TCP Part 3: Performance, Fairness, & Modern Congestion Controllers

Slides by: Ranysha Ware

Lecturer: Prof Sherry

SNAPLAB

So far, you have learned about flow control, congestion control, and how TCP Reno works.

Turn to a partner and discuss:

1. What is the difference between flow control and congestion control?
2. What does ACK clocking mean?
3. You are sending packets over a network where the bottleneck link is 50Mbps, the round trip time is 150ms, and the queue at the bottleneck link can store up to 2MB of data. How large is the largest your window can be before you will see packet loss?

Today, you will learn:

- What's good about TCP Reno?
- What's bad about TCP Reno?
- What congestion control algorithms are deployed in the Internet today?
- Is the Internet fair?

What’s good about TCP Reno?

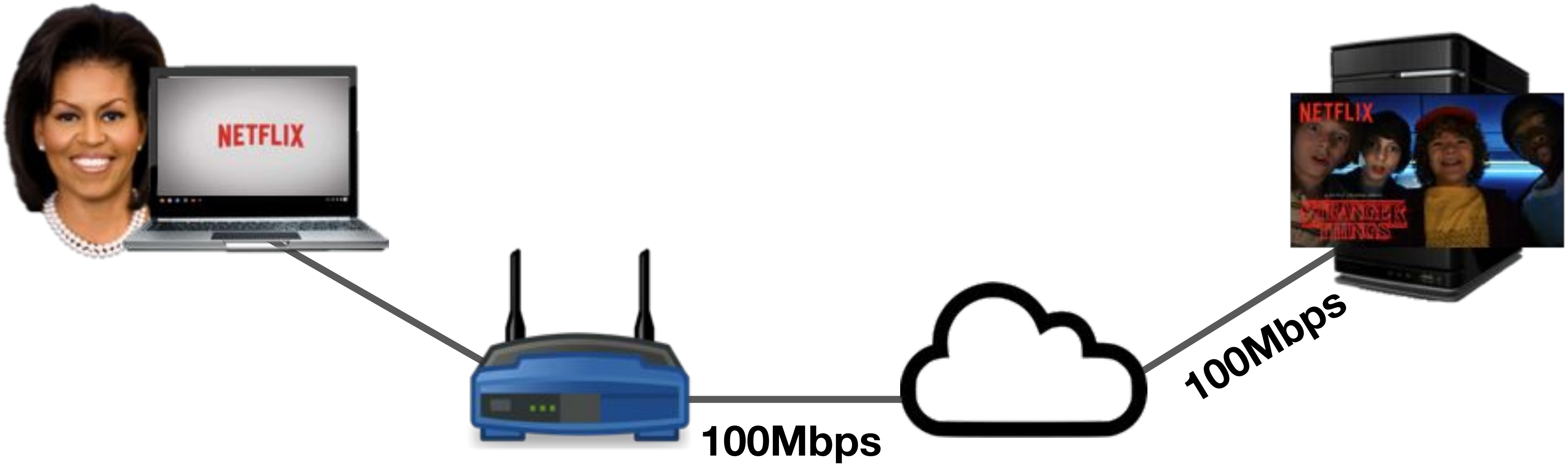
In 1989 paper, Chiu and Jain define 4 properties of a good CCA.

- Efficiency
- Fairness
- Convergence
- Distributness

In 1989 paper, Chiu and Jain define 4 properties of a good CCA.

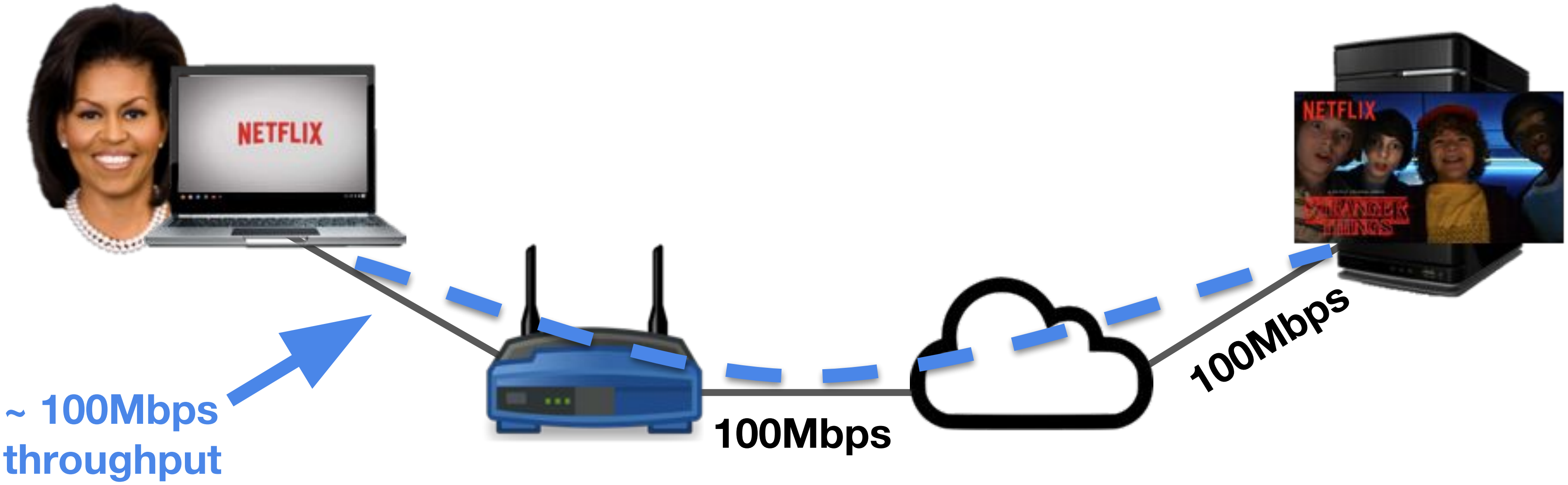
- Efficiency
- Fairness
- Convergence
- Distributness

Let’s assume: Applications always have data to send and are never limited by flow control. Only thing that affects performance is what the CCA is doing!

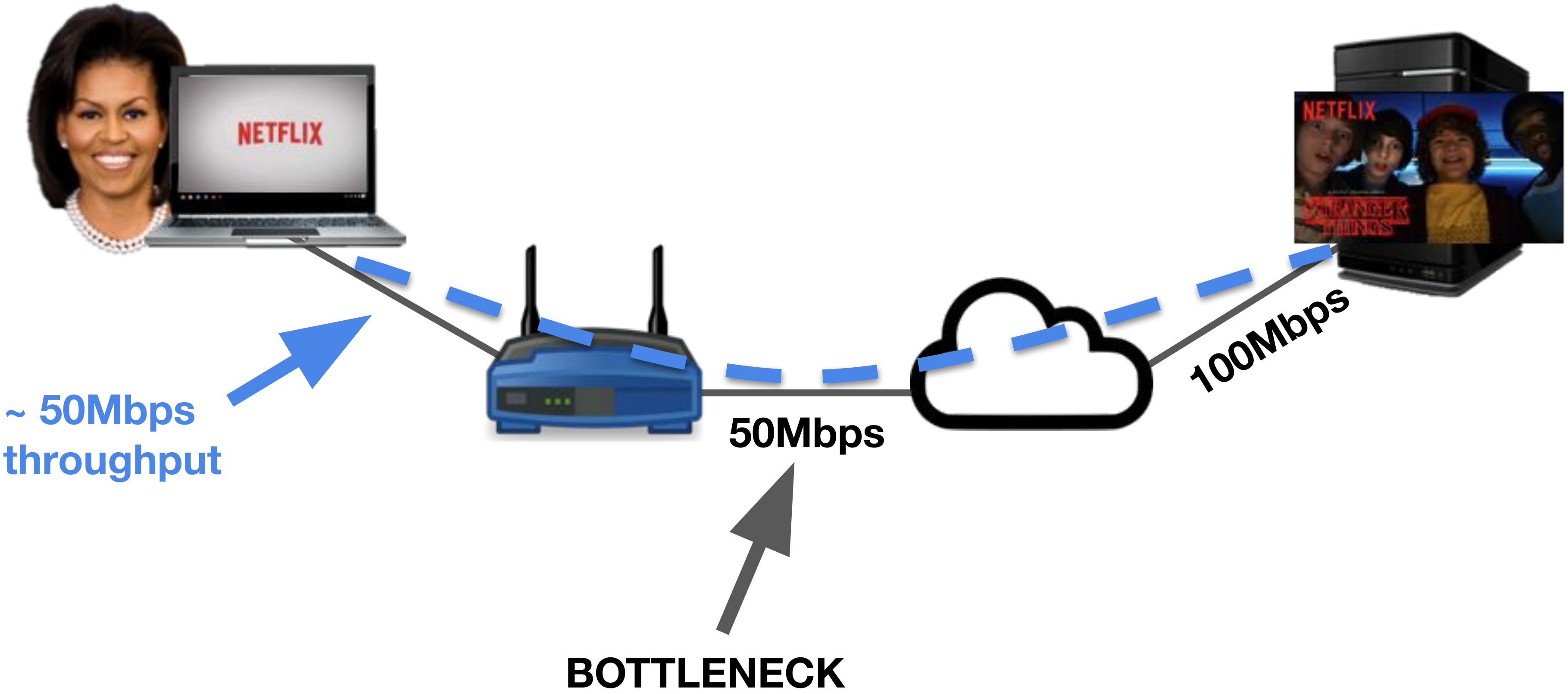


100Mbps

100Mbps



Efficiency: A good CCA should utilize available bandwidth without overloading the network.

