48].

The adoption of QUIC [39] on the Internet is another catalyst that has influenced the evolution of the Internet's CCA landscape in recent years. While the QUIC standard itself does not introduce any new CCAs, QUIC congestion control is implemented in the user space and thus makes it significantly easier to implement new CCAs and to deploy modified versions of existing CCAs. There is evidence that operators are already deploying their own variants of CCAs like CUBIC and BBR in their QUIC stacks [47]. These variants can behave very differently from their kernel counterparts.

Given that these developments have major consequences for the Internet's congestion control landscape, it is crucial to keep an eye on CCAs in the wild. Unfortunately, existing CCA identification tools [24, 31, 46, 50, 54, 63] do not work well with modern CCAs

Why is CCA identification hard?

People keep deploying *new* CCAs – we can't be ad hoc

Work well with a wider range of applications and application traffic – be client-agnostic

We can't appear hostile – we need to be as passive as possible

Why is CCA identification hard?

People keep deploying *new* CCAs – we **can’t be ad hoc**

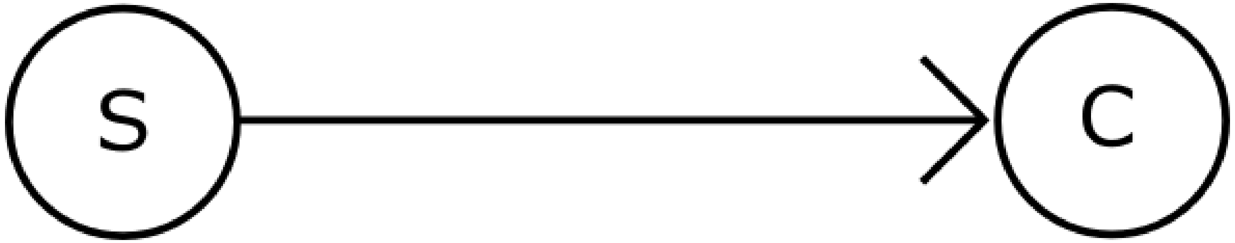
Work well with a wider range of applications and application traffic – be **client-agnostic**

We can’t appear hostile – we need to be as **passive** as possible

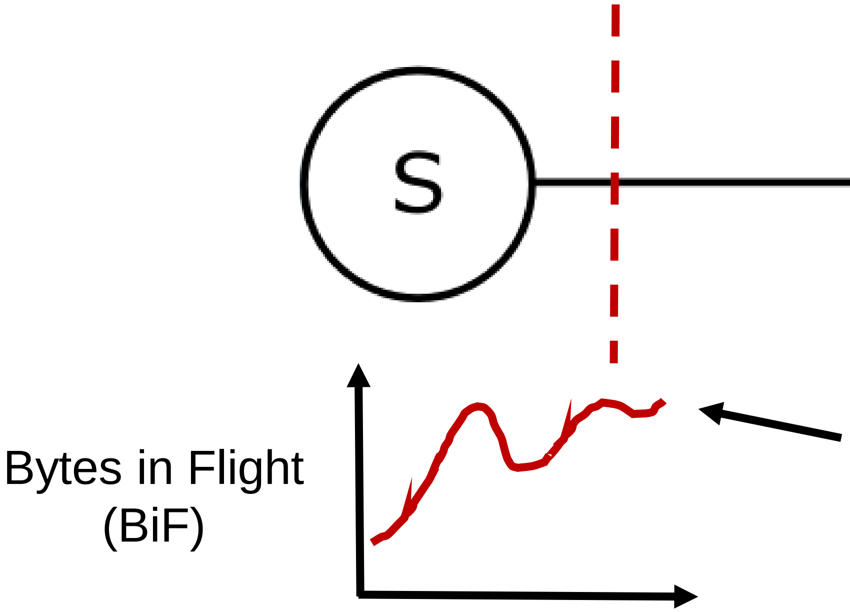
There *is* an obvious way to identify CCAs while meeting all these criteria, albeit in the controlled setting.

Let’s review the task we have at hand:

Identify the Congestion Control Algorithm run by a server

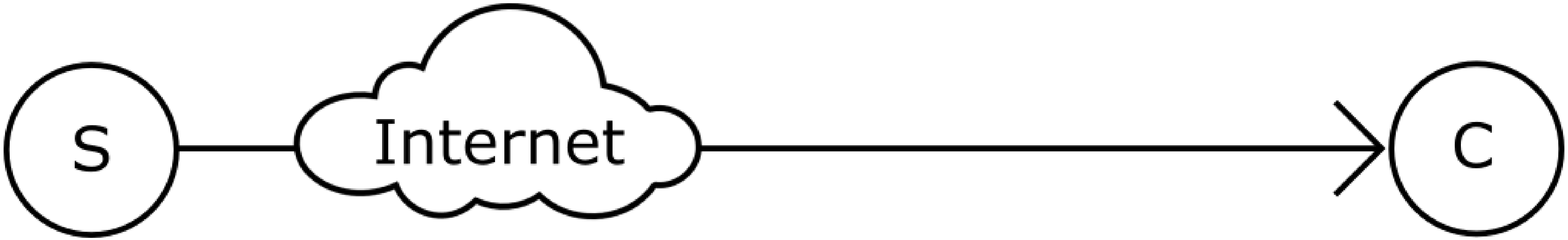


Identify the Congestion Control Algorithm run by a server



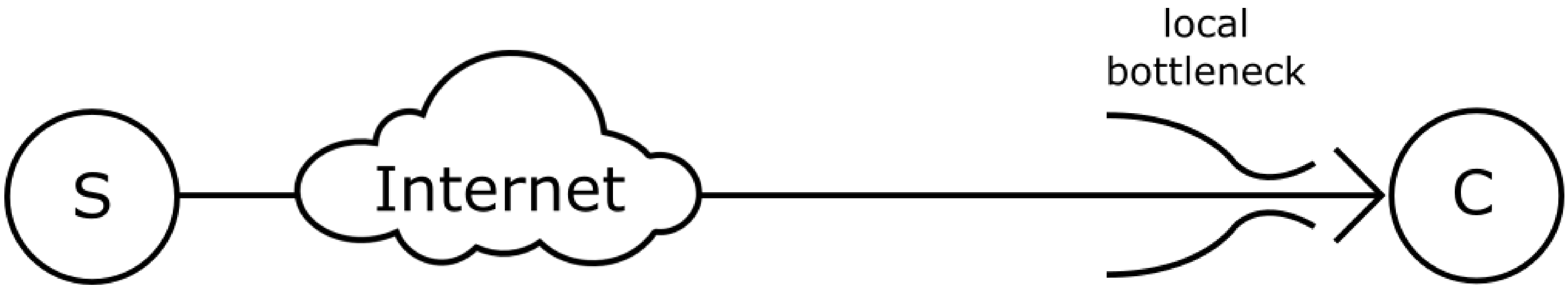
Unique for a CCA in response to a set of some specific network conditions

