The Evolution of TCP Transport Protocols

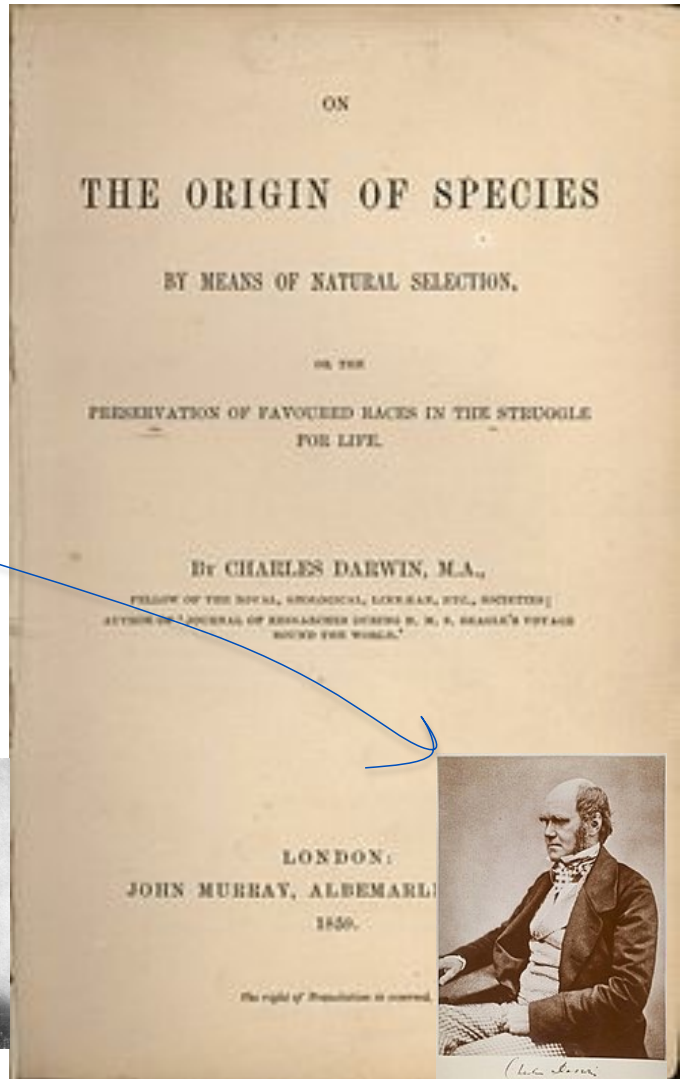
Geoff Huston AM
APNIC Labs

November 1859

Charles Darwin published a monumental work that described a theory of the origins of the diversity of life through a process of natural selection, a finding initially jointly authored in a paper by Alfred Wallace and Charles Darwin

It described a natural process that is commonly corrupted as “survival of the fittest”

It’s not just the living world where we observe these evolutionary pressures



The Evolution of Speed

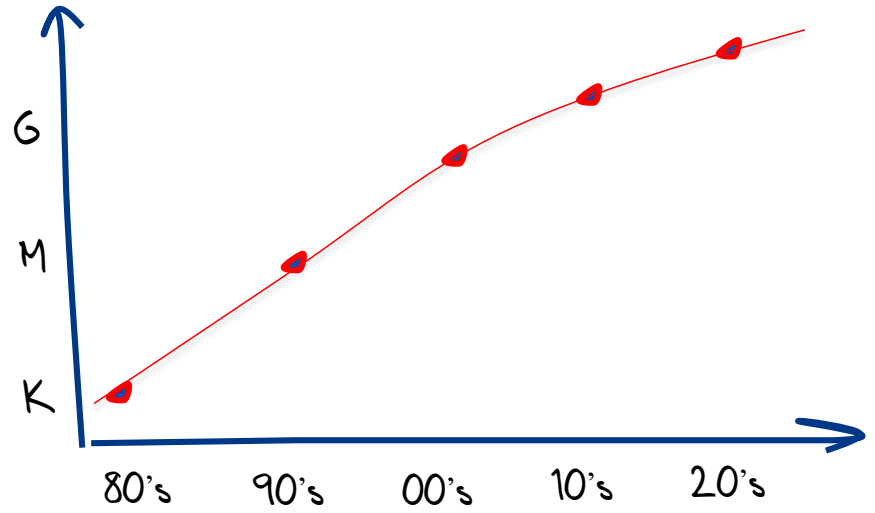
1980's
- TCP rates of Kilobits per second

1990's
- TCP rates of Megabits per second

2000's
- TCP rates of Gigabits per second

2010's
- TCP rates of tens of Gigabits per second

2020's
- TCP rates of tens of Gigabits per second



The Evolution of Speed

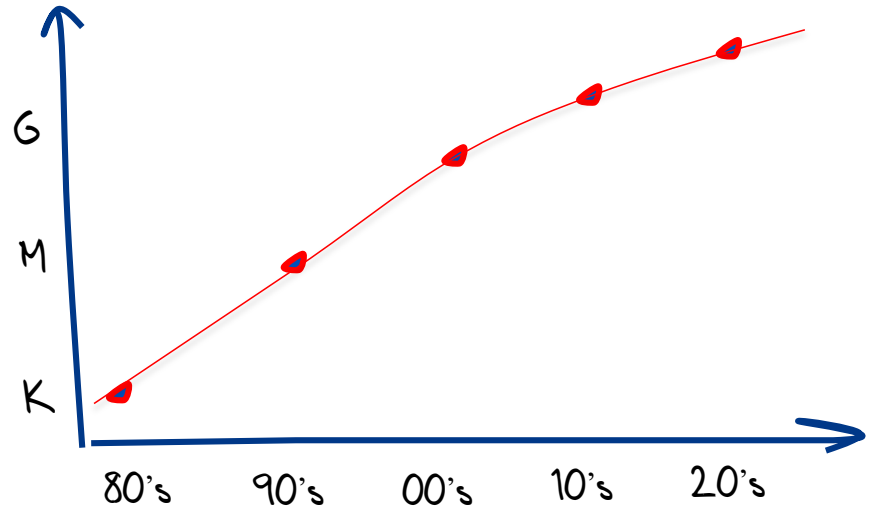
1980's
- TCP rates of Kilobits per second

1990's
- TCP rates of Megabits per second

2000's
- TCP rates of Gigabits per second

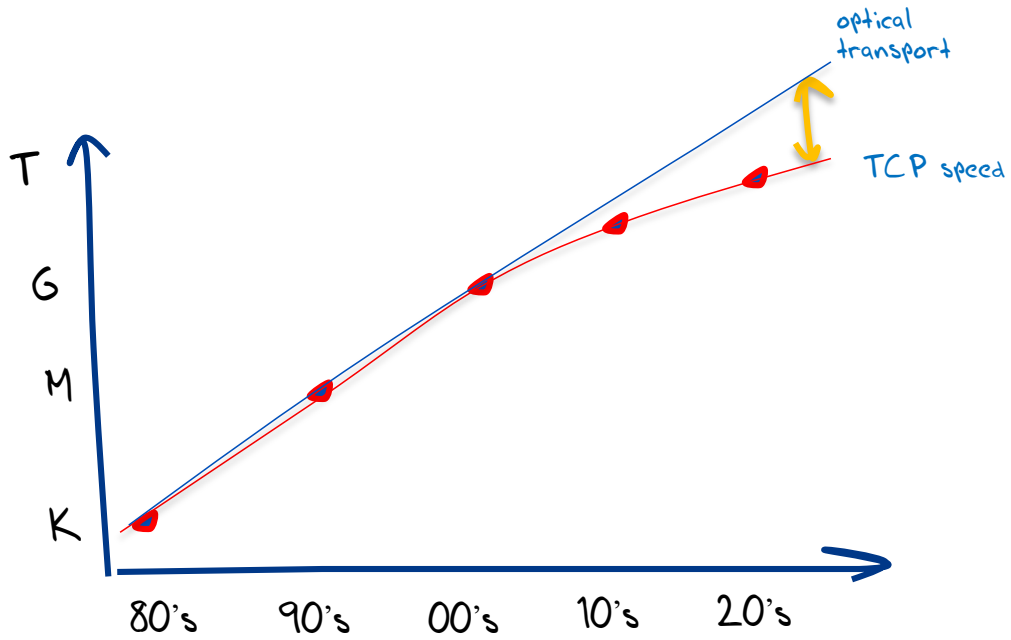
2010's
- TCP rates of tens of Gigabits per second

2020's
- TCP rates of tens of Gigabits per second



Today

- Optical transmission speeds are now edging into multi-Terabit capacity
- But peak TCP session speeds across the network are not keeping up
- Why not?