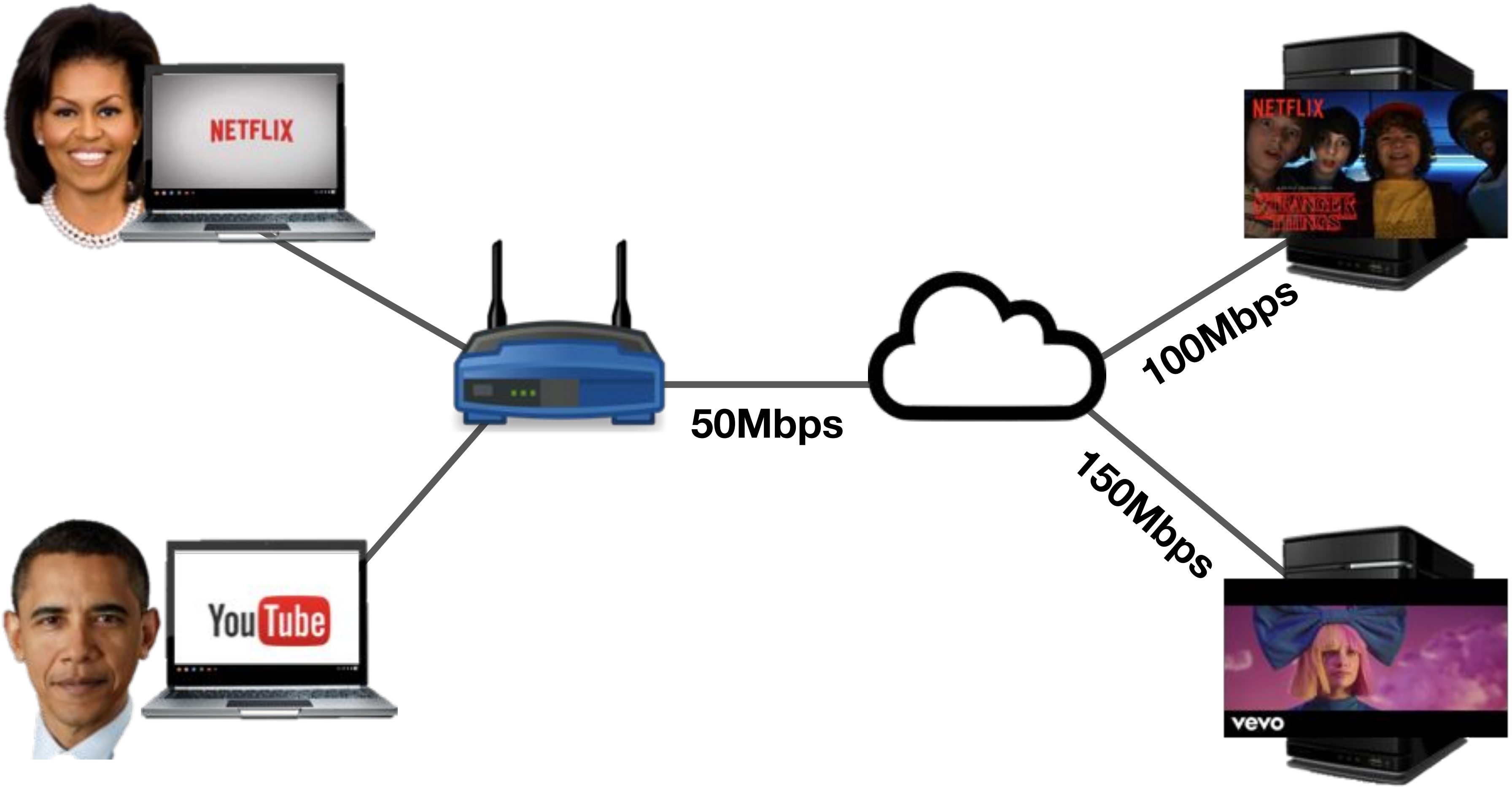


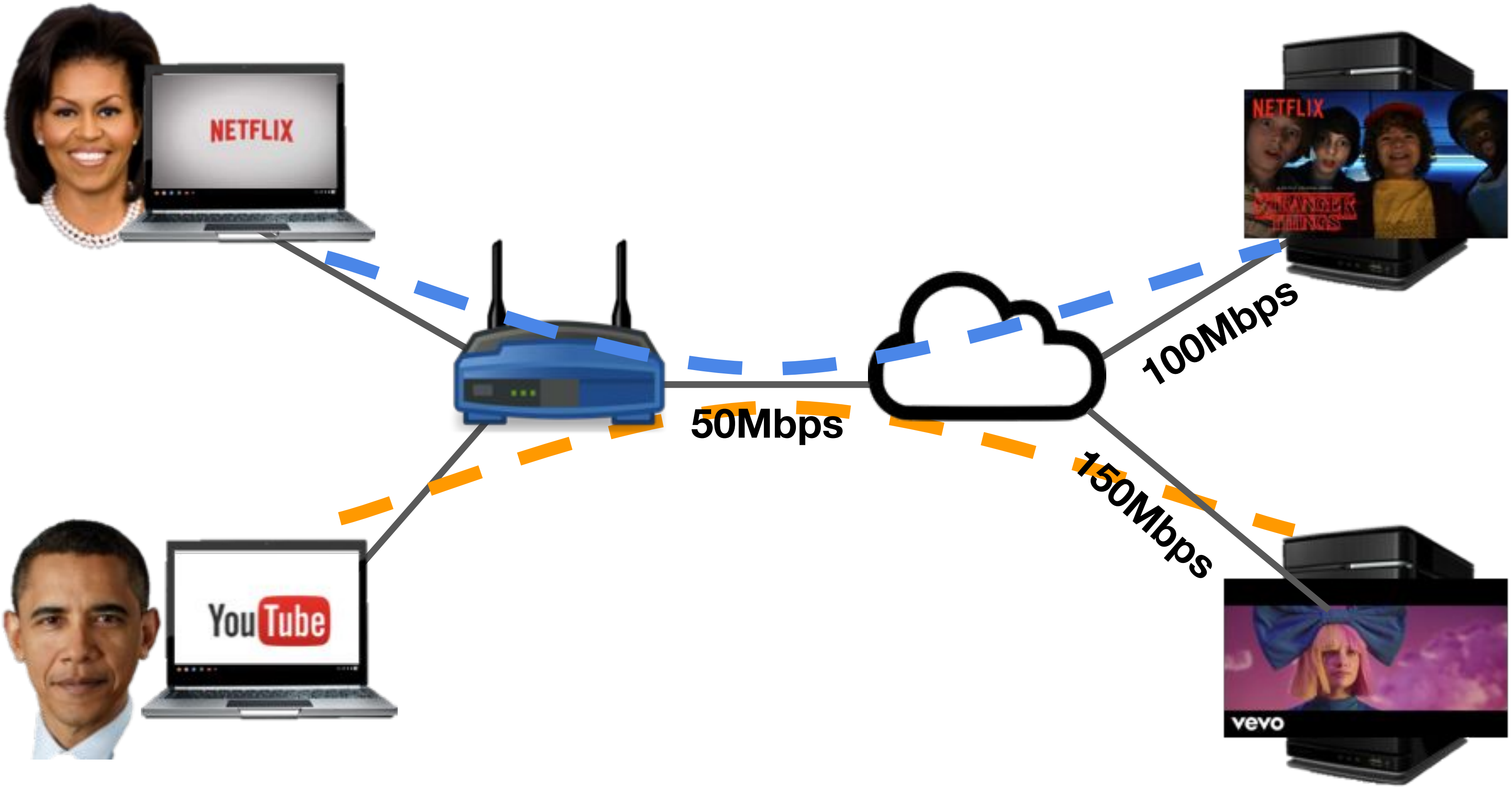
~ 50Mbps throughput

50Mbps

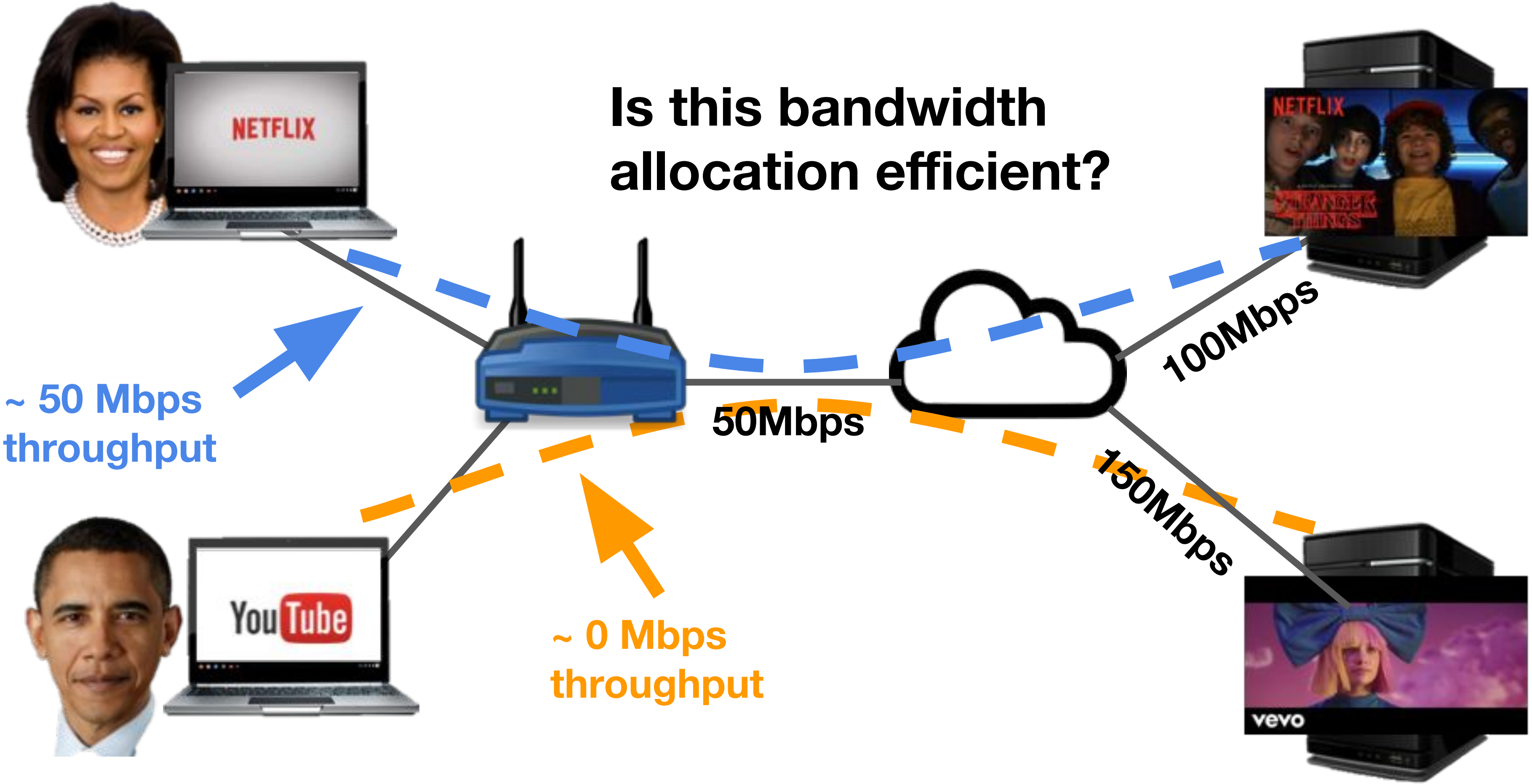
BOTTLENECK

100Mbps


NETFLIX
You Tube
50Mbps
NETFLIX
STRANGER THINGS
100Mbps
150Mbps
vevo


NETFLIX
NETFLIX
STRANGER THINGS
You Tube
vevo
50Mbps
100Mbps
150Mbps

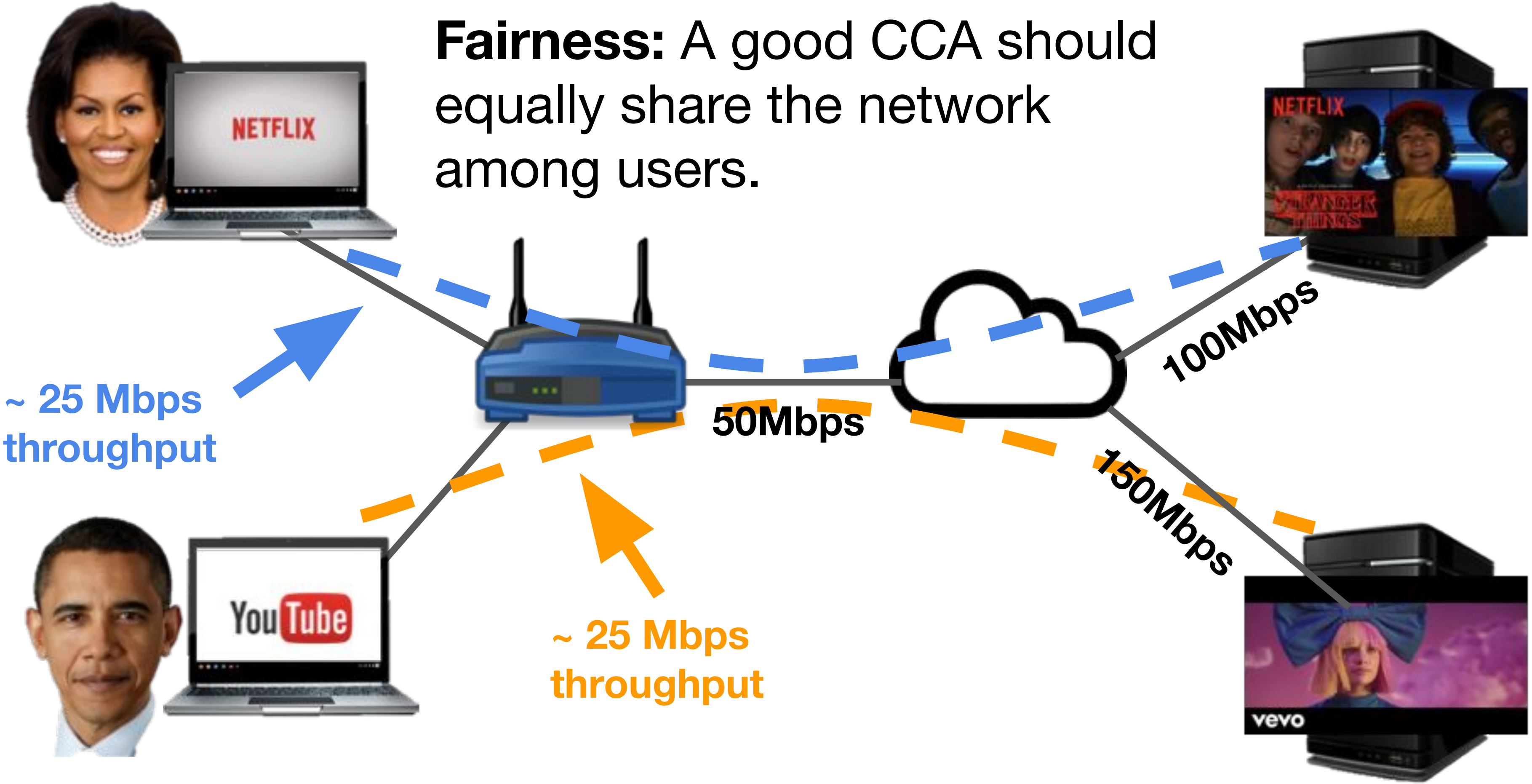
Is this bandwidth allocation efficient?



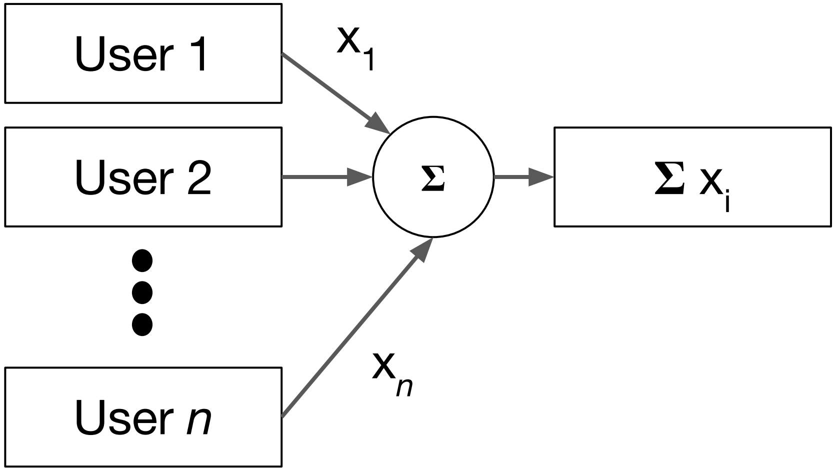
~ 50 Mbps throughput

~ 0 Mbps throughput

Fairness: A good CCA should equally share the network among users.



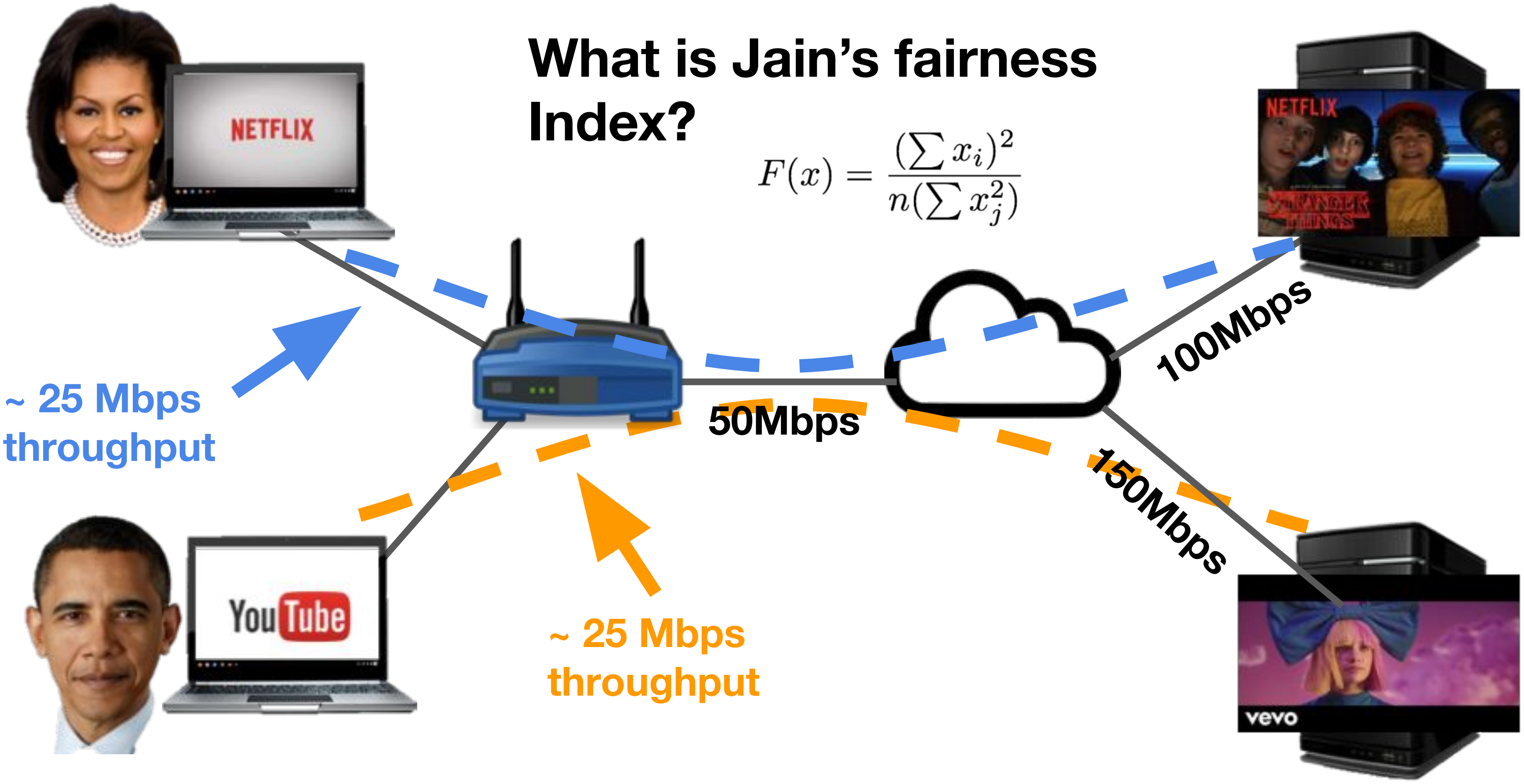
Jain’s Fairness Index is used to quantify fairness. 1 means the allocation is equal (fair) and 0 means the allocation is unfair.



$$F(x) = \frac{(\sum x_i)^2}{n(\sum x_j^2)}$$

What is Jain’s fairness Index?

$$F(x) = \frac{(\sum x_i)^2}{n(\sum x_j^2)}$$



~ 25 Mbps throughput

~ 25 Mbps throughput

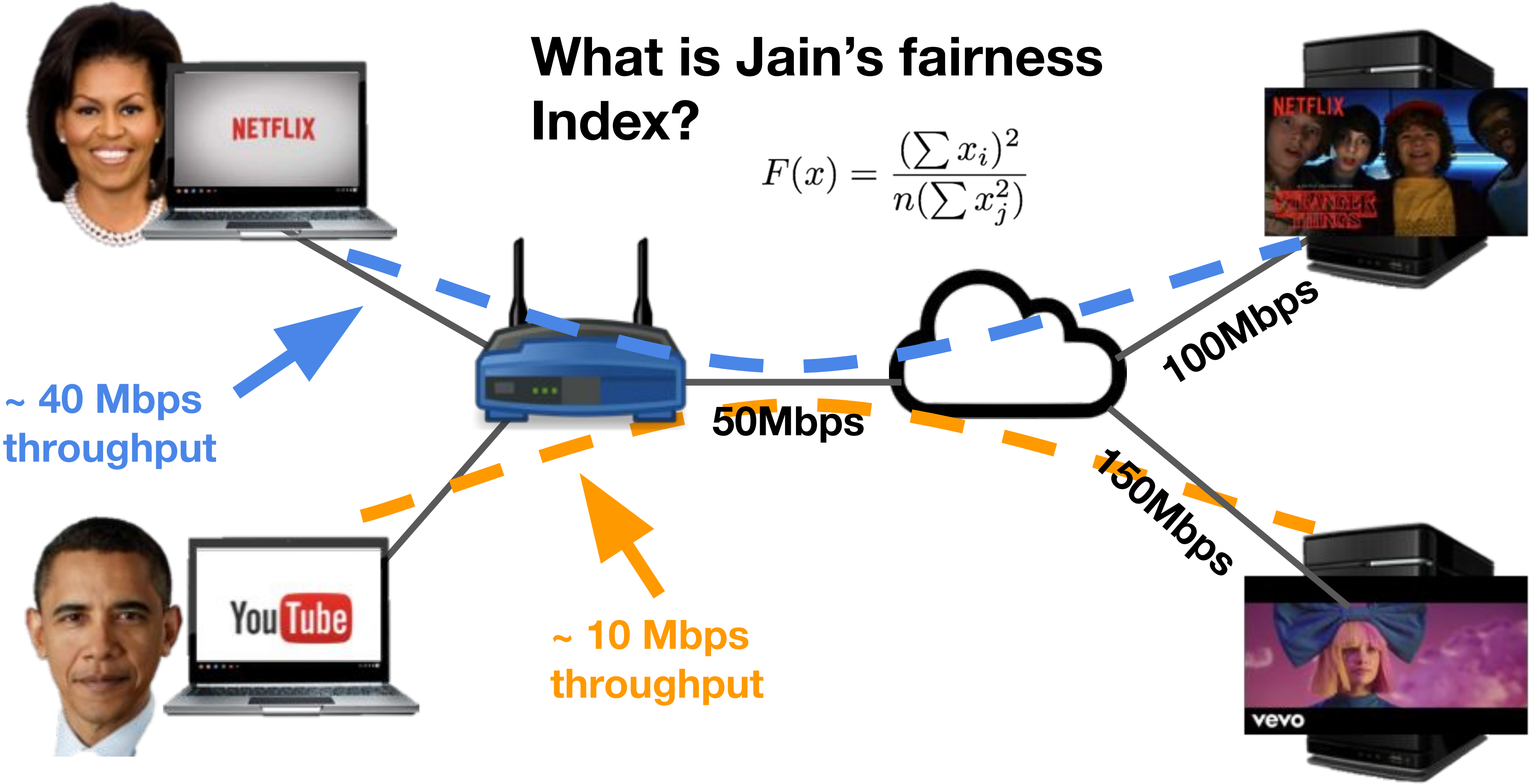
50Mbps

100Mbps

150Mbps

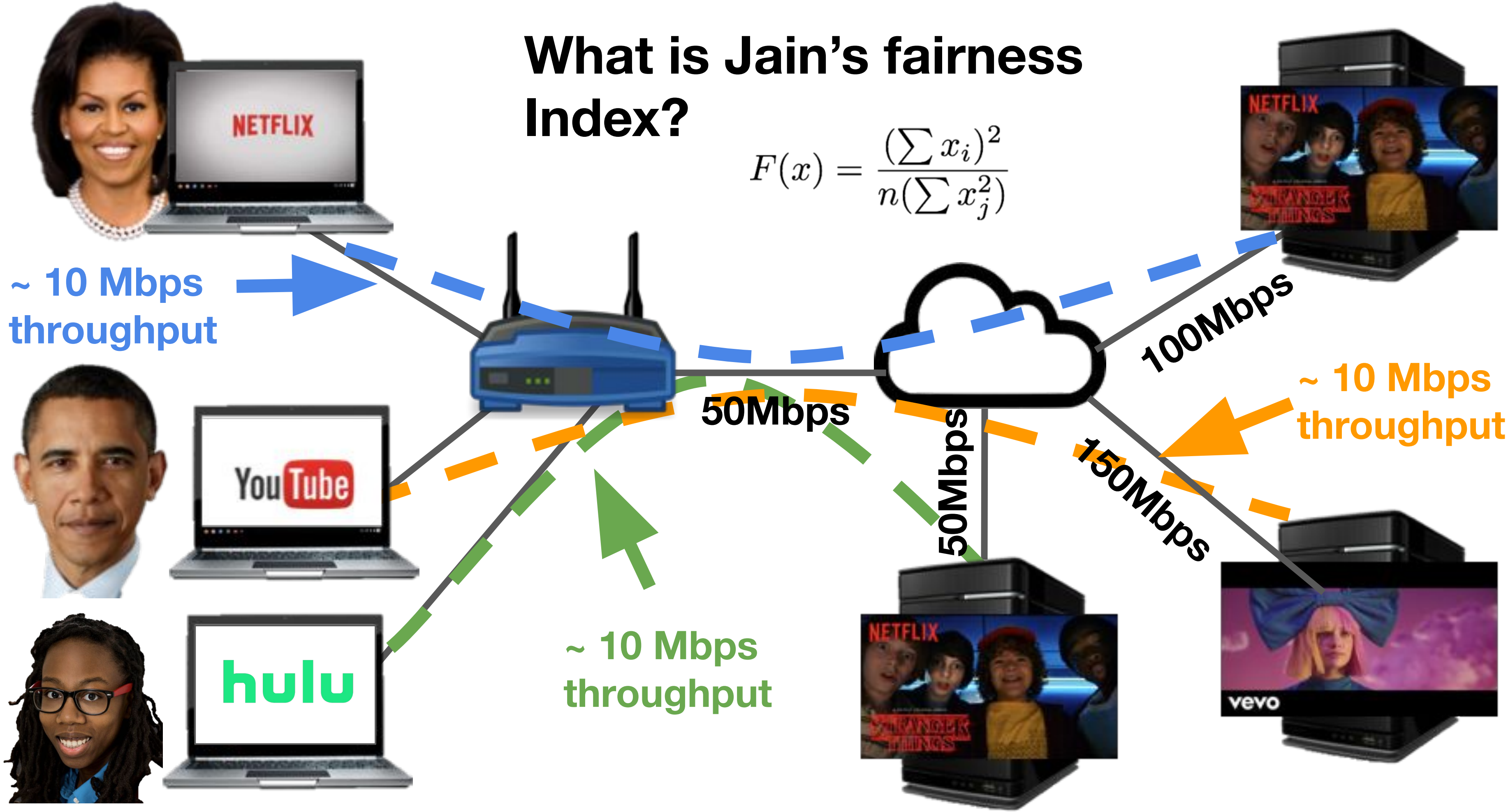
What is Jain's fairness Index?

$$F(x) = \frac{(\sum x_i)^2}{n(\sum x_j^2)}$$



What is Jain's fairness Index?

$$F(x) = \frac{(\sum x_i)^2}{n(\sum x_j^2)}$$



~ 10 Mbps throughput

50Mbps

100Mbps

50Mbps

150Mbps

~ 10 Mbps throughput

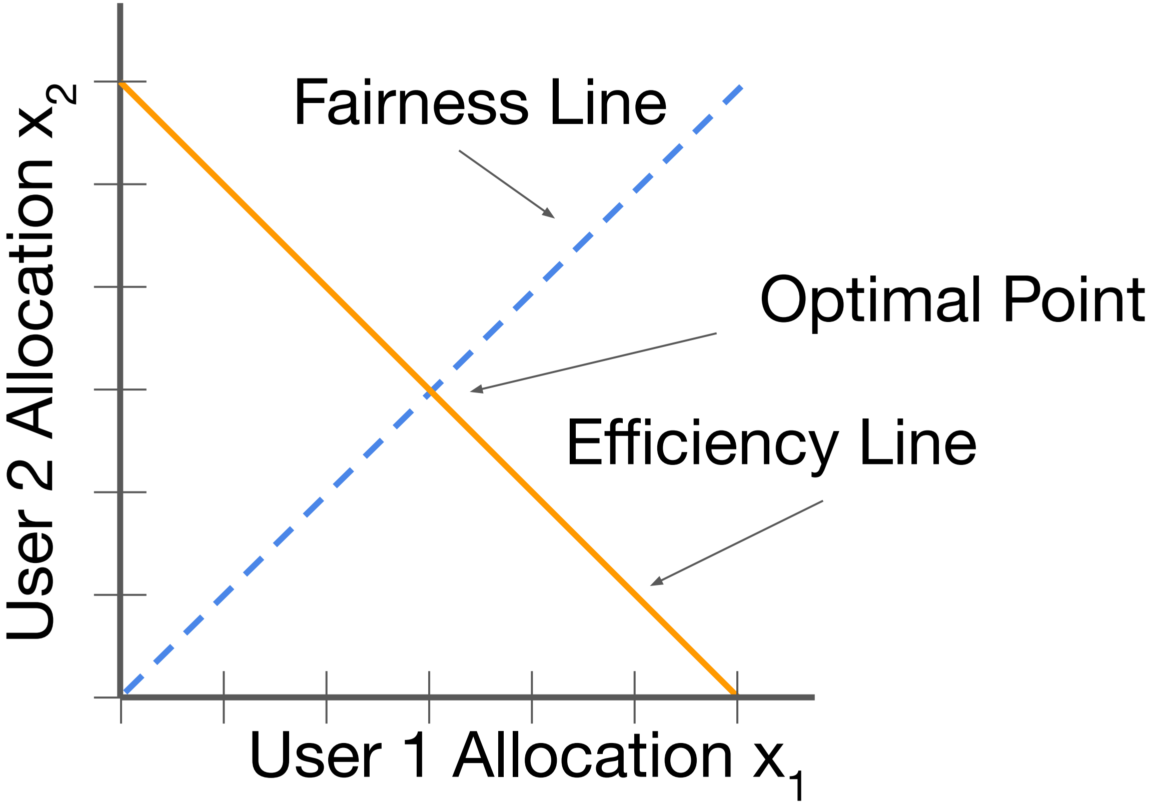
~ 10 Mbps throughput

$$F(x) = \frac{(\sum x_i)^2}{n(\sum x_j^2)}$$

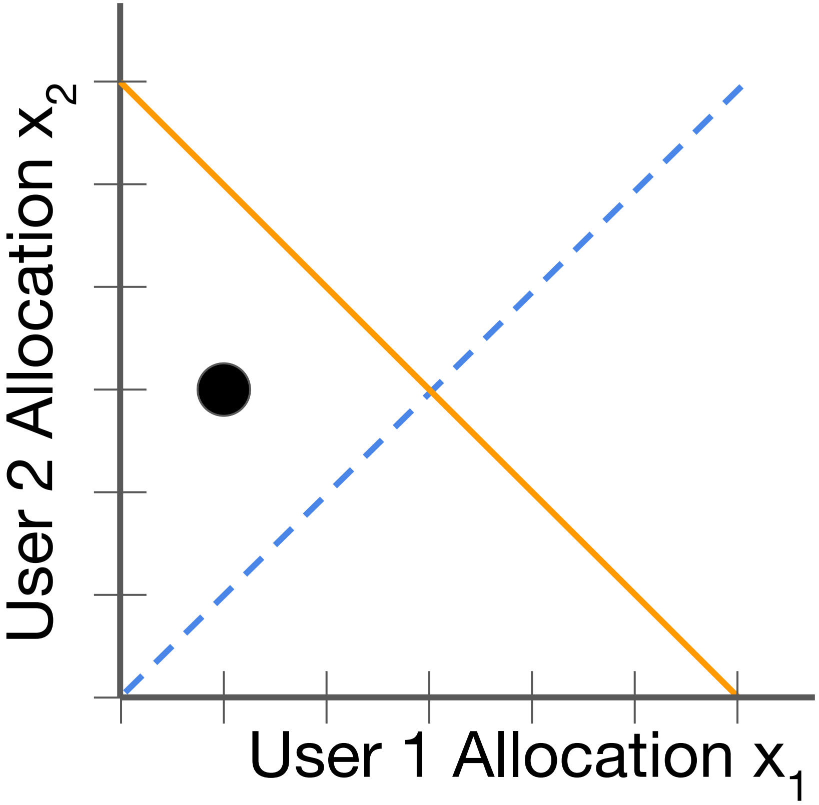
Turn to a partner and discuss:

1. What is Jain’s fairness index when User 1 gets 30mbps, User 2 get 20mbps?
2. Now consider example where when User 3 isn’t around, User 1 and User 2 get 25mbps each, but User 3 comes along and they get allocations 10mbps, 10mbps, and 10mbps respectively. What’s Jain’s fairness index for this allocation?

