4TC-Couche Transport
Transport Layer

Part 1 – TCP Generalities

R. Stanica
Févr. 2024

General Outline

- Course structure
  - 3x2h – lectures: Razvan Stanica
  - 1x2h – TD: Razvan Stanica, Frédéric Le Mouel
  - 2x4h – TP: Razvan Stanica, Frédéric Le Mouel, Victor Rebecq
- Grading system
  - Written exam, all documents authorized
  - 4TC-TCP is part of the UE RES4 and represents 1 ECTS
- Objectives
  - Understand the role and functioning of the transport layer
  - Link between “networks” and “programming”
  - Manipulation of system calls and the Sockets API

Transport Layer

- You already saw the basics in 3TC IP
  - UDP vs. TCP
  - TCP and UDP header format
  - TCP connection management
  - TCP state machine

TCP Design

- TCP follows the end-to-end principle
  - When a function can be implemented on the edge devices, it should be implemented on the edge devices.
  - If you need it at the endpoints anyway, you may as well not do it twice!
  - Keep the network simple (and cheaper).
  - The only failure you can't survive is a failure of the end points.
  - A principle challenged by technology evolutions.

TCP Design

- Flow control
  - Manage the data rate at the transmitter in order not to overwhelm a slower receiver.
- Congestion control
  - Manage transmission rate in order to avoid network congestion collapse.

TCP Vocabulary

- Segment
  - The TCP payload data unit.
  - Not a *message*, not a *packet*, not a *frame*.

- MSS – Maximum Segment Size
  - The size of the largest segment that can be transmitted or received.
  - Negotiated between end hosts.

- Receiver Window (rwnd / awnd)
  - The number of segments a host can receive at a given time.
  - Used for flow control.

- Congestion Window (cwnd)
  - The number of segments that the network can support.
  - Used for congestion control.

TCP Vocabulary

- Sequence Number
  - Each TCP segment contains a sequence number.
  - Initial sequence number (ISN) transmitted in the SYN segment.
  - It is increased by 1 for every new transmitted segment.
  - ISN originally derived from the system clock.
  - Today ISN is a function of multiple system parameters, including a random number.

- Acknowledgement
  - A segment transmitted every time a segment containing data is received.
  - It can also carry useful data.
  - In TCP, we always ACK the largest sequence number received contiguously.

TCP Design

- Basic Transmission Principle
  - At any given time, a TCP host must not transmit a segment with a sequence number higher than the sum of the highest acknowledged sequence number and the minimum value between cwnd and rwnd.

TCP Flow Control

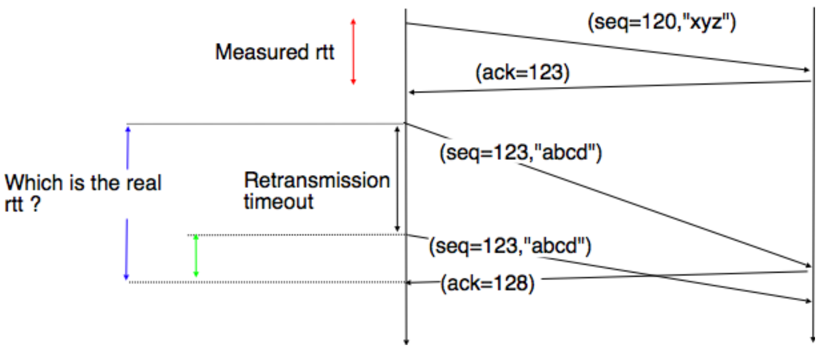
- Receiver Window - rwnd
  - Feed-back to the transmitter in the *window* field of the TCP header.
  - 16 bits field: a maximum window of 65535 bytes.
  - Serious TCP throughput limitation: 5-500 Mb/s.
  - Modern extensions allow a 32 bits window: throughput of more than 100 Gb/s.

Round Trip Time

- RTT in TCP
  - The time between the transmission of a data segment and the reception of the ACK.
  - Used to decide whether a segment is lost or not.
  - It can vary significantly during network operations.
  - TCP computes an estimator, which it updates at every received ACK.
  - Under-estimation results in useless retransmissions.
  - Over-estimation results in long waiting times.

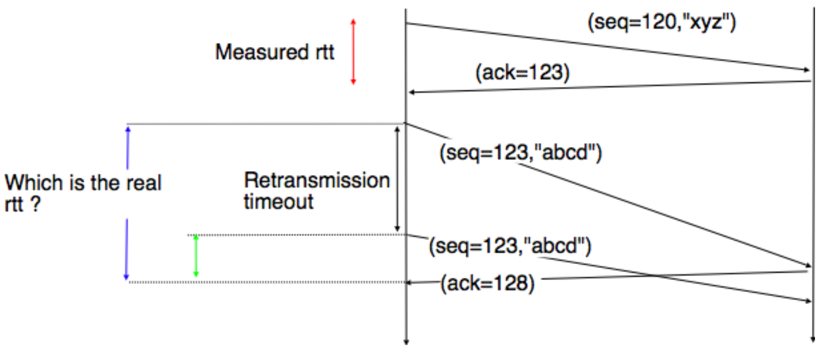
Round Trip Time

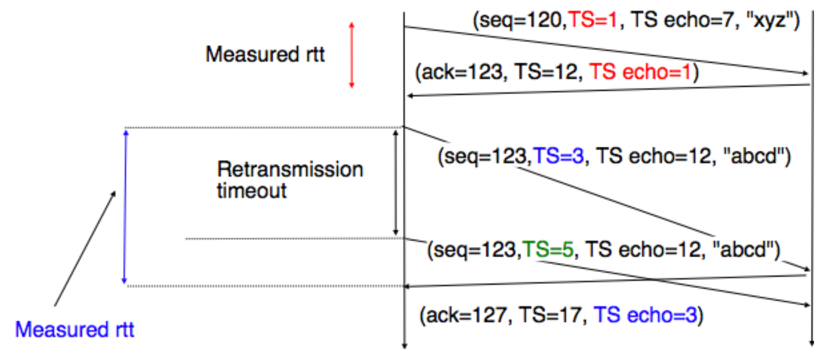
- RTT in TCP



Round Trip Time

- RTT in TCP





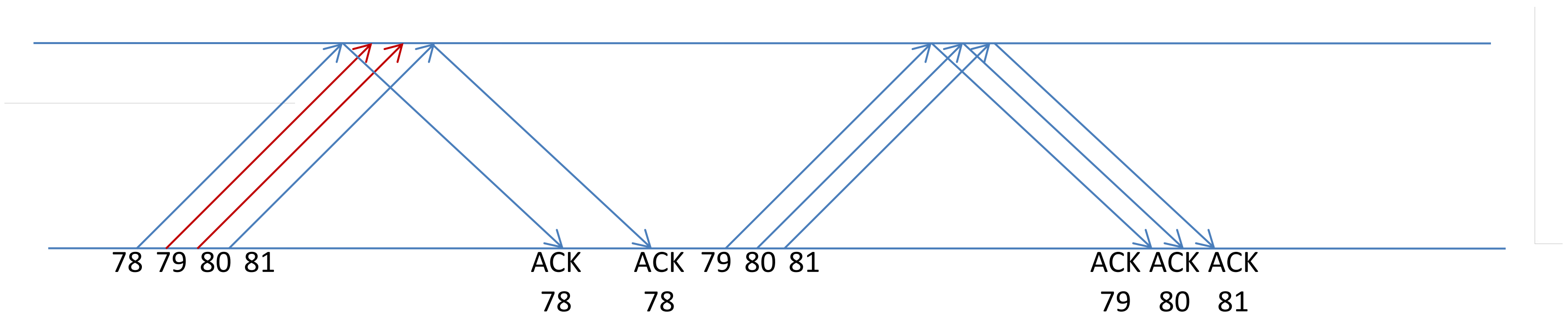
TCP Principles

- Congestion Detection
  - Based on the idea that a segment lost in the network is the results of congestion.
  - This creates significant issues with TCP on less reliable media (e.g. wireless).

