On the Impact of Network Protocols and Architectures on Network Performance

Michele Pagano

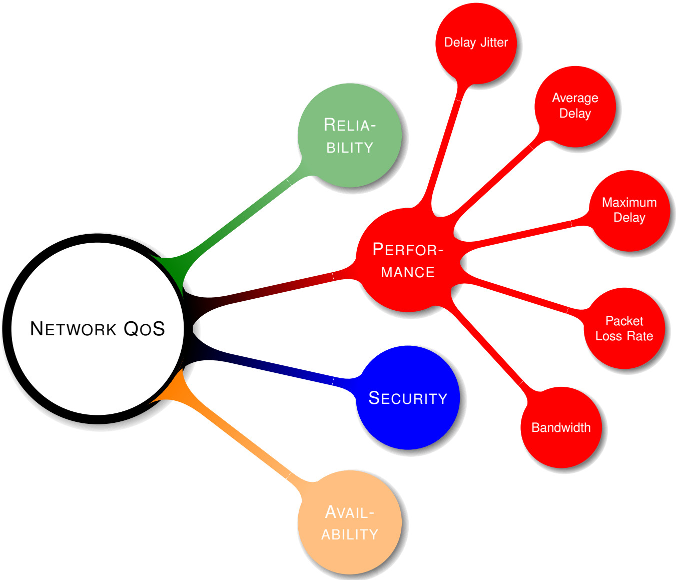
Dept. of Information Engineering, University of Pisa, Italy
michele.pagano@unipi.it

ITMM 2022
25–29 October 2022

Overview

1. Network Performance
2. Traffic Modelling
3. Routers
4. Effect of the Protocols
5. New Network Solutions
6. Conclusions

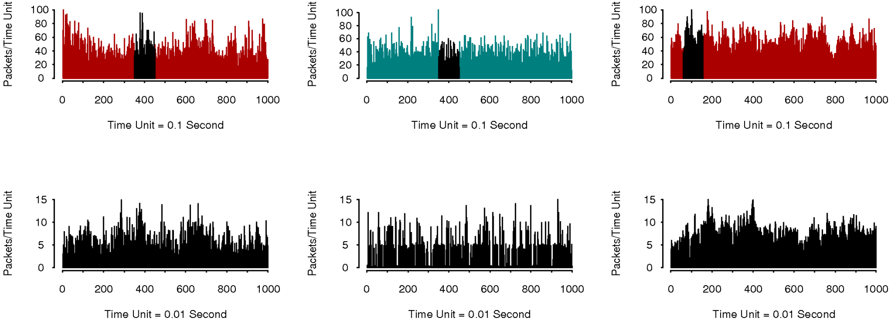
Quality of Service (QoS)



Network Performance: the *Big Picture*

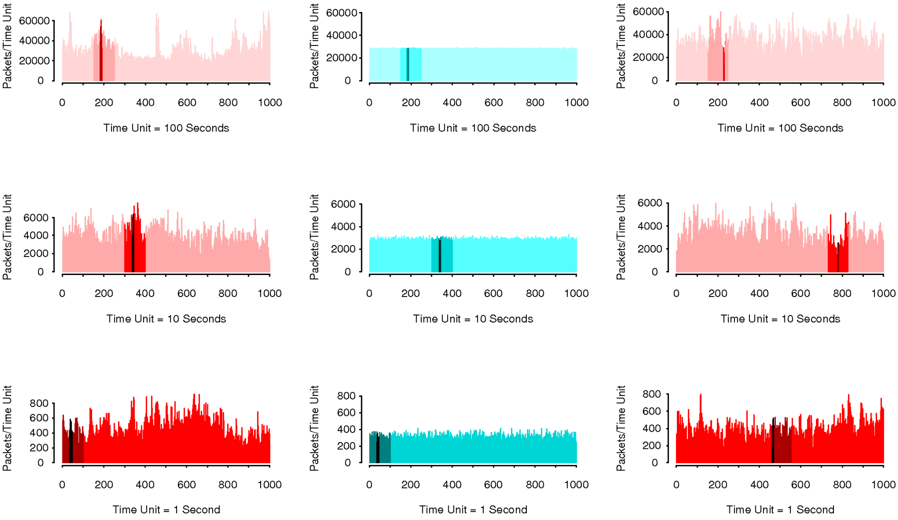
Traffic Features

Traffic Self–Similarity

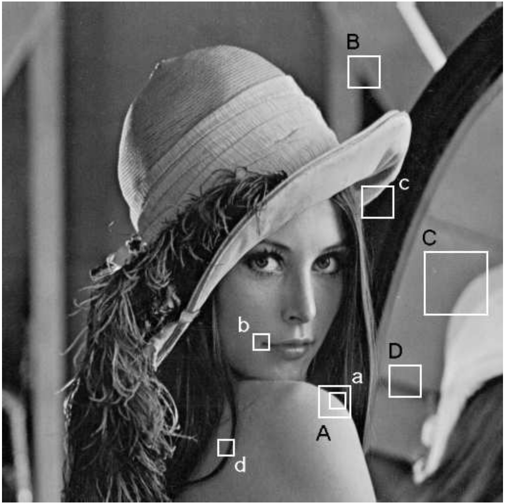


M. S. Taqqu, W. Willinger, R. Sherman *Proof of a fundamental result in self-similar traffic modeling*, Computer communication review, 1997

Traffic Self–Similarity



Fractals Everywhere!



Statistical Self–Similarity

Intuitive idea

- A dilated portion of the sample path of a self–similar process cannot be (statistically) distinguished from the whole

Self–similarity for continuous time processes

- Let $(Y_t)_t$ be a continuous time process ($t \in \mathbb{R}$)
- $(Y_t)_t$ is self–similar with self–similarity parameter H if and only if

$$c^{-H} Y_{ct} \stackrel{(d)}{=} Y_t \qquad \forall\, c > 0$$

i.e., if for any $k \geq 1$, for any $t_1, t_2, \ldots, t_k \in \mathbb{R}$ and for any $c > 0$

$$\left(Y_{ct_1}, Y_{ct_2}, \ldots, Y_{ct_k}\right) \quad \text{and} \quad \left(c^H Y_{t_1}, c^H Y_{t_2}, \ldots, c^H Y_{t_k}\right)$$

have the same distribution

Properties of self–similar processes

- If $c^{-H} Y_{ct} \stackrel{(d)}{=} Y_t$ with $H \neq 0 \quad \Rightarrow \quad (Y_t)_t$ is not stationary
  - Indeed, stationarity requires that $Y_{ct} \stackrel{(d)}{=} Y_t$
  - For the purpose of modelling time series that *look stationary*, it is possible to consider the stationary increments of a self–similar process: $X_t = Y_t - Y_{t-1}$