Identify the Congestion Control Algorithm run by a remote server on the Internet

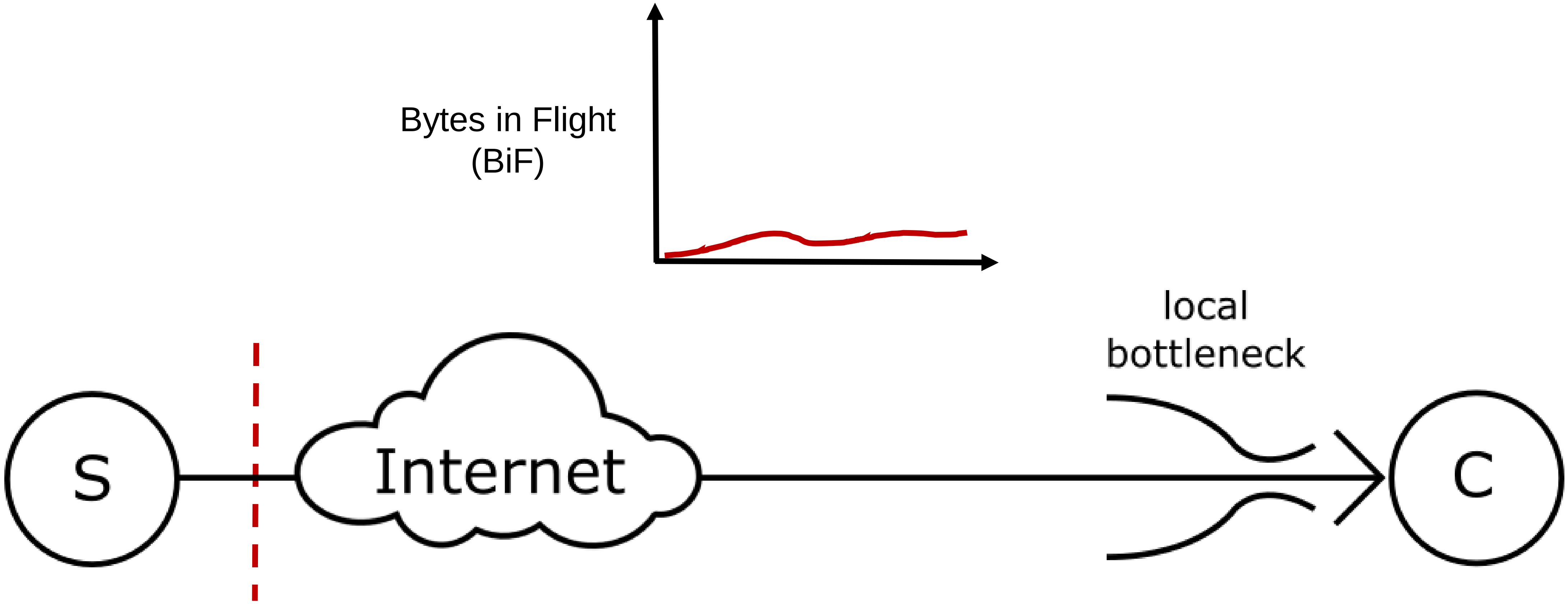


We no more have control over the network conditions

Identify the Congestion Control Algorithm run by a remote server on the Internet

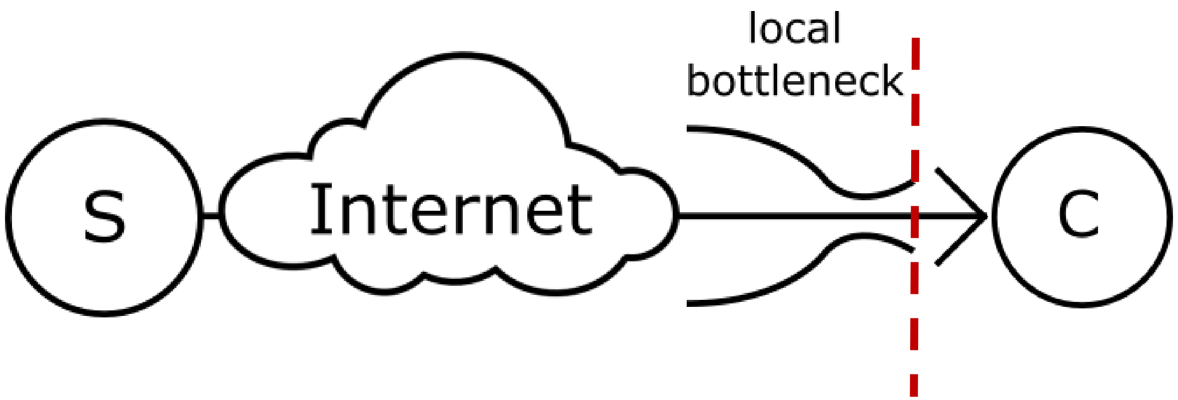


We can solve this how we have solved it before: via a localized bottleneck

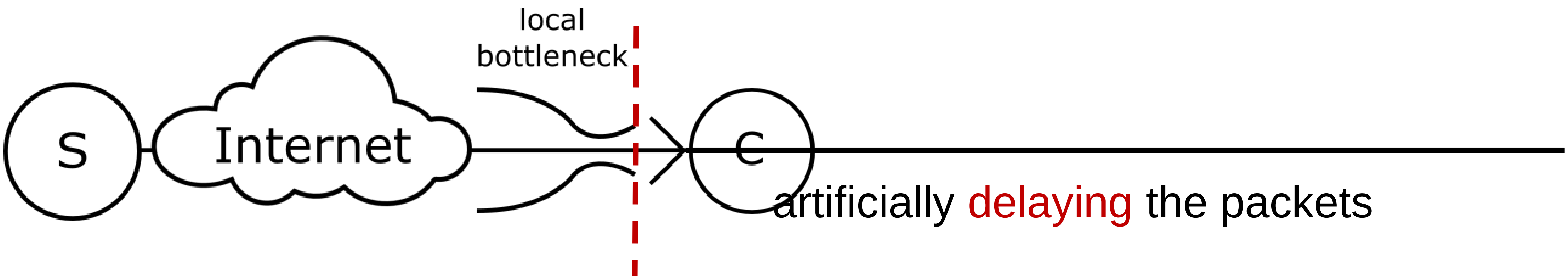

Bytes in Flight
(BiF)
local
bottleneck
S
Internet
C

What’s the solution here?

We can’t go any nearer to the server…

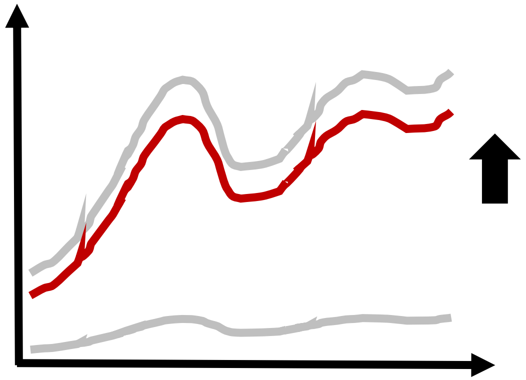


…but we can get further from the client

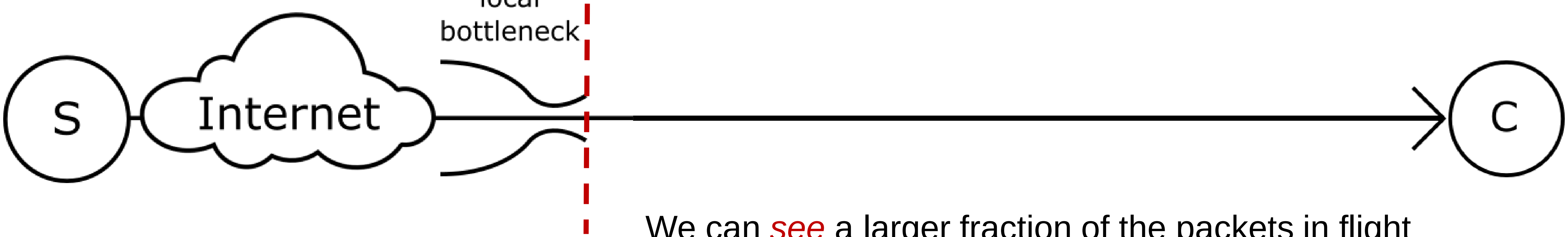


artificially delaying the packets

Bytes in Flight
(BiF)



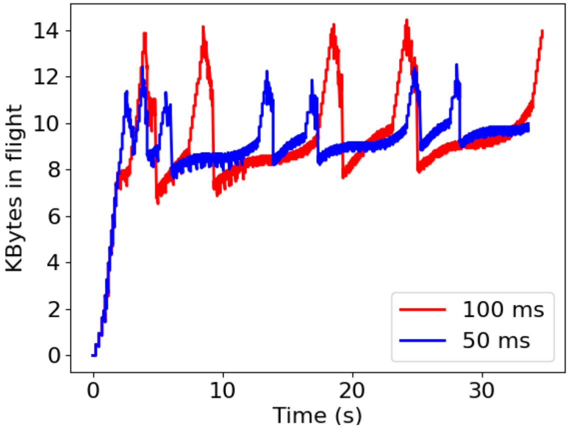
Measured BiF
closer to the actual



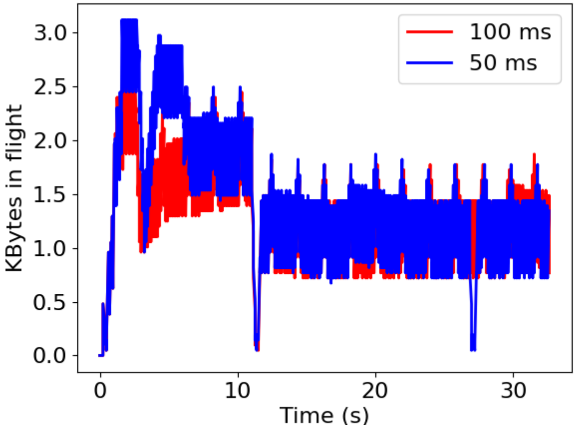
We can *see* a larger fraction of the packets in flight