- Segment Loss Detection
  - Based on the RTT estimator.
  - Based on the reception of duplicate ACKs.

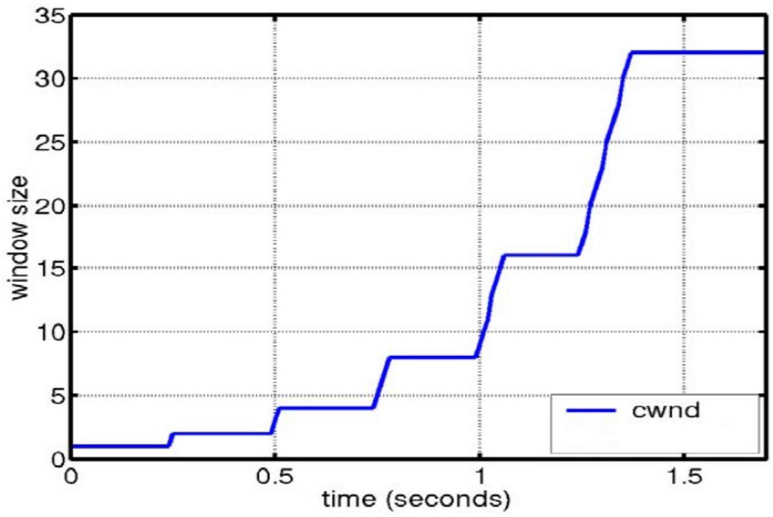
TCP Principles

- Duplicate ACKs
  - A receiver is not allowed to ACK non-contiguous segments.
  - The reception of an out-of-order segment results in the ACK of the last contiguous segment, producing a duplicate ACK.



TCP Congestion Control

- Slow Start
  - Motivation: the end hosts do not know the state of the network at the beginning of their connection.
  - Solution: start with cwnd= 1.
  - This initial value is increased in more recent versions, up to round 10 segments.
  - For every correctly received ACK: cwnd= cwnd+ 1.
  - Practically, this doubles the value of cwnd during an RTT interval.
  - Despite its name, exponential increase of cwnd.



TCP Congestion Control

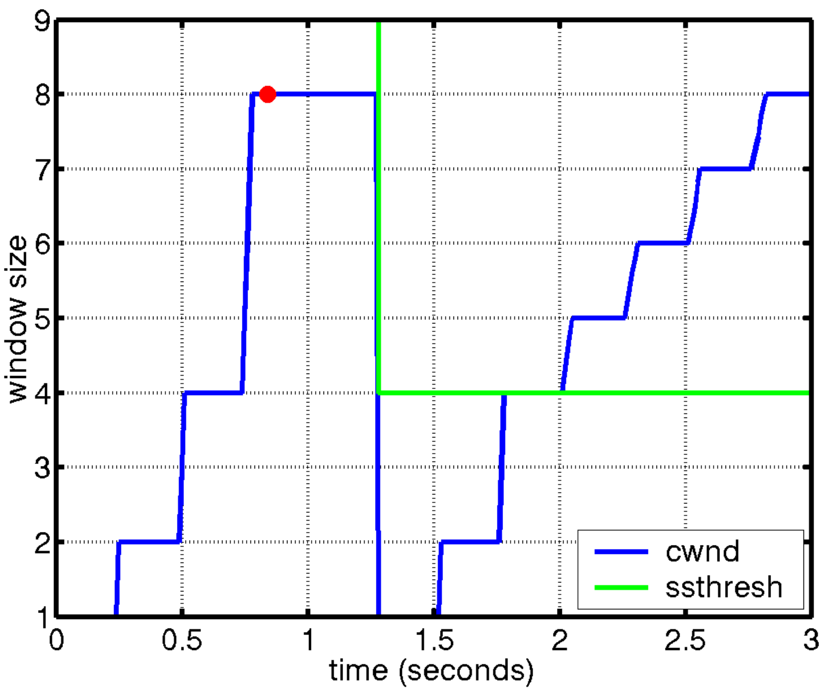
- Flight Size
  - The number of segments that have been sent, but not yet acknowledged.
  - A common mistake is to consider it equal to cwnd (this is only true when all the segments in the window have been transmitted).

TCP Congestion Control

- Slow Start Threshold - ssthresh
  - Slow start is very aggressive.
  - ssthresh decides when to stop the exponential increase.
  - Initial value – arbitrary and usually very high.
  - It should ideally follow the congestion level.
- After a lost segment
  - ssthresh= FlightSize/2
  - cwnd= 1
  - Retransmit the lost segment.
  - Restart the Slow Start mode.

TCP Congestion Control

- Congestion Avoidance
  - A host enters in this mode when cwnd > ssthresh
  - For every correctly received ACK: cwnd= cwnd+ 1/cwnd.
  - Practically, during an RTT interval: cwnd= cwnd+1.

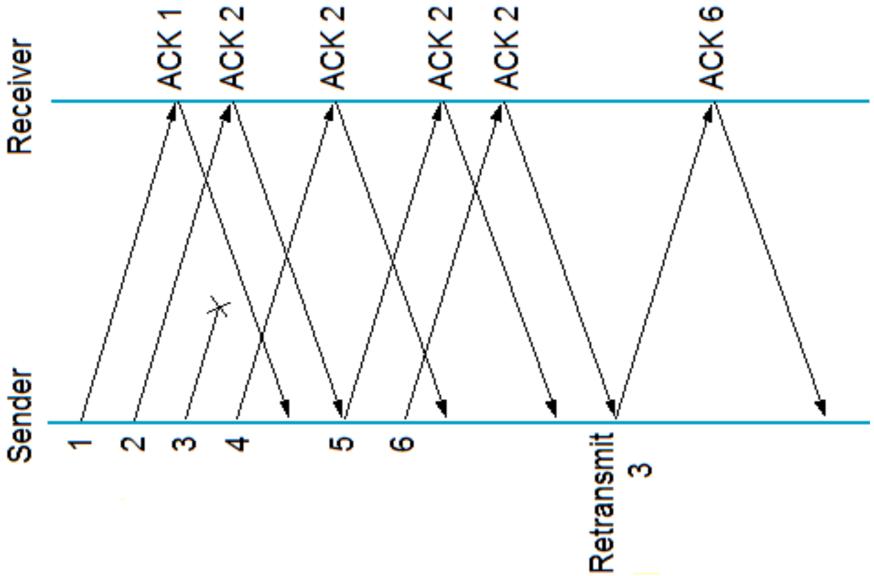


TCP Congestion Control

- Congestion Avoidance
  - Idea: be less aggressive than in Slow Start
  - Once a congestion has been detected, the transmitter tries to “avoid” reaching the congested state once again.
  - A static approach can miss the opportunity of an increased throughput if the network congestion reduces.
  - The slow cwnd increase delays the next congestion, while still testing for transmission opportunities.

TCP Congestion Control

- Fast Retransmit
  - An ACK timeout indicates a loss, but it can be long.
  - Idea: use duplicate ACKs to detect congestion.
  - A duplicate ACK can have multiple reasons: congestion, segments following different paths, reordered ACKs.
  - Considering a segment lost after the first duplicate ACK is too aggressive.
  - TCP declares a segment lost after 3 duplicate ACKs (i.e. 4 consecutive ACKs of the same segment).