- Let Y_t be a self–similar process with
  - $H > 0$
  - $\mathbb{E} Y_t = 0$
  - $Y_0 = 0$ with probability 1

$\Rightarrow$ By definition of self–similarity, its covariance function is

$$r_Y(t, s) = \frac{\sigma^2}{2}\left[|t|^{2H} - |t-s|^{2H} + |s|^{2H}\right]$$

where $\sigma^2 = \mathbb{E}\left[(Y_t - Y_{t-1})^2\right]$

Stationary increments of a self–similar process

- The increment process $X_n = Y_n - Y_{n-1}$ is a second order (discrete time) stationary process

Aggregated process

$$X \stackrel{(d)}{=} m^{1-H} X^{(m)} \qquad \forall\, m \in \mathbb{N}$$

where

$$X_k^{(n)} = \frac{1}{n} \sum_{i=(k-1)n+1}^{kn} X_i$$

Autocorrelation function of X_n

$$\rho(k) \triangleq \frac{r(k)}{r(0)} = \frac{1}{2}\left[|k+1|^{2H} - 2\,|k|^{2H} + |k-1|^{2H}\right]$$

$\Rightarrow$ If $H = 1/2$, the increments are uncorrelated: $\rho(k) = \delta_0$

Asymptotic properties of X_n (for $H \neq 1/2$)

$$\lim_{k\to\infty} \frac{\rho(k)}{k^{2H-2}} = H(2H-1)$$

$\rho(k) \sim k^{-\alpha}$ as $k \to \infty$ where $\alpha = 2 - 2H$

Short Range Dependence (SRD)

- If $0 < H < 1/2 \quad \Rightarrow \quad X_n$ has SRD (actually $\sum \rho_k = 0$)

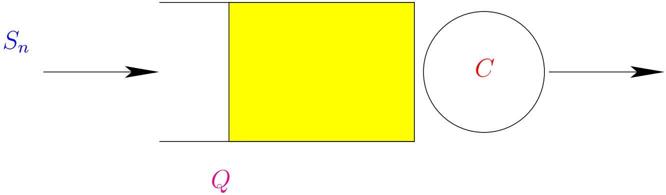
Long Range Dependence (LRD)

- If $1/2 < H < 1$, then $0 < \alpha < 1 \quad \Rightarrow \quad X_n$ has LRD

$$\sum_k |\rho(k)| = \infty \quad \text{and} \quad \mathrm{Var}\, X^{(n)} \sim n^{-\alpha}$$

- If $H = 1 \quad \Rightarrow \quad \rho(k) = 1 \quad \forall k$
- If $H > 1 \quad \Rightarrow \quad \rho(k)$ diverges

Steady–state overflow probability



- Single server queue in discrete time
- Deterministic service rate C
- Cumulative arrival process $S_n = \sum_{k=0}^{n} A_k$
- Limiting cumulant generating function $\Lambda(\theta)$
- Infinite buffer
- Lindley's recursion: $Q_n \; = \; (Q_{n-1} + A_n - C)^+$

Overflow probability

SRD traffic

$$\mathbb{P}(Q > b) \approx e^{-\delta b}$$

Effective Bandwidth Approximation

$$\delta = \inf_{\tau > 0} \tau \Lambda^*(C + \frac{1}{\tau})$$

$$\Lambda(\theta) = \lim_{n\to\infty} \frac{1}{n} \log \mathbb{E} e^{\theta S_n}$$

LRD traffic

$$\mathbb{P}(Q > b) \approx e^{-\delta b^{2-2H}}$$

$$\delta = \inf_{\tau > 0} \tau^{2-2H} \Lambda^*(C + \frac{1}{\tau})$$

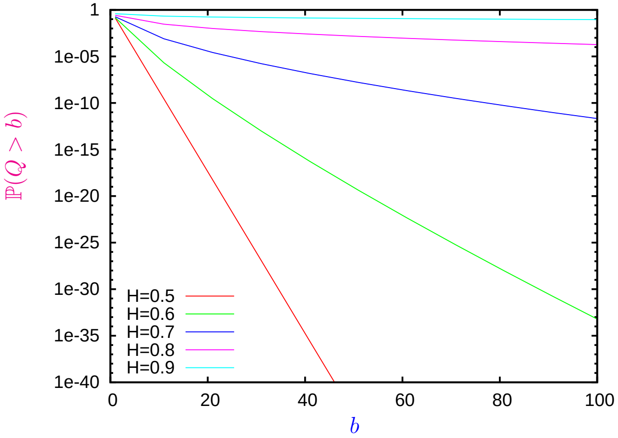
$$\Lambda(\theta) = \lim_{n\to\infty} \frac{1}{a(n)} \log \mathbb{E} e^{\frac{\nu_n}{n}\theta S_n}$$

$$\nu_n = n^2/\text{Var}[S_n] = n^{2(1-H)}$$

Fenchel-Legendre transform

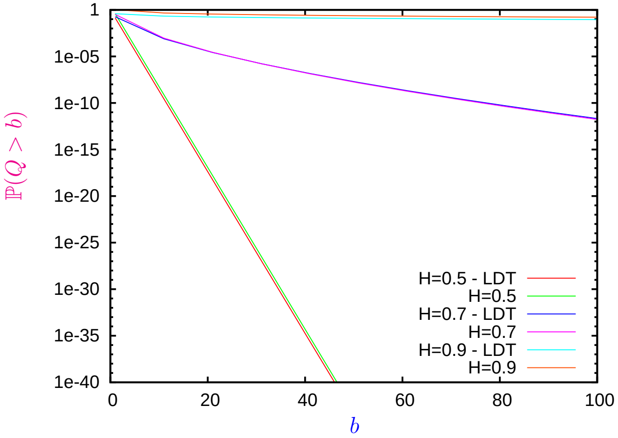
$$\Lambda^*(x) \triangleq \sup_{\theta\in\mathbb{R}} (x\theta - \Lambda(\theta))$$