A good CCA needs to be **fair** and **efficient.**



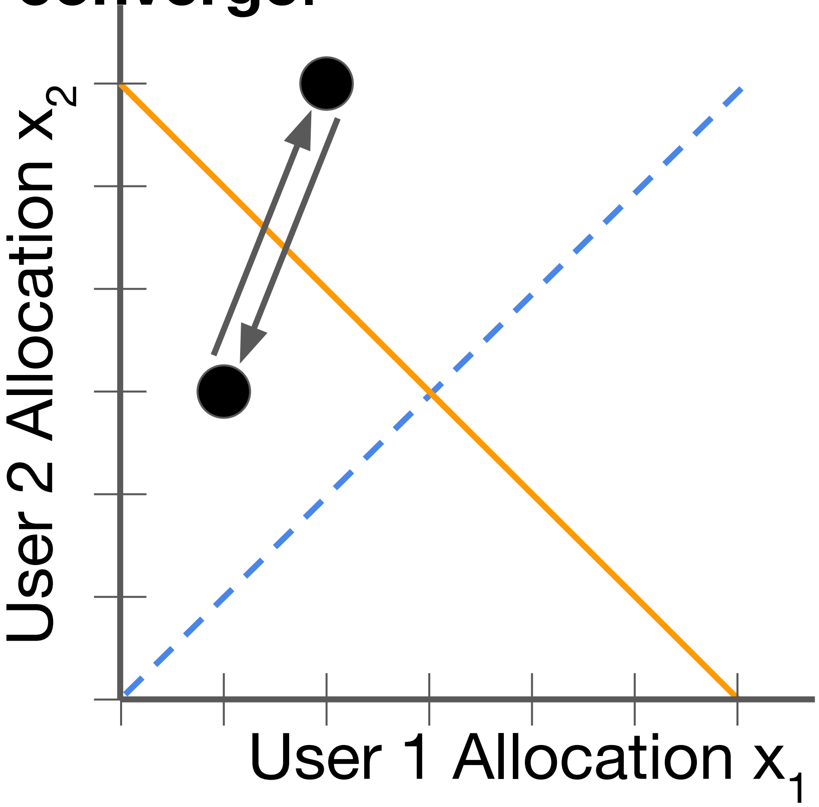
What happens when senders are using MIMD?



Assume: Max capacity is 6. When over capacity user’s experience loss at the same time. (RTT is the same)

time	x_1	x_2
1	1	3
2		

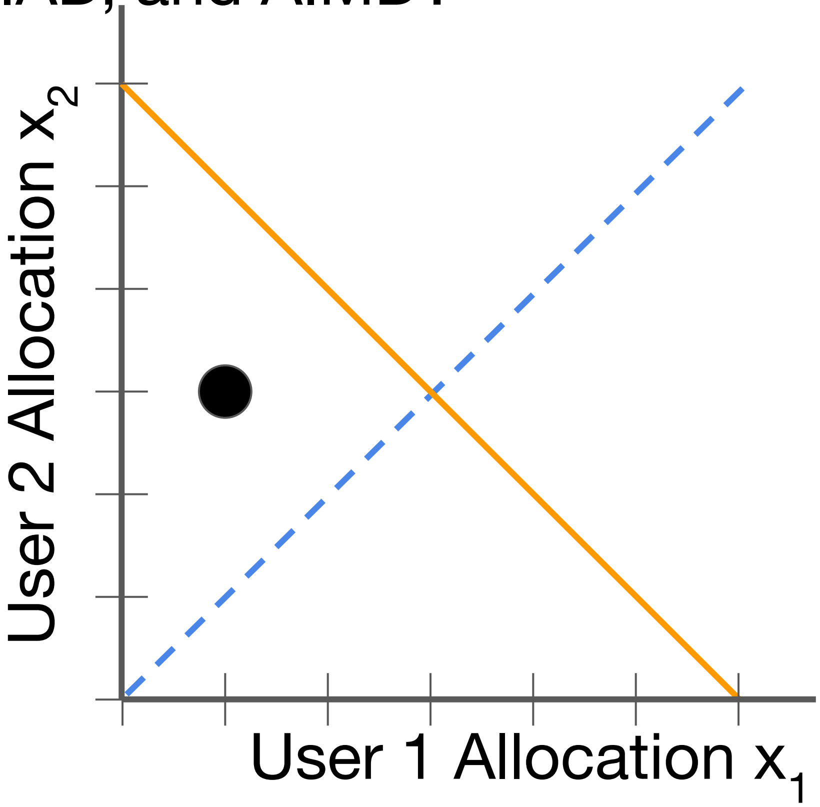
MIMD never converges to optimal point! A good CCA needs to **converge**.



Assume: Max capacity is 6. When over capacity user's experience loss at the same time. (RTT is the same)

time	x_1	x_2
1	1	3
2	2	6
3	1	3
4	2	6

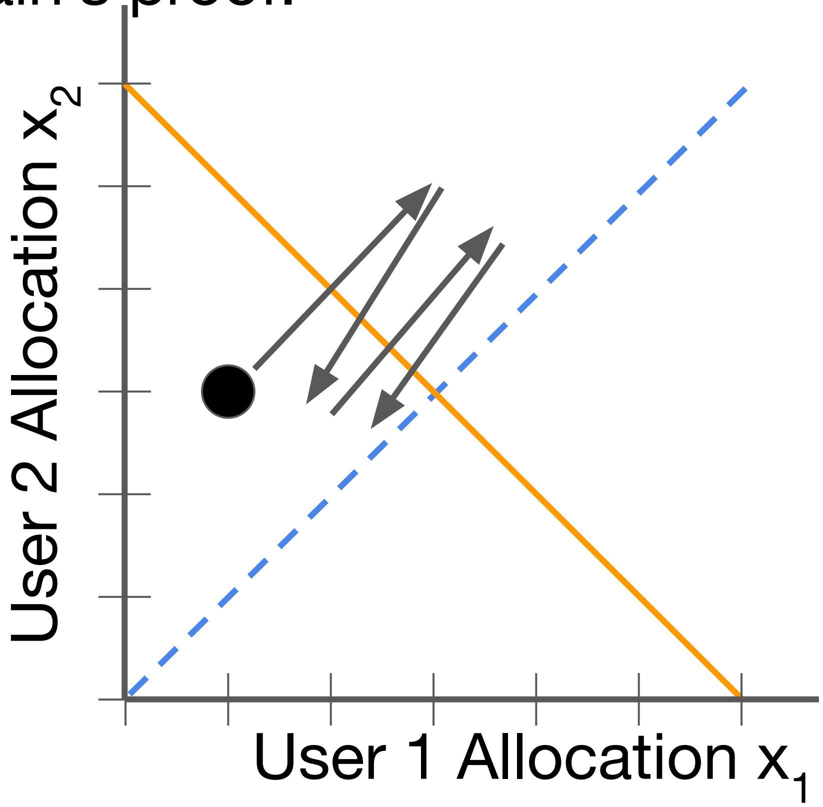
Turn to a partner: What happens when senders use AIAD, MIAD, and AIMD?



Assume: Max capacity is 6. When over capacity user's experience loss at the same time. (RTT is the same).

time	x_1	x_2
1	1	3
...		
10		

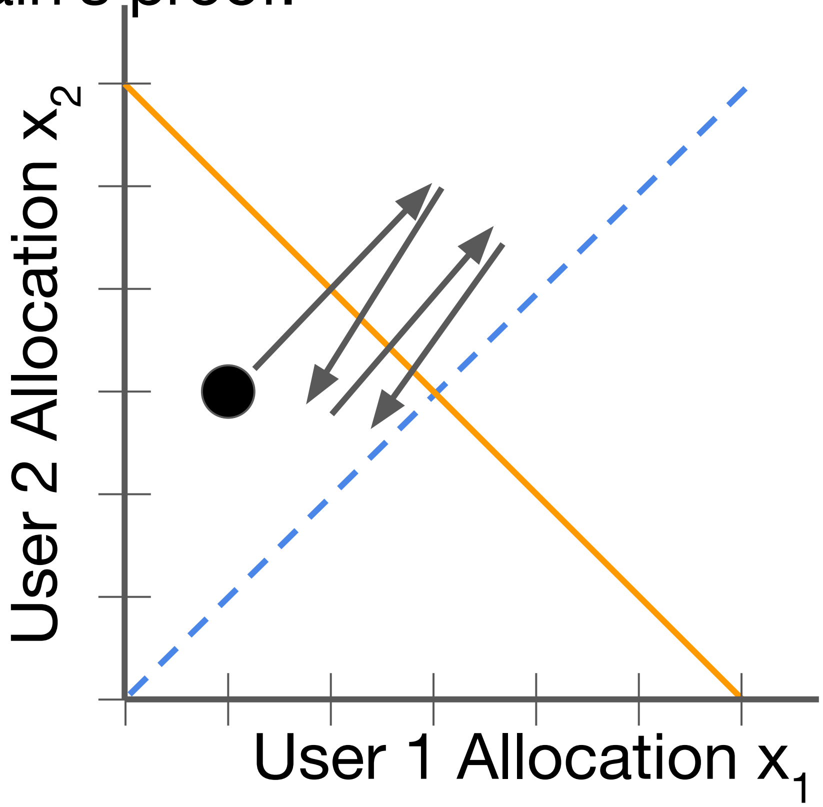
AIMD converges around the optimal point. This is Chiu and Jain's proof!



Assume: Max capacity is 6. When over capacity user's experience loss at the same time. (RTT is the same)

time	x_1	x_2
1	1	3
3	3	5
6	3.5	4.5
9	3.75	4.25

AIMD converges around the optimal point. This is Chiu and Jain's proof!



Assume: Max capacity is 6. When over capacity user's experience loss at the same time. (RTT is the same). Cutoff decimals (1.75=1)

time	x_1	x_2
1	1	3
3	3	5
6	3	4
9	2	3

A CCA is **distributed** if it doesn’t require cooperation between users or the network to operate well.

Why did Chiu and Jain think a good CCA needs to be distributed?

What’s good about TCP Reno?

- It meets the 4 criteria of a good CCA!

In 1989 paper, Chiu and Jain define 4 properties of a good CCA.

- **Efficiency:** TCP Reno can utilize available bandwidth.
- **Fairness:** TCP Reno is fair when competing with itself.
- **Convergence:** TCP Reno converges to an equal and efficient bandwidth allocation among users.
- **Distributness:** TCP Reno is an end-to-end CCA, which doesn’t require cooperation between users or the network to meet other 3 criteria.

What’s bad about TCP Reno?

In today's high speed networks, TCP Reno's additive increase is too slow, and multiplicative decrease is too aggressive.

- TCP RTT unfairness
- TCP throughput model
- TCP in highspeed networks & lossy networks
- TCP & bufferbloat

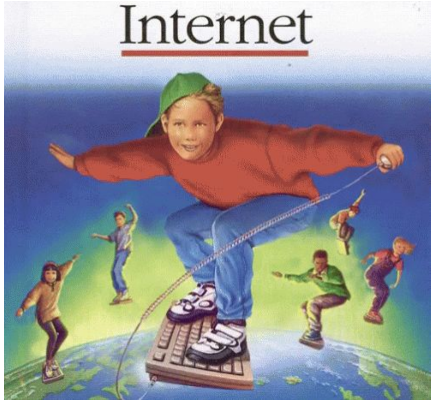
TCP Reno is great. It’s **efficient**, **fair**, **converges**, and is **distributed**! What more could you want!?

Raj Jain

TCP Reno is great. It’s **efficient**, **fair**, **converges**, and is **distributed**! What more could you want!?

Raj Jain

Lies! These things are only true under certain conditions and I’ve evolved since 1989, man!

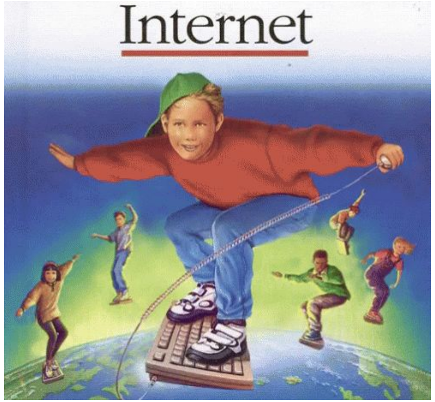


The Internet

TCP Reno is fair when competing with itself. I proved it!

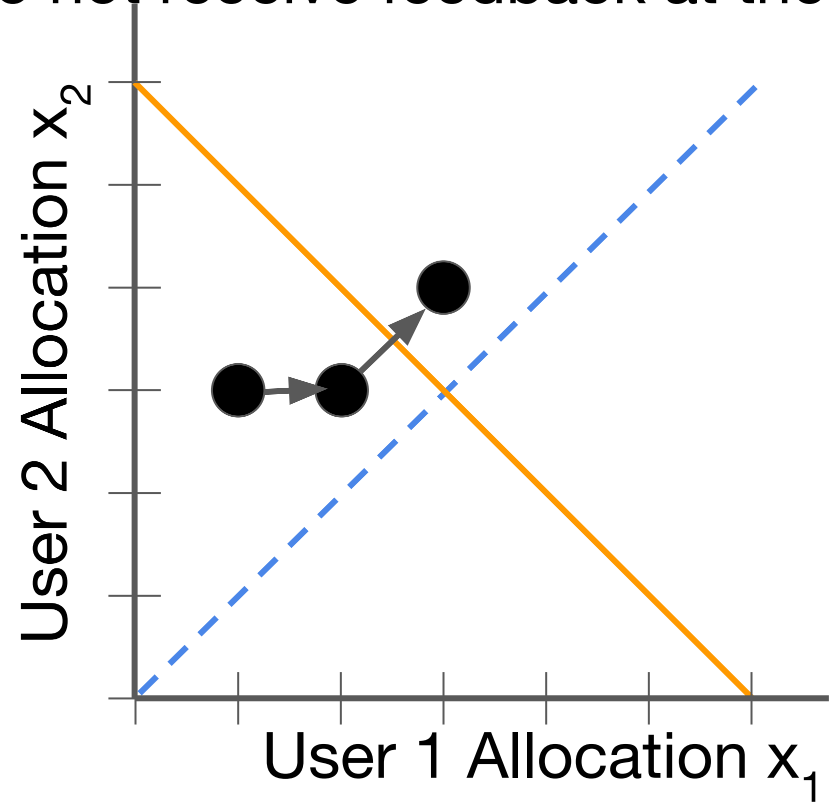
What about when the users don't experience loss at the same time?

Raj Jain



The Internet

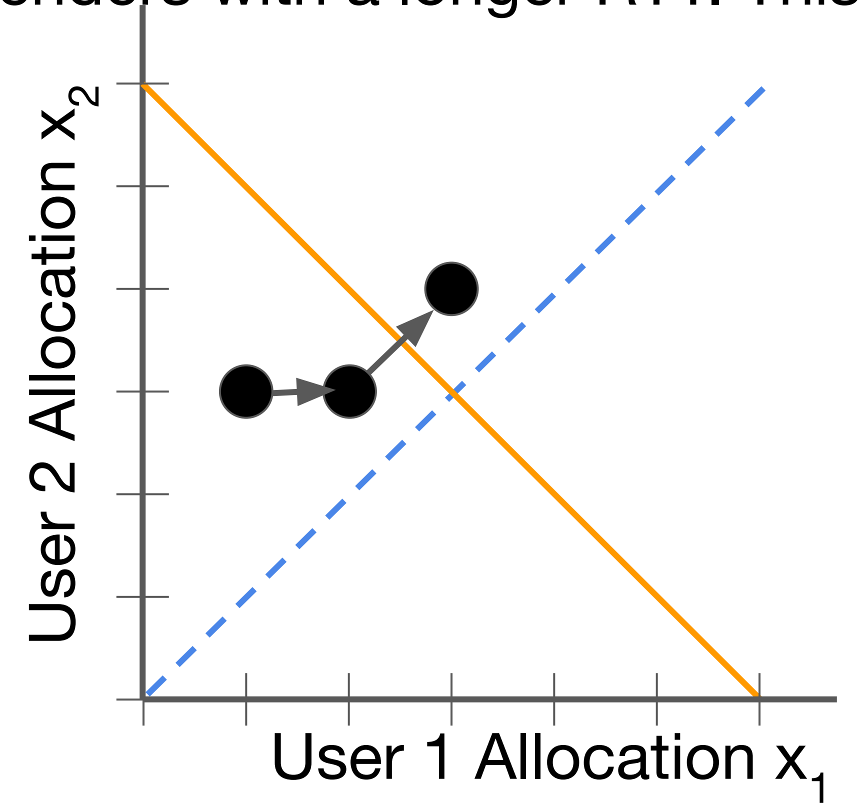
Turn to a partner: What happens when senders use AIMD but do not receive feedback at the same time?



Assume: Max capacity is 6. User x_1 receives updates every time interval, User x_2 updates every 2 time intervals.

time	x_1	x_2
1	1	3
2	2	
3	3	4
...		

AIMD senders with a shorter RTT can update faster than senders with a longer RTT. This is **RTT unfairness**.



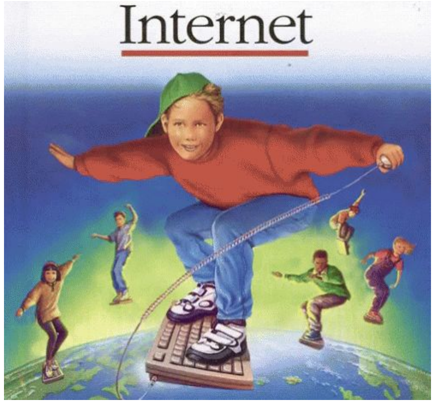
Assume: Max capacity is 6. User x_1 receives updates every time interval, User x_2 updates every 2 time intervals.

time	x_1	x_2
1	1	3
2	2	
3	3	4
...		