TCP Congestion Control

- Fast Retransmit
  - This mechanism generally eliminates half of the TCP timeouts.
  - This yields roughly a 20% increase in throughput.
  - It does not work when the transmission window is too small to allow the reception of three duplicate ACKs.
  - It does not work when multiple segments are lost in the same window.

TCP Congestion Control

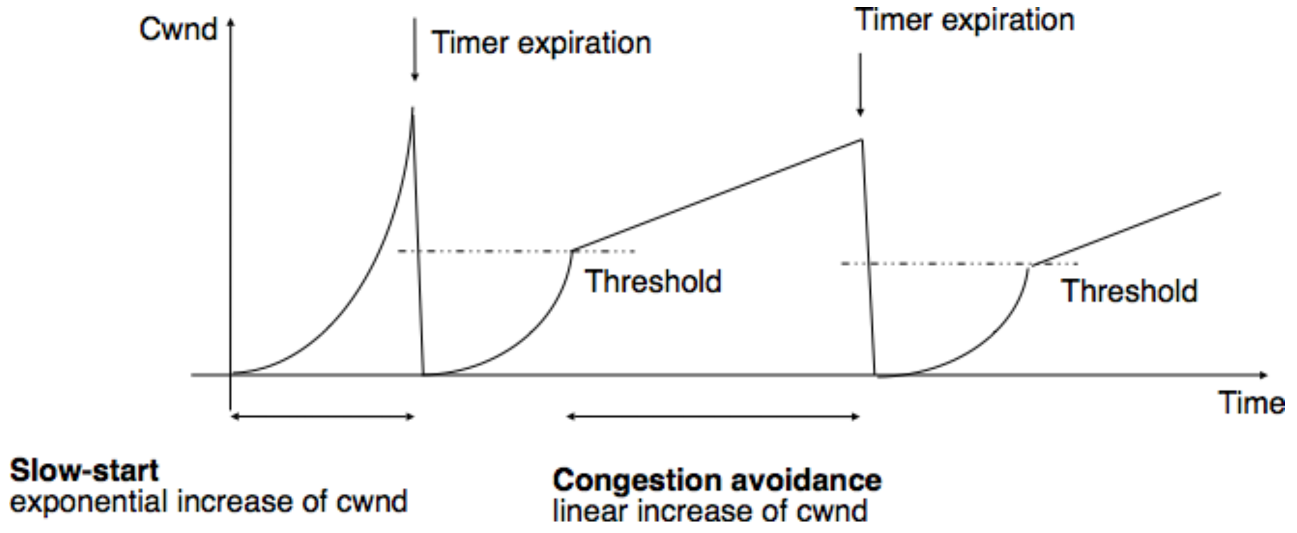
- Fast Recovery
  - Complement of Fast Retransmit.
  - The reception of duplicate ACKs also means that network connectivity exists, despite a lost segment.
  - Entering Slow Start is a bad idea in this case, as the congested state might have disappeared.
  - The mechanism allows for higher throughput in case of moderate congestion.

TCP Congestion Control

- Fast Recovery
  - Mode entered after Fast Retransmit (3 duplicate ACKs detected).
  - As usual: ssthresh= FlightSize/2.
  - Retransmit lost segment.
  - Window inflation: cwnd= ssthresh+ ndup (number of duplicate ACKs received).
  - This allows the transmission of new segments.
  - Window deflation after the reception of the missing ACK.
  - Practically, we skip Slow Start, go directly to Congestion Avoidance.

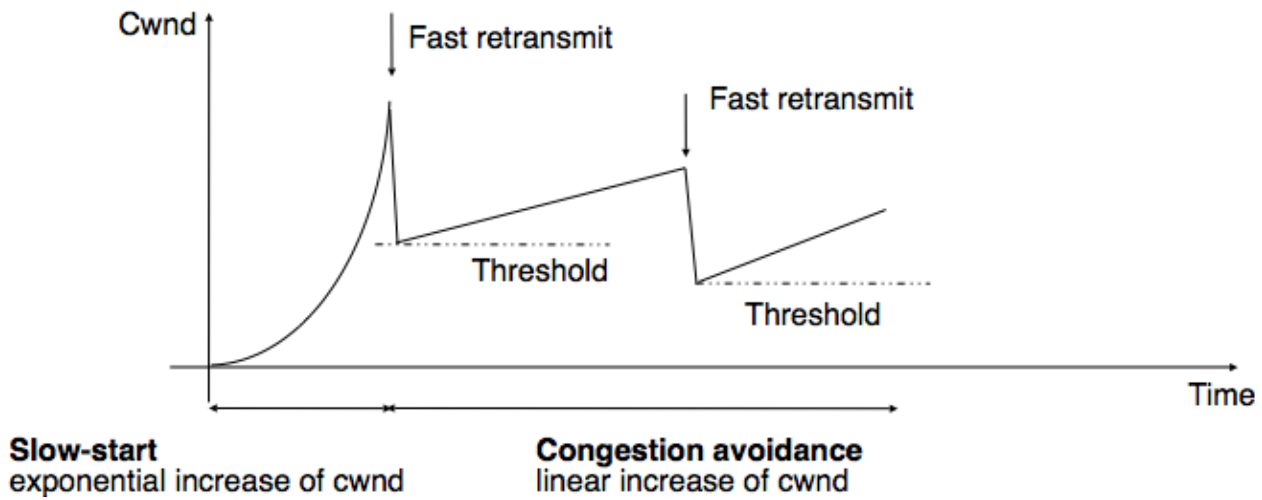
TCP Congestion Control

- Severe Congestion



TCP Congestion Control

- Mild Congestion



TCP Congestion Control

- Selective Acknowledgements
  - The receiver can only acknowledge contiguous segments.
  - No ACK for segments correctly received after a lost segment.
  - The sender has no feed-back regarding correctly received segments and it retransmits them.
  - Ideally, the sender should retransmit only the missing segments.
  - With SACK, the receiver provides this feed-back to the sender, through an extension of the TCP header.

TCP Mechanisms

- Delayed ACKs
  - Many applications imply bidirectional communications (e.g. HTTP).
  - Wait for applications layer response, in order to combine ACK and data.
  - Reduce overhead by combining multiple ACKs in one segment.
  - Delay ACK by up to 500 ms.
  - An ACK is still mandatory for every two received segments.

TCP Mechanisms

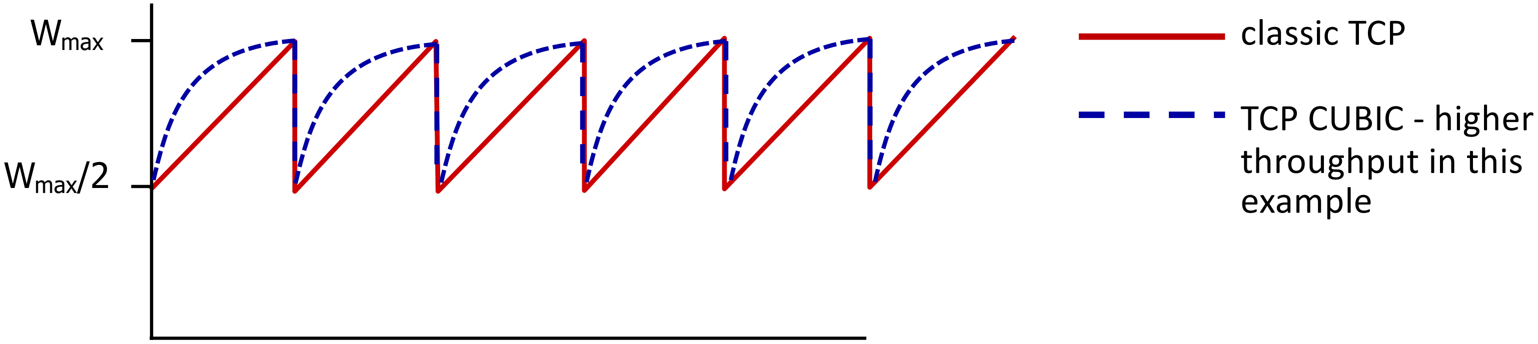
- Nagle's Algorithm
  - Small segment problem.
  - Wait for the application layer to provide more data and combine small outgoing data in one single segment.
  - Until ACK is not received, keep buffering output data until a full size segment can be sent.
  - Poor interaction with Delayed ACKs.