Nebby's measurement methodology is built on this key insight

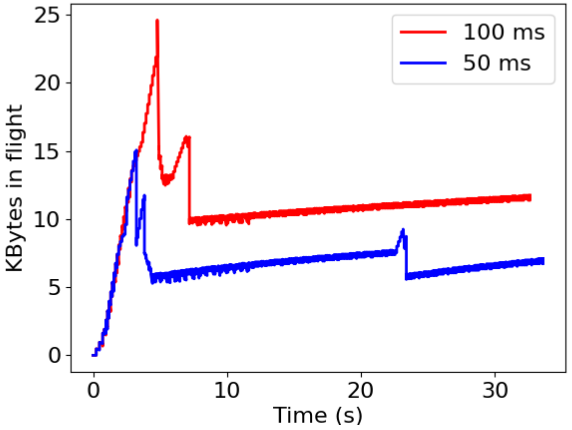
This simple strategy is good enough to capture distinct BiF traces for most CCAs



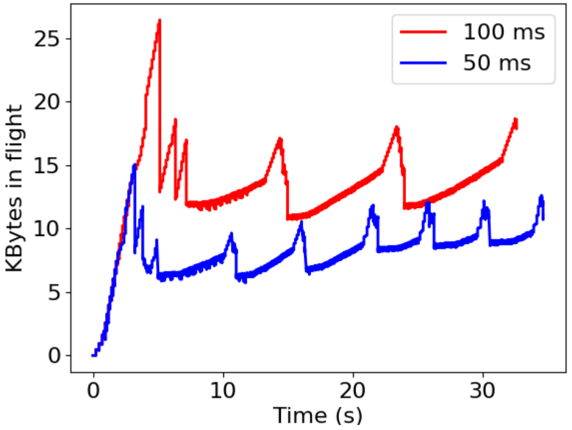
TCP CUBIC



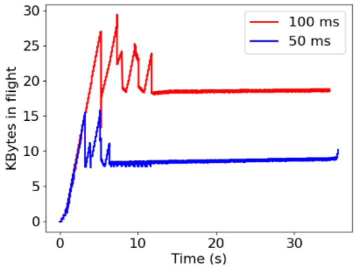
TCP BBR



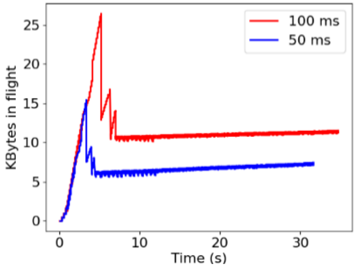
TCP New Reno



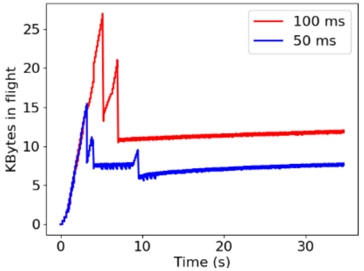
HTCP



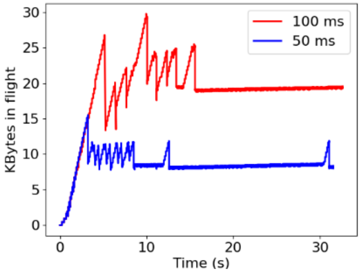
TCP BIC



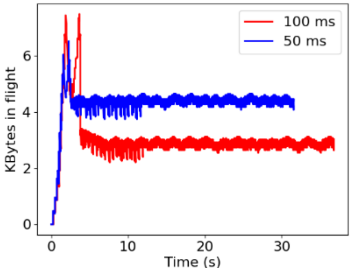
TCP Highspeed



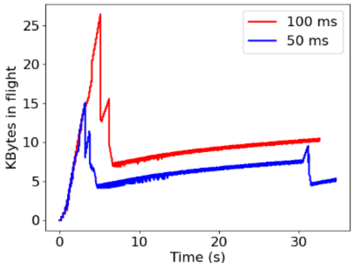
TCP Illinois



TCP Scalable

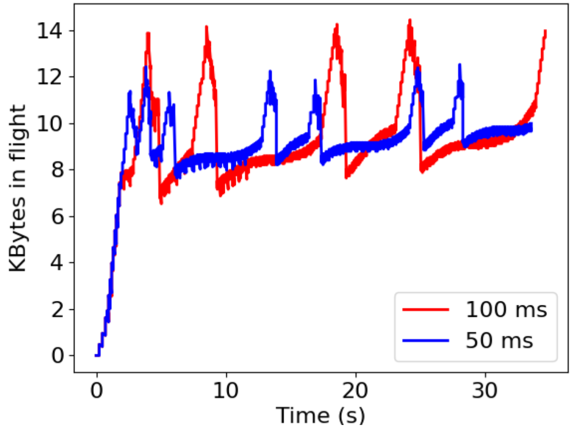


TCP Vegas

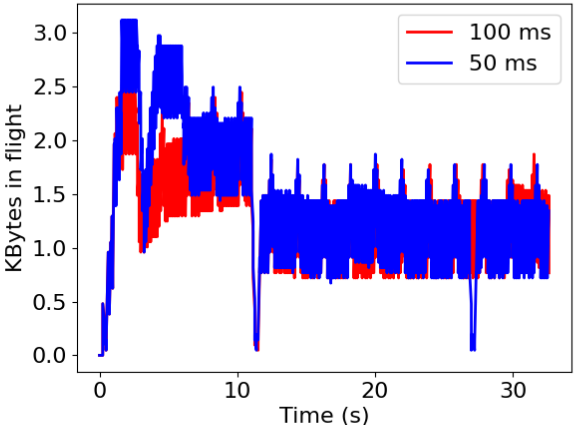


TCP Westwood

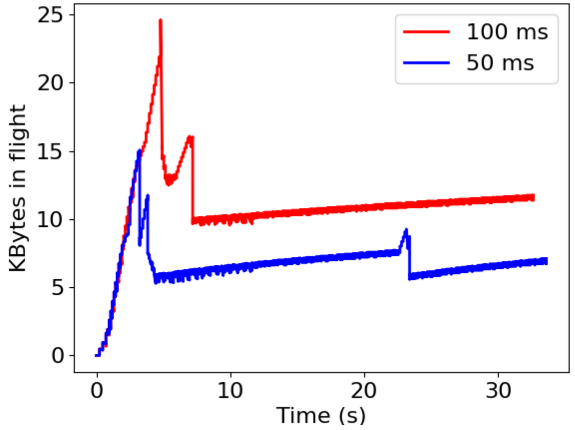
This simple strategy is good enough to capture distinct BiF traces for most CCAs