Well maybe it is not always fair, but TCP Reno is definitely efficient!

What about when the bandwidth is really large? Or the RTT is really large?

Raj Jain



The Internet

In 1997 paper, Mathis derived a simple model for a TCP flow’s throughput.

$$BW < \left(\frac{MSS}{RTT}\right)\frac{1}{\sqrt{p}}$$

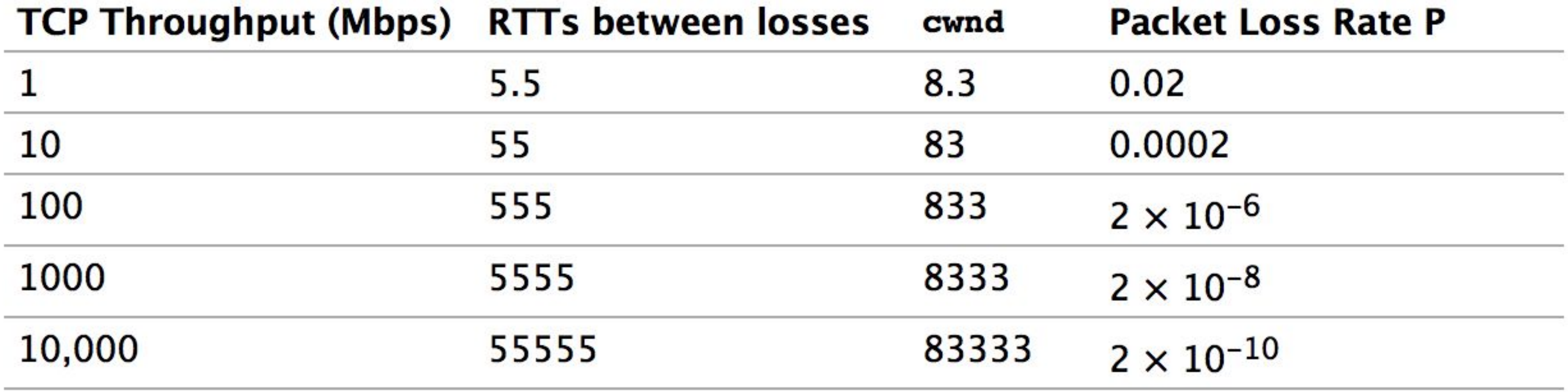
In 1997 paper, Mathis derived a simple model for a TCP flow’s throughput.

$$BW < \left(\frac{MSS}{RTT}\right)\frac{1}{\sqrt{p}}$$

Assume, MSS is 1500 bytes and RTT is 100ms. Turn to a partner discuss:

- How big does TCP Reno’s cwnd need to be to utilize 10 Gbps available BW?
- To achieve 10 Gbps throughput, what does the loss probability have to be?

The Mathis equation shows why TCP Reno does not work well in highspeed networks and lossy networks, which are common today! This is **TCP’s high bandwidth problem**.

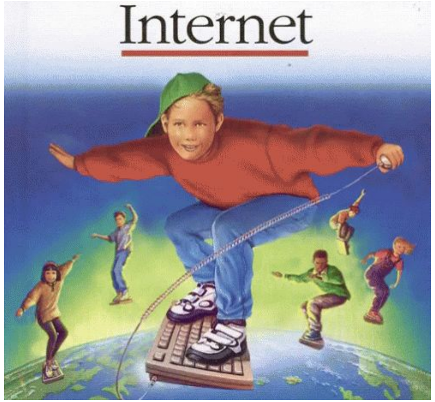


TCP Throughput (Mbps)	RTTs between losses	cwnd	Packet Loss Rate P
1	5.5	8.3	0.02
10	55	83	0.0002
100	555	833	2×10^{-6}
1000	5555	8333	2×10^{-8}
10,000	55555	83333	2×10^{-10}

Well maybe it is not always fair, but TCP Reno is definitely efficient!

Raj Jain

And what about in lossy networks like Wi-Fi?

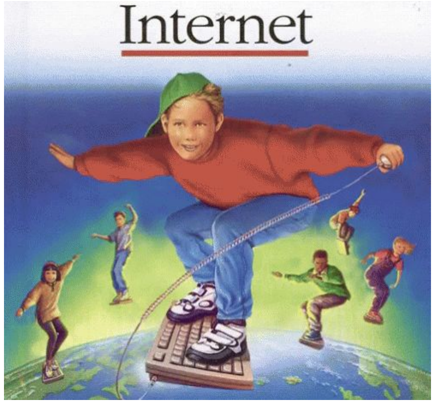


The Internet

Well maybe it is not always fair, but TCP Reno is definitely efficient!

Raj Jain

TCP Reno assumes EVERY packet loss is because of congestion! That’s wack!

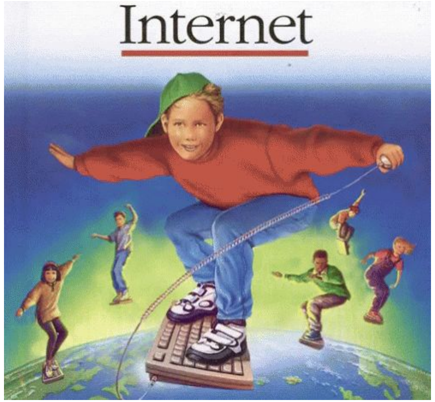


The Internet

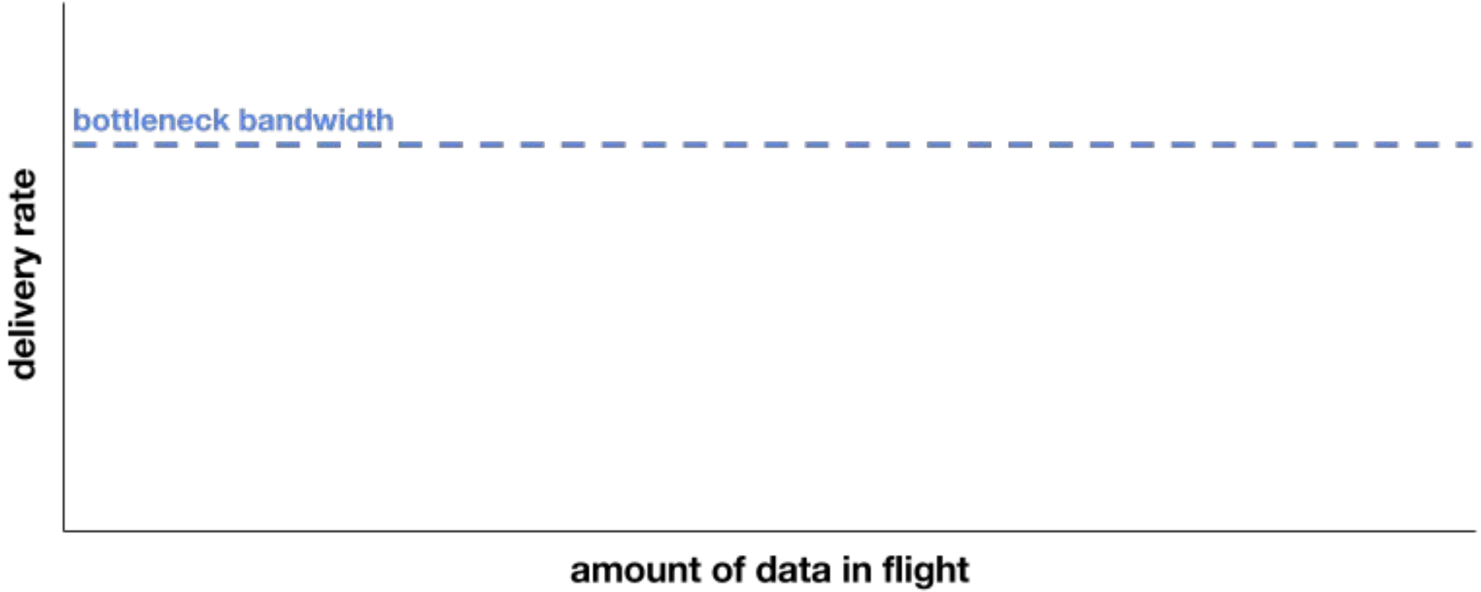
Fine. But, there’s no way you can tell me Reno doesn’t converge to the optimal point though!

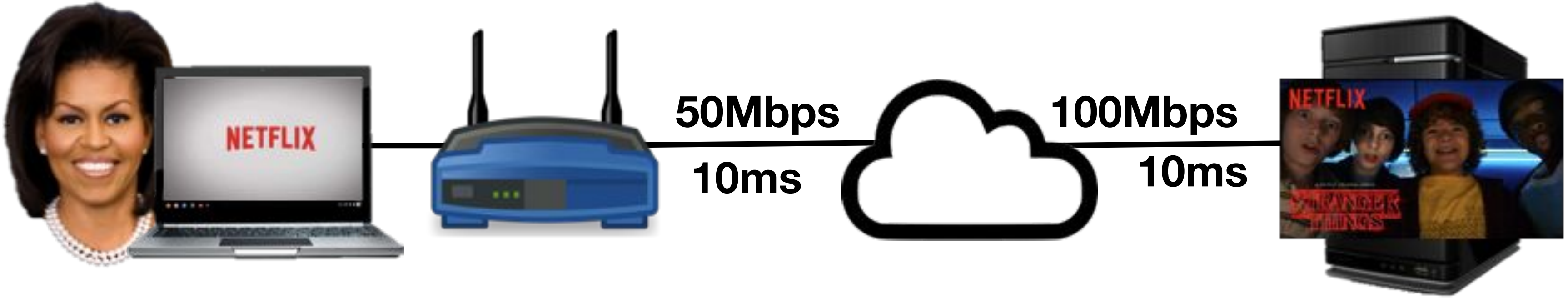
Is it REALLY optimal though?