TCP Versions

- Different TCP versions, depending on the implemented mechanisms
  - TCP Tahoe
  - TCP Reno
  - TCP New Reno
  - TCP Vegas
- Major requirement: backwards compatibility.
- Other requirement: TCP friendliness.
- Common principle: AIMD – Additive Increase Multiplicative Decrease.
- Window behaviour is a function of the received ACKs (known as ACK pacing).

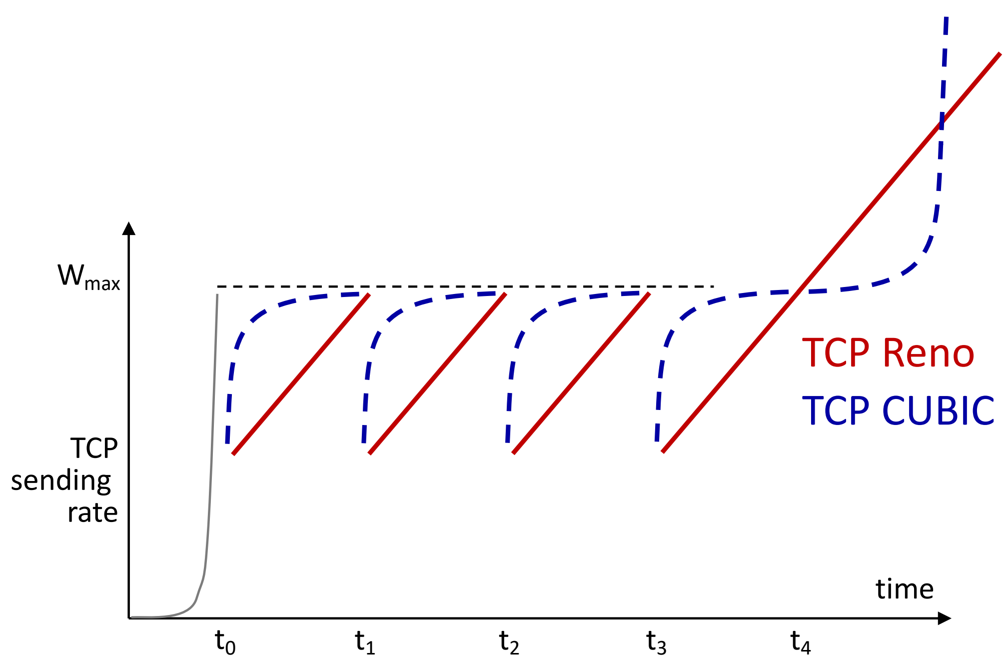
TCP CUBIC

- Philosophy shift: window as a function of time since last congestion.
- Do better than AIMD.
- Assumption: the congestion level does not change a lot.
- Congestion happens at cwnd= Wmax.
- Decrease cwnd= Wmax/2
- Aggressive cwnd increase.



TCP CUBIC

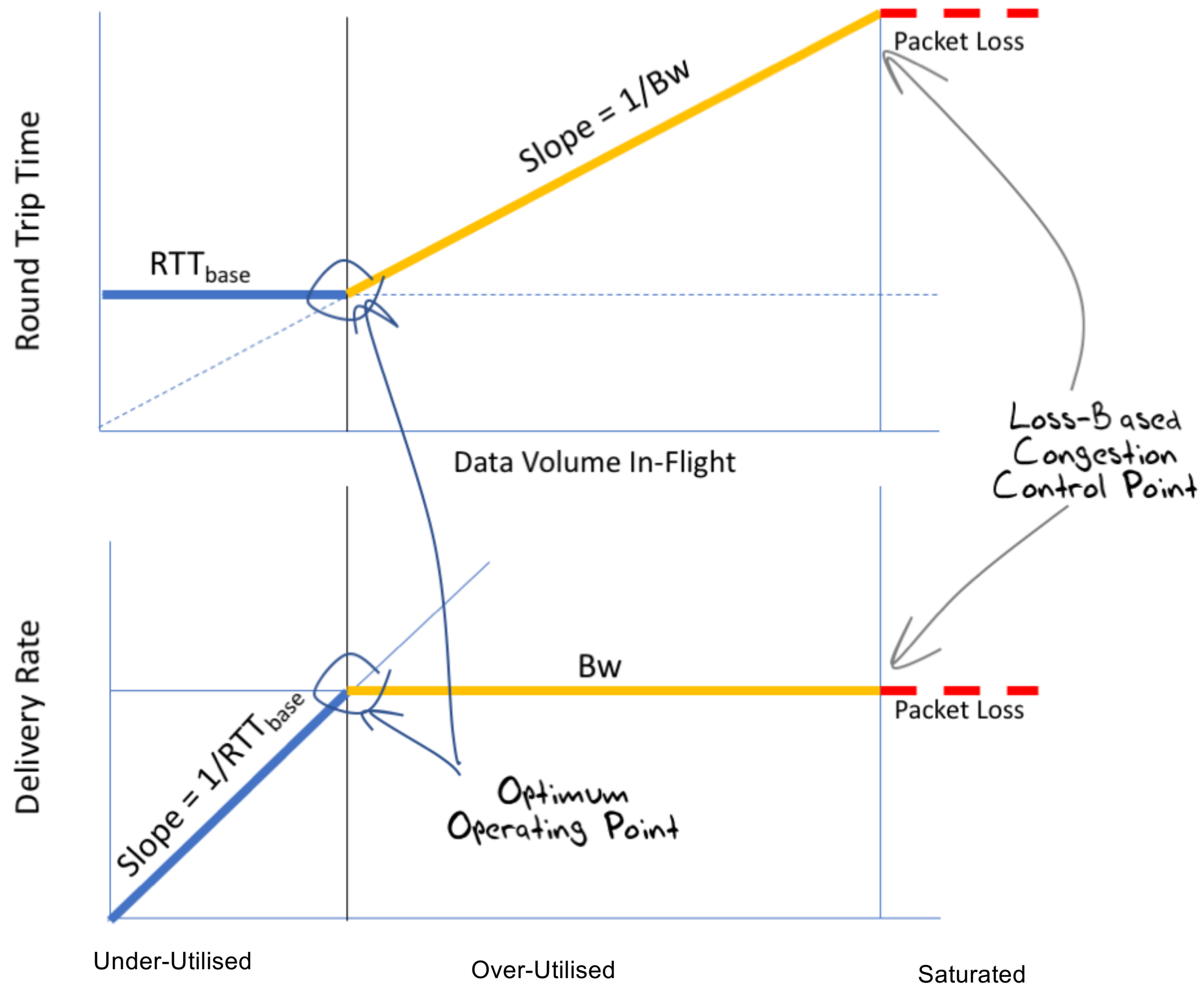
- Slow increase when cwnd is close to Wmax, the window value at the last congestion.
- If no congestion around Wmax, there is no way to know the new congestion level.
- Back to a very aggressive increase.
- Window value varies with time, not with received ACKs.
- End result: a cubic window function as a function of time.