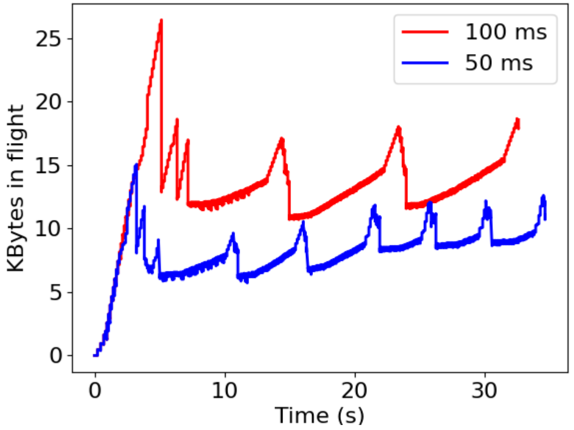
TCP CUBIC



TCP BBR

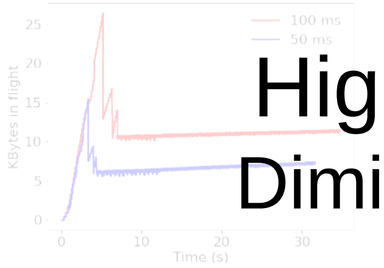


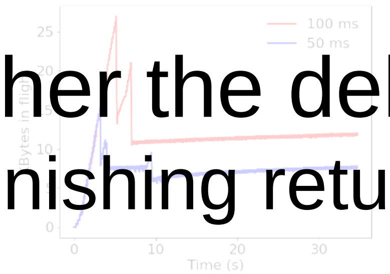
TCP New Reno

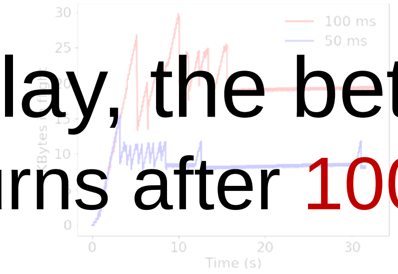


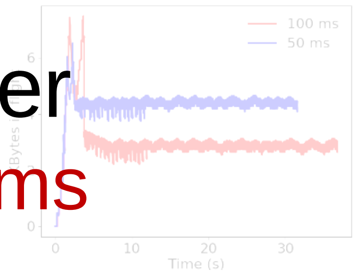
HTCP

TCP BIC


TCP Highspeed


TCP Illinois


TCP Scalable

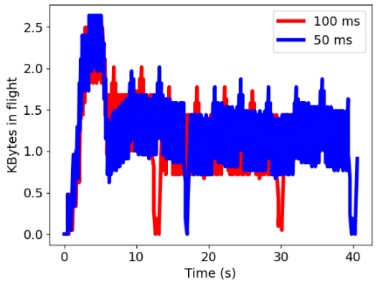

TCP Vegas

TCP Westwood

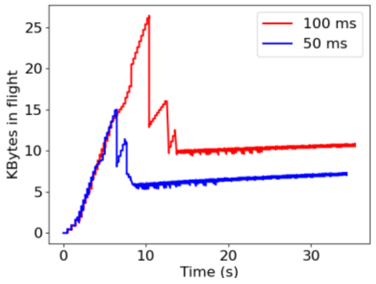
Higher the delay, the better

Diminishing returns after 100ms

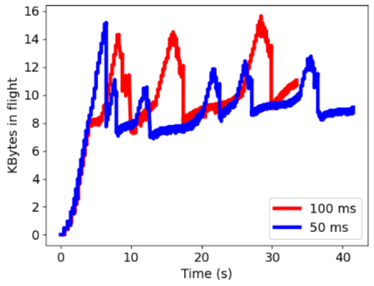
Moreover, since Nebby is mostly passive, it can work with QUIC and real browser traffic too