Fractional Brownian traffic – LDT asymptotics



$$\lim_{b\to\infty} \frac{\ln \mathbb{P}(Q > b)}{\left(-\frac{1}{2}\left(\frac{b}{1-H}\right)^{2-2H}\left(\frac{r}{H}\right)^{2H}\right)} = 1 \qquad \text{where} \quad r = C - \mathbb{E}A_n$$

Fractional Brownian traffic – More precise results



$$\lim_{b\to\infty} \frac{\mathbb{P}(Q > b)}{b^{2H-3+1/H} \exp\left(-\frac{1}{2}\left(\frac{b}{1-H}\right)^{2-2H}\left(\frac{r}{H}\right)^{2H}\right)} = \frac{\alpha(H,r)}{\sqrt{2\pi}\beta(H,r)}$$

Routers

Network Performance
New Network Solutions
Middle-boxes
SDN
CDN
BBR TCP
Caching
QUIC and HTTP 3.0
TLS 1.3
Protocols
Layering
TCP Three-Way Handshake
TCP Congestion Control
Multiple Access Protocols
Traffic Features
Long-Range Dependence
Heavy-Tails
Self-Similar Processes
Correlation Horizon
Routers
Node Delay
Queue Size
Scheduling discipline
Active Queue Management

Scheduler

- A key component for QoS enabling networks
- Selects which next packet to transmit, and when, on the basis of some expected performance
- Different scheduling algorithms have been devised, which exhibit different fairness and latency properties at different worst-case per-packet complexity

Scheduling Algorithms

- WFQ (Weighted Fair Queueing) or PGPS (Packetized GPS)
- WF^2Q (Worst-case Fair Weighted Fair Queueing)
- SCFQ (Self Clocked Fair Queueing)
- WRR (Weighted Round Robin)
- DRR (Deficit Round Robin)
- MDRR (Modified Deficit Round Robin)

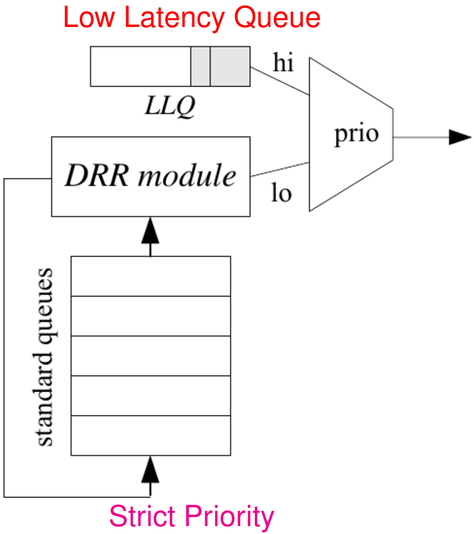
Deficit Round Robin

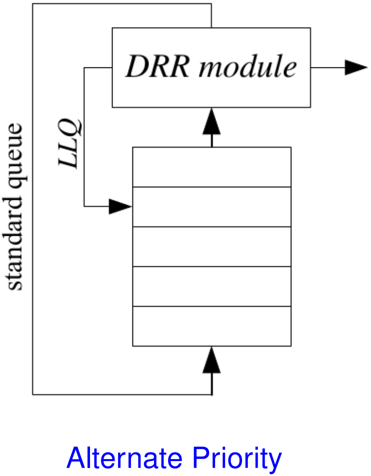
- Achieves $O(1)$ per–packet complexity
- Each queue i is characterized by
  - A quantum of ϕ_i bits: the quantity of packets that queue i should ideally transmit during a round
  - A deficit variable Δ_i
- A backlogged queue is allowed to transmit a burst of packets of an overall length not exceeding $\phi_i + \Delta_i$
- The deficit variable Δ_i is managed as follows
  - Reset to zero when the queue is not backlogged
  - Increased by ϕ_i when the queue is selected for service
  - Decreased by the packet length when a packet is transmitted
- The minimum guaranteed rate of queue i is

$$R_i = \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} C$$

where C is the link capacity

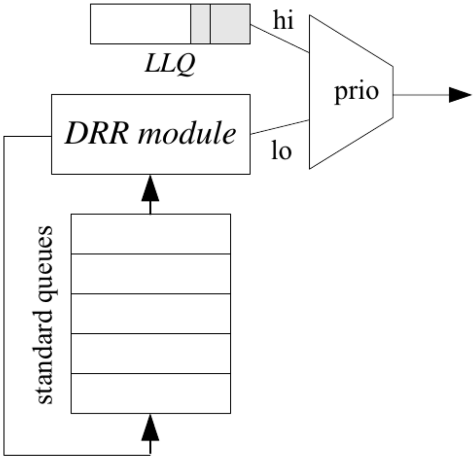
C-MDRR (Cisco 12000 routers)





C-MDRR (Cisco 12000 routers)

Low Latency Queue



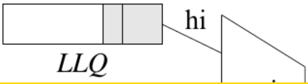
Strict Priority

Strict priority mode

- The LLQ is always serviced in exhaustive, non preemptive priority mode
- The other queues are serviced cyclically, as in DRR, whenever the LLQ is empty
- A standard queue can have its service turn interrupted by the arrival of a packet in the LLQ

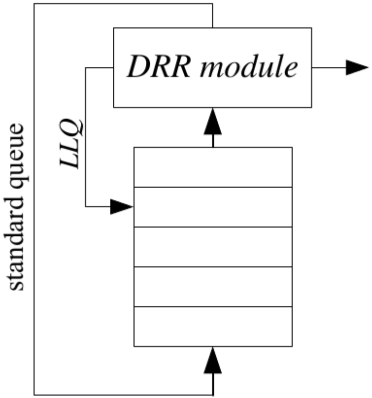
C-MDRR (Cisco 12000 routers)