TCP is the Internet

- The Transmission Control Protocol is an end-to-end protocol that creates a reliable stream protocol from the underlying IP datagram device
- This single protocol is the “beating heart” at the core of the Internet
- TCP operates as an adaptive rate control protocol that attempts to operate **efficiently** and **fairly**

TCP Performance Objectives

To maintain an average flow which is both **Efficient** and **Fair**

Efficient:

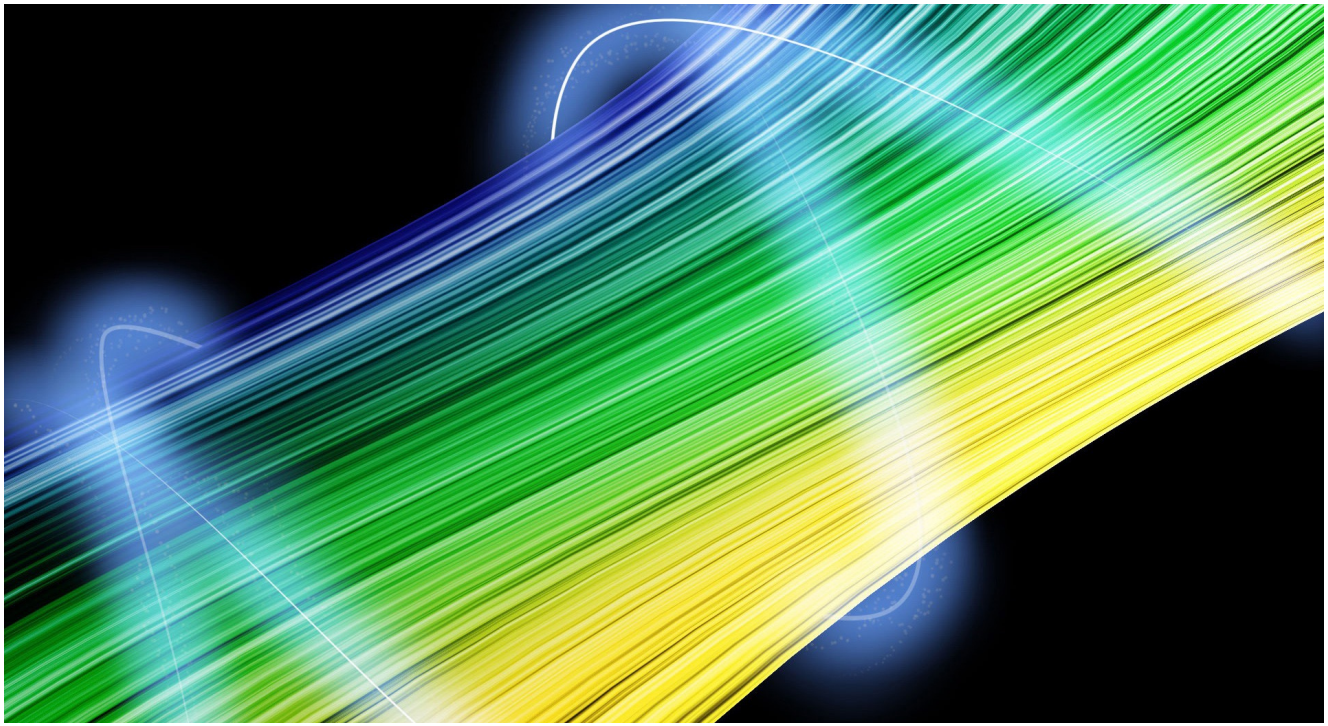
- Minimise packet loss
- Minimise packet re-ordering
- Do not leave unused path bandwidth on the table!

Fair:

- Do not crowd out other TCP sessions
- Over time, take an average 1/N of the path capacity when there are N other TCP sessions sharing the same path

It's a Flow Control process

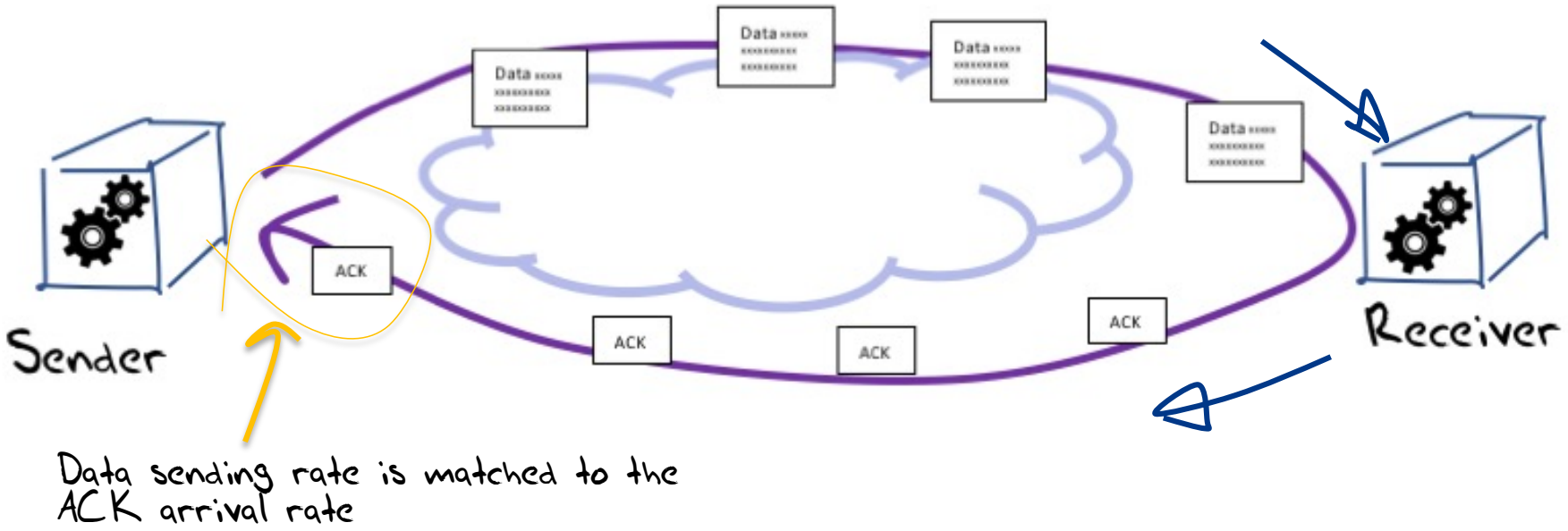
- Think of this as a multi-flow fluid dynamics problem
- Each flow has to gently exert pressure on the other flows to signal them to provide a fair share of the network, and be responsive to the pressure from all other flows



TCP Control

TCP is an ***ACK Pacing*** protocol

If the sender sends one packet each time it receives an ACK, then the sender will maintain a steady number of packets in flight within the network



