Transport Layer Protocols

Unit 08 - Hands-On Networking - 2018

Prof. Dr.-Ing. Thorsten Herfet, Andreas Schmidt, Pablo Gil Pereira

Telecommunications Lab, Saarland Informatics Campus, 21st Feb. 2018

Recap

- Transport layer can provide:
  - Process-to-Process Communication
  - Connections
  - Ordered Delivery
  - Segmentation
  - Congestion Control
  - Flow Control
  - Reliable Communication
  - Pacing
- Go-Back-N and Selective Repeat are two approaches for pipelined reliable communication.

User Datagram Protocol (UDP)

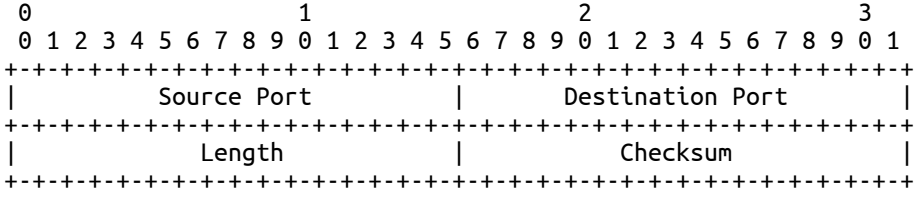
Source

UDP | Introduction

Properties

- Minimalistic, lightweight transport layer protocol (RFC768).
- Message oriented. Simple. Stateless.
- Fast (no limit by protocol, send as quickly as network interface card can do).
- No reliability (except validity check through checksum).
- Connection-less (no need to establish connection, just point at peer)

Header (Size: 8 Bytes)



- Ports: 2 Byte (`short` each).
- Length: 2 Byte (max. theoretical size 65507 = 65535 - 8B (UDP hdr) - 20B (IP hdr)).
- Checksum: 2 Byte (see later, optional in IPv4, mandatory in IPv6).

UDP | Typical Services

❓ Why/when to use UDP for your service?

- Protocol has inherent message types.
- Reliability might be required but only on a packet level.
- Time-awareness (e.g. packets expire).
- Throughput-constrained (e.g. need 1Mbps all the time).
- Loss-tolerant (e.g. multimedia).
- Small overhead (short header).

🏷 Examples

- DNS (Port `53`): see U05.
  - Request/Response messages are well-defined.
  - Resend complete packets if lost.
- Dynamic Host Conf. Protocol (DHCP) (Ports `67`, `68`): details see later.
  - Automatically configure hosts when they join a network.
  - No need for users to fiddle around with IP addresses.
- RTP: Low latency required. Many small packets desired. Loss-tolerant.

CRC32 | Internet Checksum

- Used to spot errors in the datagram (e.g. flipped bits). (*How could this happen?*)
- Calculated over complete IP pseudo-header, UDP header and payload.

↪ Sender

- Treat segment contents (and header) as sequence of 16-bit integers.
- Calculate checksum as negation of one's complement sum of segment contents (carry-bit wraps around).
- Checksum entered into UDP checksum field.

↩ Receiver

- Compute sum of received segment contents.
- Check if computed checksum equals -0 (1111...1111):
  - No - error detected.
  - Yes - no error detected.

⚡ Error might be present but don't get noticed! See later...

CRC32 | Example

```
A:    1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
B:    1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

-----------------------------------

W:  1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1

-----------------------------------

S:    1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
C:    0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
```

Transmission Control Protocol (TCP)

TCP | Introduction

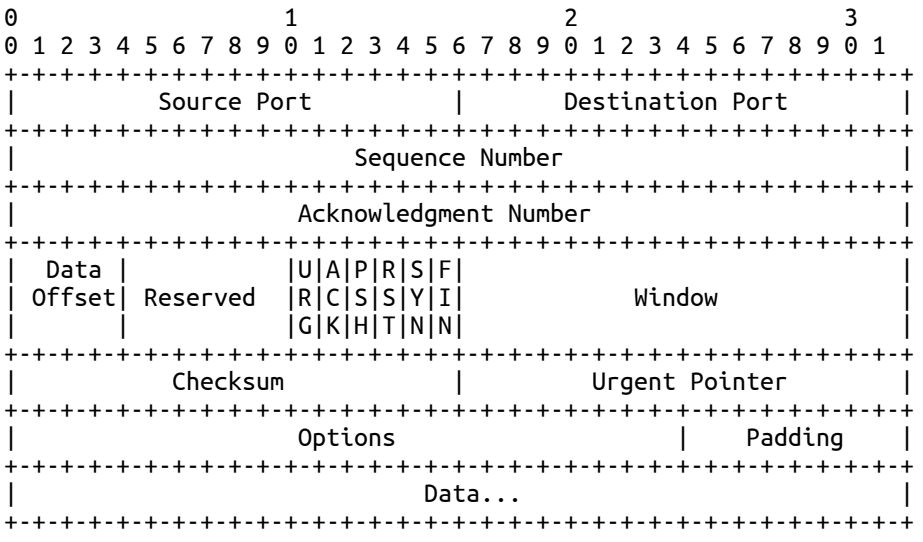
Motivation

- UDP: Simple protocol (send & pray).
- Many applications require:
  - *Reliability*: All data arrives.
  - *Receiver Buffer Awareness*: Not more data than fits in the buffer is sent at a time.
  - *Congestion-Free Network*: To ensure data is not lost in transit.