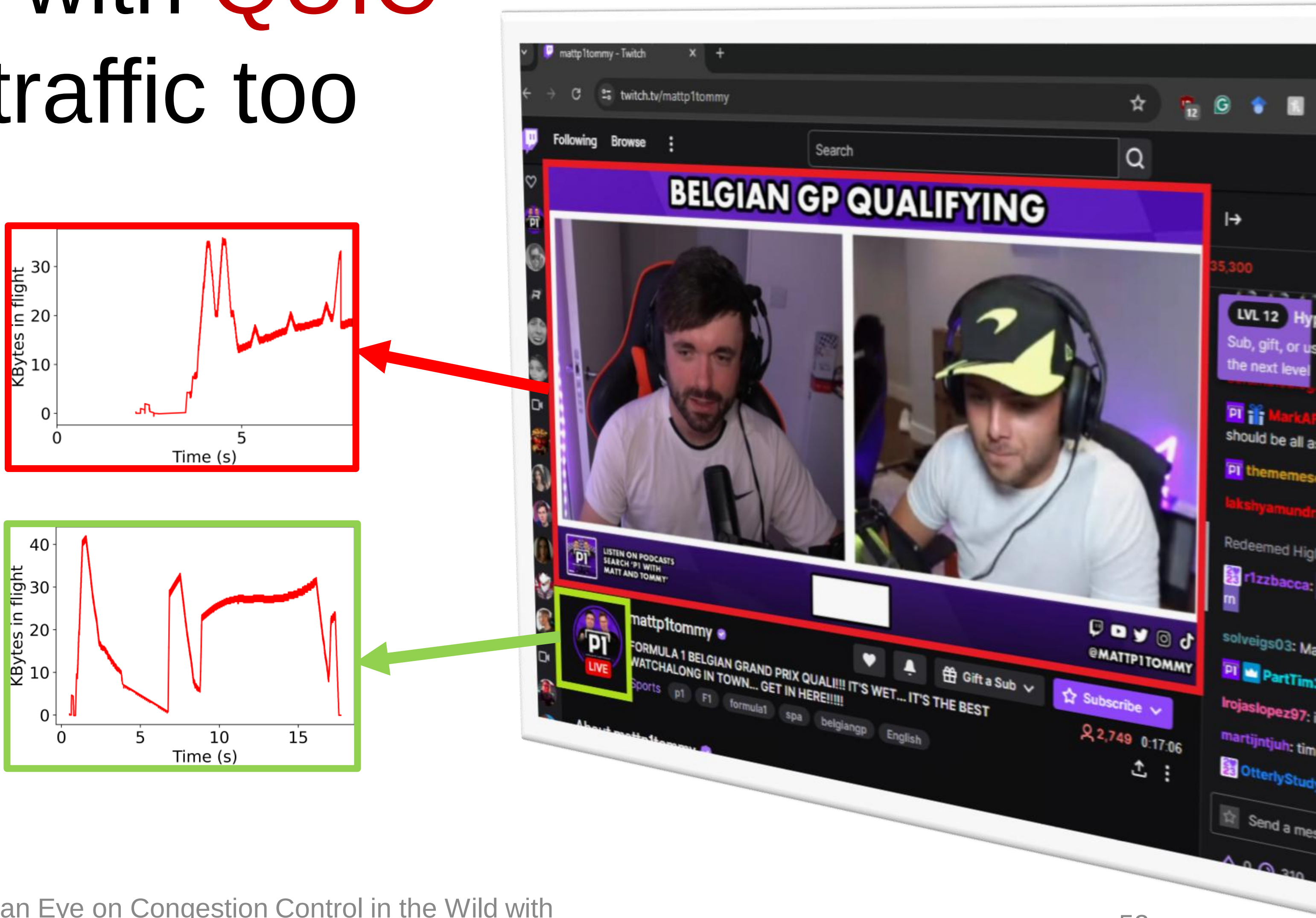
mvfst BBR



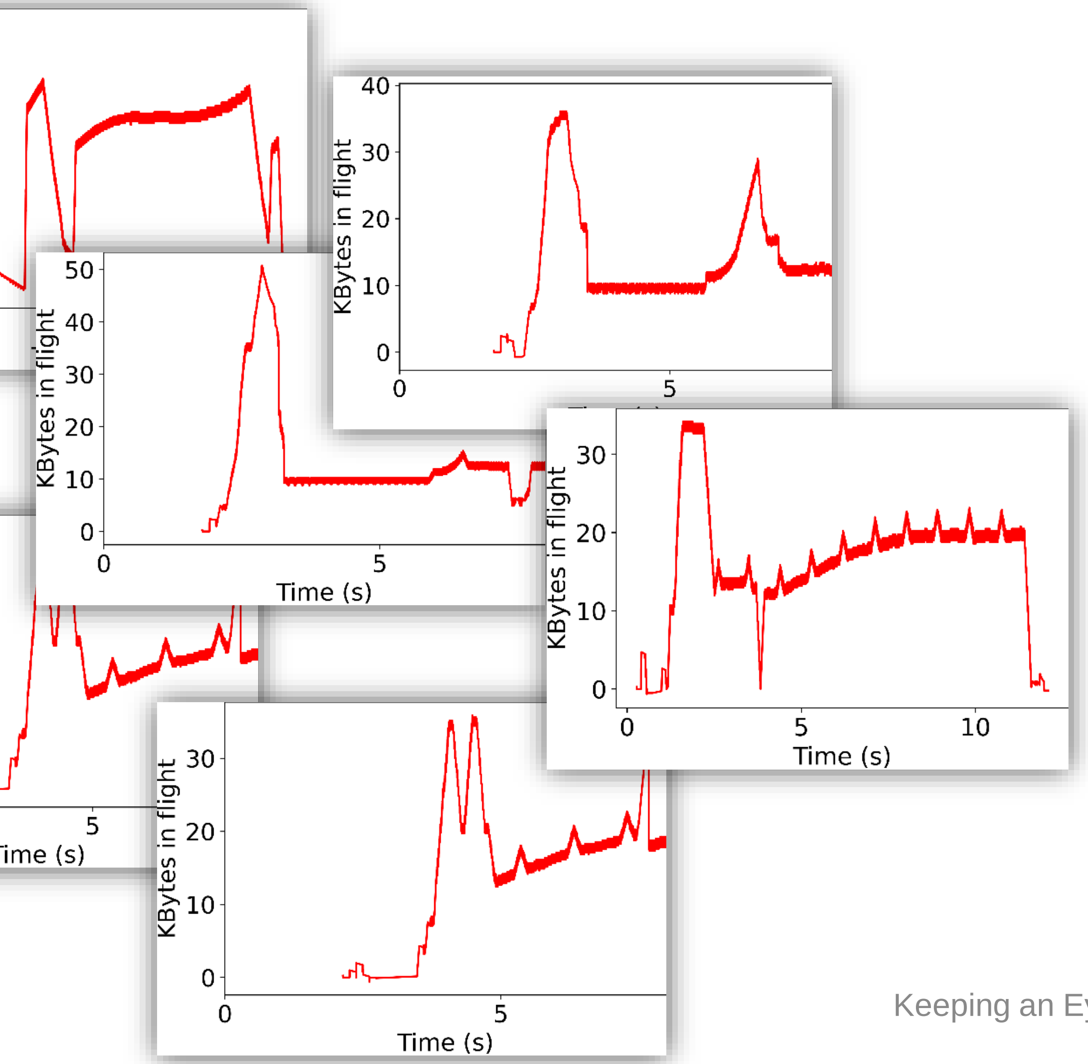
quicgo BBR



quiche CUBIC



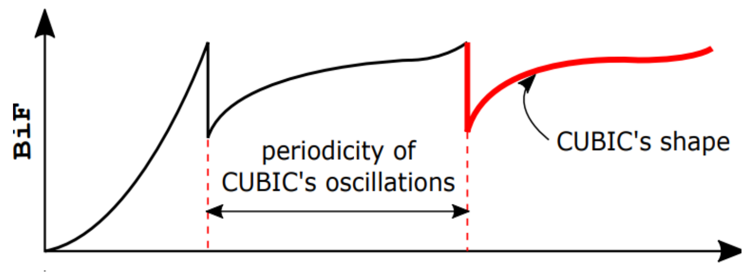
How do we build a classifier for all these traces?



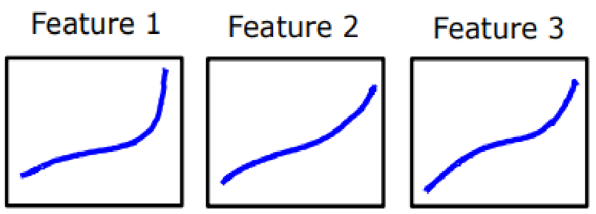
Avoiding a common pitfall: ML-based classifier

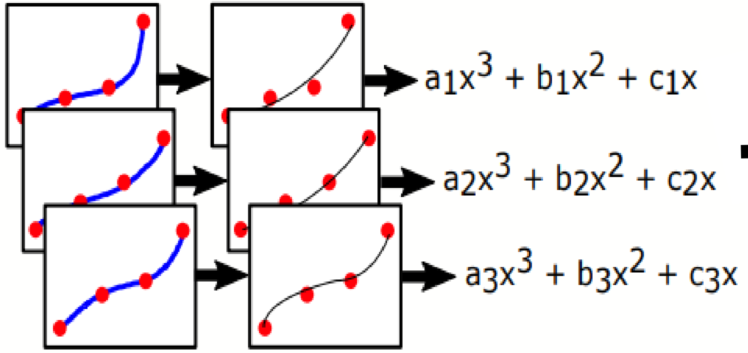
Key insight:
All *reasonable* CCAs are feedback loops with periodic probes and oscillations in the *congestion avoidance* phase.

Using a shape-based Classifier

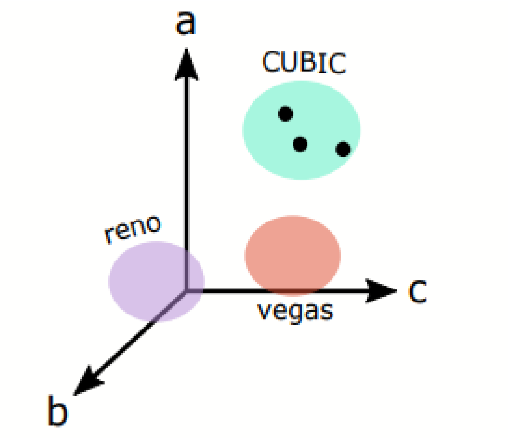


Extract these periodic regions: duration and frequency





Fit Polynomials and compare them with the polynomials for oscillations of known CCAs

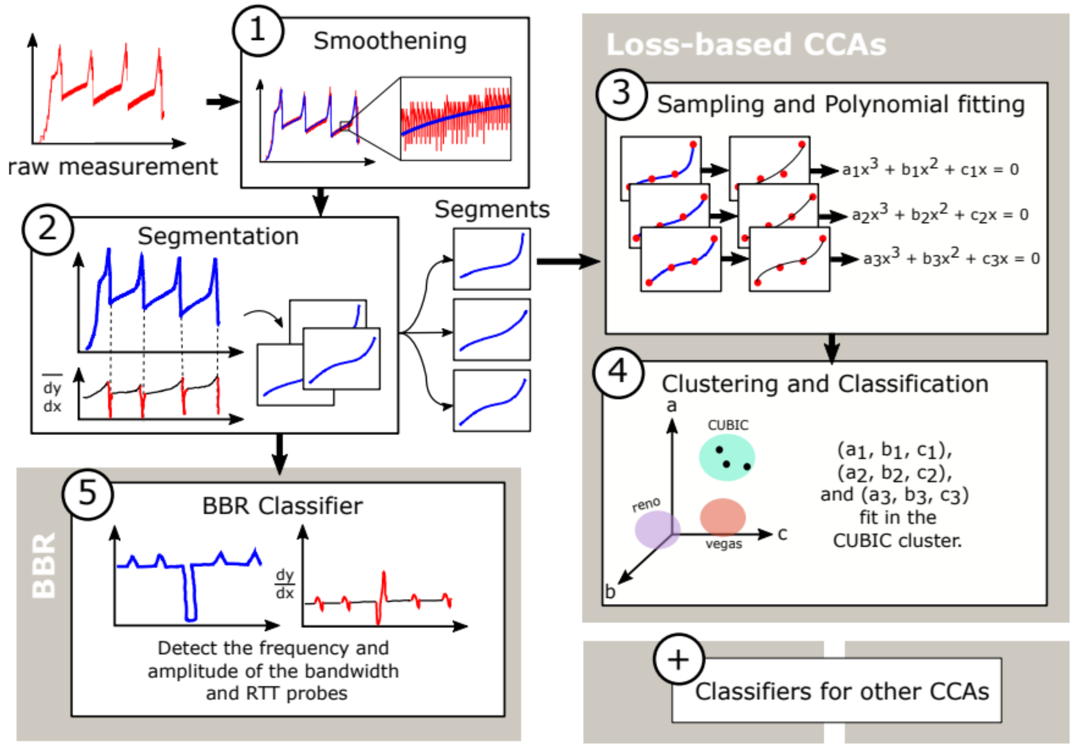


(a_1, b_1, c_1), (a_2, b_2, c_2), and (a_3, b_3, c_3) fit in the CUBIC cluster.