TCP BBR

- The network can be in three states.
- Under-utilized: the flow rate is below the link capacity and no queues form in the routers.
- Over-utilized: the flow rate is above the link capacity and queues form in the routers.
- Saturated: Queues are filled and packets are dropped.
- Loss-based TCP versions probe towards the saturated stated and drop quickly to under-utilized.
- And then go up again to saturated…

TCP BBR

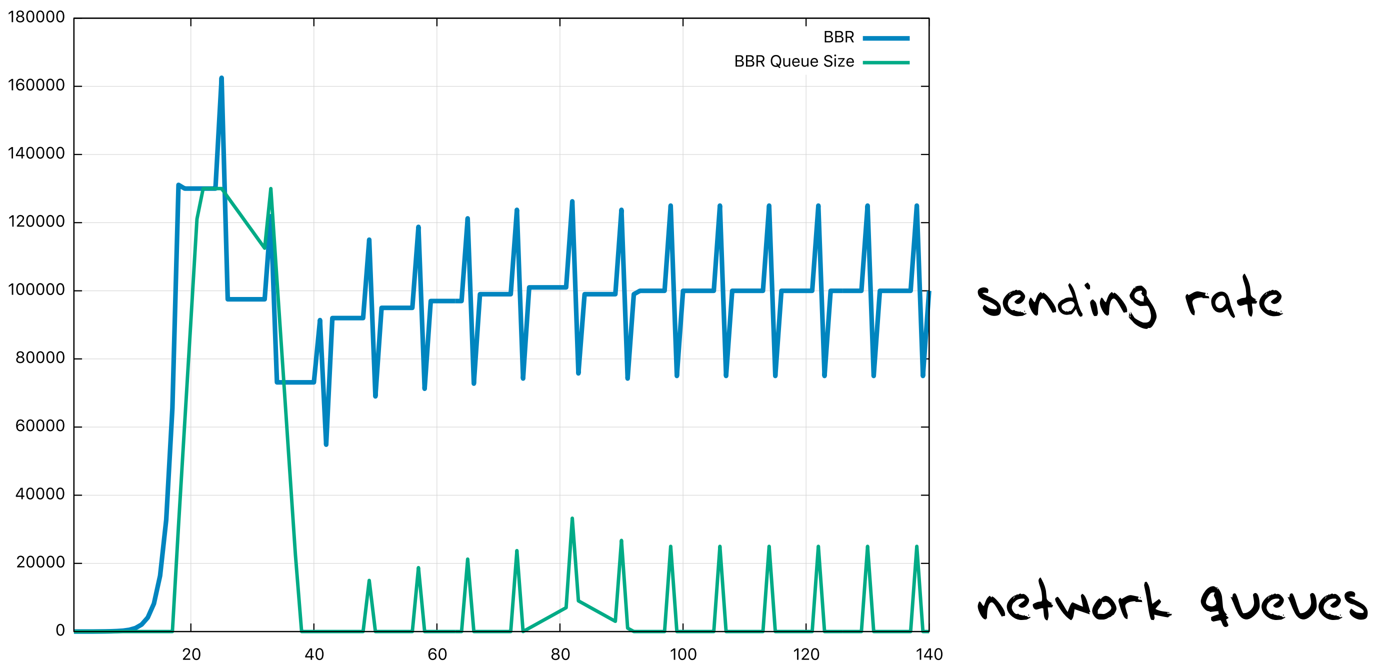

Round Trip Time
RTT_base
Slope = 1/Bw
Packet Loss
Data Volume In-Flight
Loss-Based
Congestion
Control Point
Delivery Rate
Slope = 1/RTT_base
Bw
Packet Loss
Optimum
Operating Point
Under-Utilised
Over-Utilised
Saturated

TCP BBR

- BBR – Bottleneck Bandwidth and Round-trip propagation time
- Idea: detect over-utilization by carefully monitoring the RTT.
- Probe the RTT intermittently.
- If the RTT is similar to the previously measured RTT, the network is under-utilized and we can increase the transmission rate.
- If the RTT increases, there is likely an over-utilization.
- Pace the transmission rate to stay close to the optimal operating point.
- Not very TCP friendly.

TCP BBR

- Ideal BBR behaviour



Network-assisted solutions

- Beyond the end-to-end principle.
- Packets are dropped in the network.
- Advanced techniques can be implemented at the router level.

Random Early Detection

- RED manages router queues and drops packets based on a queue threshold.
- Once the queue is over the threshold, the router drops packets with a certain probability.
- Only the affected TCP senders will enter Slow Start or Congestion Avoidance, reducing the network usage before the actual congestion.

Explicit Congestion Notification

- ECN is based on a queue threshold parameter, just as RED.
- As opposed to RED, ECN only marks packets instead of dropping them.
- Routers mark 2 bits in the IP header (Type of Service field) to signal that the queue size is above the threshold.
- Through cross-layer mechanisms, TCP learns the information and reduces the congestion window.
- ECN avoids packet drops and reduces the delay created by retransmissions.

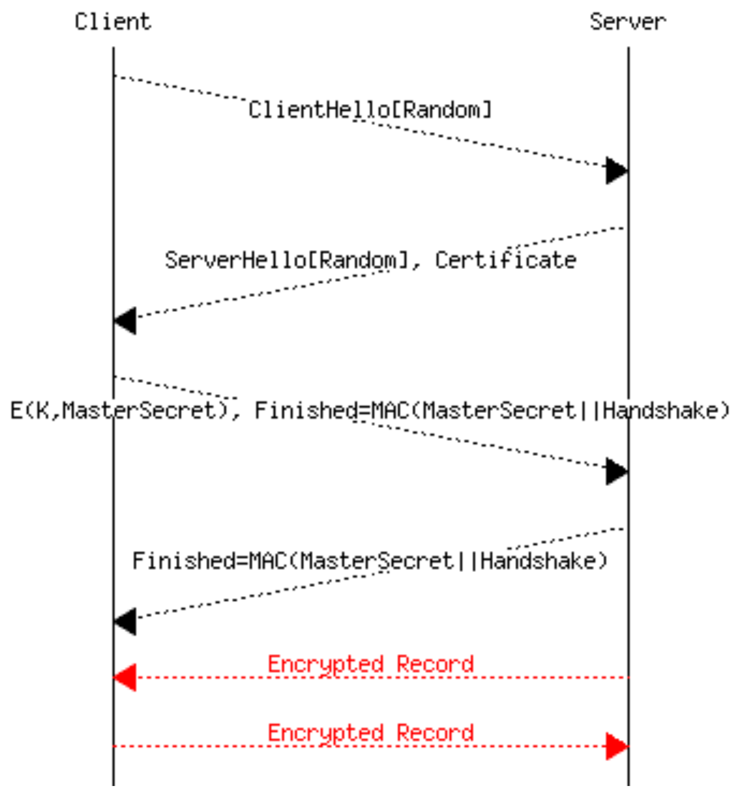
Transport layer security

- TCP (and UDP) do not have any innate security functions.
- The underlying idea is that security should be implemented at the application layer.
- Not scalable with the democratization of application development.
- Security is needed at the operating system level.