TCP Control

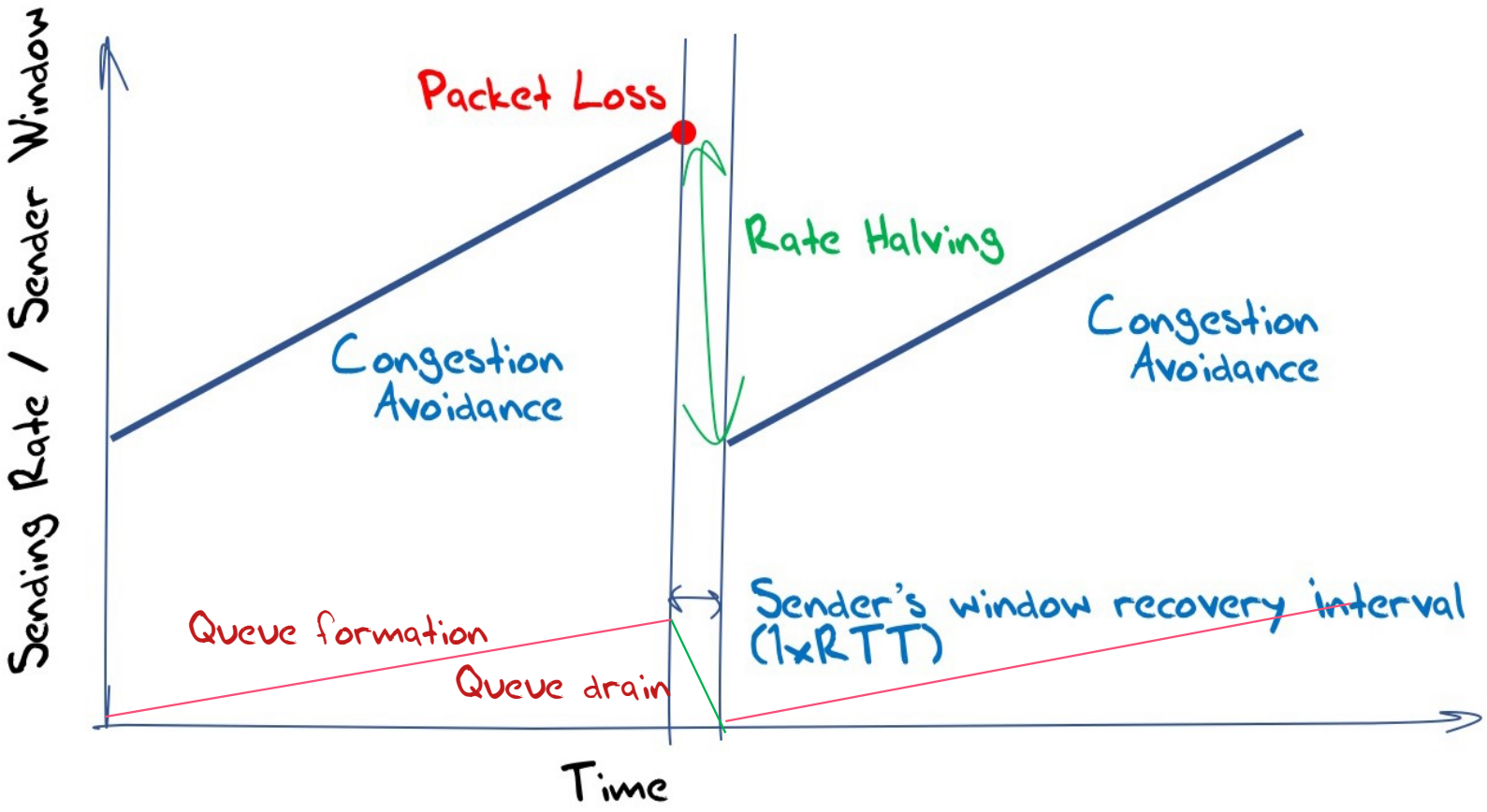
- Ideally TCP would send packets at a fair share of available network capacity. But the TCP sender has no idea what "available network capacity" means.
- So, TCP uses '**rate adaptation**' to probe into network, increasing the sending rate until it receives a signal that the sending rate is 'too fast'
- We've been experimenting with various forms of TCP **rate adaptation** for decades!

"Classic TCP" - TCP Reno

- Additive Increase Multiplicative Decrease (AIMD)
  - While there is no packet loss, increase the sending rate by one segment (MSS) each RTT interval
  - If there is packet loss (detected by duplicate ACKs) pause for 1xRTT and decrease the sending rate by 50% over the next RTT Interval by halving the sender's send window

- Start Up
  - Each RTT interval, double the sending rate
  - We call this "slow start" – probably because its anything but slow!!!

The Classic TCP Picture



Changing TCP's control algorithm

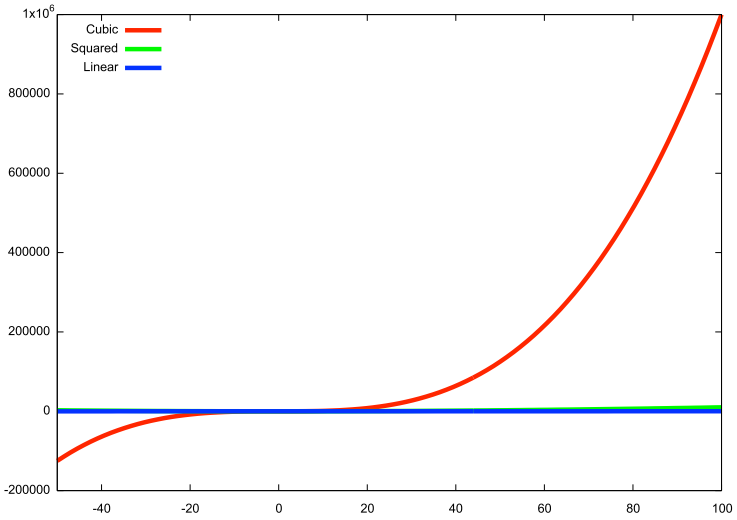
- The TCP packet format is invariant
- But the control algorithm can vary
- What defines a “fitter” control algorithm?
  - Be no less ‘aggressive’ than everyone else
  - Try to exploit opportunities that others do not
  - But don’t destroy the environment (network)