Low Latency Queue



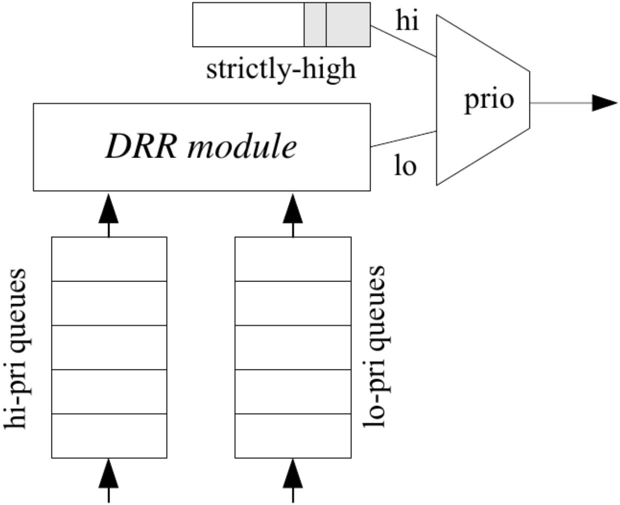
Alternate priority mode

- The LLQ is assigned a quantum
- Whenever non empty, the LLQ is serviced for its whole quantum every second service turn
- If N standard queues (SQ$_1$, . . . , SQ$_N$) are defined, and all queues (including the LLQ) are backlogged, the service order in a round is: LLQ, SQ$_1$, LLQ, SQ$_2$, . . . , LLQ, SQ$_N$



Alternate Priority

J-MDRR (Juniper M–Series)



J-MDRR (Juniper M–Series)

- A queue can have a low, high or strictly-high priority
- A strictly-high priority queue is serviced whenever it is non empty, like the LLQ in strict priority mode in C-MDRR
- Both high and low priority queues are serviced for a quantum on each round, and they carry on their deficit to the subsequent round if they are still backlogged
- In a round, the active list of high-priority queues is serviced first, until either it is empty or all high-priority queues have been serviced for their quantum
- Low-priority queues are serviced next
- Unlike C-MDRR, low and high priority queues transmit one packet at a time
- A queue can be serviced more than once per round

Random Early Detection (RED)

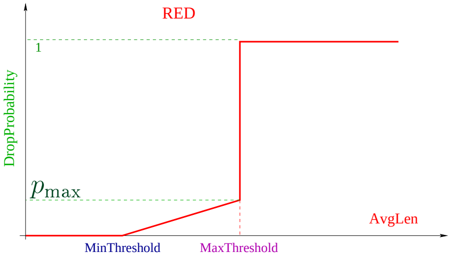
- Router–centric congestion avoidance approach
- Early Drop: rather than wait for queue to become full, drop each arriving packet with *some drop probability* whenever the queue length exceeds *some drop level* $\Rightarrow$ Active queue management
- Notification is implicit: just drop the packet (TCP will timeout)
- ECN-RED: notification could make explicit by marking the packet (ECN – Explicit Congestion Notification)

The decision is based on the *average queue length*

AvgLen = (1 - w) AvgLen + w SampleLen $\qquad 0 < w < 1$

- Weighted running average
- SampleLen is the queue length each time a packet arrives

RED Drop Probability curve



Dropping mechanism based on *two queue length thresholds*

if (AvgLen $\leq$ MinThreshold) then enqueue the packet

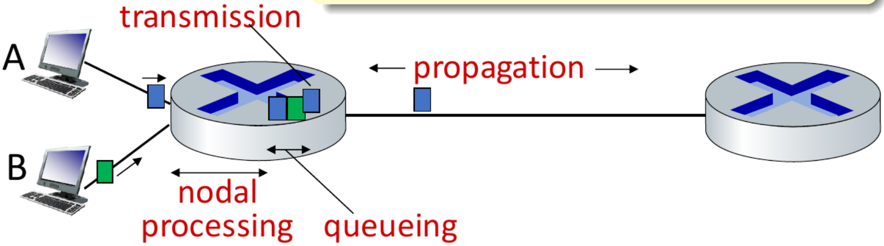
if (MinThreshold $<$ AvgLen $<$ MaxThreshold) then

calculate probability P
drop arriving packet with probability P

if (MaxThreshold $\leq$ AvgLen) then *drop arriving packet*

Nodal delay

$$d_{\text{nodal}} = d_{\text{prop}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{proc}}$$



Processing delay d_{proc}

- The time required to examine the packet's header and determine where to direct the packet
- It can also include the time needed to check for bit–level errors
- In high-speed routers, typically on the order of μs or less

Nodal delay

Queueing delay d_{queue}

- It is the delay between the time a packet is assigned to a queue for transmission and the time it starts being transmitted
- The queueing delays can vary significantly from packet to packet and can be on the order of μs to ms in practice

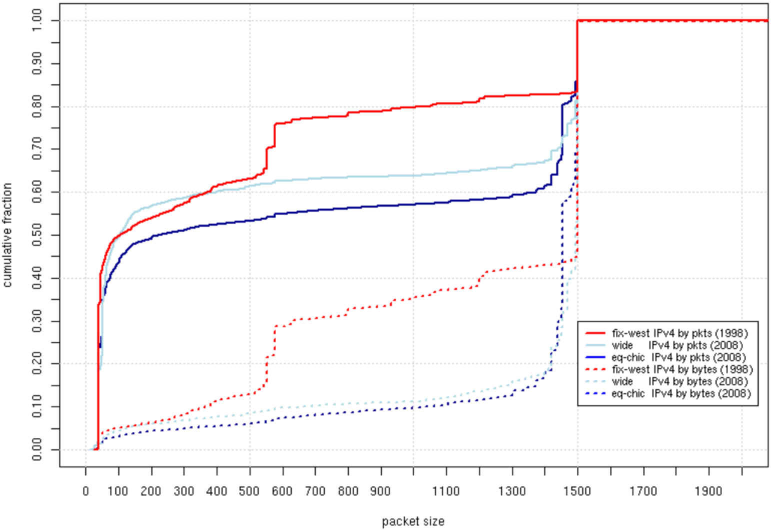
Transmission delay d_{trans}

- It is the delay between the times that the first and the last bits of the packet are transmitted
- Transmission delays are typically on the order of μs to ms (hundreds of ms in case of low-speed dial-up modem links)

$$d_{\text{trans}} = L/R$$

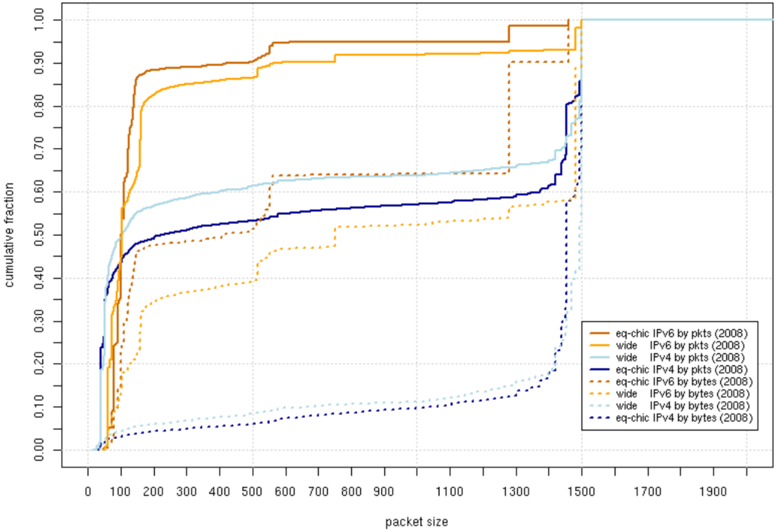
where L is the packet length and R is the link transmission rate