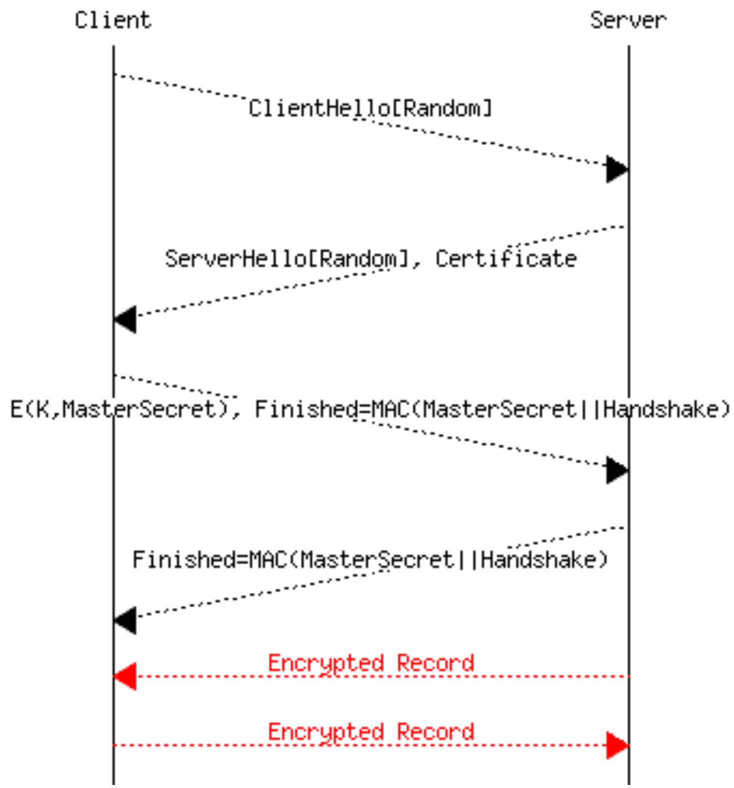
Transport layer security

- TLS – Transport Layer Security
- Evolution of SSL – Secure Socket Layer
- Current version: TLS 1.3
- Responsible for authentication and encryption
- On top of a reliable TCP connection
- Two implementation possibilities:
  - Use a different port for the secure connection: HTTP over port 80, HTTPS over port 443
  - Use the generic TCP port and a specific message to trigger the TLS connection: STARTTLS in SMTP

Transport layer security

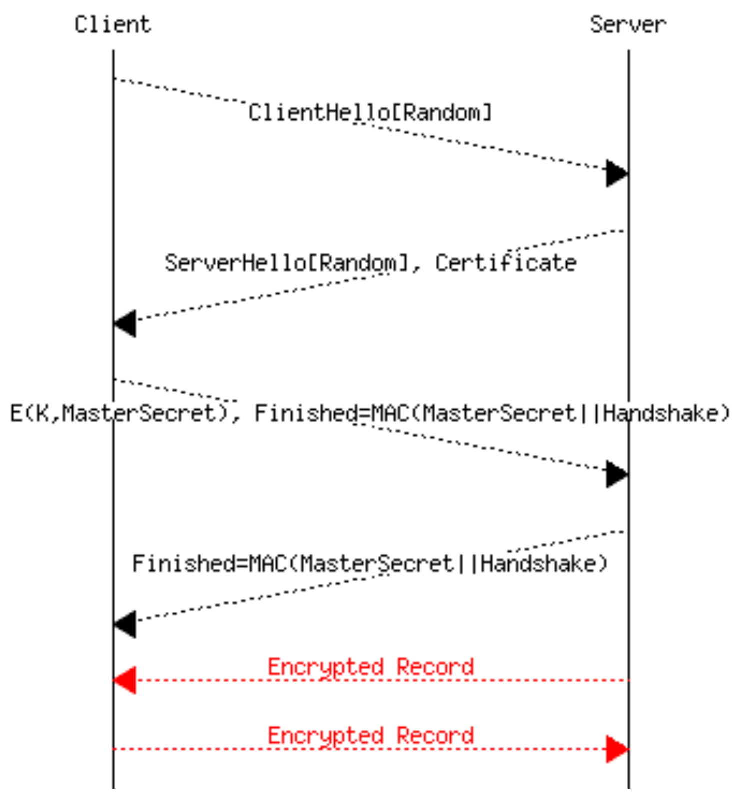


Transport layer security



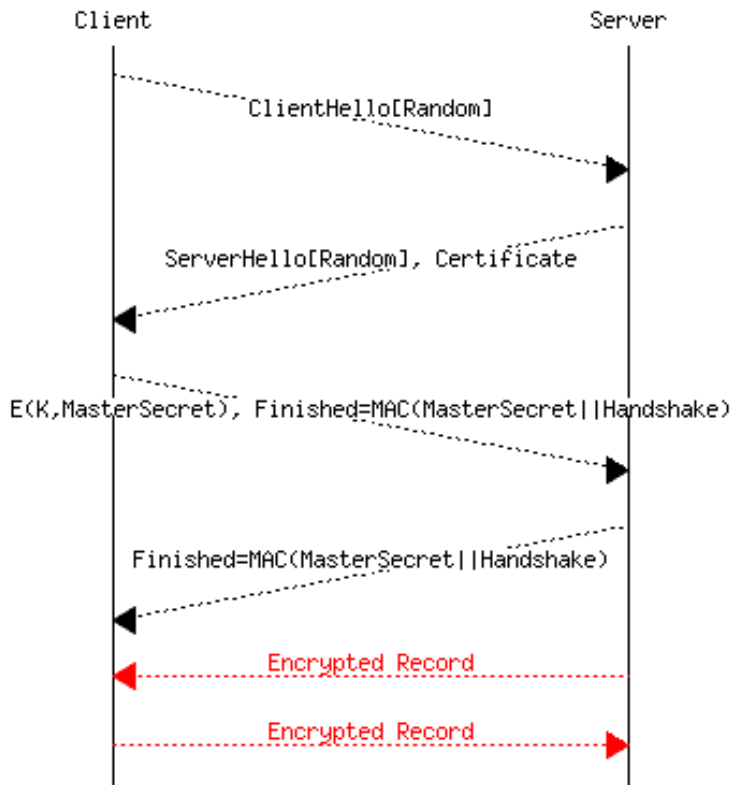
- ClientHello
  - Protocol version
  - Random number – later used to generate the keys
  - Cipher suites – supported cryptographic algorithms
  - Compression algorithms – usually disabled
  - Extensions – e.g. SNI – Server Name Indication

Transport layer security



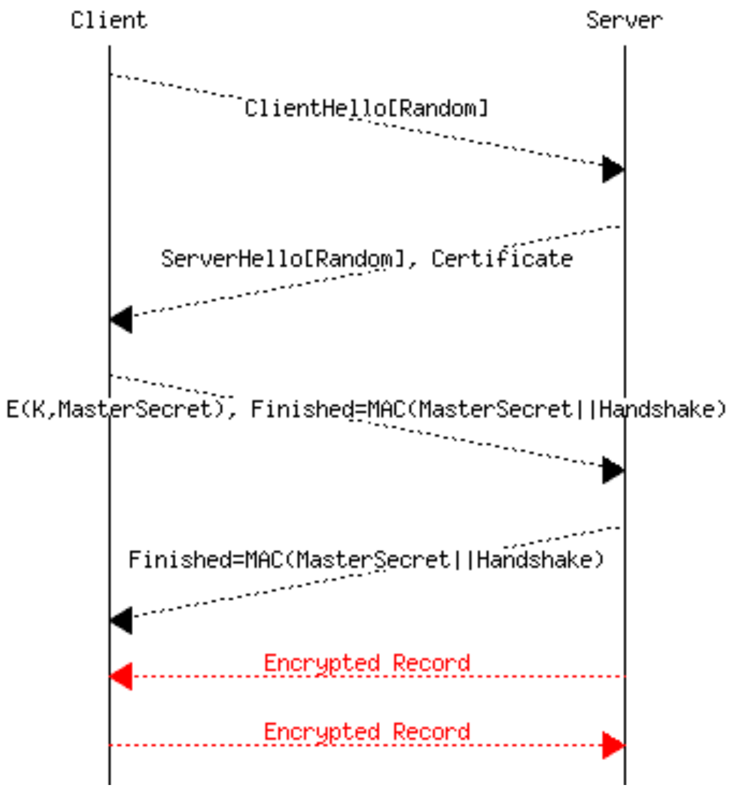
- ServerHello
  - Answers to ClientHello
  - Random number – later used to generate the keys
  - Session ID – allows resuming a session