Carriage Service Challenges

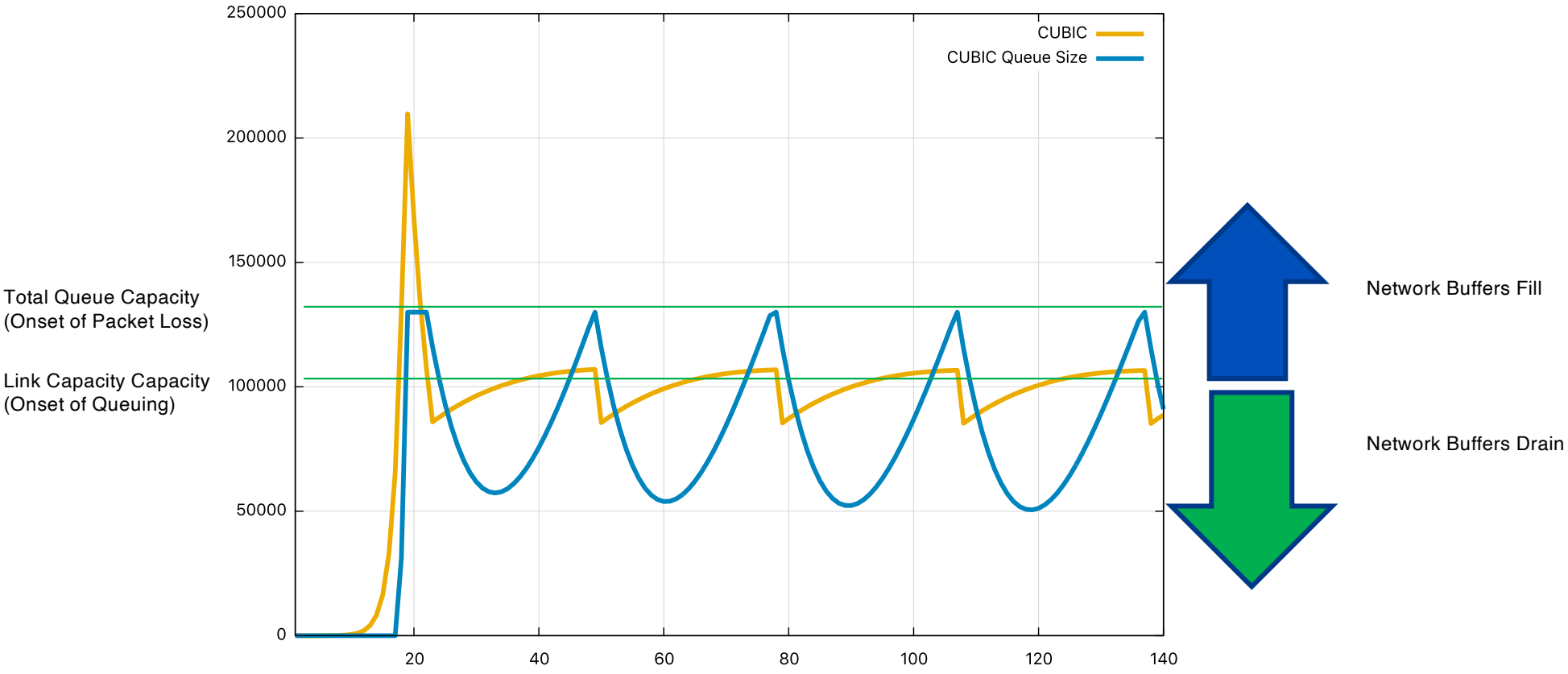
- Radio system with non-congestion loss behaviours
- LEO satellite services with very high jitter elements
- Very high bandwidth services pose a challenge to linear rate increase
- How to take advantage of equal-cost multi path frameworks
- Session “pulsing” used by streaming services

CUBIC

- CUBIC is designed to be useful for high-speed sessions while still being ‘fair’ to other sessions and also be efficient even at lower speeds
- Rather than probe in a linear manner for the sending rate that triggers packet loss, CUBIC uses a non-linear (cubic) search algorithm



CUBIC and Queue formation



CUBIC assessment

- Can react quickly to available capacity in the network
- Tends to sit for extended periods in the phase of queue formation
- Can react efficiently to long fat pipes and rapidly scale up the sending rate
- Operates in a manner that tends to exacerbate ‘buffer bloat’ conditions

And there's a whole lot more…

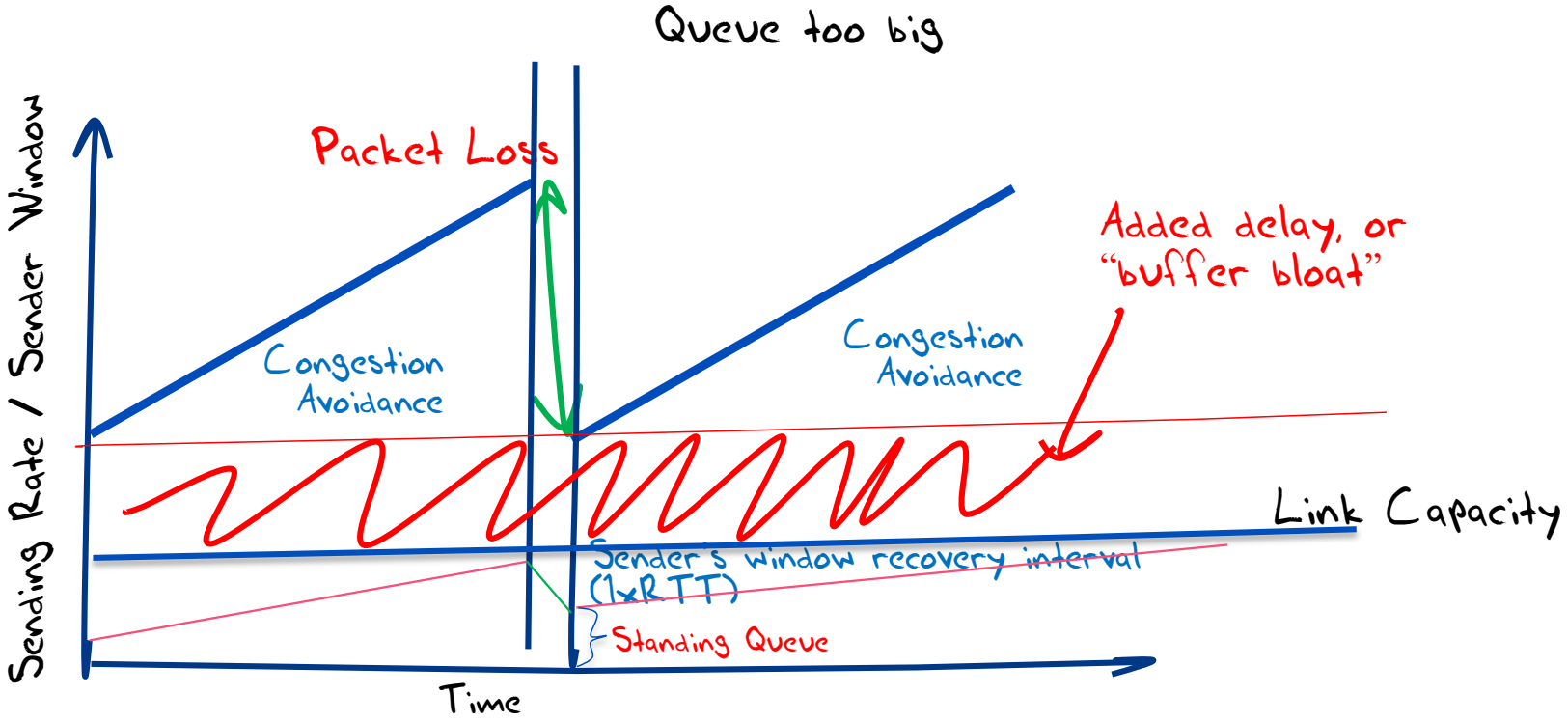
TCP Variant	*Feedback*	
RENO	Loss	AIMD
Vegas	Delay	
High Speed TCP	Loss	
BIC	Loss	Binary Increase
CUBIC	Loss	Cubic function increase - Linux-Adopted
Agile-TCP	Loss	High Speed - Low Delay
H-TCP	Loss	High Speed
Fast	Delay	Akamai Propriatary
Compound TCP	Loss/Delay	Microsoft Adopted
Westwood	Loss	Dynamic setting of Slow Start Threshold
Elastic TCP	Loss/Delay	High Speed - High Delay

TCP and Buffers - the Theory

- When a sender receives a loss signal it repairs the loss and halves it's sending window
- This will cause the sender to pause for the amount of time to drain half the outstanding data in the network (1xRTT interval)
- Ideally, this exactly matches the amount of time taken for the queue to drain
- At the time the queue is drained the sender resumes its sending (at half the rate) and the buffer has fully drained
- For this to work efficiently, the queue size for a link should equal the delay bandwidth product of the link it drives