Using a shape-based Classifier

Shape-based classifier can successfully classify all CCAs in the Linux kernel and BBRv2 with an average accuracy of 96%

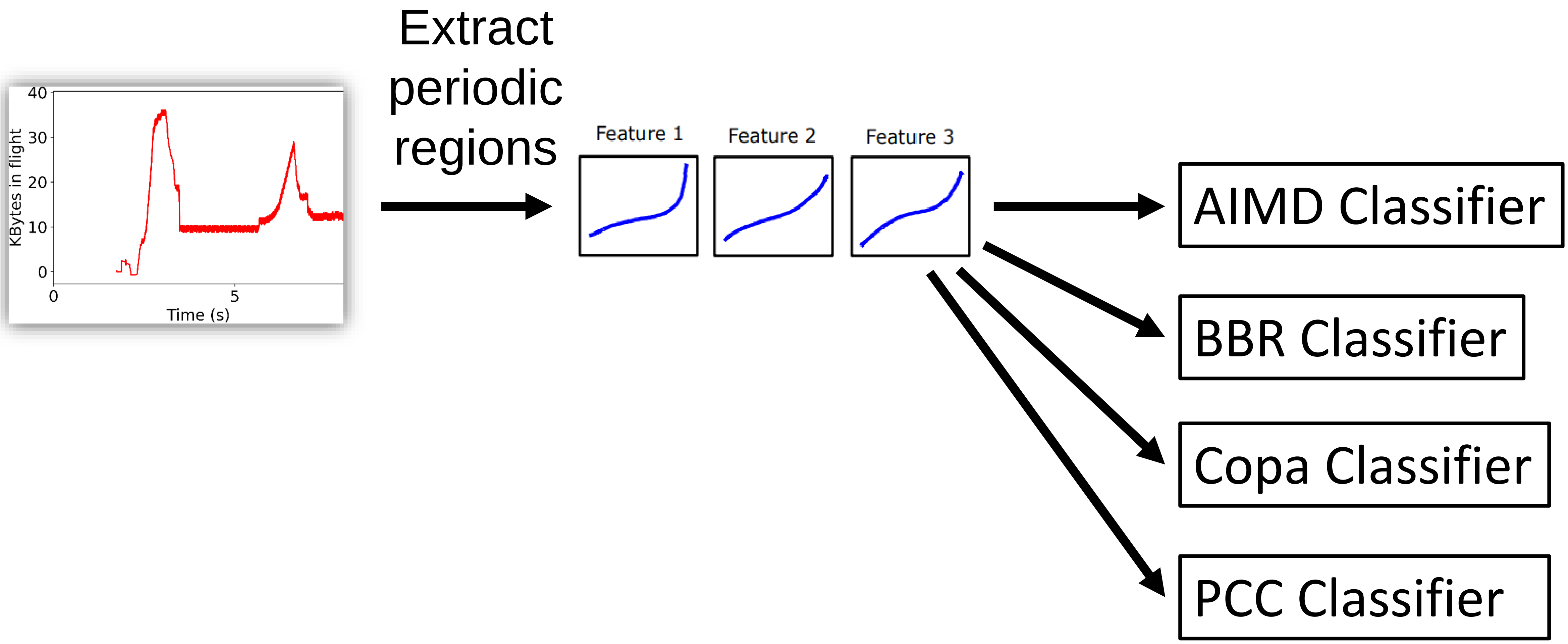
Nebby is demonstrably extensible, with support for CCAs like Copa and PCC. Can detect unknown CCAs deployed by popular websites.



Why is a shape-based classifier good enough?

All “reasonable” CCAs have a *congestion avoidance* phase

Achieving Extensibility



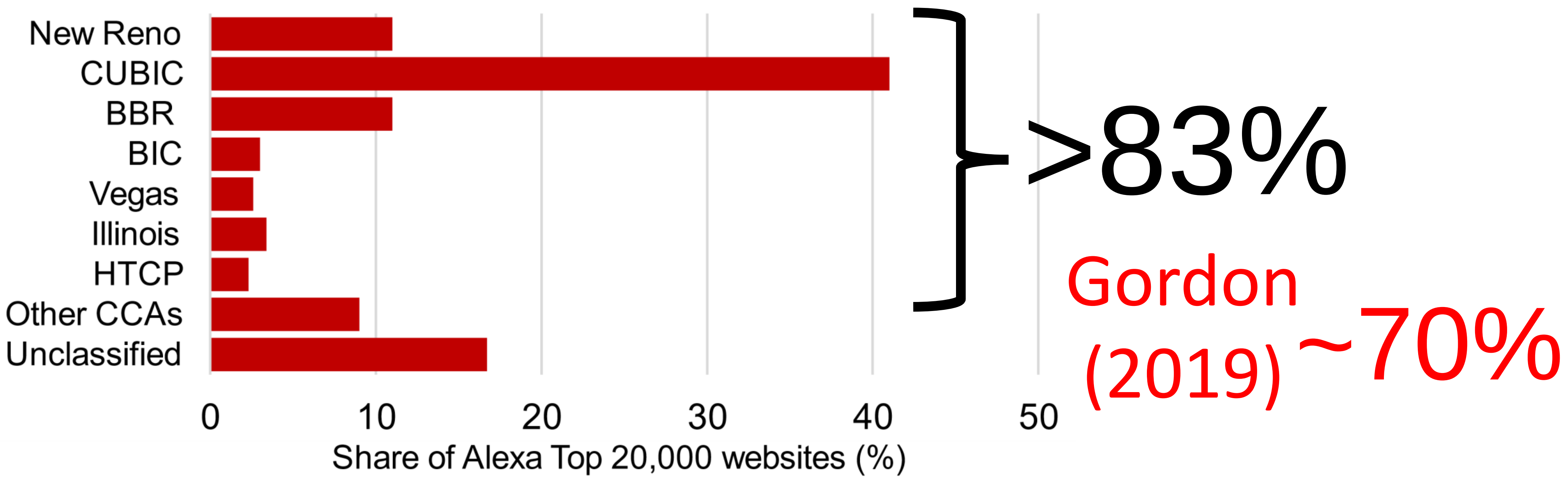
Internet Measurement Results

We made our measurements from `aws` instances in Ohio, Paris, Mumbai, Singapore, and Sao Paulo

We measured the Alexa Top 20,000 websites over a single TCP connection (`wget`) and a single QUIC connection (`quiche`)

We also measured a selection of websites that stream video, audio, and other dynamic content via a chromium web browser.

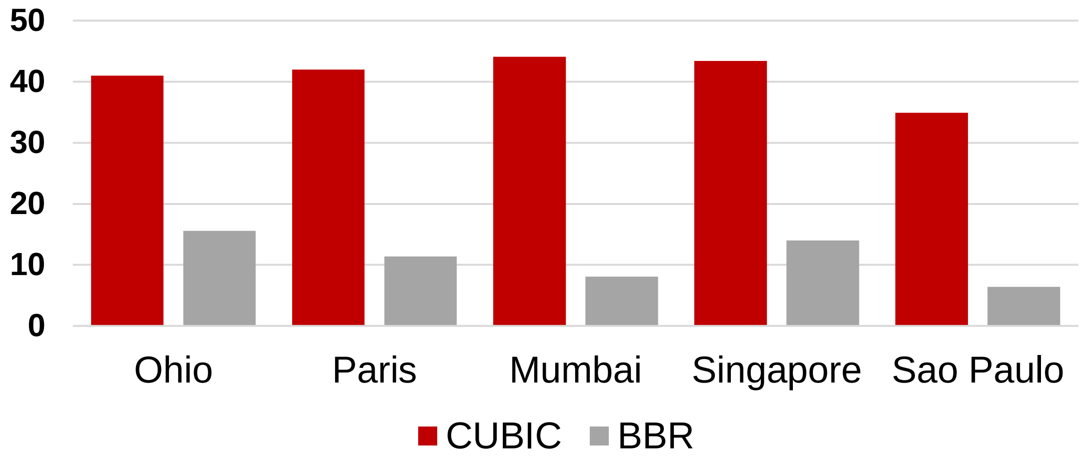
Internet Measurement Results

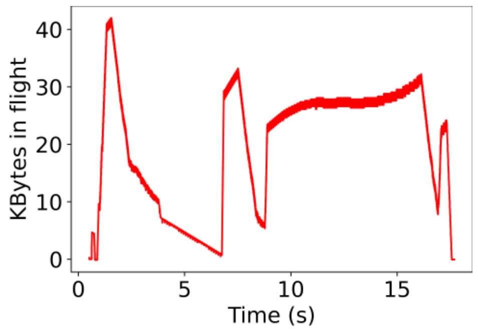


5 key findings

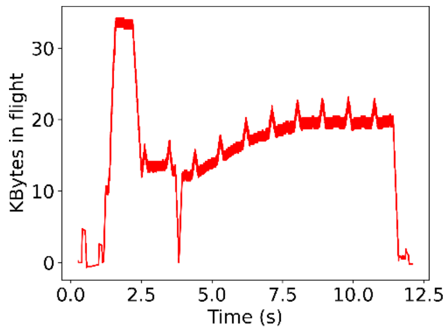
…and many more nuggets covered in the paper

#1 CCA Deployment differs by region





Mumbai



Ohio

While CUBIC and BBR remain the two most dominant CCAs on the Internet, their deployment differs by region

Some websites deploy different CCAs in different regions. For example, `amazon.in` uses CUBIC in Mumbai and BBRv1 in Ohio