Transport layer security



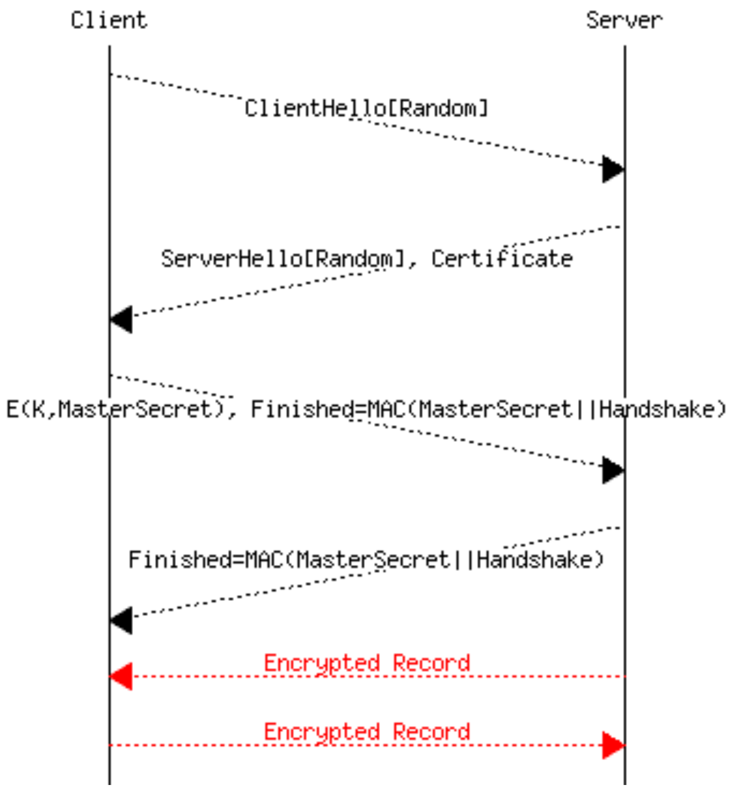
- Certificate
  - Authenticates the server
  - Based on a public key infrastructure
  - Requires a Root Certification Authority

Transport layer security



- MasterSecret
  - Depends on the selected key exchange algorithm
  - Encrypted with the public key of the server
  - Used to derive the session keys

Transport layer security

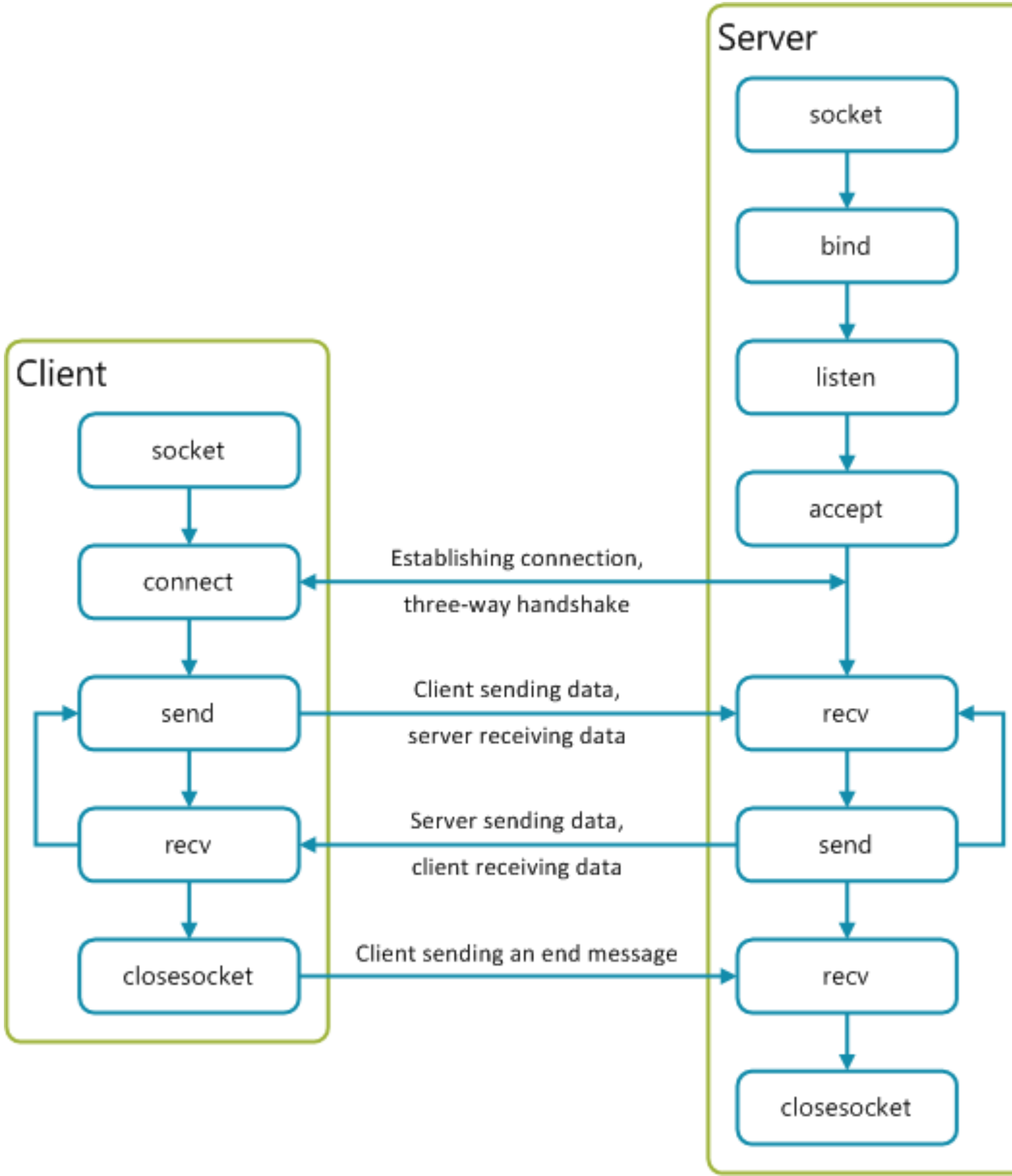


- Finished
  - Message authentication code
  - Ensures messages were not modified