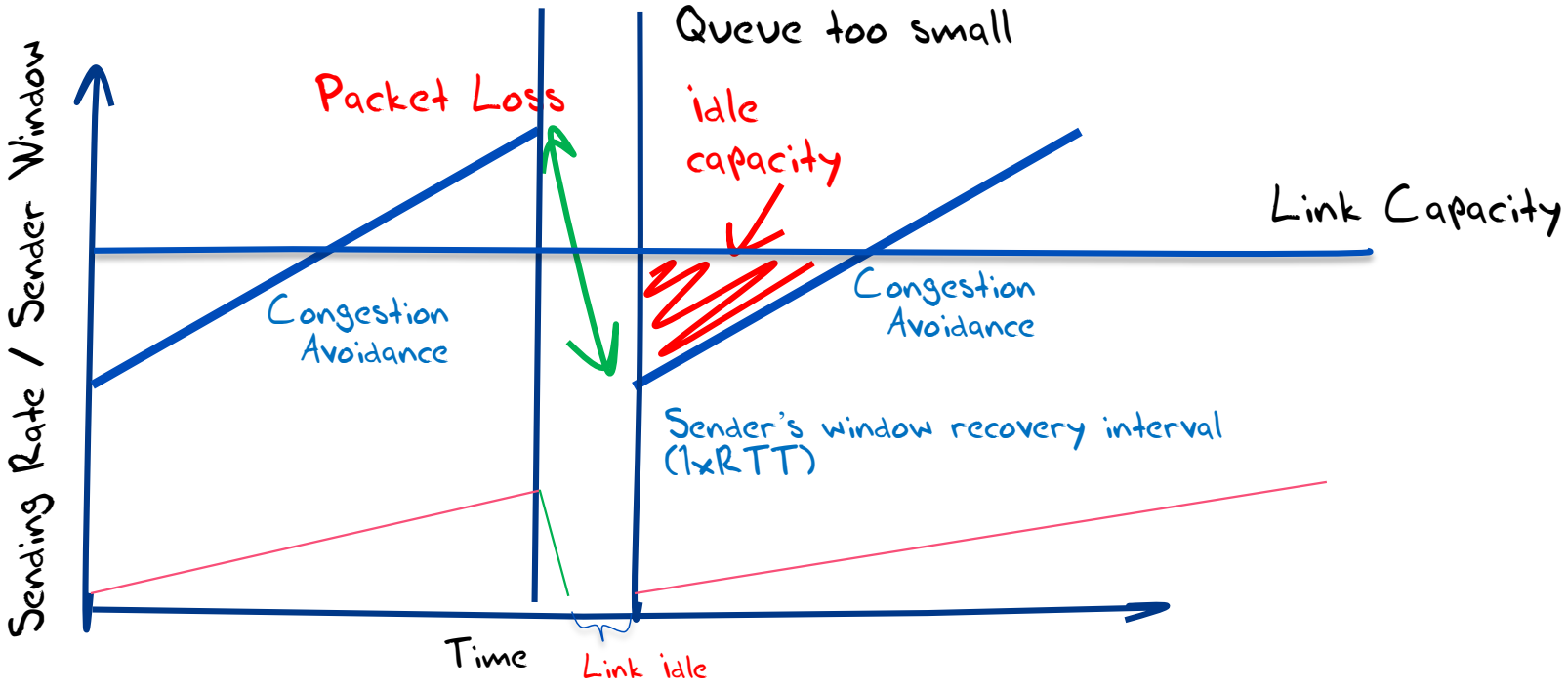
TCP and Buffers

Buffer Too Big: The queue never drains, so part of the buffer just adds delay to the connection



TCP and Buffers

Buffer Too Small: The queue drains, and the sender operates below bottleneck speed – so the link is under-used



TCP and Buffer Size

The “general” rule of thumb for configuring the buffer size in a router is:

$$\textbf{Size} = (\boldsymbol{BW} \cdot \boldsymbol{RTT})$$

Using the bandwidth and the roundtrip delay of the link being driven

TCP and Buffer Size

The “general” rule of thumb for configuring the buffer size in a router is:

e:

All this works with an assumption of a single queue and a single flow

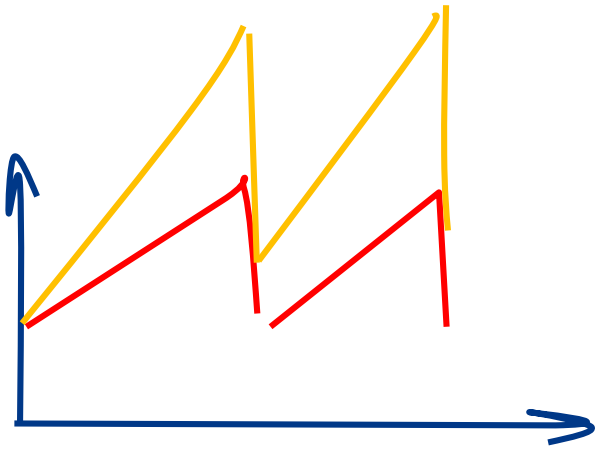
...e bandwidth and the roundtrip delay of the link being driven

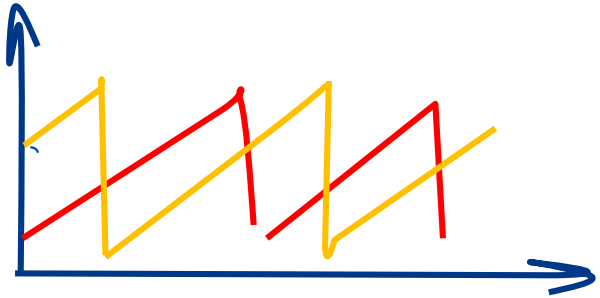
From 1 to N - Scaling Switching

- This finding of buffer size relates to a single flow through a single bottleneck resource
- What happens to buffers with more simultaneous flows and faster transmission systems?

Flow Mixing

- If 2 flows use a single buffer and they resonate precisely then the buffer still needs to be delay-bandwidth size
- If they are precisely out of phase the common buffer requirement is reduced by 25%





Smaller Buffers?

- What about the case of N de-synchronised flows?

$$\textbf{Size} = (\boldsymbol{BW} \cdot \boldsymbol{RTT}) / \sqrt{\boldsymbol{N}}$$

Assuming that the component flows manage to achieve a fair outcome of obtaining 1/N of the resource in a non-synchronised manner, then the peak buffer resource is inversely proportionate to the square root of N

("Sizing Router Buffers", Appenzeller, McKeown, Keslassy, SIGCOM'04)

The Role of Buffers

- Buffers in a network serve two essential roles:
  - smooth sender burstiness
  - Multiplexing N inputs to 1 output

Sender Pacing (Fair Queuing)

- Distribute *cwnd* data across the entire RTT interval
- Removes burst adaptation pressure on network buffers

```
net.core.default_qdisc=fq
```

Tiny Buffers?

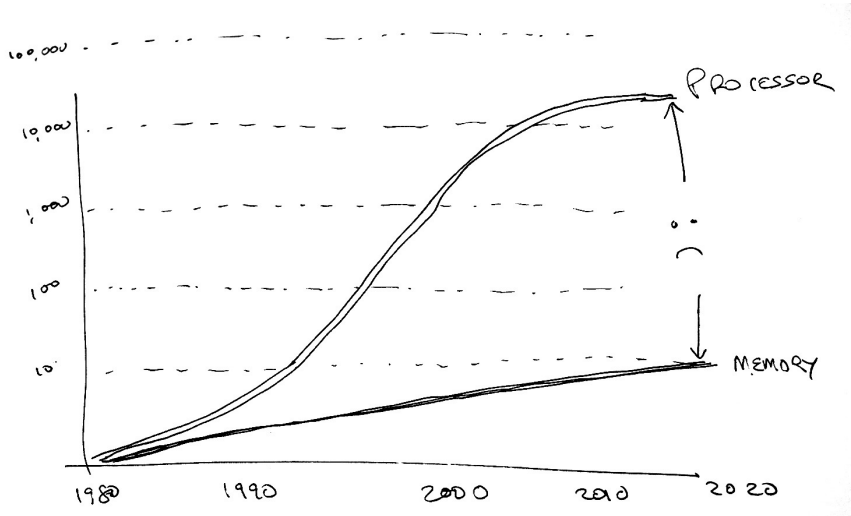
- If all senders ‘paced’ their sending to avoid bursting, and were sensitive to the formation of standing queues then we would likely have a residual multiplexing requirement for buffers where:

  B >= O(log W)

  where W is the average flow window size

Why is this important?