Cumulative IPv4 packet size distribution



CAIDA - Cooperative Association for Internet Data Analysis

Cumulative IPv4 and IPv6 packet size distribution



CAIDA - Cooperative Association for Internet Data Analysis

Nodal delay

Propagation delay d_{prop}

- It is the delay between the time a bit is transmitted at the head node of the link and the time it is received at the tail node
- The bits propagate at the propagation speed s of the link, which depends on the physical medium and is in the range of $2 \cdot 10^8\ m/s\ -\ 3 \cdot 10^8\ m/s$
- d_{prop} can range from a couple of μs (two routers on the same university campus) to hundreds of ms (two routers interconnected by a geostationary satellite link)

$$d_{\text{prop}} = d/s$$

where d is the distance between the two routers

$$d_{\text{nodal}} = d_{\text{prop}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{proc}}$$

Effect of the Protocols

Transmission Control Protocol (TCP)

- TCP is based on concepts first described in *V.Cerf, R. Kahn, "A Protocol for Packet Network Intercommunication", IEEE TCOM, May 1974"*
- In IETF world originally defined in RFC 793 (September 1981)

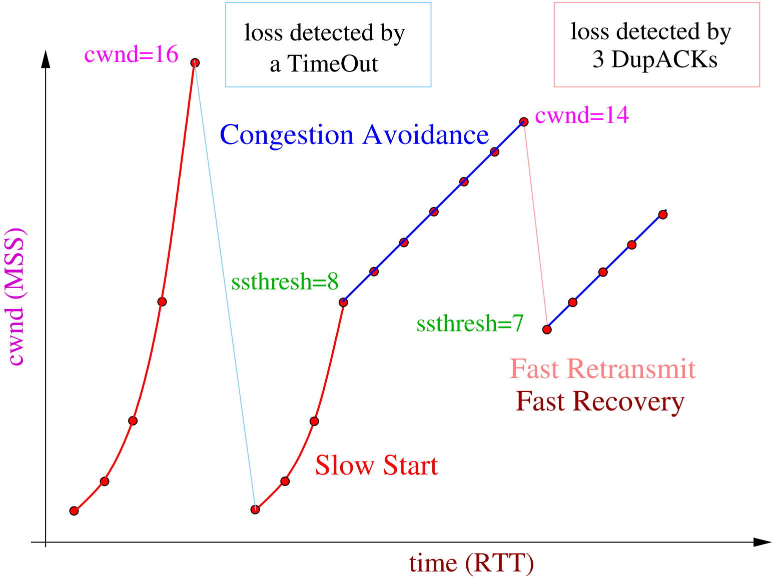
Key features

- Full duplex (piggyback of ACKs)
- Connection-oriented (Establishment and teardown of the connections)
- Multiplexing/Demultiplexing (through Source and Destination Port numbers)
- Reliability (through Sequence Numbers, Checksum, ACKs and timers)
- Flow Control (through Advertized Window)
- Congestion Control, making TCP sensitive to network conditions

TCP Congestion Control

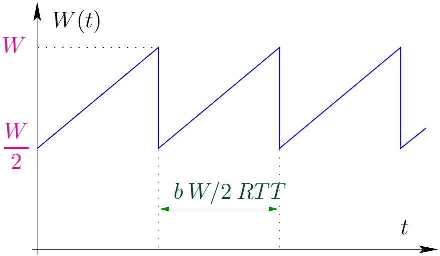
- TCP congestion control (CC) mechanisms seek to
  - Achieve high utilization
  - *Control* congestion
  - Share bandwidth
- TCP CC introduced in the late 1980s by Van Jacobson
  - In October 1986, the Internet had the first of what became a series of congestion collapses (sudden factor-of-thousand drop in bandwidth)
  - window-based mechanism: TCP maintains a state variable cwnd, used by the source to limit how much data it is allowed to have in transit at a given time
  - Slow Start, Congestion Avoidance and Fast Retransmit
  - *round-trip variance estimation*
- Differentiation between major and minor congestion events
  - Introduction of Fast Recovery (april 1990)

Classical TCP Congestion Control (TCP Reno)



Simple deterministic model of TCP Reno

- TCP source running over a lossy path with sufficient bandwidth and sufficiently low competing traffic
- Assume that the link introduces one drop after the successful delivery of $1/p$ consecutive packets
- No ACK loss



- Periodic evolution of cwnd
  - W: maximum value of cwnd reached at the equilibrium
  - cwnd is backed off to $W/2$ after each loss, starting a new cycle

Simple deterministic model – Main results

Mean throughput

$$\mathcal{B} = \frac{A_{\text{cycle}}}{T_{\text{cycle}}} = \frac{MSS \cdot b\frac{3}{8}W^2}{RTT \cdot \frac{b}{2}W} = \sqrt{\frac{3}{2b}} \cdot \frac{MSS}{RTT\sqrt{p}}$$

- The throughput is proportional to the packet size
- The throughput is inversely proportional to RTT (unfair behavior) and to the square root of loss probability
- Slightly different proportionality constant in other models

Limitations

- The timeout mechanisms is not taken into account
- Optimistic estimate of the bandwidth of a TCP connection
- Accurate in the range of small loss probabilities
- Not suitable to determine performance of TCP over slow-speed line (few packets in transit)