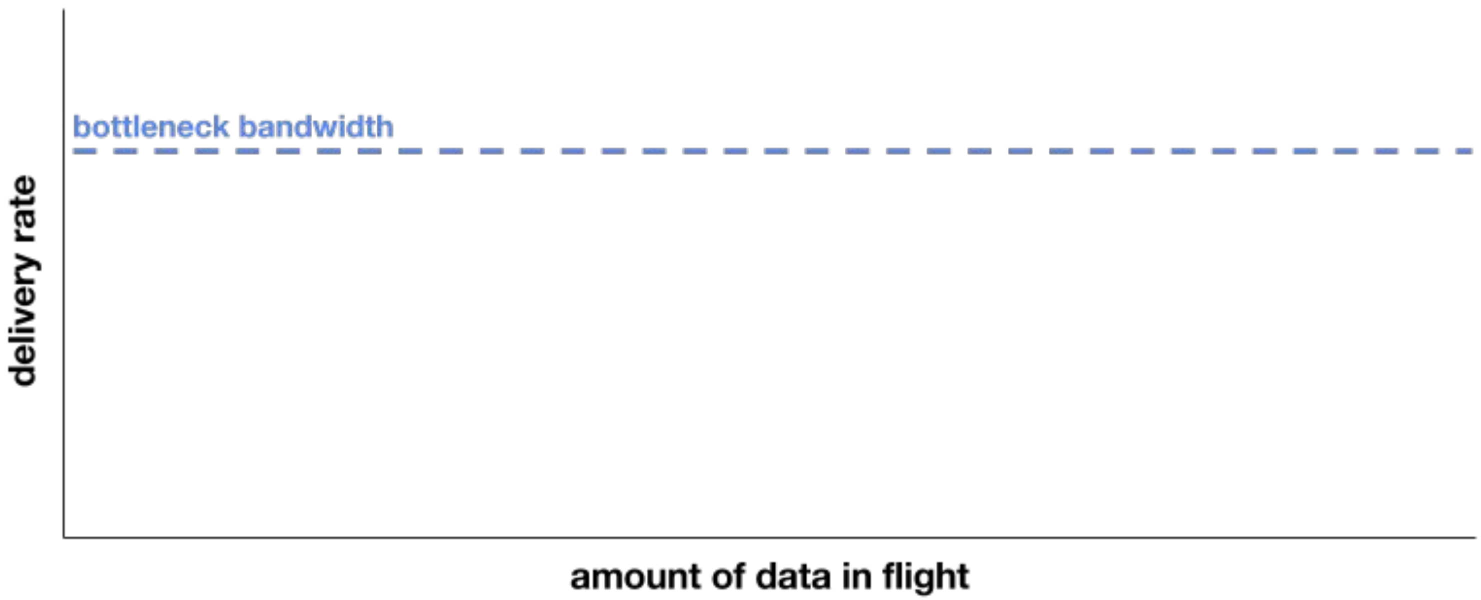
Raj Jain

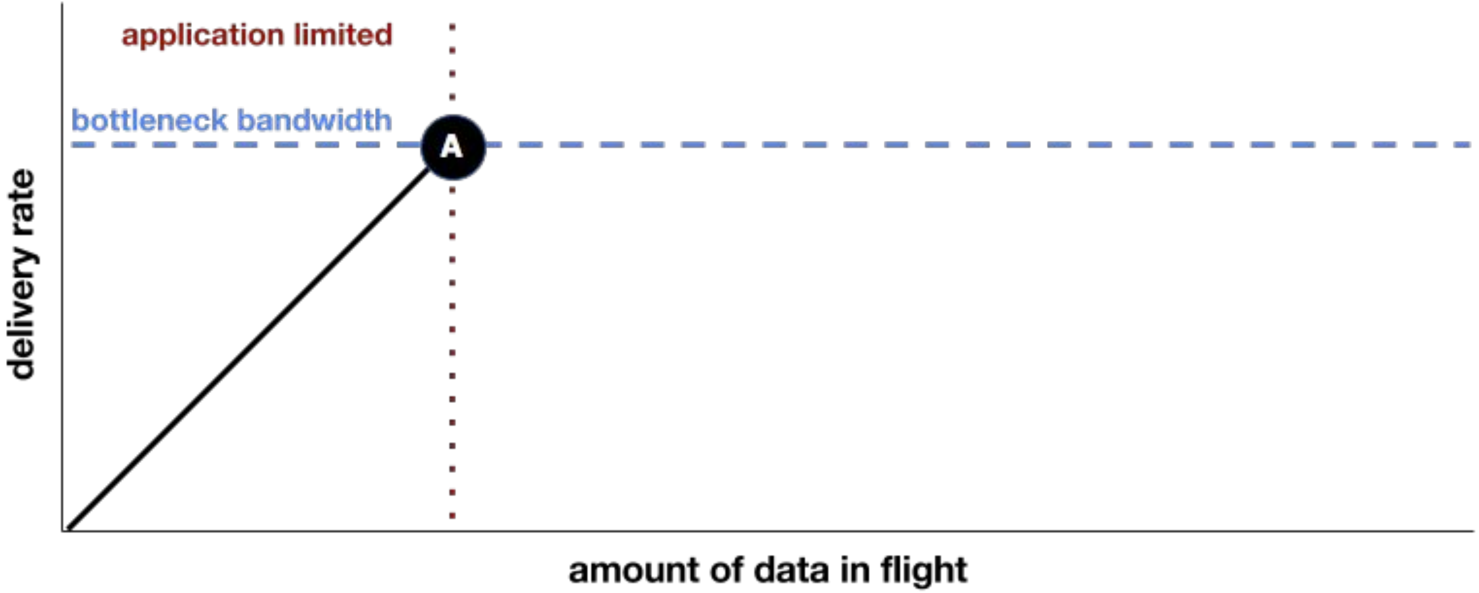


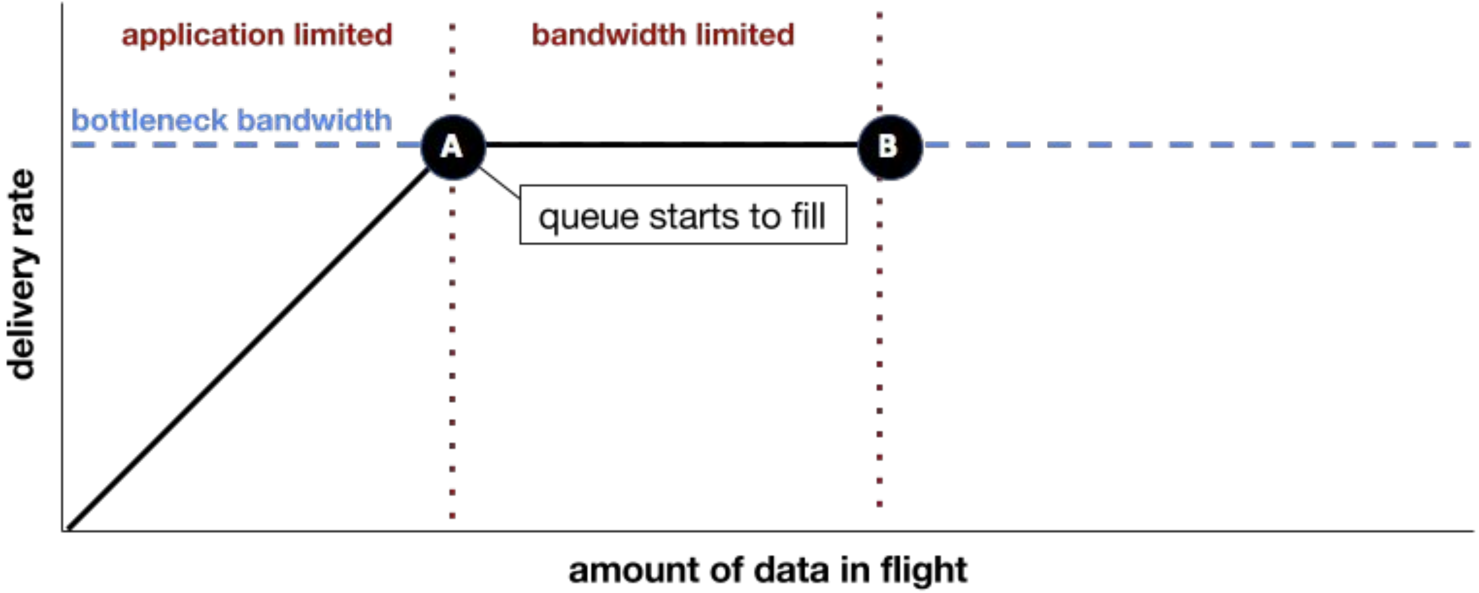
The Internet

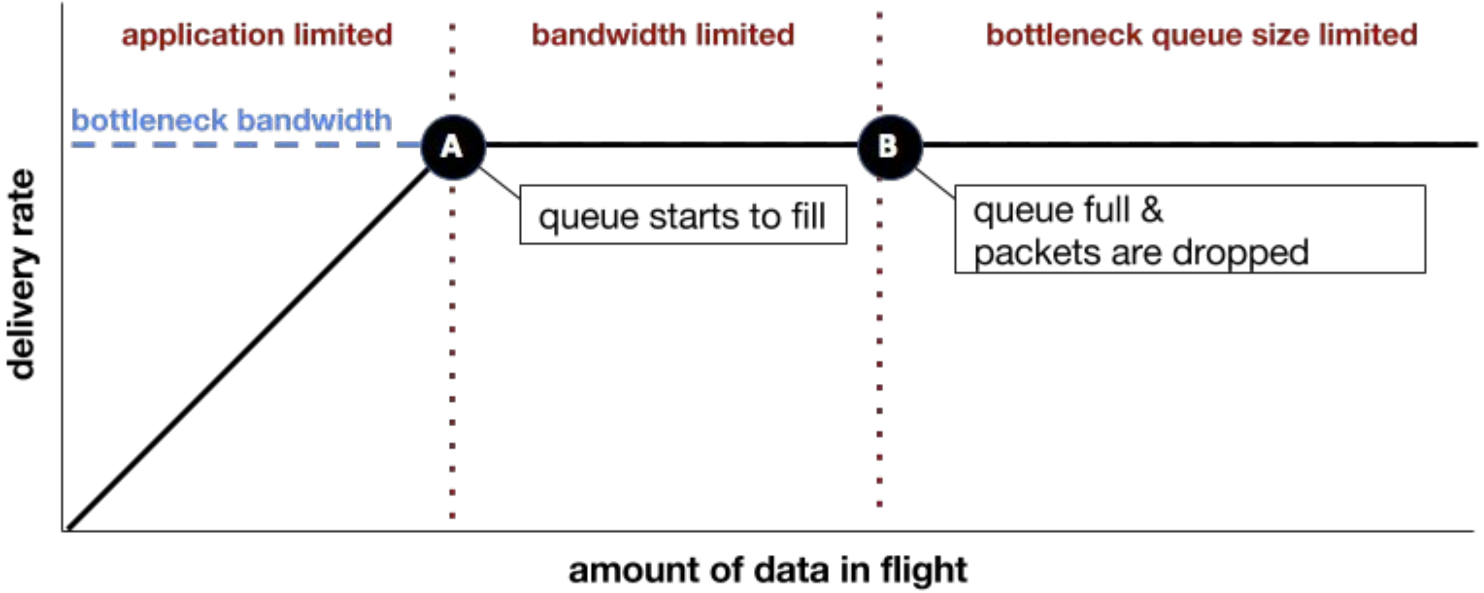

bottleneck bandwidth
delivery rate
amount of data in flight


NETFLIX
50Mbps
10ms
100Mbps
10ms
NETFLIX
STRANGER THINGS


bottleneck bandwidth
delivery rate
amount of data in flight

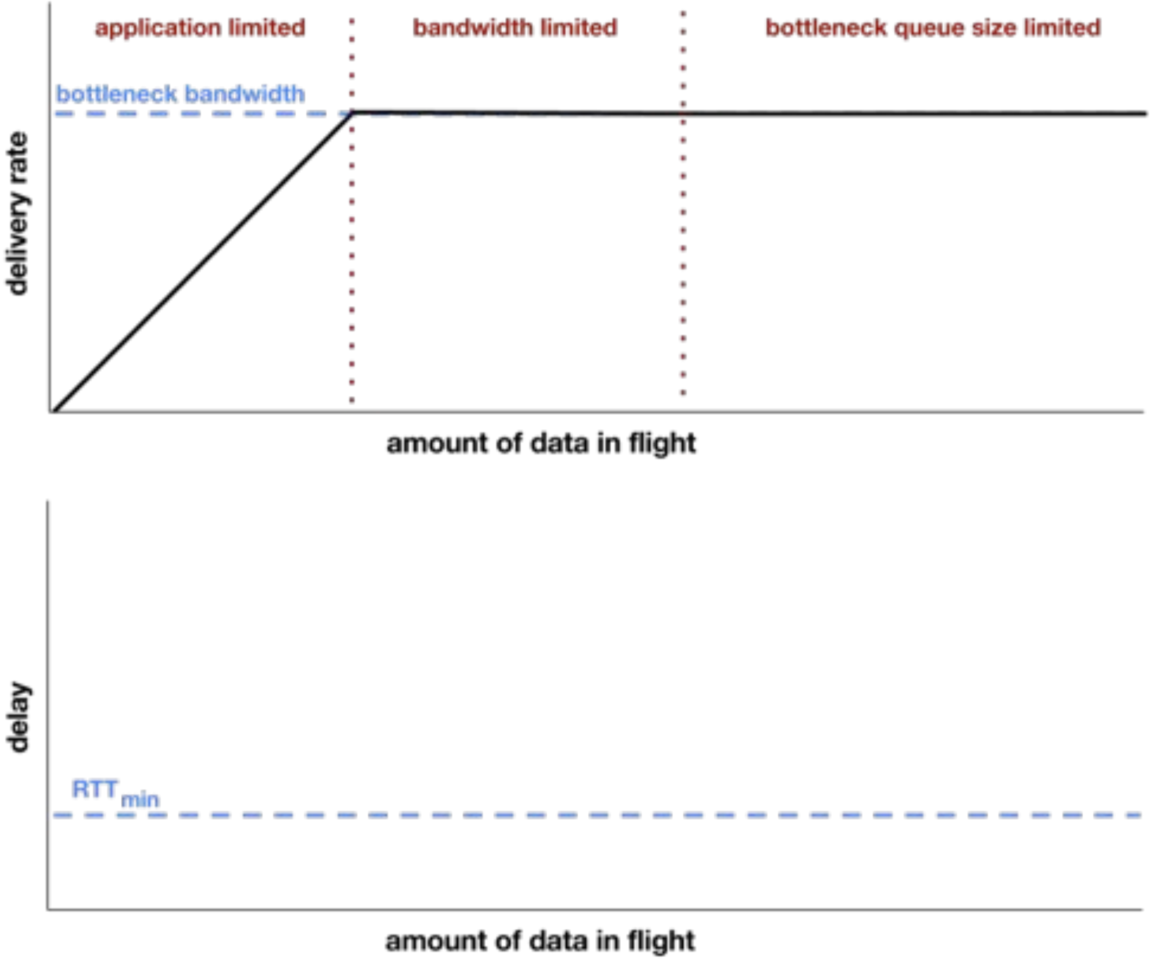

application limited
bottleneck bandwidth
A
delivery rate
amount of data in flight


application limited
bandwidth limited
bottleneck bandwidth
A
B
queue starts to fill
delivery rate
amount of data in flight

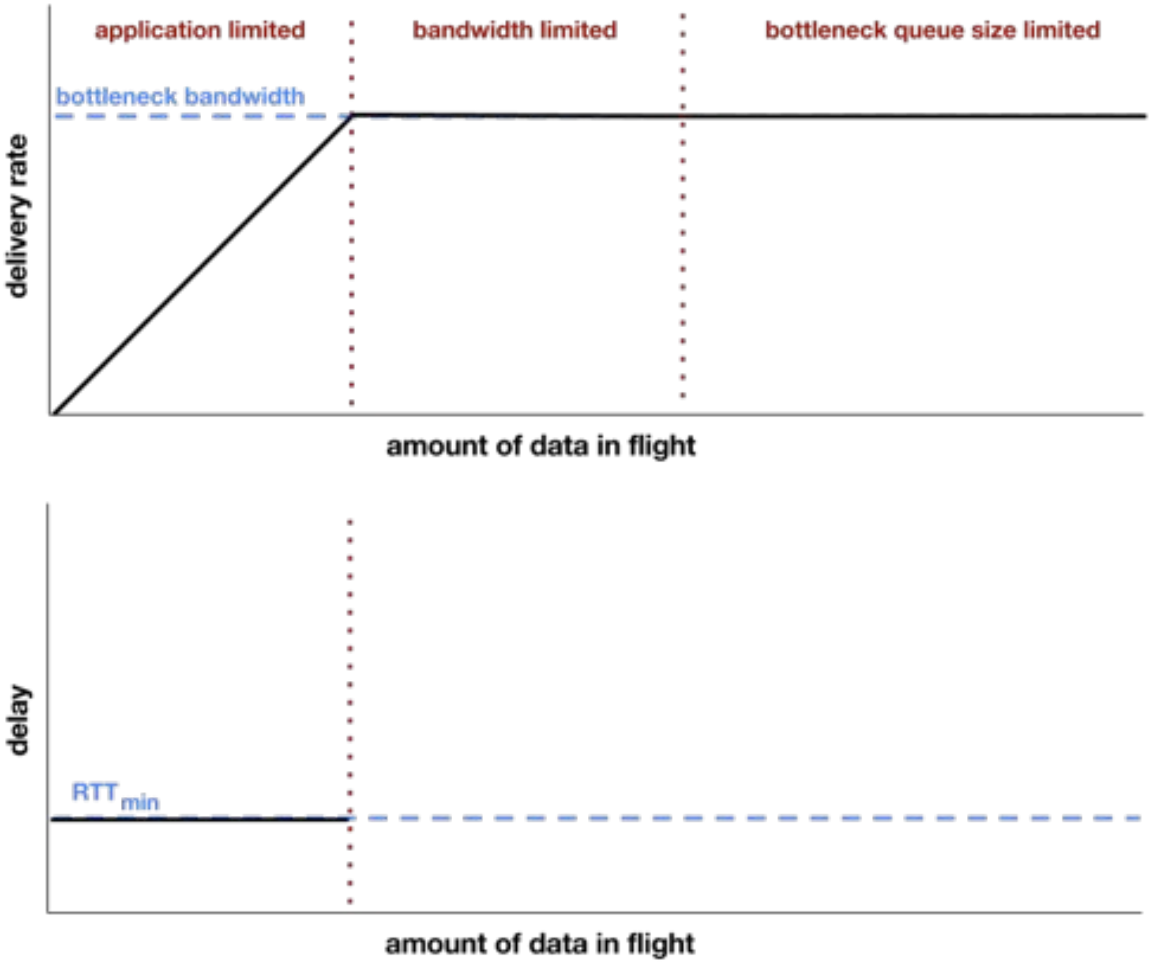


Filling the bottleneck queue, let's a flow fully utilize the available bottleneck bandwidth.

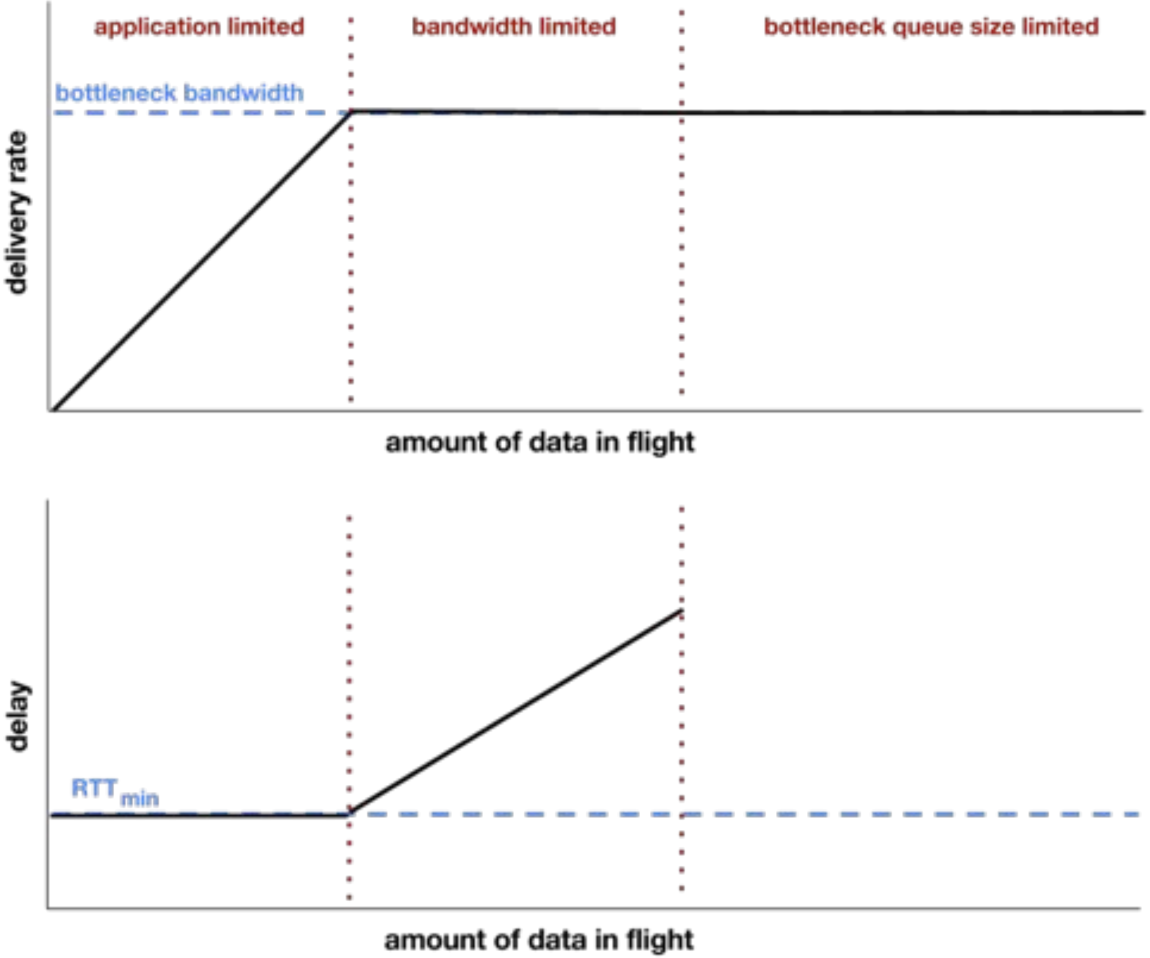
But what happens to delay when the bottleneck queue is full?



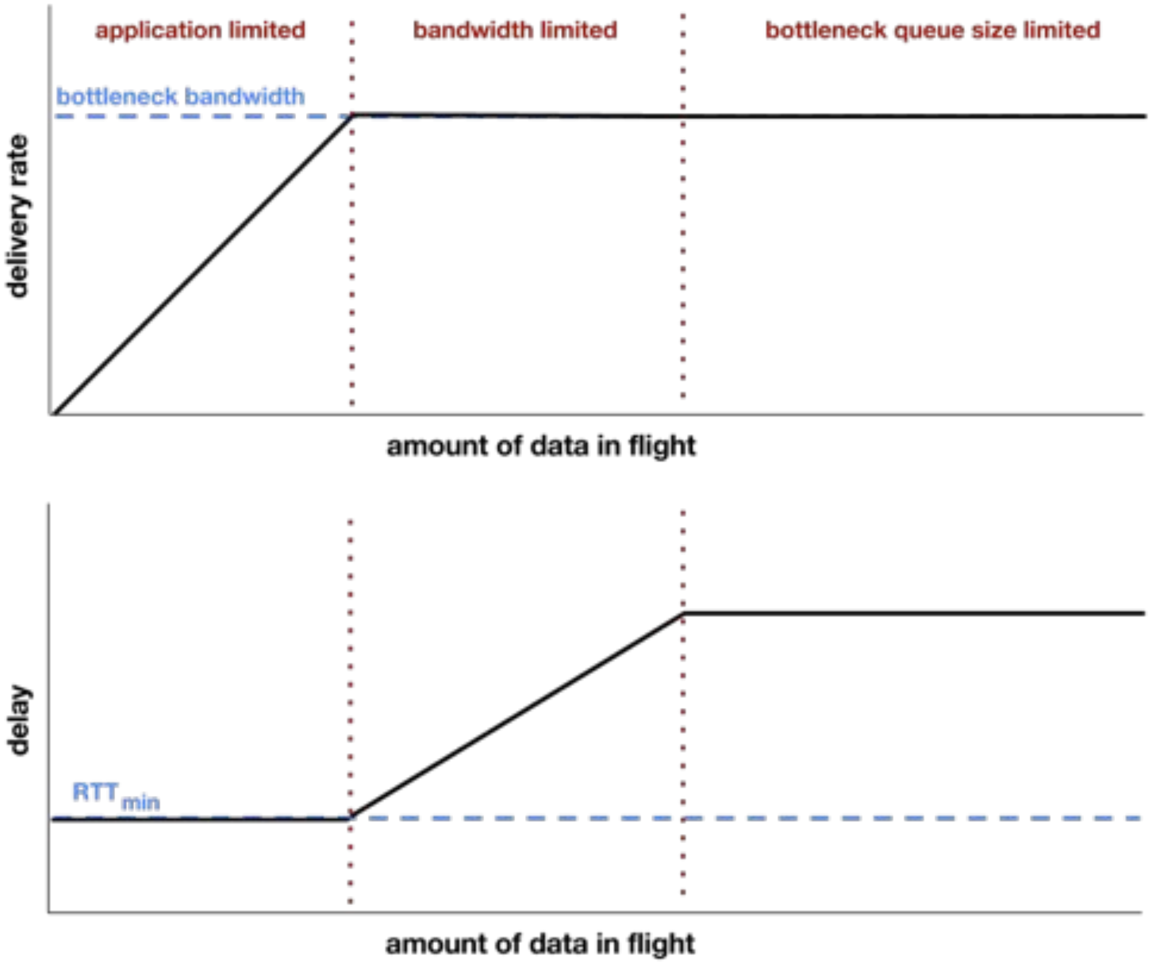
But what happens to delay when the bottleneck queue is full?



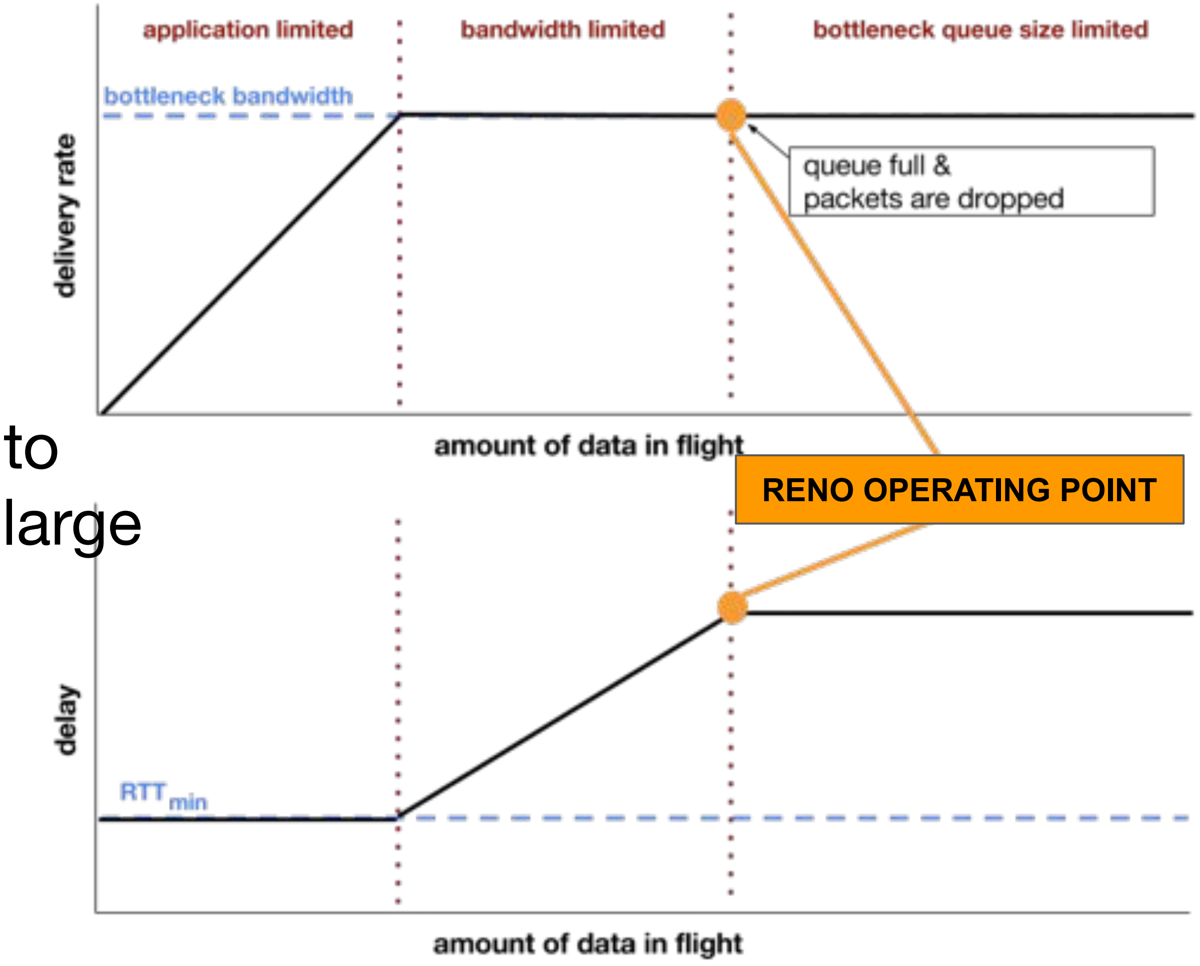
But what happens to delay when the bottleneck queue is full?



But what happens to delay when the bottleneck queue is full?