- Because memory speed is not scaling at the same rate as transmission or switching
- Further capacity and speed improvements in the network mandate reduced memory demands within the switch



Switching Chip Design TradeOffs

- On-Chip memory is faster, but limited to between ~16M to ~64M
- A chip design can include an interface to external memory banks but the memory interface/controller also takes up chip space and the external memory is slower

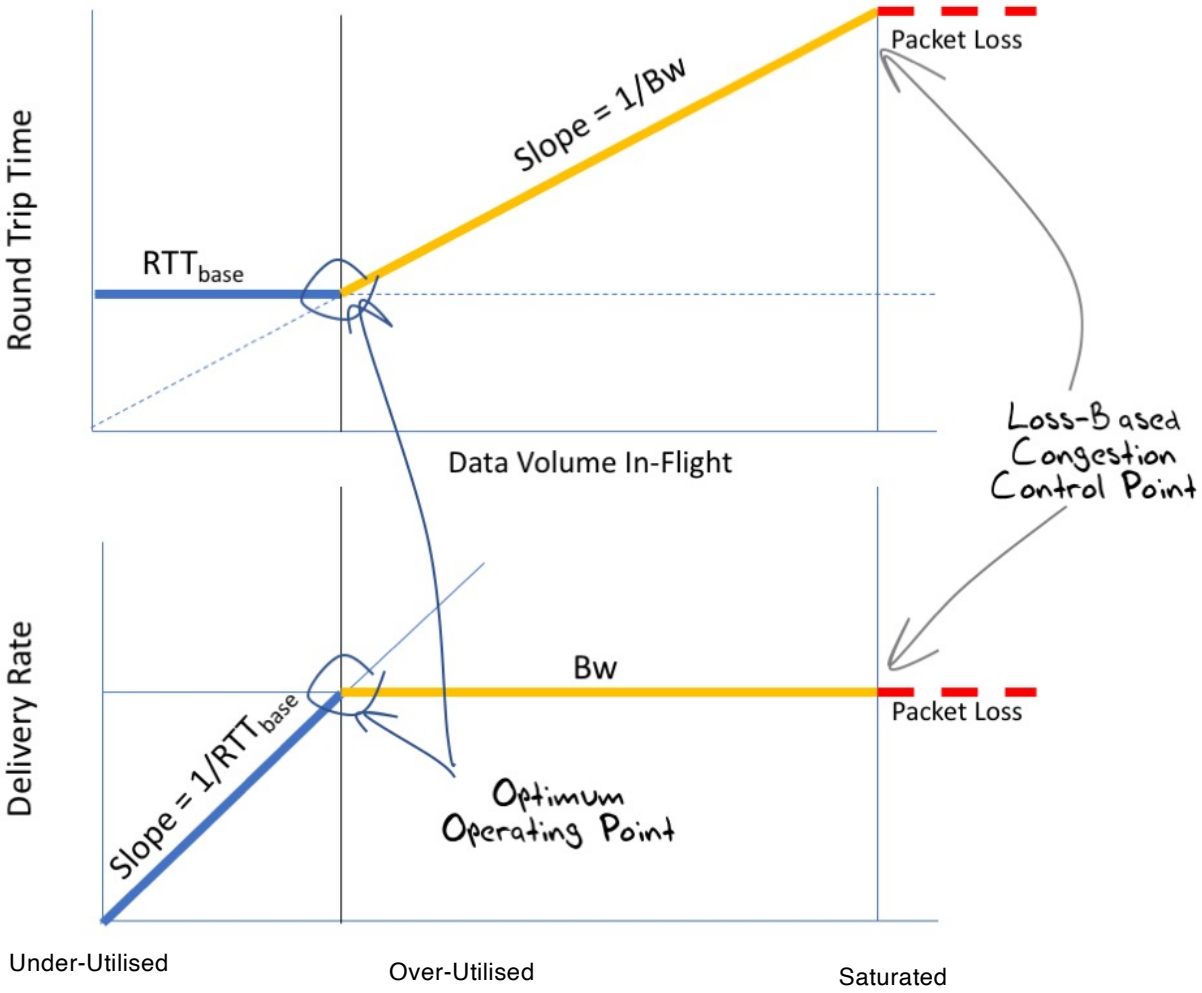
- Between 20% to 60% of switch chip real estate is devoted to memory / memory control
- Small memory buffers in switch design allows for larger switch fabric implementations on the chip

Optimising Flow State

- There are three ‘states’ of flow management:
  - **Under-Utilised** – where the flow rate is below the link capacity and no queues form
  - **Over-Utilised** – where the flow rate is greater that the link capacity and queues form
  - **Saturated** – where the queue is filled and packet loss occurs

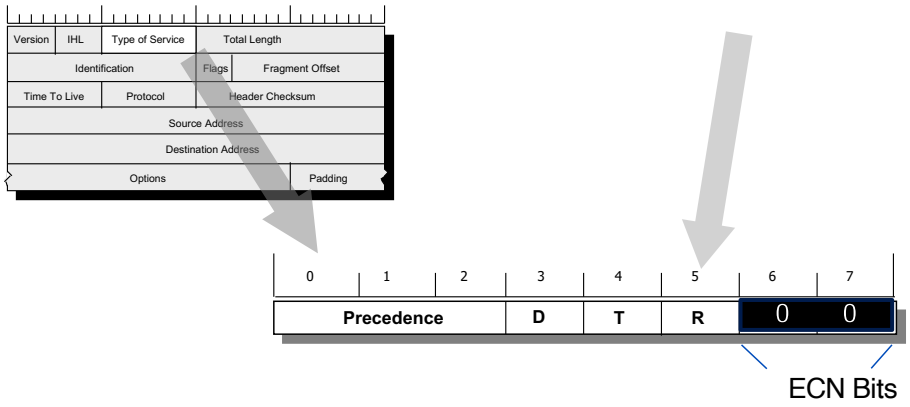
- Loss-based control systems probe upward to the Saturated point, and back off quickly to what they guess is the Under-Utilised state in order to the let the queues drain
- But the optimal operational point for any flow is at the point of state change from Under to Over-utilised, not at the Saturated point

RTT and Delivery Rate with Queuing



How to detect the onset of queuing?

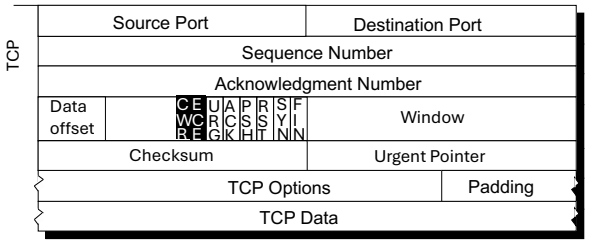
- By getting the network's routers to report when queues are forming!