TCP Variants

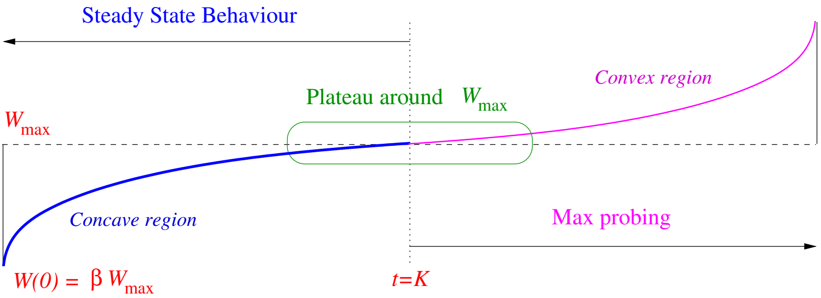
Long-distance (Long) and High-speed (Fat) Networks

- Conservative behavior of TCP Reno in adjusting its cwnd
- Congestion control parameters depend on current cwnd
- Queueing delay as a *secondary* congestion signal
- Impact of multiple losses $\Rightarrow$ Use of SACK

- Different mechanisms are necessary for congestion control in heterogeneous networks

High BDP	Wireless	Satellite	Inter-DC	Intra-DC
BIC H-TCP Compound CUBIC FAST TCP	Westwood Vegas Veno	Hybla STAR	Illinois SABUL	ICTCP DCTCP

TCP CUBIC

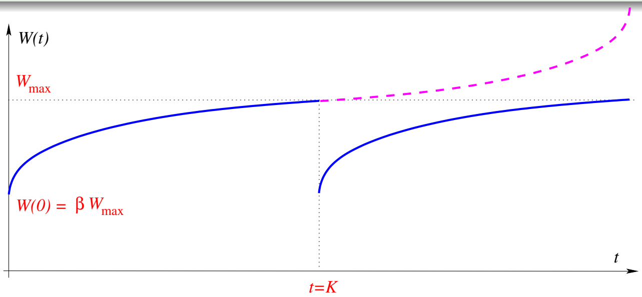


Window groth after a *congestion event*

- CUBIC registers the window size W_{max}
- It performs a multiplicative decrease of congestion window by a factor of β (suggested value: $\beta = 0.7$)
- It starts to increase the window using the concave profile
- The concave growth continues until W_{max}
- After that, the convex window growth begins

Simple deterministic model of TCP CUBIC

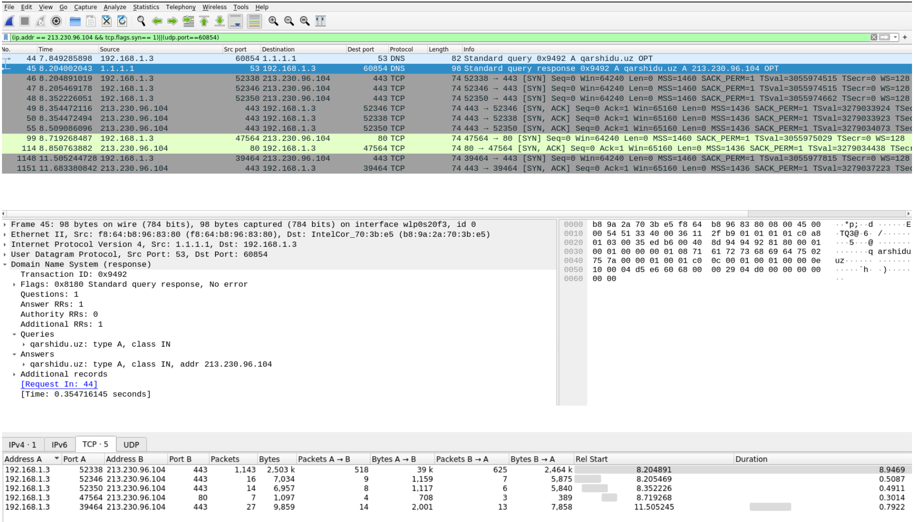
- The number of packets between two successive losses is 1/p
- CUBIC always operates with the concave window profile
- cwnd has a periodic evolution



Average cwnd size

$$\mathbb{E}W_{\text{CUBIC}} = \sqrt[4]{\frac{C\,(3+\beta)}{4\,(1-\beta)}\left(\frac{RTT}{p}\right)^3}$$

Opening a web connection . . .



Opening a web connection . . .

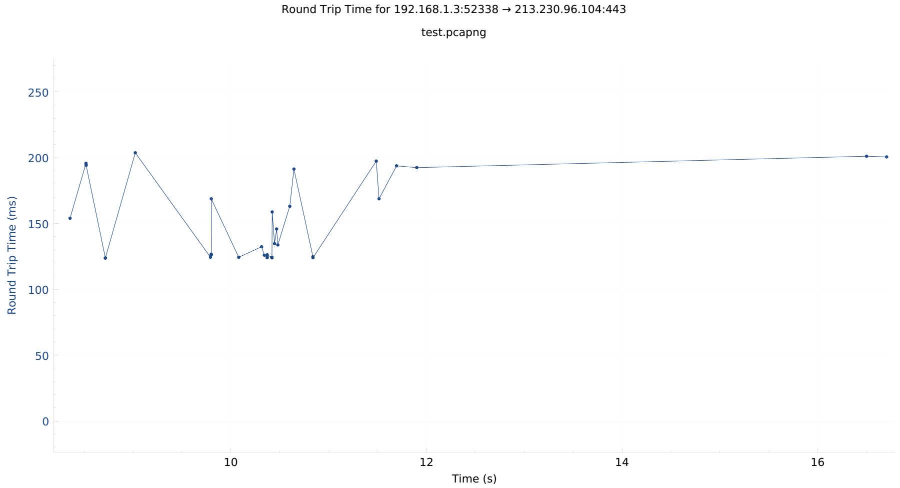
ip.addr == 213.230.96.104 && tcp.port ==39464