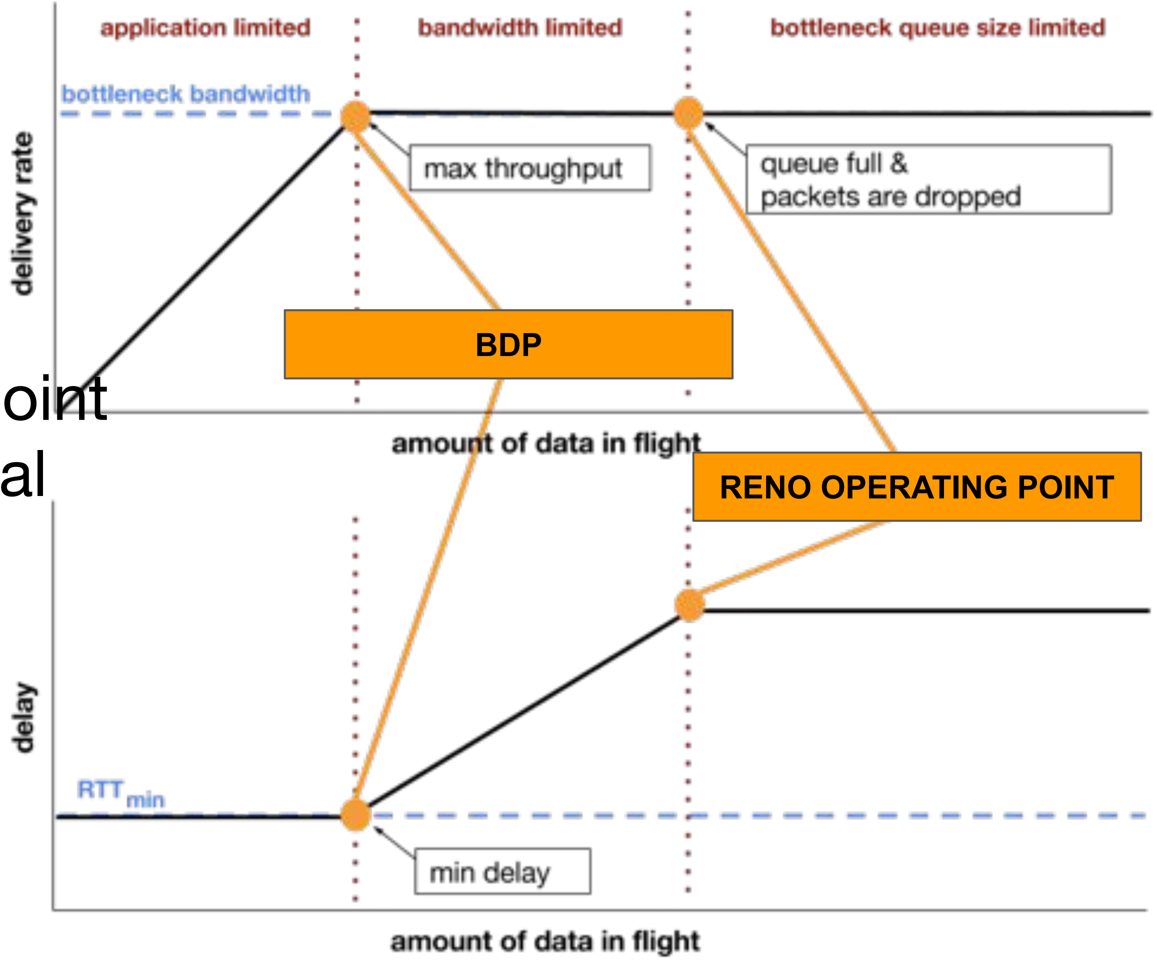
TCP Reno fills the bottleneck queue to find BDP causing large queueing delays. (bufferbloat)



queue full & packets are dropped

RENO OPERATING POINT

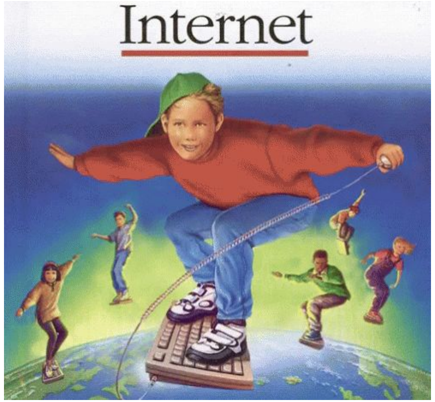
In 1976, Leonard Kleinrock showed optimal operating point for a CCA is maximal throughput and minimal delay.



Gah! How could any CCA possibly work in all scenarios!?

Good question! I don’t know, man. I just deliver packets.

Raj Jain



The Internet

What’s bad about TCP Reno?

- It’s performance sucks!

TCP Reno’s implicit assumptions hurt performance in modern networks!

- Reno takes too long to find BDP in large BDP networks
- Reno assumes every loss is because of packet loss (bad for Wi-Fi networks and high speed networks)
- Reno’s cwnd update speed is proportional to RTT - super slow for long RTT’s (satellite networks), causes RTT unfairness (can lead to starvation when RTT diff is 100:1)
- Reno fills queues causing large queuing delays (delay sensitive applications like Web apps, gaming)

What CCAs are deployed in the Internet today?

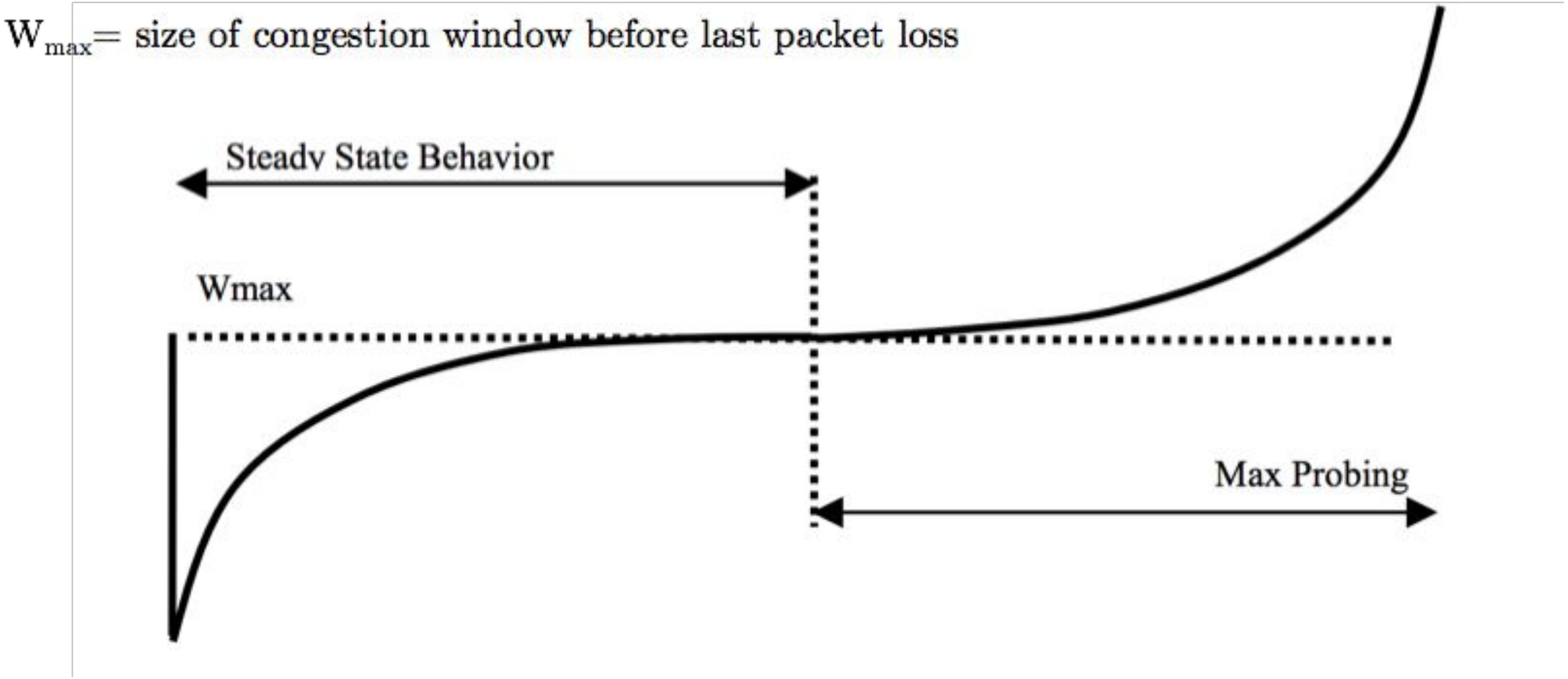
The CCA in TCP is plug-and-play. It does not have to be Reno! In Linux, you can actually change the CCA per socket!

```
#include <netinet/in.h>
#include <netinet/tcp.h>
...
char * cong_algorithm = "bbr";
int slen = strlen( cong_algorithm ) + 1;
int rc = setsockopt( sock, IPPROTO_TCP,
TCP_CONGESTION, cong_algorithm, slen);
if (rc < 0) { /* error */ }
```

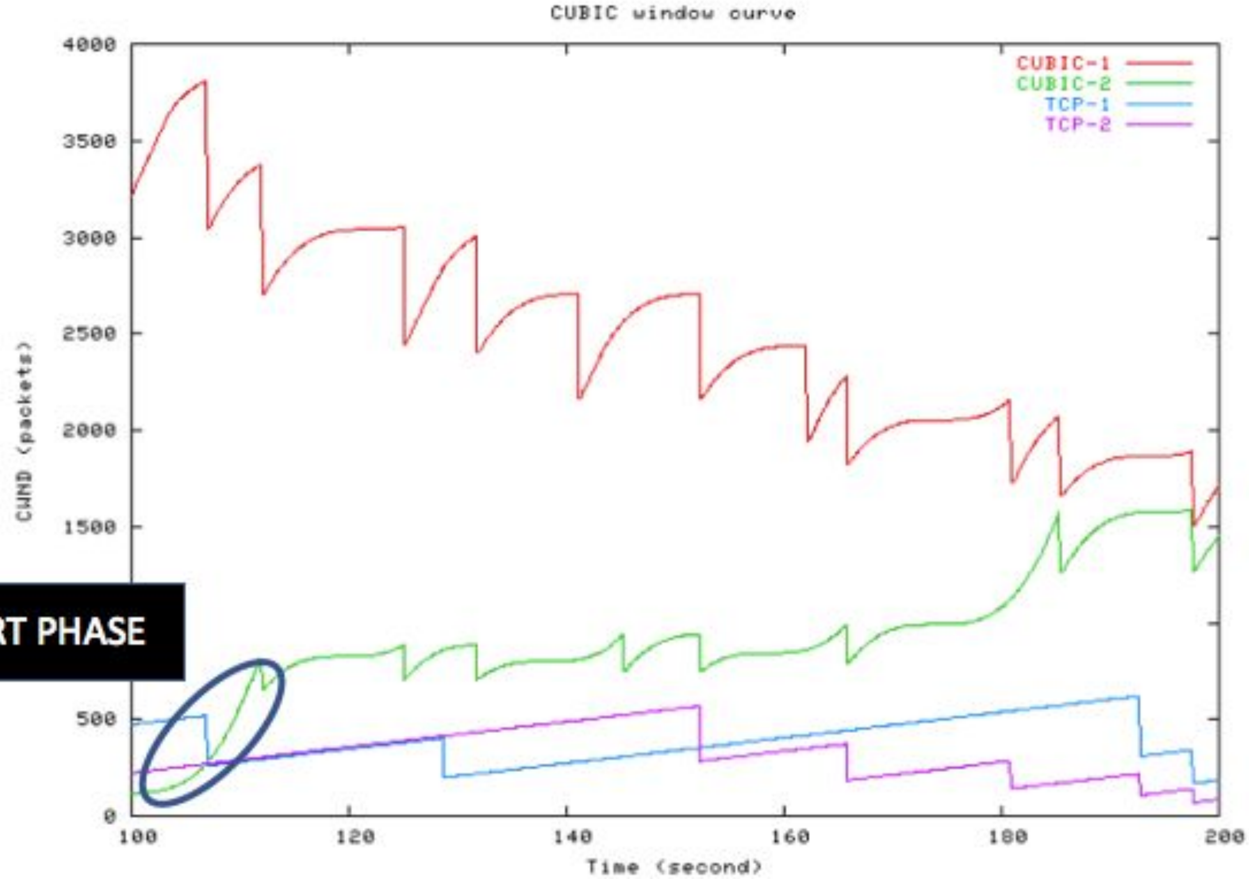
Over the past 30 years, there have been an alphabet soup of alternatives to AIMD/TCP Reno proposed and deployed.

- **Fix highspeed TCP problem:** Cubic, CompoundTCP
- **Fix bufferbloat problem:** BBR
- **Network-assisted CCAs and Active Queue Management (AQM)** popular in datacenters: RED, RCP, XCP, ECN, DCTCP, TIMELY
- F**ancy machine learning approaches** (not really deployed yet thought): PCC, Remy

TCP Cubic is similar to TCP Reno but its window growth function is cubic instead of linear.

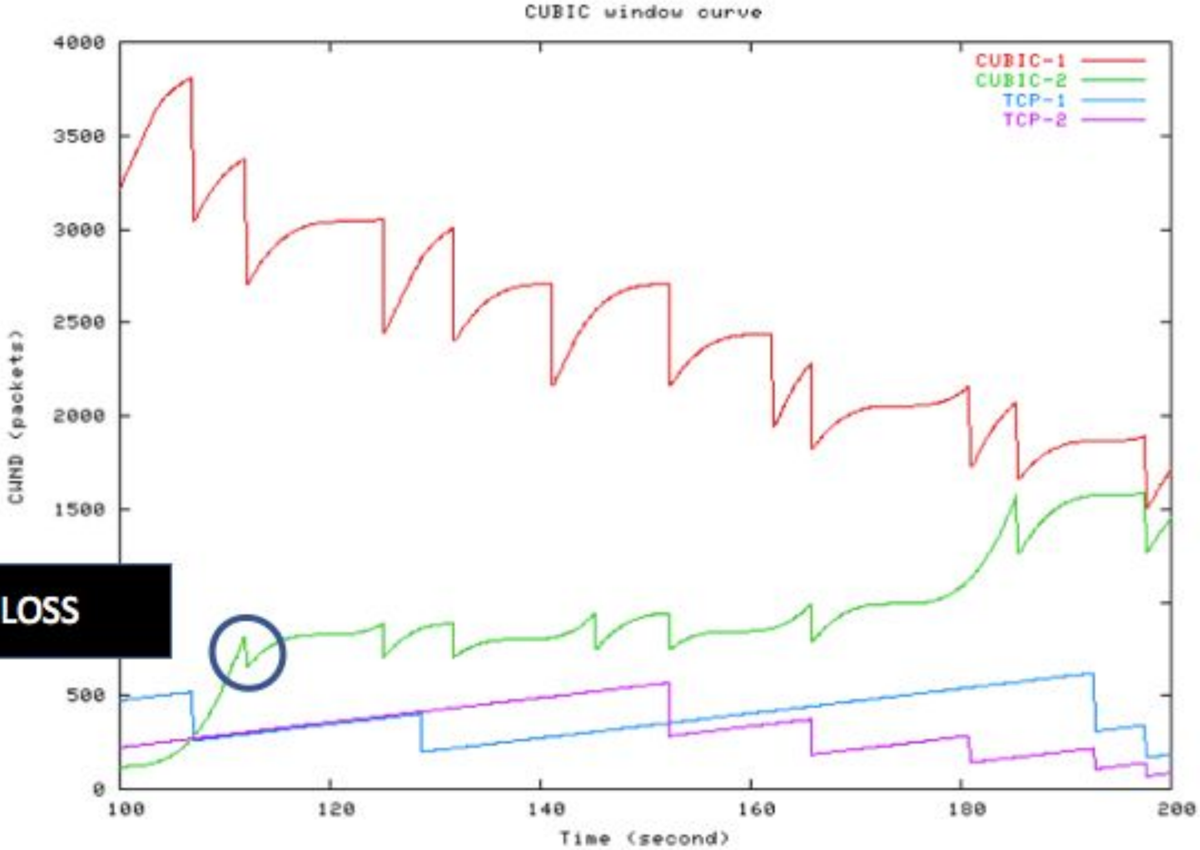

W_{max}= size of congestion window before last packet loss
Steadv State Behavior
Wmax
Max Probing

Initially, TCP Cubic sending rate rapidly approaches available capacity.



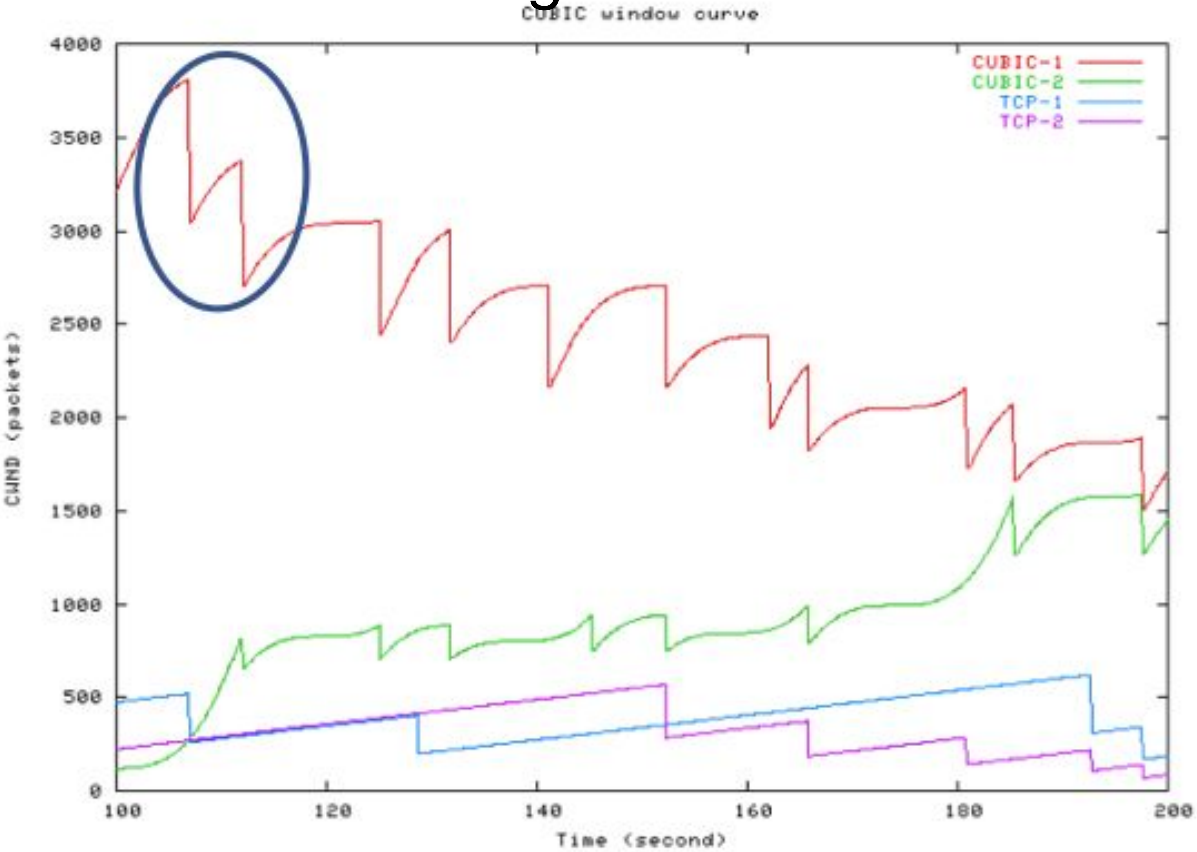
SLOW START PHASE

Cubic responds to packet loss by reducing the congestion window by 20%.

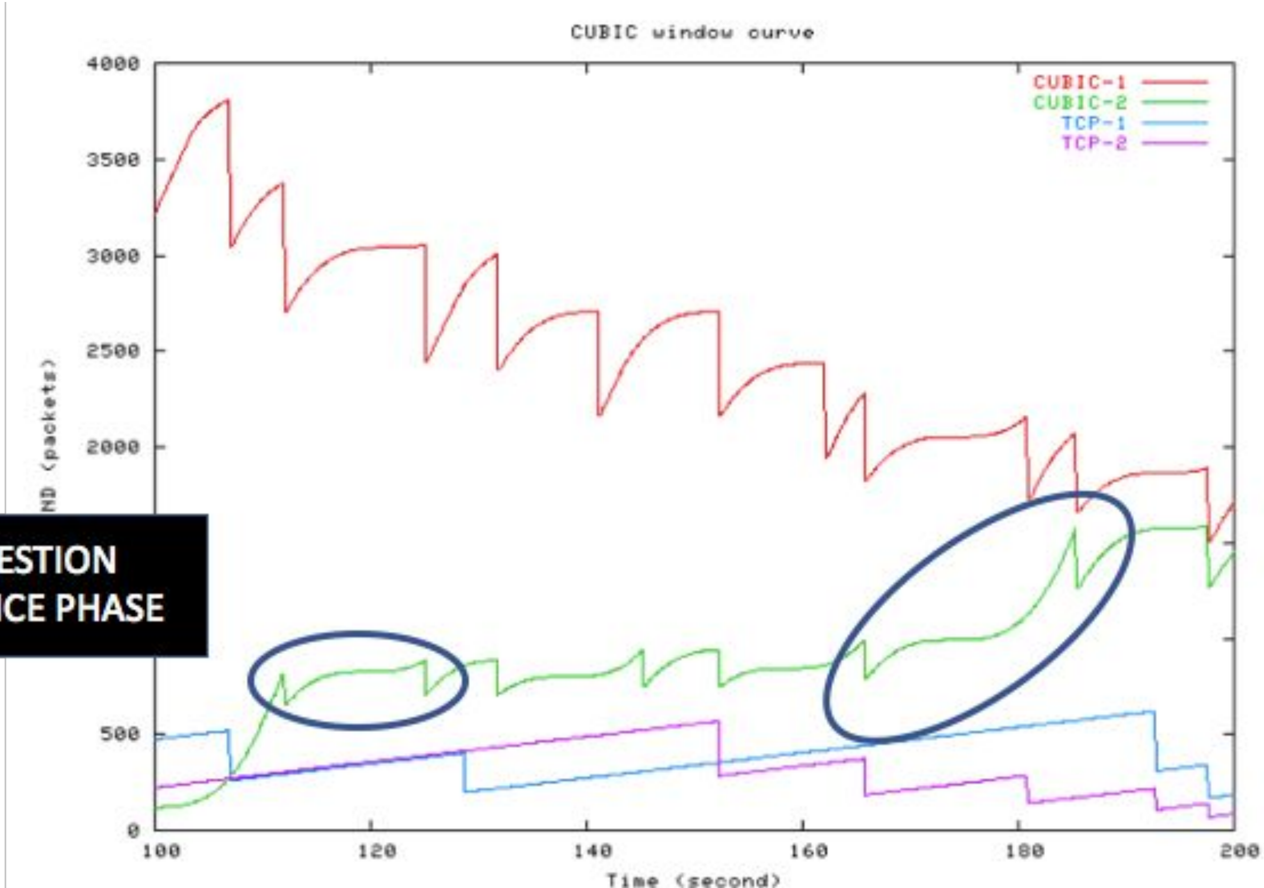


PACKET LOSS

To ensure fairness, Cubic reduces window by 40% if it’s estimation of maximum cwnd grows smaller.