IP

0 0 – Non-ECN Capable Transport
0 1 – ECN Capable Transport
1 0 - ECN Capable Transport
1 1 – Congestion Experienced

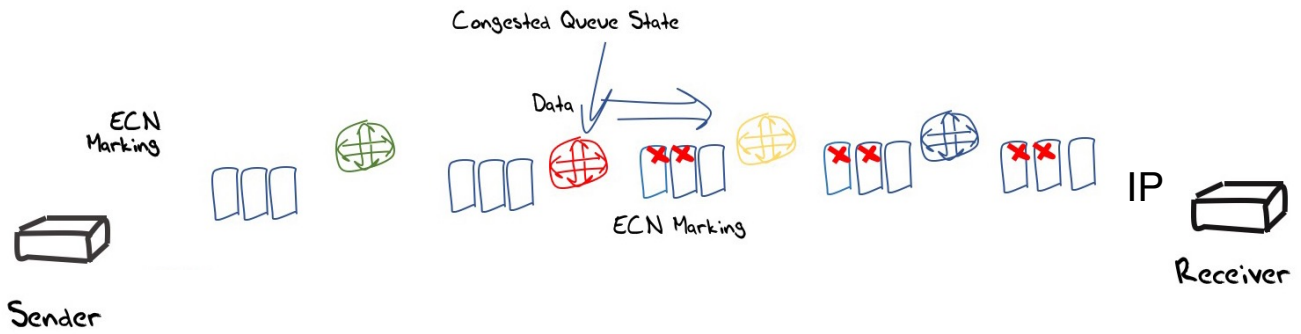
TCP



ECE – receiver back to sender – CE received
CWR – sender to receiver – Congestion Window Reduced

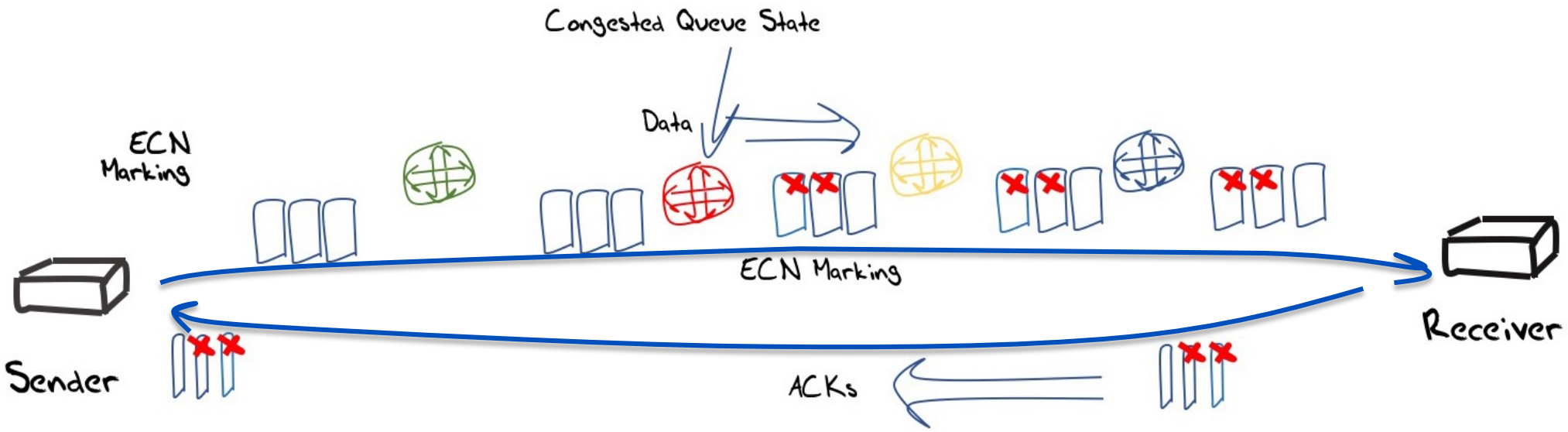
SYN+ECE+CWR – ECN capable on session start
SYN+ACK+ECE – ECN capable response

ECN Control Loop



- A router “marks” IP packets at the onset of queue formation with a bit signal
- The Receiver echoes this bit up into the transport protocol reverse flow
- The sender reduces its sending window size (and notifies the receiver that it was performed this window reduction)

Explicit Congestion Notification



Explicit Congestion Notification

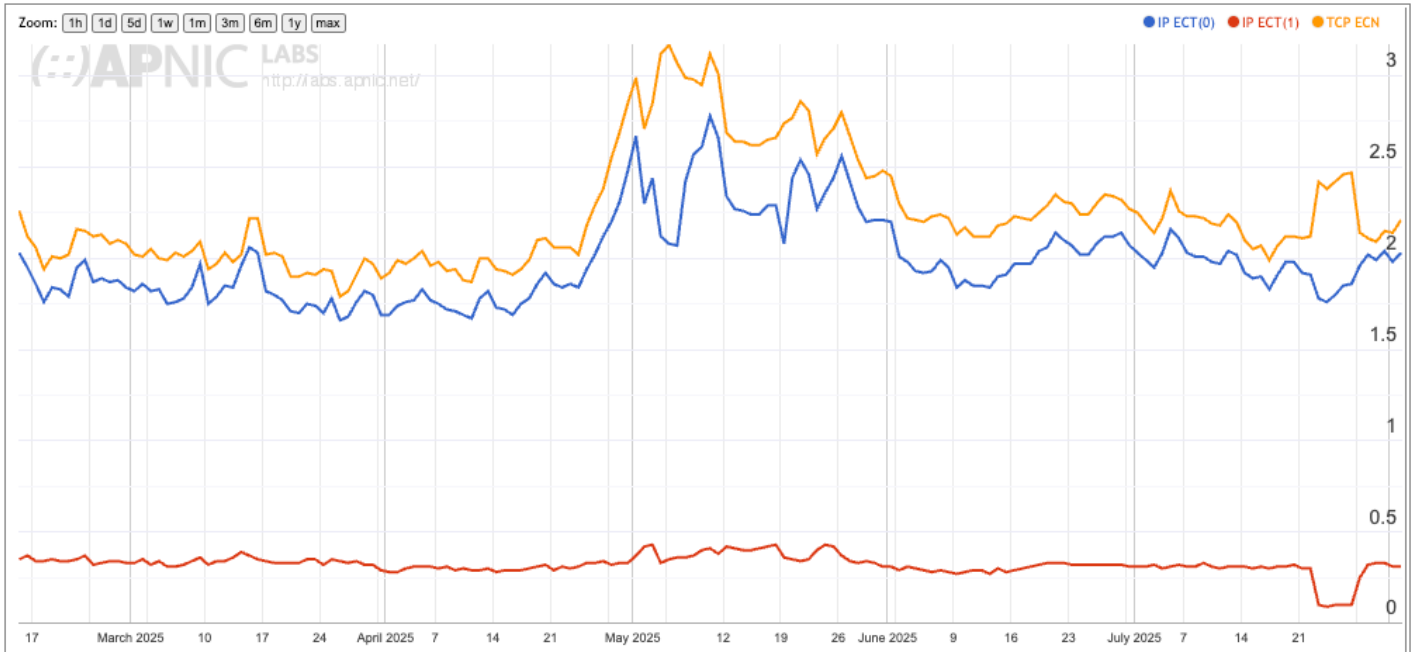
- Sparse signal (single bit)
- Both hosts and routers need to be ECN aware
- IP level marking requires end host protocol surgery at both ends:
    - Receivers need to reflect ECN bits
    - Senders need to pass IP CE up to the TCP session to signal a need to reduce the sending rate

ECN Issues

- It would be good if **everyone** did it!
  - That probably means every router and every end host running TCP (and QUIC)
  - How are we doing in deploying ECN?

ECN Issues

ECN Use in World (XA)



How to detect the onset of queuing?