No.	Time	Source	Src port	Destination	Dest port	Protocol	Length	Info
1148	11.505244728	192.168.1.3	39464	213.230.96.104	443	TCP	74	39464 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3055977815 TSecr=0 WS
1151	11.683380842	213.230.96.104	443	192.168.1.3	39464	TCP	74	443 → 39464 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1436 SACK_PERM=1 TSval=3279037223
1154	11.683444097	192.168.1.3	39464	213.230.96.104	443	TCP	66	39464 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3055977993 TSecr=3279037223
1163	11.687536020	192.168.1.3	39464	213.230.96.104	443	TLSv1.3	583	Client Hello
1166	11.887868670	213.230.96.104	443	192.168.1.3	39464	TCP	66	443 → 39464 [ACK] Seq=1 Ack=518 Win=64768 Len=0 TSval=3279037407 TSecr=3055977997
1167	11.887868987	213.230.96.104	443	192.168.1.3	39464	TLSv1.3	2914	Server Hello, Change Cipher Spec, Application Data
1169	11.887912976	192.168.1.3	39464	213.230.96.104	443	TCP	66	39464 → 443 [ACK] Seq=518 Ack=2849 Win=61440 Len=0 TSval=3055978198 TSecr=3279037416
1170	11.887869064	213.230.96.104	443	192.168.1.3	39464	TLSv1.3	2599	Application Data, Application Data, Application Data, Application Data
1171	11.887948871	192.168.1.3	39464	213.230.96.104	443	TCP	66	39464 → 443 [ACK] Seq=518 Ack=5382 Win=59008 Len=0 TSval=3055978198 TSecr=3279037416
1173	11.892628665	192.168.1.3	39464	213.230.96.104	443	TLSv1.3	130	Change Cipher Spec, Application Data
1174	11.893392180	192.168.1.3	39464	213.230.96.104	443	TLSv1.3	164	Application Data
1175	11.893764666	192.168.1.3	39464	213.230.96.104	443	TLSv1.3	425	Application Data
1177	12.092989721	213.230.96.104	443	192.168.1.3	39464	TCP	66	443 → 39464 [ACK] Seq=5382 Ack=582 Win=64768 Len=0 TSval=3279037611 TSecr=3055978202
1179	12.092990172	213.230.96.104	443	192.168.1.3	39464	TLSv1.3	121	Application Data
1181	12.093050795	192.168.1.3	39464	213.230.96.104	443	TCP	66	39464 → 443 [ACK] Seq=1039 Ack=5437 Win=64128 Len=0 TSval=3055978403 TSecr=3279037611
1183	12.092990341	213.230.96.104	443	192.168.1.3	39464	TCP	66	443 → 39464 [ACK] Seq=5437 Ack=680 Win=64768 Len=0 TSval=3279037611 TSecr=3055978203
1184	12.092990436	213.230.96.104	443	192.168.1.3	39464	TLSv1.3	101	Application Data
1185	12.093116980	192.168.1.3	39464	213.230.96.104	443	TCP	66	39464 → 443 [ACK] Seq=1039 Ack=5472 Win=64128 Len=0 TSval=3055978403 TSecr=3279037611
1187	12.092990519	213.230.96.104	443	192.168.1.3	39464	TLSv1.3	97	Application Data
1189	12.093140835	192.168.1.3	39464	213.230.96.104	443	TCP	66	39464 → 443 [ACK] Seq=1039 Ack=5503 Win=64128 Len=0 TSval=3055978403 TSecr=3279037611
1190	12.092990608	213.230.96.104	443	192.168.1.3	39464	TCP	66	443 → 39464 [ACK] Seq=5503 Ack=1039 Win=64512 Len=0 TSval=3279037612 TSecr=3055978204
1191	12.092990702	213.230.96.104	443	192.168.1.3	39464	TLSv1.3	209	Application Data
1192	12.093169309	192.168.1.3	39464	213.230.96.104	443	TCP	66	39464 → 443 [ACK] Seq=1039 Ack=5646 Win=64000 Len=0 TSval=3055978403 TSecr=3279037613

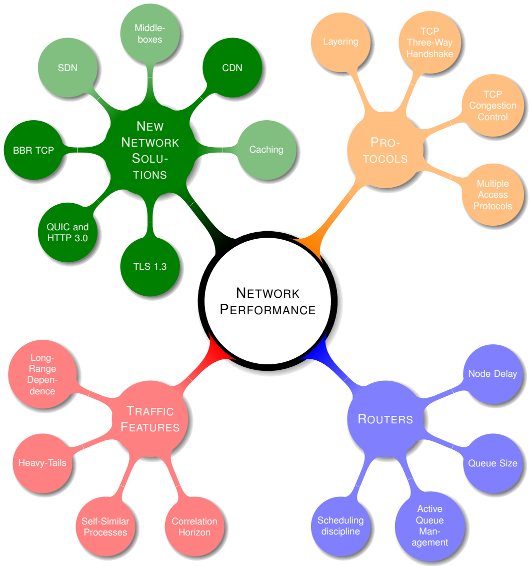
```
Transmission Control Protocol, Src Port: 39464, Dst Port: 443, Seq: 0, Len: 0
  Source Port: 39464
  Destination Port: 443
  [Stream index: 10]
  [TCP Segment Len: 0]
  Sequence number: 0    (relative sequence number)
  Sequence number (raw): 440046424
  [Next sequence number: 1    (relative sequence number)]
  Acknowledgment number: 0
  Acknowledgment number (raw): 0
  1010 .... = Header Length: 40 bytes (10)
  Flags: 0x002 (SYN)
  Window size value: 64240
  [Calculated window size: 64240]
  Checksum: 0xf828 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
```

```
Transmission Control Protocol, Src Port: 443, Dst Port: 39464, Seq: 1, Ack: 518, Len: 2848
Transport Layer Security
  TLSv1.3 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 122
    Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 118
      Version: TLS 1.2 (0x0303)
      Random: 88a73abfa9b2f4ea4e8478b066ac17dea32fff7e6c1434a8…
      Session ID Length: 32
      Session ID: 1915498050431ae9598389a5fac77643147bde9aebd40c8b…
      Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
      Compression Method: null (0)
      Extensions Length: 46
      Extension: supported_versions (len=2)
      Extension: key_share (len=36)
  TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
```

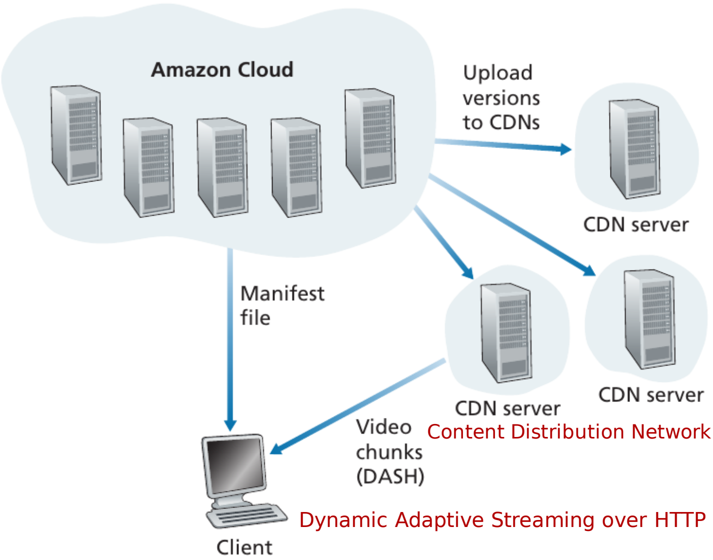
Opening a web connection . . .