#2 Slow Migration to BBRv2

Since the last measurement study, Google has proposed BBRv2, a more fairness-conscious alternative to BBRv1

However, despite this, about 98% of websites that deployed BBRv1 in 2019 have either stuck to BBRv1 or switched to CUBIC

Most websites that deploy BBRv2 are new adopters of BBR

#3 QUIC still has a long way to go

We saw a surprisingly small number of websites in the Alexa Top 20,000 websites respond to QUIC

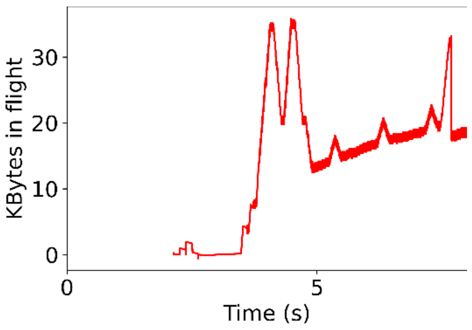
About only 8% of measured websites deployed QUIC

CUBIC and BBR were equally popular amongst websites deploying QUIC

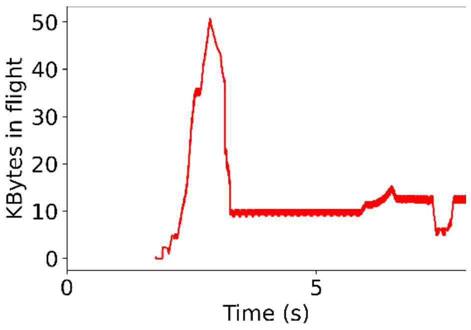
#4 Unknown CCAs on the Internet

Nebby found about 24% of websites deploy CCAs that did not resemble any CCAs in the Linux kernel or BBRv2

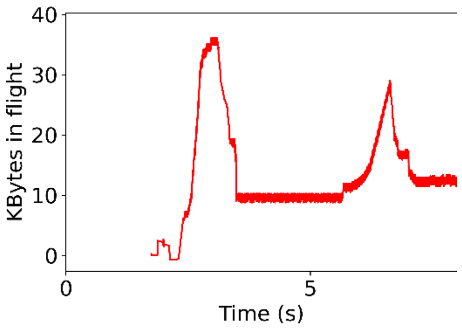
About 200 of these websites were Google domains(!)



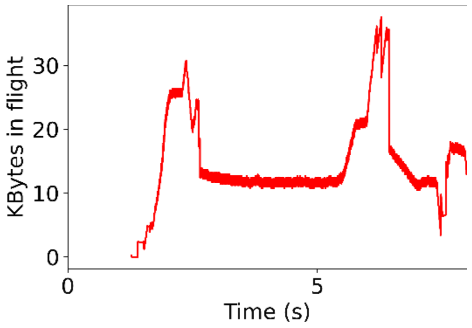
BBRv1



BBRv2



`google.com`
(June 2023)

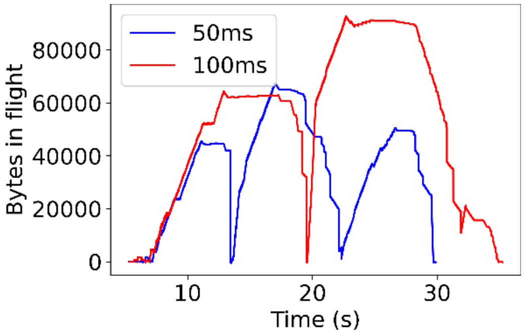


BBRv3
(released Aug 2023)

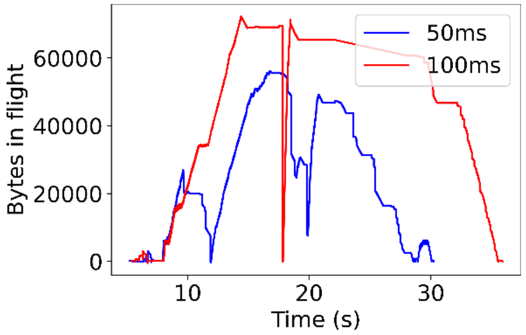
#4 Unknown CCAs on the Internet

Nebby found about 24% of websites deploy CCAs that did not resemble any CCAs in the Linux kernel or BBRv2

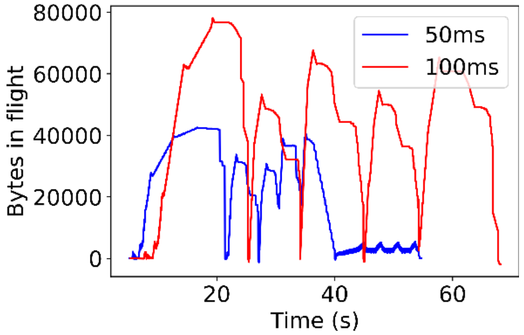
A large proportion of these websites hosted on Akamai were also deploying their own CCAs



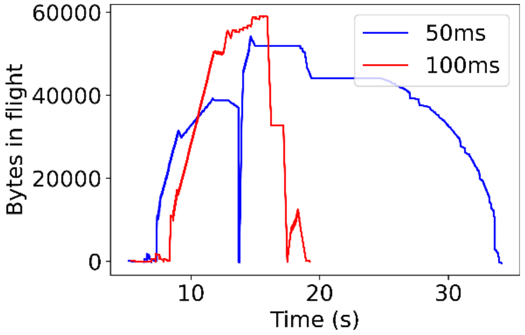
apple.com



hulu.com



pornhub.com

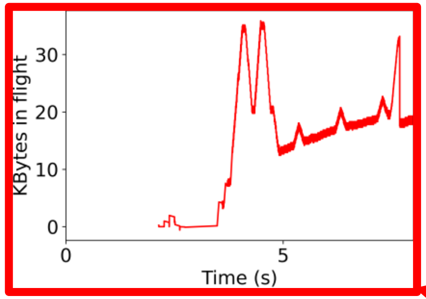


tiktok.com

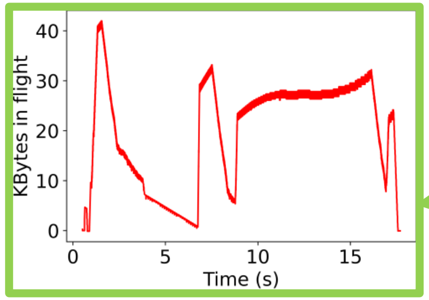
#5 Deployment differs by asset type

Websites like `twitch.tv` and `appletv.com` chose to deploy different CCAs for serving different content

In general, BBR is more popular with websites serving video



BBR



CUBIC

Nebby’s Limitations

The current weakest link is the classifier – even though it is modular, it requires a new module for new CCAs

The classifier is also only as good as the selected network profile. However, there is scope for generating specific network profiles for new CCAs