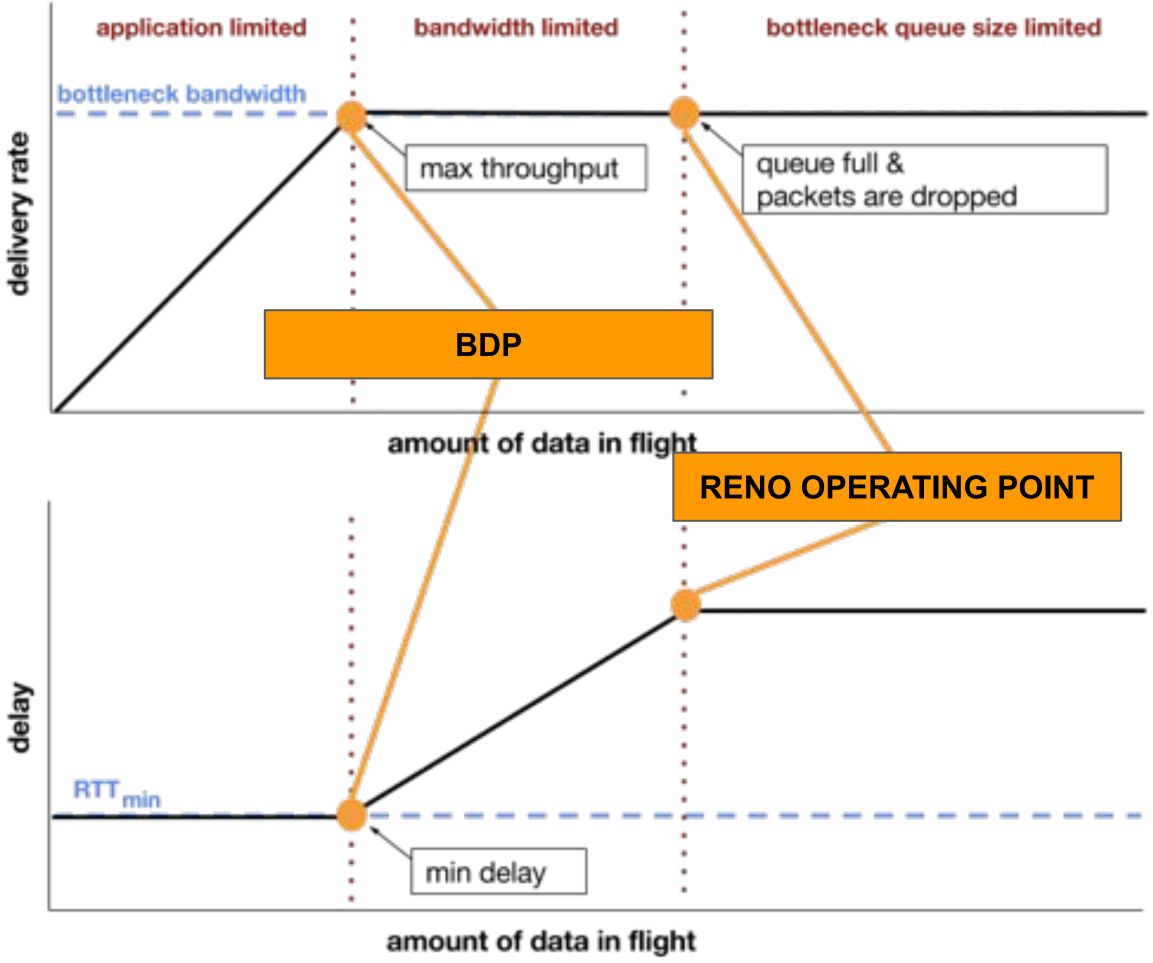


After packet loss, Cubic’s window growth function is cubic.

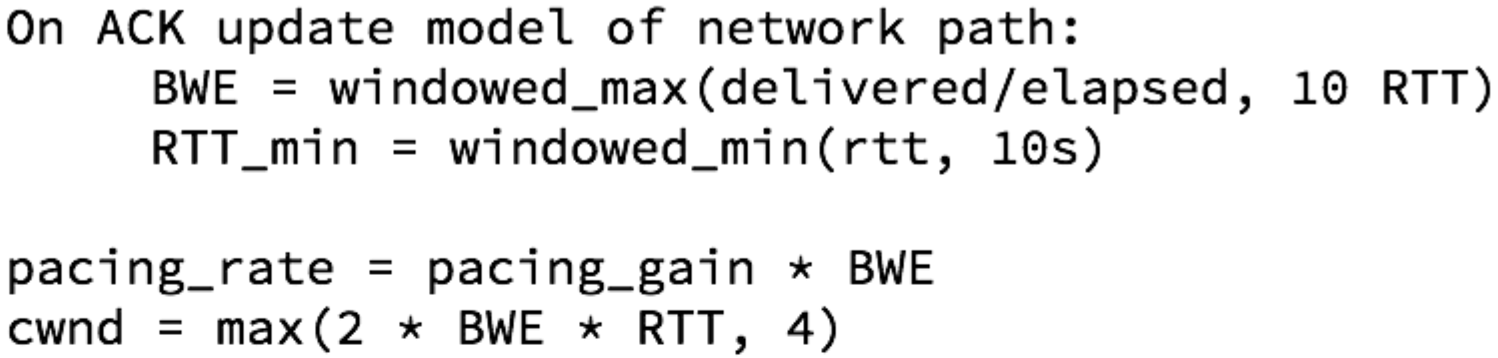


CONGESTION AVOIDANCE PHASE

BBR aims to minimize delay and maximize throughput by sending data at BDP rate.



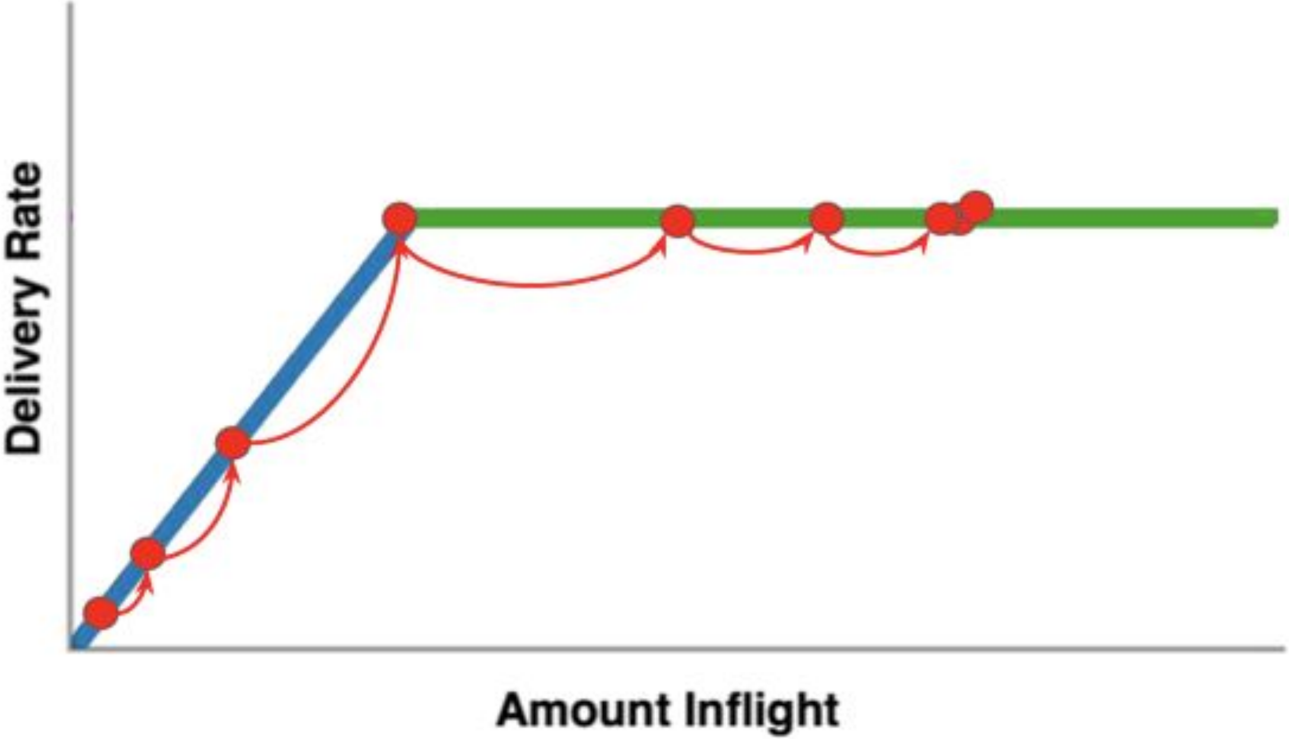
BBR’s core algorithm builds a ‘model’ of the network path and tries to send at bottleneck bandwidth rate, with no more than 2BDP packets in flight.



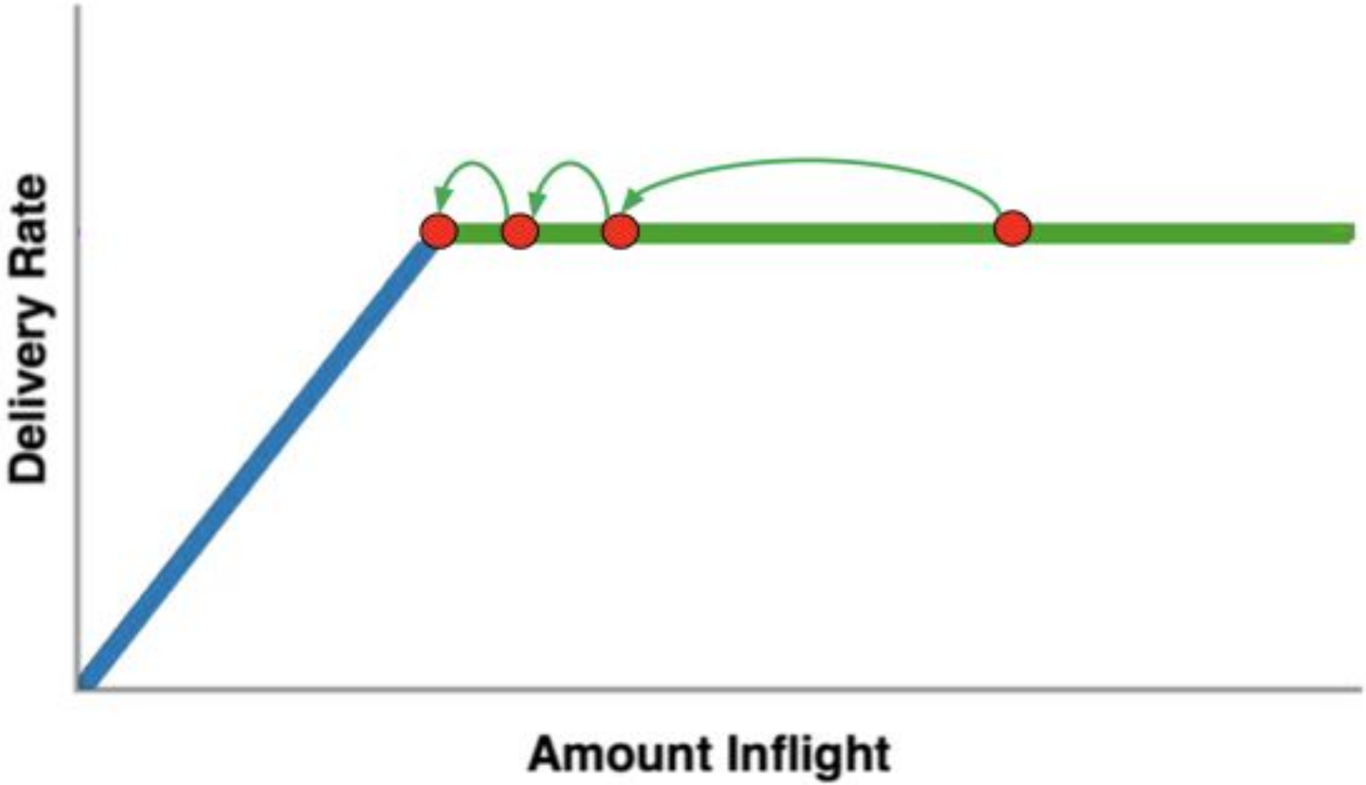
```
On ACK update model of network path:
     BWE = windowed_max(delivered/elapsed, 10 RTT)
     RTT_min = windowed_min(rtt, 10s)

pacing_rate = pacing_gain * BWE
cwnd = max(2 * BWE * RTT, 4)
```

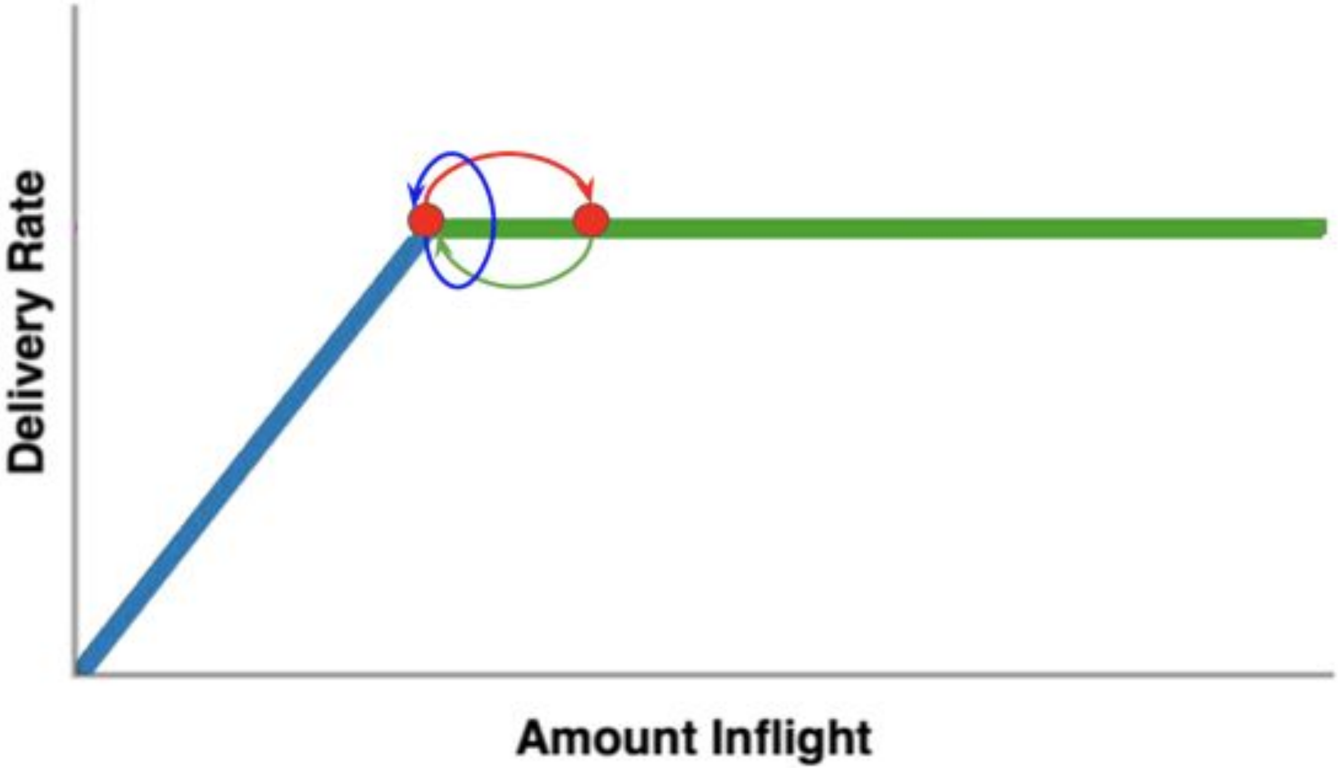
Initially, BBR increases the sending rate exponentially to estimate bandwidth.



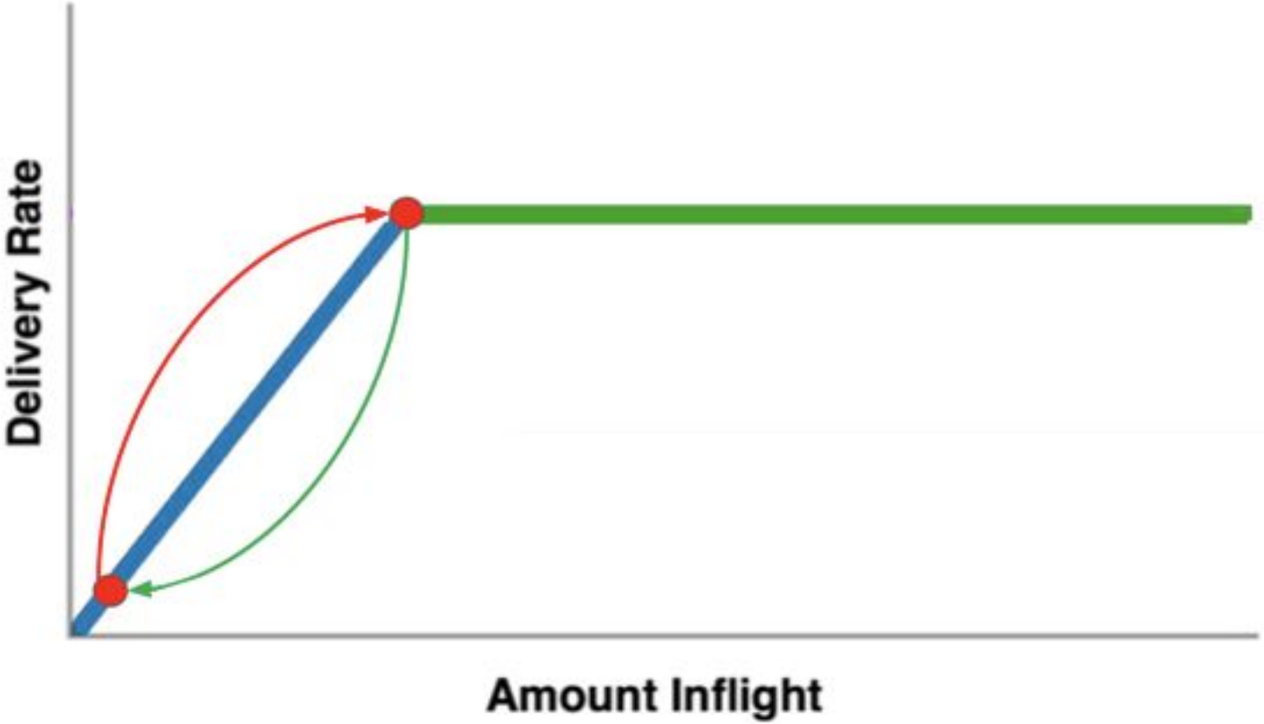
BBR reduces sending rate to drain the queue that could have built up during STARTUP phase.



BBR reduces sending rate to drain the queue that could have built up during STARTUP phase.



To ensure fairness, every 10s, if the RTTmin estimate hasn’t gotten smaller, then BBR reduces the cwnd to 4 and updates RTTmin estimate.