Transmission Control Protocol (RFC793)

- Fulfills requirements mentioned on the left.
- *Connection-oriented*: Handshake for establishment, handshake for termination.
- *Stream-oriented*: Single bytes are transmitted in variable size segments. Message framing is open to application.
- *Throughput-constrained*: Flow and congestion control (see later).
- *Fully reliable*: Combination of Selective Repeat and Go-Back-N.

TCP | Header



Fields and Sizes

- Src / Dst Port: 16 bits each
- Data Offset: 4 bits
- Flags: 6 bits
- Checksum: 16 bits
- Urgent Pointer: 16 bits
- Sequence Number: 32 bits
- Acknowledgment Number: 32 bits
- Reserved: 6 bits
- Window: 16 bits

TCP | Typical Services

❓ Why/when to use TCP for your service?

- Transport should be reliable.
- Transport is not time constrained.
- Throughput can be elastic.

🏷 Examples

- HTTP (`80`): see U04.
- File Transfer Protocol (FTP) (`21`):
  - Safely transmit large files.
  - Elastic throughput requirements (done when done).
- Secure SHell (SSH) (`22`):
  - Cryptographic network protocol for remote login.
  - Transmission must be reliable.
  - Low latency appreciated but not always necessary.

Reserved Bit Pattern

Context:

- When specifying a protocol, sizes have to be chosen and headers be defined.
- Requirements of protocols evolve over time and might need more meta-data fields.
- Adding fields is hard, because backward-compatibility is risked.

Implementation:

- Design headers in a way that bits in certain areas are reserved.
- Leave some bits intentionally unused.
- Advise implementations to set them to zero by default.

Benefits:

- Protocols stay backward-compatible, when client and servers repurpose reserved bits for transmitting new information.
- Protocols have evolvability, as fields can be added.

Drawbacks:

- Performance penalties (sent but unused bits).
- Reserved bits: No unlimited resource.

Similar Patterns:

- Overdimensioned fields.
- Repurpose fields (e.g. TCP cookies).

Reserved Bits - "Mis"-Uses

Reserved bits are an important topic for security research and can be used for:

Exploits / Denial-of-Service

- Poorly written server software might not consider other values for reserved bits than 0.
- If really poorly written, this can cause crashes.

Covert Channels

- Transmit data with the intent of hiding it.
- Compare Johnson, Jajodia 1998 "Steganalysis".

Fingerprinting / OS-Detection

- Content of reserved bits not specified or mandated.
- Implementations free to overwrite bits or leave unchanged.
- Developer choices used to identify implementations.
- `nmap` and other tools exploit this.

Robustness Principle

Motivation

- Reserved bits might cause problems (see before).
- Protocol versions of sender and receiver might not match.
- Implementations might not follow standards wholeheartedly.
- Standards might be incomplete.

Approach

Robustness Principle by Jon Postel - Author of RFC793: TCP.

- Be conservative in what you do.
  Follow standards as closely as possible.
- Be liberal in what you accept from others.
  Don't expect others to follow it and accept things that do not contradict the standard.

You might also consider this as a *Pattern* for designing robust communication.

TCP | Connection Establishment

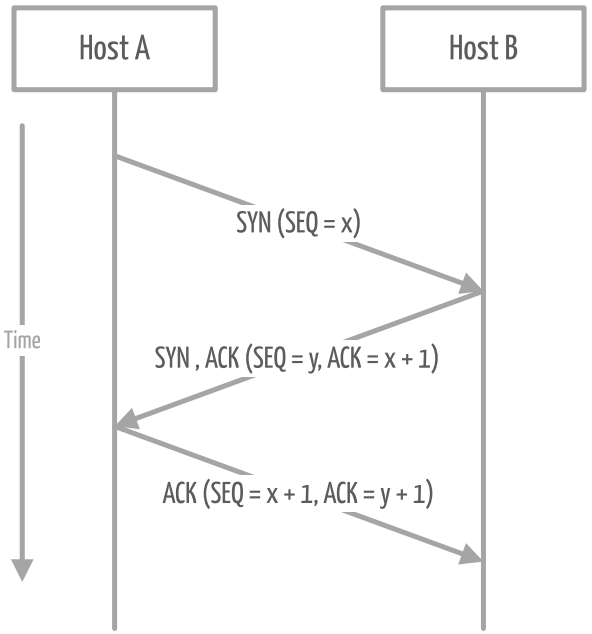
Three-Way Handshake

1. Host A: Connection Request (TCP segment with SYN flag set).
2. Host B: Acknowledge SYN (ACK) and connects back (SYN).
3. Host A: Acknowledge SYN (ACK).