New Network Solutions



Netflix video streaming platform

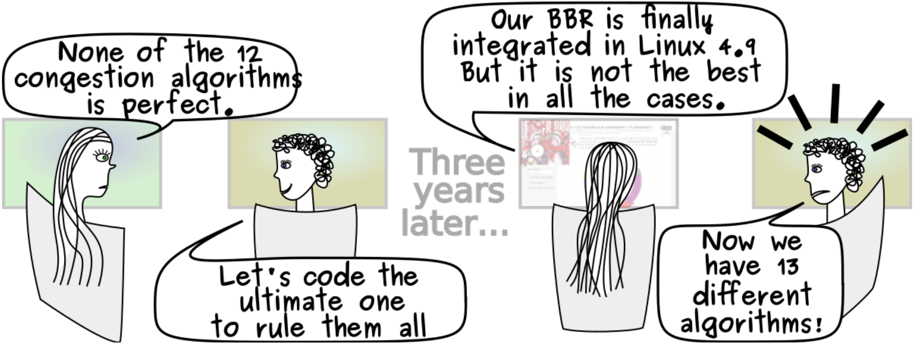


Content Distribution Network

Dynamic Adaptive Streaming over HTTP

BBR: Bottleneck Bandwidth and Round-trip propagation time

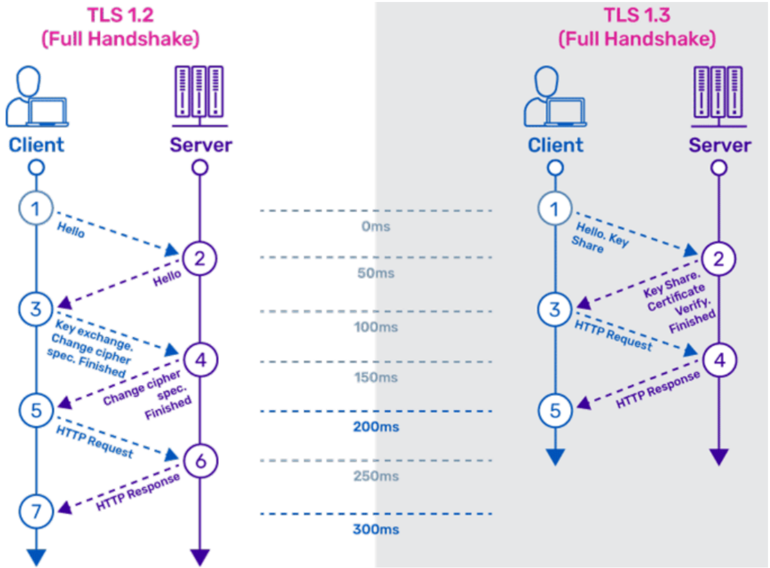
N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson
"**BBR: Congestion-Based Congestion Control**", ACM Queue, Oct. 2016



BBR v2 – A Model-based Congestion Control

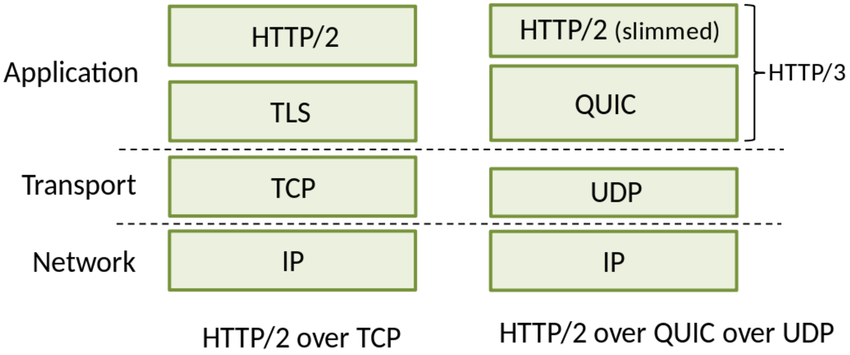
N. Cardwell, Y. Cheng, S. H. Yeganeh, I. Swett, V. Vasiliev, P. Jha, Y. Seung, M. Mathis, V. Jacobson, IETF 104, Prague, March 2019
https://groups.google.com/d/forum/bbr-dev

TLS 1.3 – Faster TLS Handshake



QUIC and HTTP/3

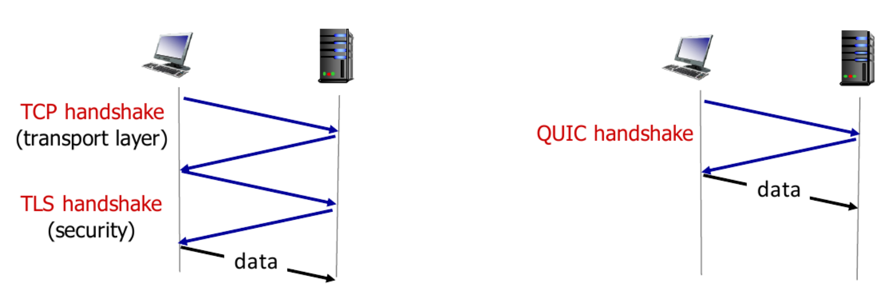
- QUIC: Quick UPD Internet Connections
- Application–layer protocol, on top of UDP
- Deployed on many Google servers and apps



QUIC's major features

- Connection–oriented and Secure
- Application-level *streams*
- Reliable, *TCP-friendly* congestion–controlled data transfer

QUIC vs. TCP with TLS



Conclusions

- Future killer applications and their traffic features
- New versions of TCP
- TCP or QUIC?
- Effect of CDNs
- Role of Middleboxes
- SDN controller
- Mobile users
- IoT and IIoT
- QoS vs. QoE


СПАСИБО ЗА ВНИМАНИЕ
И ПОПЫТКУ ПОНИМАНИЯ