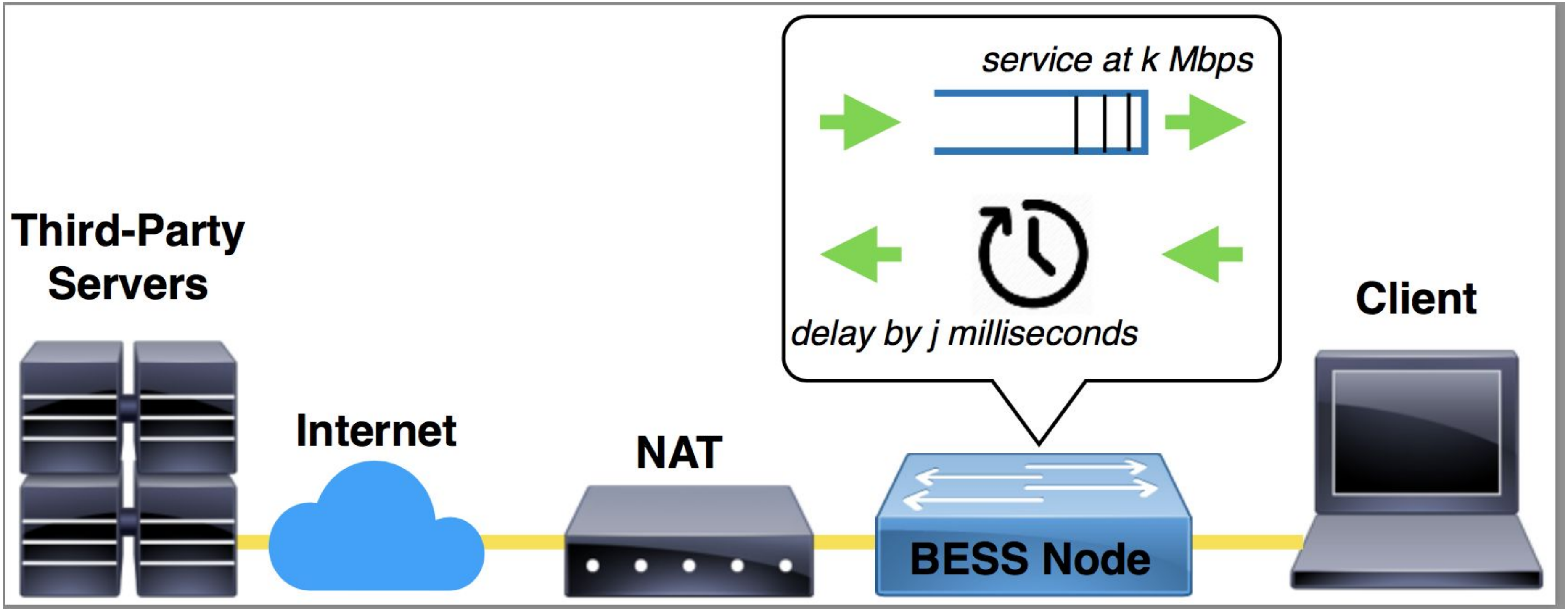
What CCAs are deployed in the Internet today?

- We can guess but we don’t really know.
- Lots of possible algorithms floating out there in the Internet.

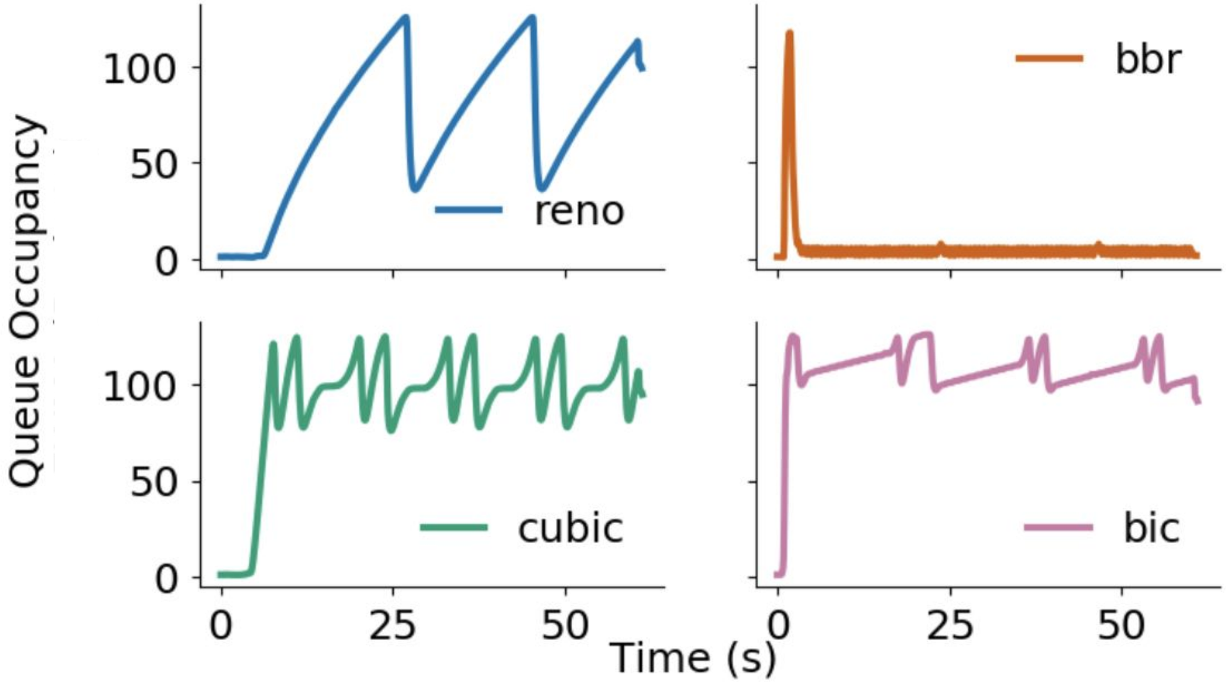
My research attempts to answer this question through empirical measurement! We conduct a census of what CCAs are deployed on some popular websites.

Downloading a large file from a website, how can we determine what CCA the website is using?

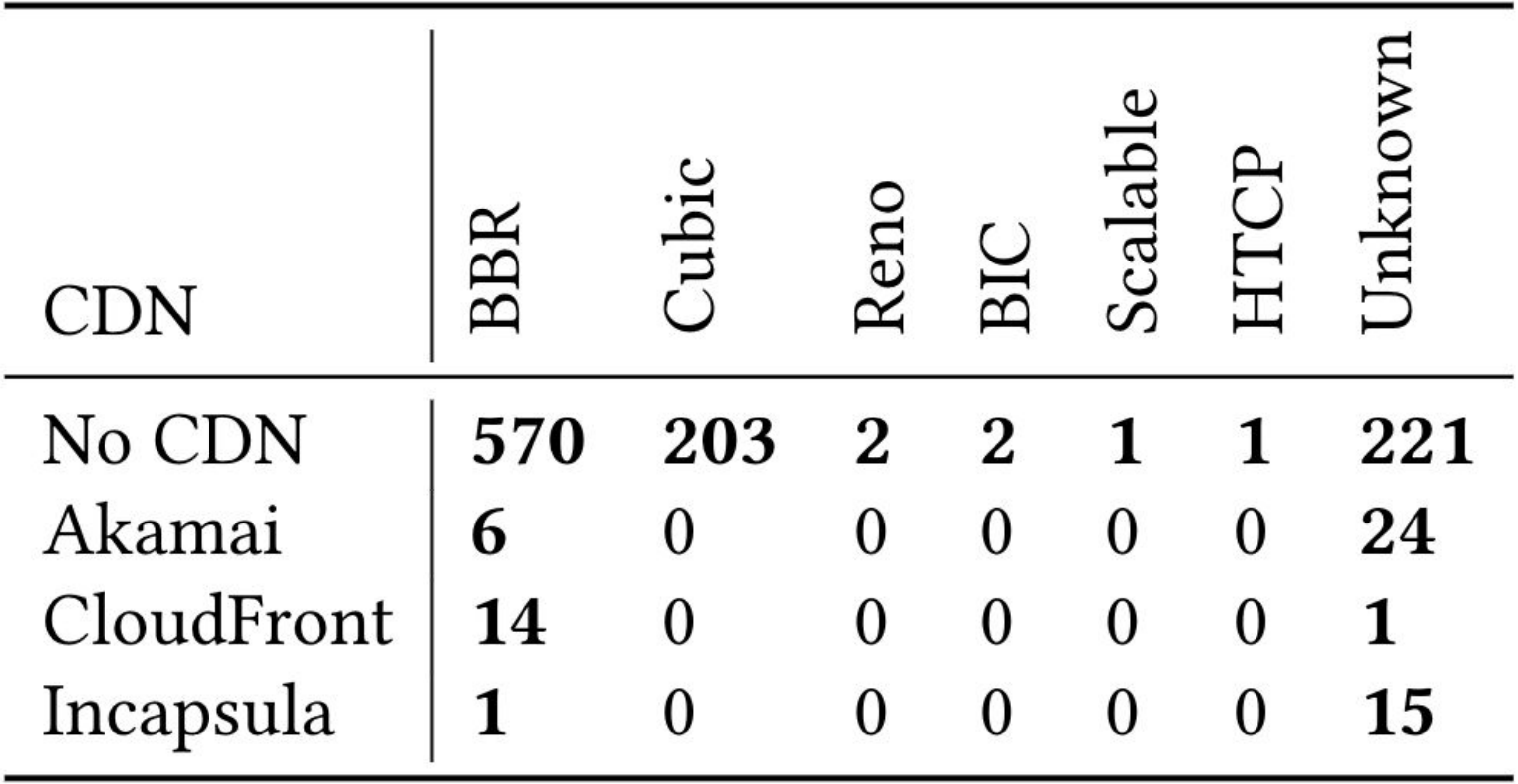
We build a testbed that allows us to control the bottleneck queue and see a TCP sender's queue occupancy over time.



We build a testbed that allows us to control the bottleneck queue and see a TCP sender's queue occupancy over time.



Our census results show BBR is popular, Reno is not.



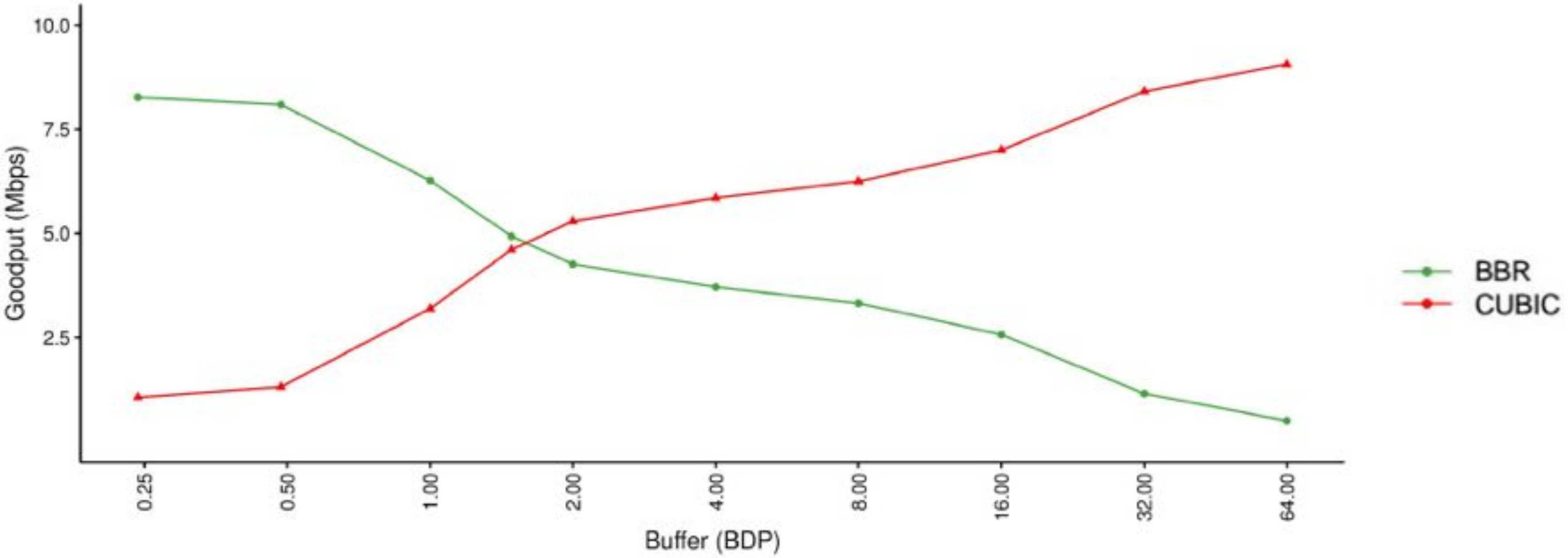
CDN	BBR	Cubic	Reno	BIC	Scalable	HTCP	Unknown
No CDN	**570**	**203**	**2**	**2**	**1**	**1**	**221**
Akamai	**6**	0	0	0	0	0	**24**
CloudFront	**14**	0	0	0	0	0	**1**
Incapsula	**1**	0	0	0	0	0	**15**

Is the Internet fair?

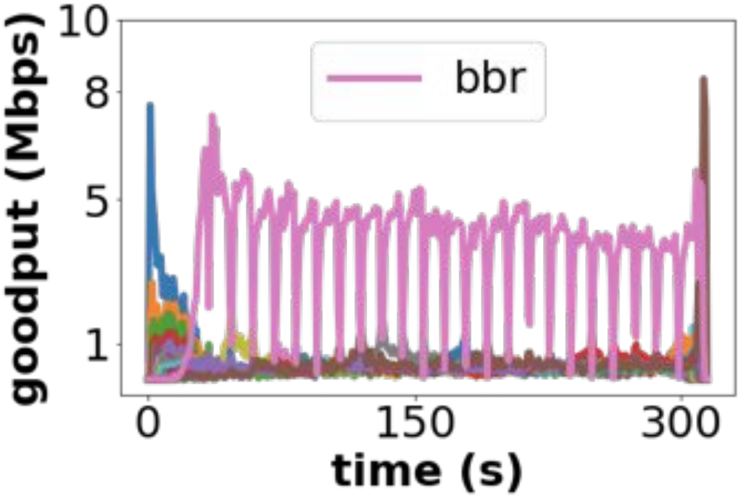
- This is my research!
- You are not expected to know the material in this section for a test or homework.

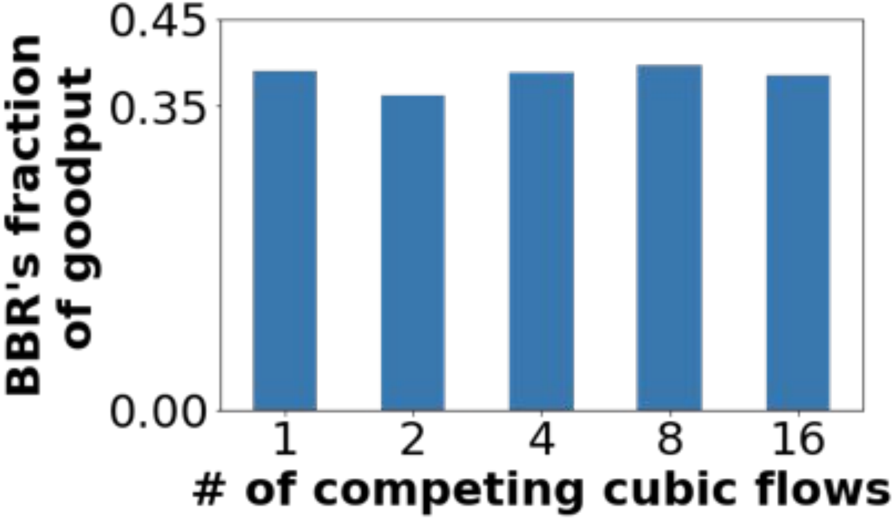
When heterogenous algorithms compete, fairness can be a problem.

CUBIC vs BBR goodput: bw = 10Mbps, RTT = 40ms, 4 min. bulk xfer, varying buffer sizes

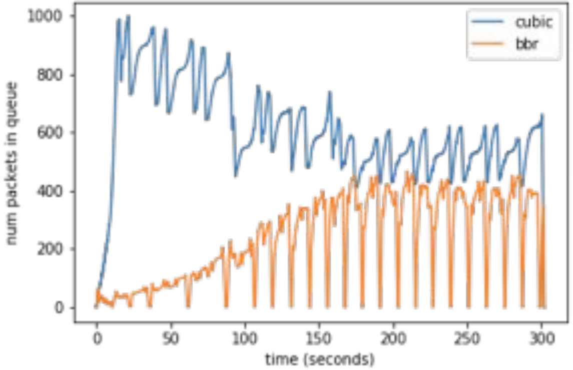


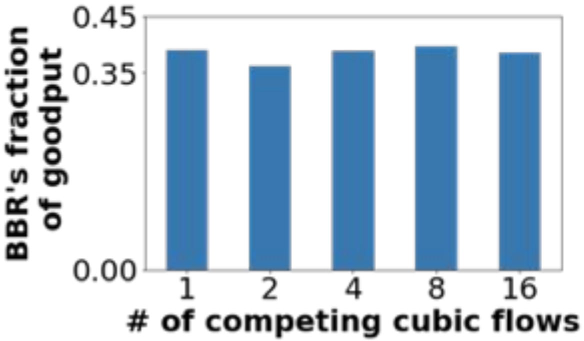
What happens when Cubic flows compete with 1 BBR flow?

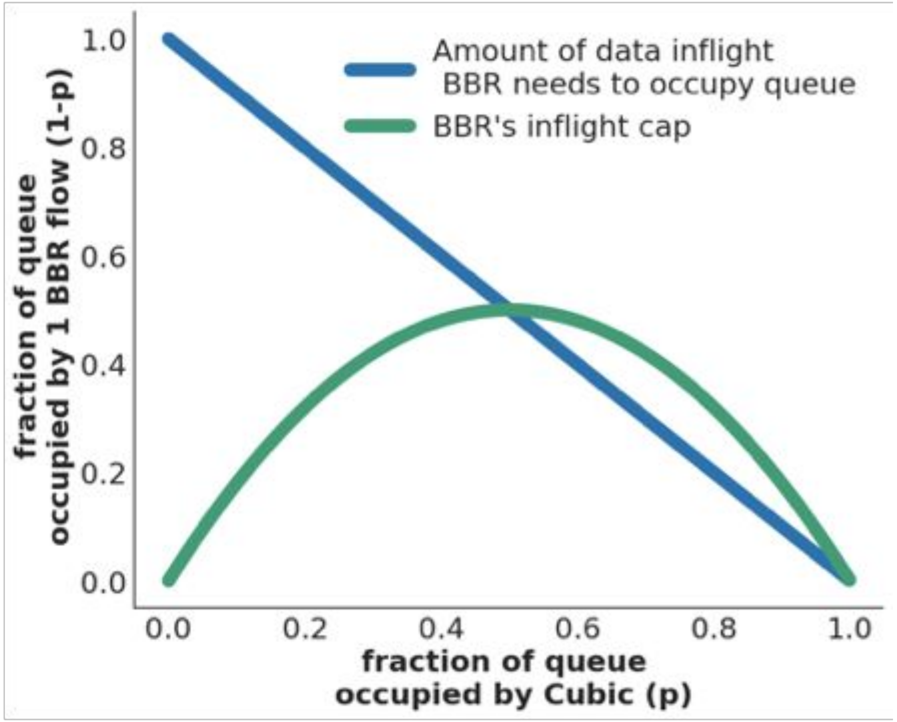




We define a model that explains BBR's behavior when competing with loss-based CCAs.







Is the Internet fair?

- Maybe? It depends.

Congestion control is one of the oldest topics in networking, and yet is still an active area of research!

Congestion Control Hall of Fame

Van Jacobson

Raj Jain

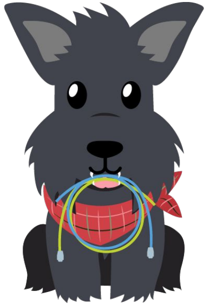
Matt Mathis

Sally Floyd

Leonard Kleinrock

Nandita Dukkipati

Ranysha Ware



You?

Today, you learned:

- **What's good about Reno:** Reno meets 4 criteria of a good CCA as defined by Chiu and Jain: efficiency, fairness, distributedness, and convergence.
- **What's bad about Reno:** Reno's use of packet loss as congestive signal hurts performance in modern networks.
- **What's CCA's are deployed today:** Many. Examples include Cubic, the default CCA in Linux and Google's BBR.
- **Is the Internet fair:** Not always. When heterogenous algorithms compete, fairness can be a problem.