- By getting the network's routers to report when queues are forming!

OR

- By detecting the onset of queue-based delays in the measured RTT

Flow Control Evolution

- Current flow control systems make small continual adjustments every RTT interval and a massive adjustment at irregular intervals
  - As the flow rate increases the CA adjustments of 1 segment per RTT become too small
  - Rate halving is a massive response

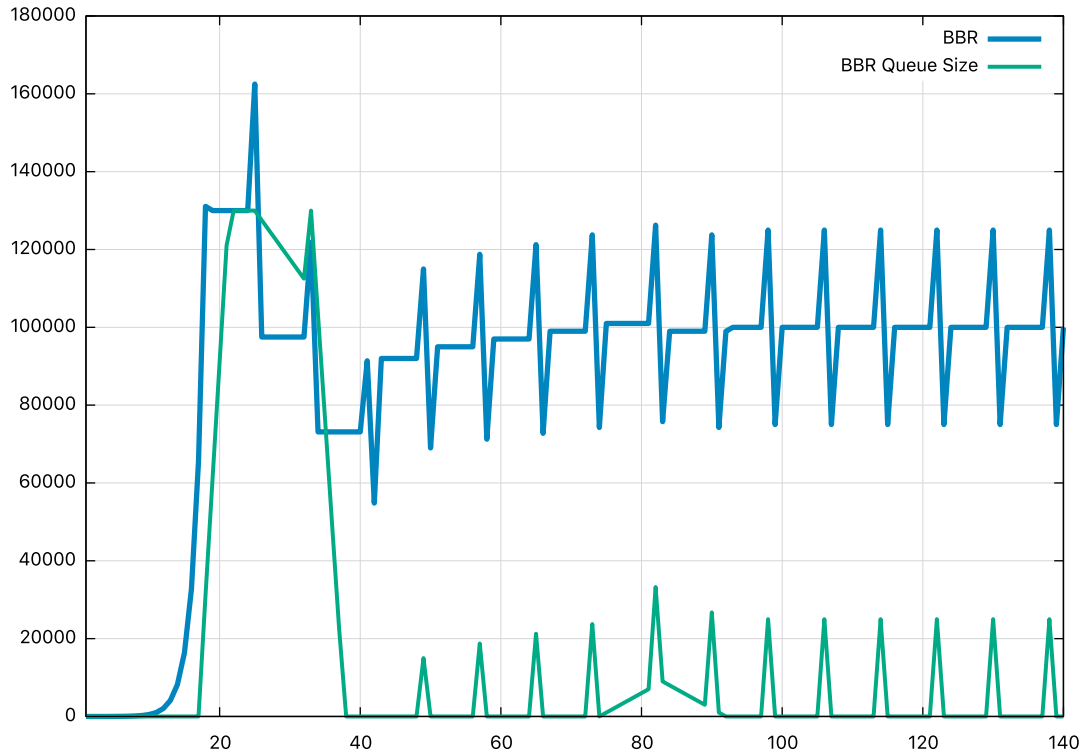
OR

- We could use a system that only made periodic adjustments every *n* RTT intervals based on delay probing
  - And set the adjustment to be proportionate to the current flow rate

BBR Design Principles

- Pace the sending packets to avoid the need for network buffer rate adaptation
- Probe the path capacity only intermittently (every *8th* RTT)
- Probe the path capacity by increasing the sending rate by *25%* for an RTT interval and then drop the rate to drain the queue:
  - If the RTT of the probe interval equals the RTT of the previous state, then there is available path bandwidth that could be utilised
  - If the RTT of the probe rises, then the path is likely to be at the onset of queuing and no further path bandwidth is available
- Do not alter the path bandwidth estimate in response to packet loss!

Idealised BBR profile

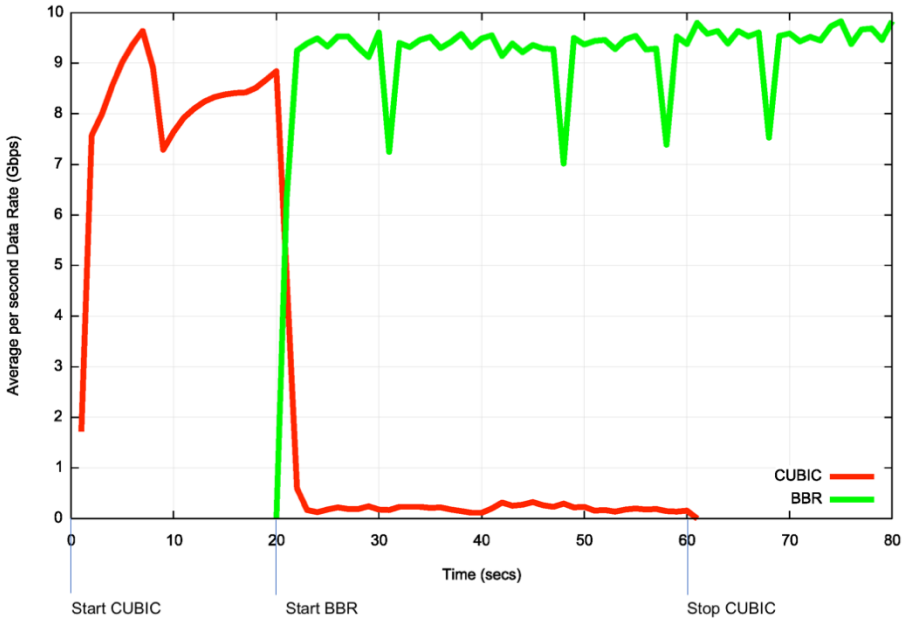


sending rate

network queues

BBR Politeness?

- BBR will probably not constantly pull back when simultaneous loss-based protocols exert pressure on the path's queues
- BBR tries to make minimal demands on the queue size, and does not rely on a large dynamic range of queue occupancy during a flow

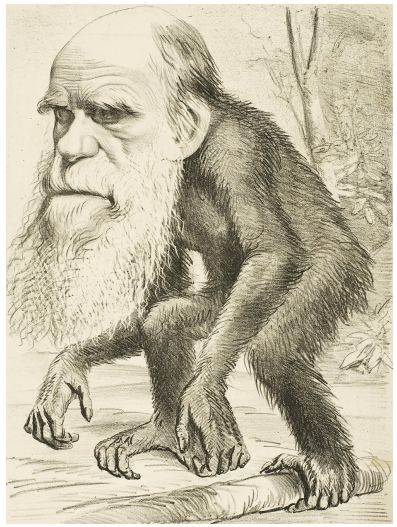


Our Environment…

It's a pretty comprehensive mess:

- A diverse mix of e-2-e TCP control protocols
  CUBIC, NewRENO, LEDBAT, Fast, BBR, Compound
- A mix of traffic models
  Buffer-filling streamers, flash bursts, bulk data
- A mix of active queue management disciplines
  RED, WRED, CODEL, FQ, none
- A mix of media
  Wire line, mobile, WiFi
- A mix of buffer size deployments
- Sporadic ECN marking

Protocol Darwinism?



What “wins” in this diverse environment?

- ***Efficiency*** is perhaps more critical than ***fairness*** as a “survival fitness” strategy
- I suspect that protocols that make minimal assumptions about the network will be more robust than those that require certain network characteristics to operate efficiently
- Protocols that operate with regular feedback mechanisms appear to be more robust than irregular “shock” treatment protocols

What is all this telling us?

- We actually don't know all that much about fine-grained behaviour of large-scale high capacity switching systems.
- Some of our cherished assumptions about network design may be mistaken
- Moving large data sets over very high-speed networks requires an entirely different approach to what we are doing today

The Internet still contains a large set of important unsolved problems!

That's it!