Evolution of the Internet Congestion Control Landscape

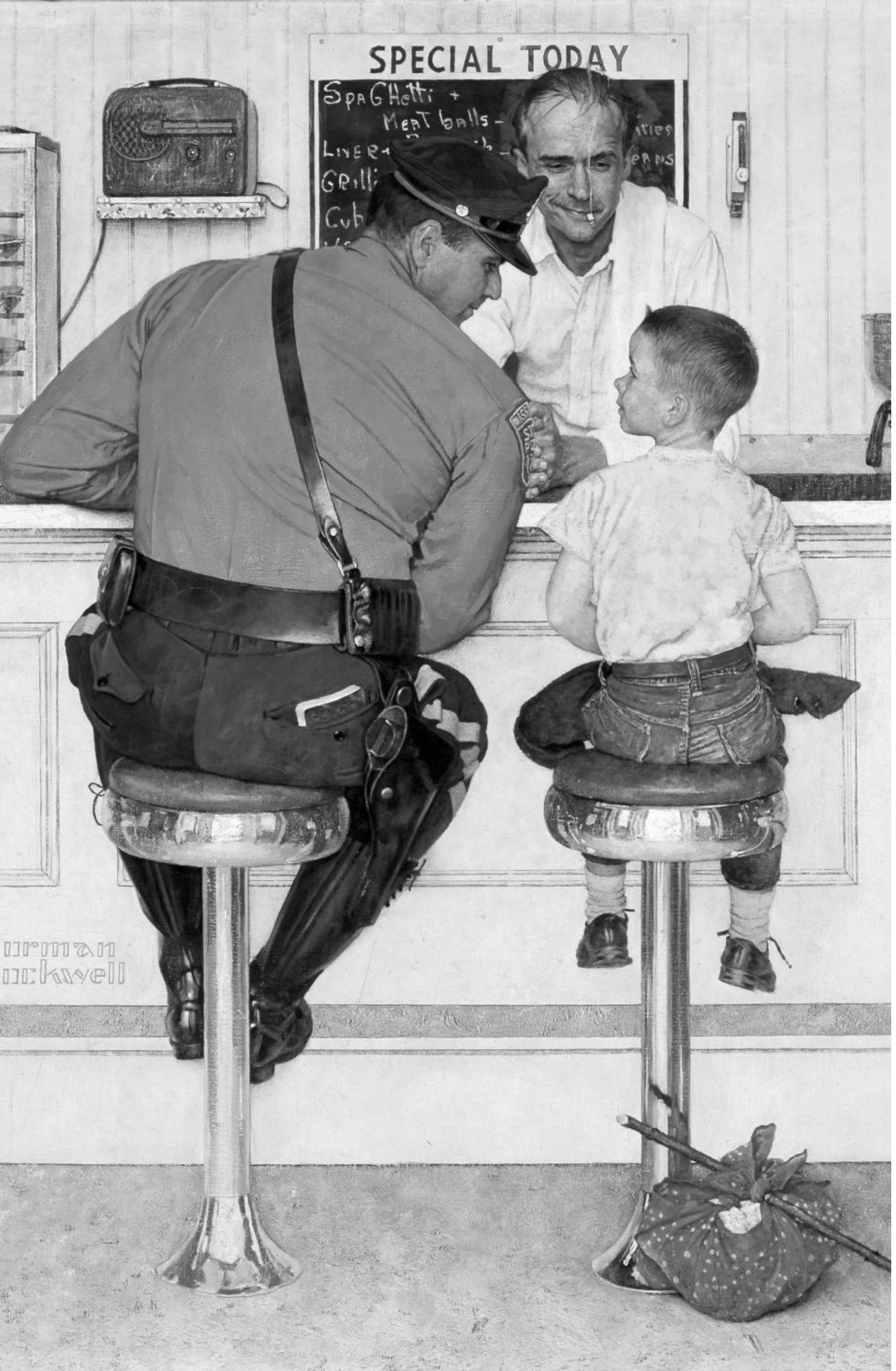
Class	CCA	2001 [54]	2004 [41]	2011 [63]	2019 [50]	2023, *Nebby*
Loss-based AIMD	New Reno [55]	35% (1,571)	25% (21,266)		0.8% (160)	11.1% (11,157)
	Reno [40]	21% (945)	5% (4,115)	12.5% (623)	-	-
	Tahoe	26% (1,211)	3% (2,164)		-	-
Loss-based MIMD	CUBIC [34]			22.3% (1,115)	30.7% (6,139)	41% (41,085)
	BIC [62]	-	-	10.6% (531)	0.9% (181)	3% (2,985)
	HSTCP [28]			7.4% (369)	R	0% (0)
	Scalable [43]			1.4% (69)	0.2% (39)	0% (24)
Delay-based AIMD	Vegas [19]	-	-	1.2% (58)	2.8% (564)	2.6% (2,637)
	Westwood [23]			2% (104)	0% (0)	0.4% (371)
Delay-based MIMD	CTCP [59]			6.7% (334)	5.7% (1,148)	C
	Illinois [45]			0.6% (28)		3.4% (3,380)
	Veno [29]	-	-	0.9% (45)	V	0.6% (578)
	YeAH [18]			1.4% (72)	5.8% (1,162)	0.4% (388)
	HTCP [44]			0.4% (18)	2.8% (560)	2.3% (2,259)
Rate-based	BBRv1 [21]				17.8% (3,550)	10% (9,985)
	BBR G1.1 [50]	-	-	-	0.8% (167)	-
	BBRv2 [22]				-	1.1% (1,151)
	BBRv3				-	0.2% (204)
Unclassified		17.3% (792)	53% (44,950)	4% (198)	12.2% (2,432)	16.7% (16,733)
AkamaiCC		-	-	-	5.5% (1,103)	7.2% (7,117)
Short Flows		-	-	26% (1,300)	7.5% (1,493)	-
Unresponsive		0.7% (30)	14% (11,529)	-	6.5% (1,302)	-
Abnormal SS*		-	-	2.9% (144)	-	-
Total hosts		100% (4,550)	100% (84,394)	100% (5,000)	100% (10,000)	100% (100,000)

R Classified together with New Reno

V Classified together with Vegas

C CTCP has been deprecated in Windows

* Websites identified by CAAI as having Abnormal Slow Starts



In Summary…

We introduce a fresh methodology for studying and identifying CCAs on the Internet for TCP, QUIC, and live clients.

We show that while BBR’s adoption has slowed down, most bandwidth intensive applications still opt for BBR.

BBRv3 and AkamaiCC are case studies in using Nebby to catch the deployment of unknown congestion control algorithms.

Nebby is open source and available on GitHub!

Lessons from a 5 year study of the Internet's CCA landscape

The mix of CCAs on the Internet is more heterogeneous than it has ever been in the past

The heterogeneity created by the deployment of BBR is likely to remain given its diminishing performance benefits over CUBIC.

We can expect the deployment of modified and new CCAs on the Internet thanks to QUIC

Lessons from a 5 year study of the Internet's CCA landscape

The **mix of CCAs** on the Internet is more **heterogeneous** than it has ever been in the past

But what about **fairness**, **stability**, and **coexistence**?

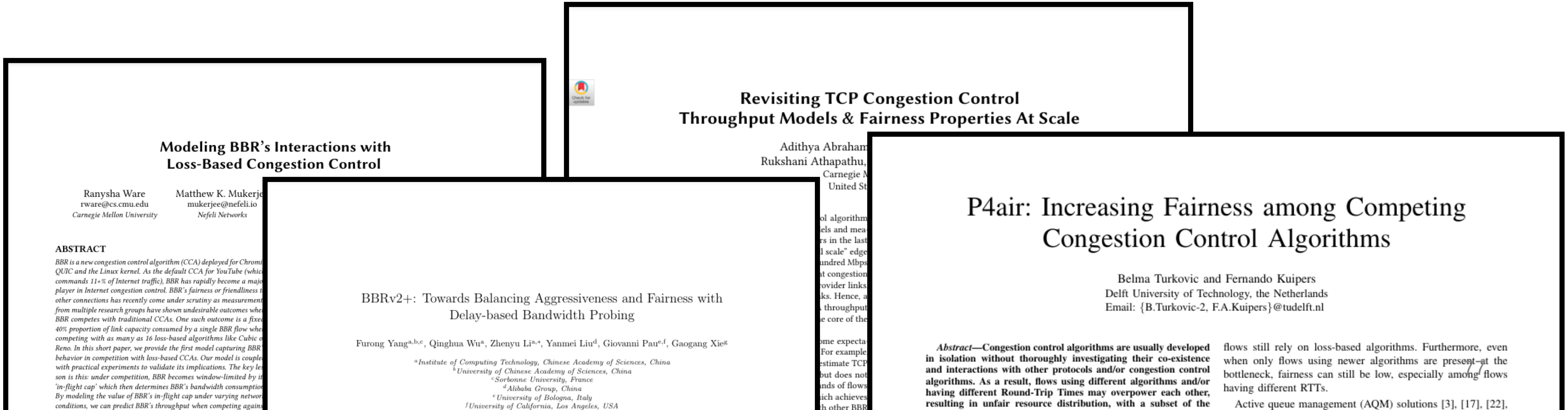
The heterogeneity created by the deployment of BBR is likely to remain given its **diminishing performance** benefits over CUBIC.

We can expect the **deployment of modified** and **new CCAs** on the Internet thanks to **QUIC**

The knee-jerk reaction

Enforce bandwidth fairness!

Through in-network solutions, modifications (and harsh critique) to existing offenders



Redefining *Coexistence*

This heterogeneity in the Internet's congestion control landscape has not been created in a vacuum.

People develop and chose to deploy different CCAs because they want different things.

Since the heterogeneity on the Internet is app driven, so should our ideas of fairness and coexistence.

Our response to this heterogeneity needs to be two-fold

#1 We need to come up with in-network mechanisms that allow flows with competing needs to co-exist

In-network isolation?

#2 We need reactive end-host congestion control algorithms

CCAs need to react not just to the network, but also who they compete with