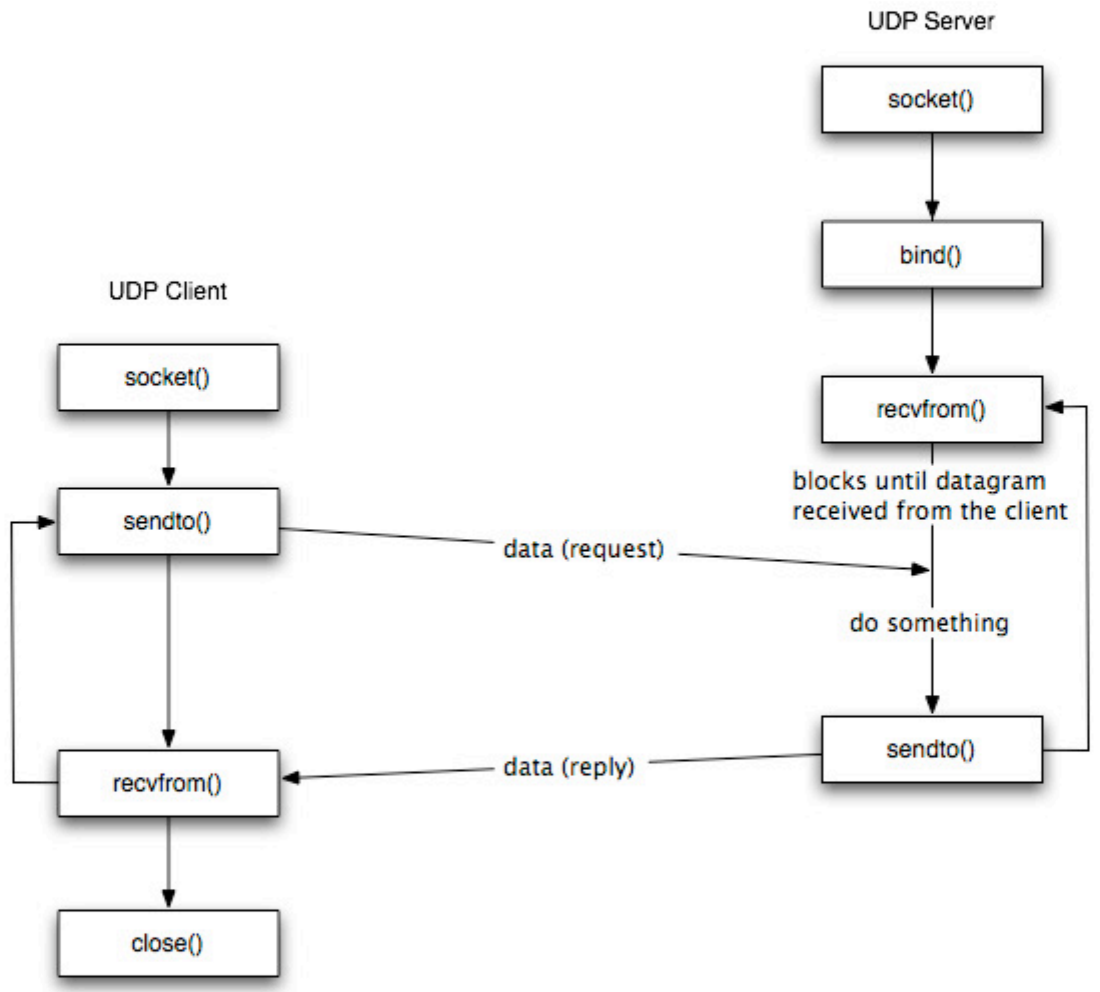
TCP implementation

- TCP (and UDP) are implemented in the operating system, at the kernel level.
- A series of libraries in the OS.
- Functions exposed by the Socket API.
- A socket is represented as a file handler in Unix-like operating systems.

TCP implementation



UDP implementation



Internet ossification

- Numerous intermediary machines on the Internet, so called middle-boxes.
- Proxies, VPN servers, NATs, firewalls, etc.
- A tendency towards security policies based on traffic filtering.
- Difficult to propose new backward-compatible TCP versions.
- Practically impossible to implement a new transport protocol at the OS level.

Head-of-line blocking

- In current web, HTTP multiplexes multiple streams over one TCP connection.
- With TCP congestion control, losing a segment from one stream blocks the other streams.

QUIC

- The Google answer to all problems above.
- Integrated with TLS to reduce the connection delay.
- Independent control of streams to solve HOL blocking.
- Implemented in the user space over UDP to escape middleboxes.
- No longer based on the Socket API.

QUIC

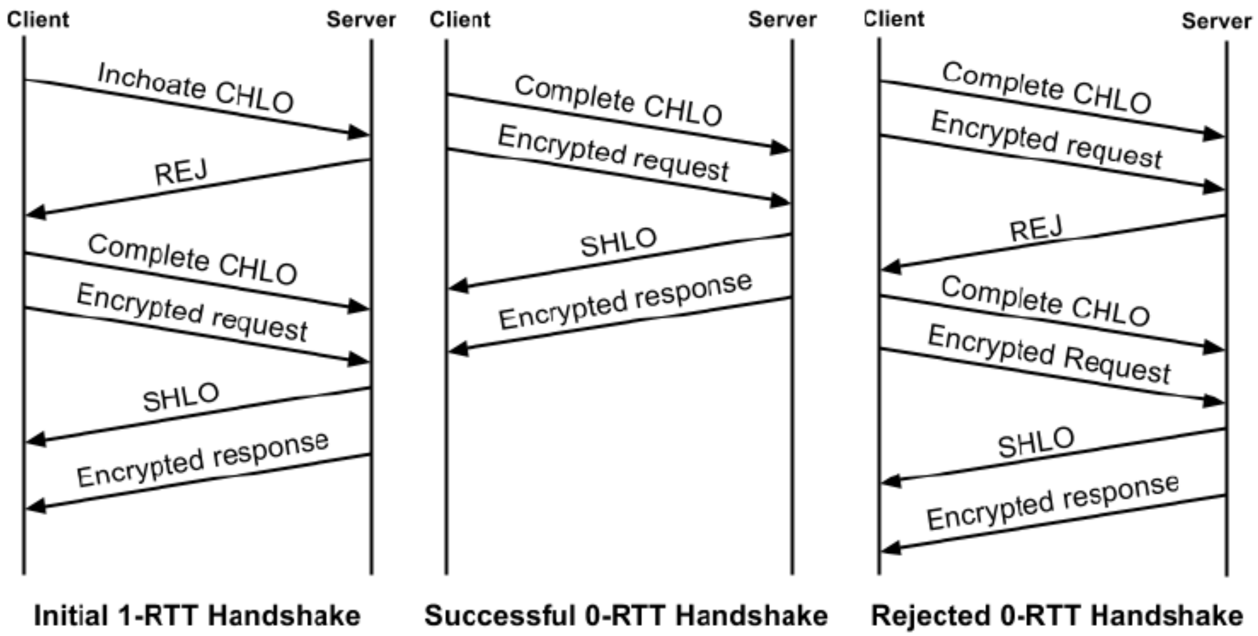
- Connection ID – unique connection identifier, instead of the socket.
- Allows migrating a connection between IP addresses and network interfaces.
- Allows resuming a stale connection.

QUIC

- Multiple streams transmitted over the same connection (similar to TCP).
- Streams are controlled both independently (flow control) and per connection.
- TLS exchanges are directly integrated.
- Unique segment identifier, even for retransmission, easing RTT measurement.
- Classical TCP congestion control mechanisms.

QUIC connection setup

- 0-RTT – to a known server, with previous QUIC and TLS connection.
- 1-RTT – if the TLS keys can be reused.
- 2-RTT – if everything needs to be set up.



QUIC

- 8% of the websites
- Google, Facebook, Cloudflare
- Chrome, Firefox, curl
- 5% reduction in search time
- 15% reduction in video rebuffering
- 85% of the connections are 0-RTT