Result

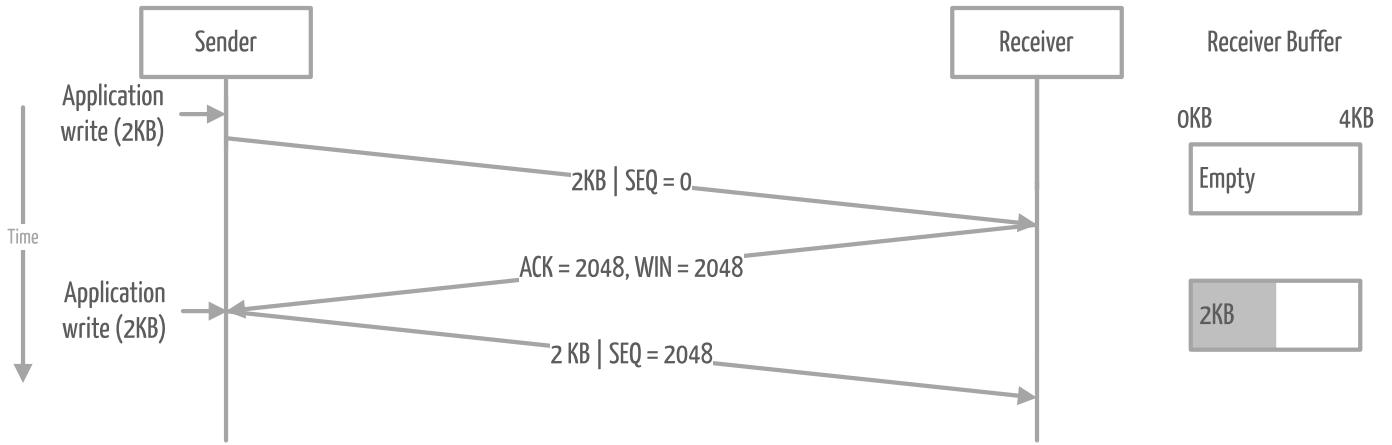
- Bi-directional connection.
- Two independent directions.
  - A to B and B to A.
  - Sequence Number and Window Size maintained per direction.

Process



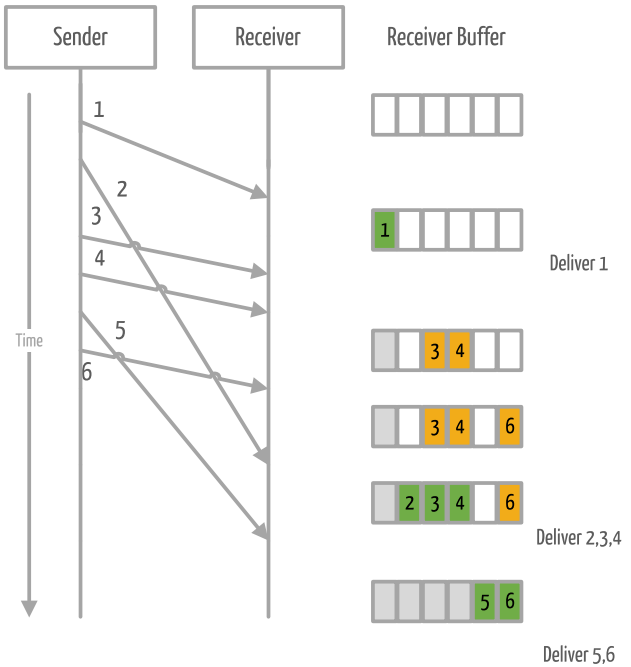
TCP | Sequence Numbers

- TCP makes use of sequence numbers to implement Go-Back-N.
- Sender specifies sequence number in segments.
- Initial sequence number per direction is chosen at random in handshake (🕵).
- Per byte sent, the sender increases the sequence number.
- The receiver ACKs the sequence number until which it received all previous bytes.

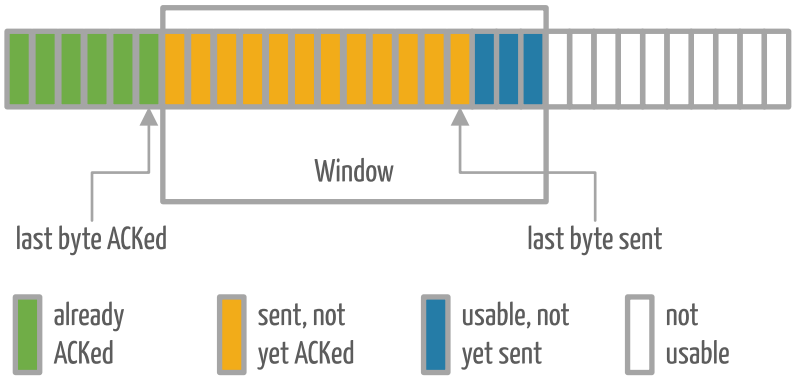


TCP | Ordered Delivery

- TCP sockets store packets to be retrieved by app layer.
- Packets above the currently awaited sequence number are buffered.
- Delivery to app layer starts when currently awaited byte is present.
- Data delivered until the next not-yet arrived byte.



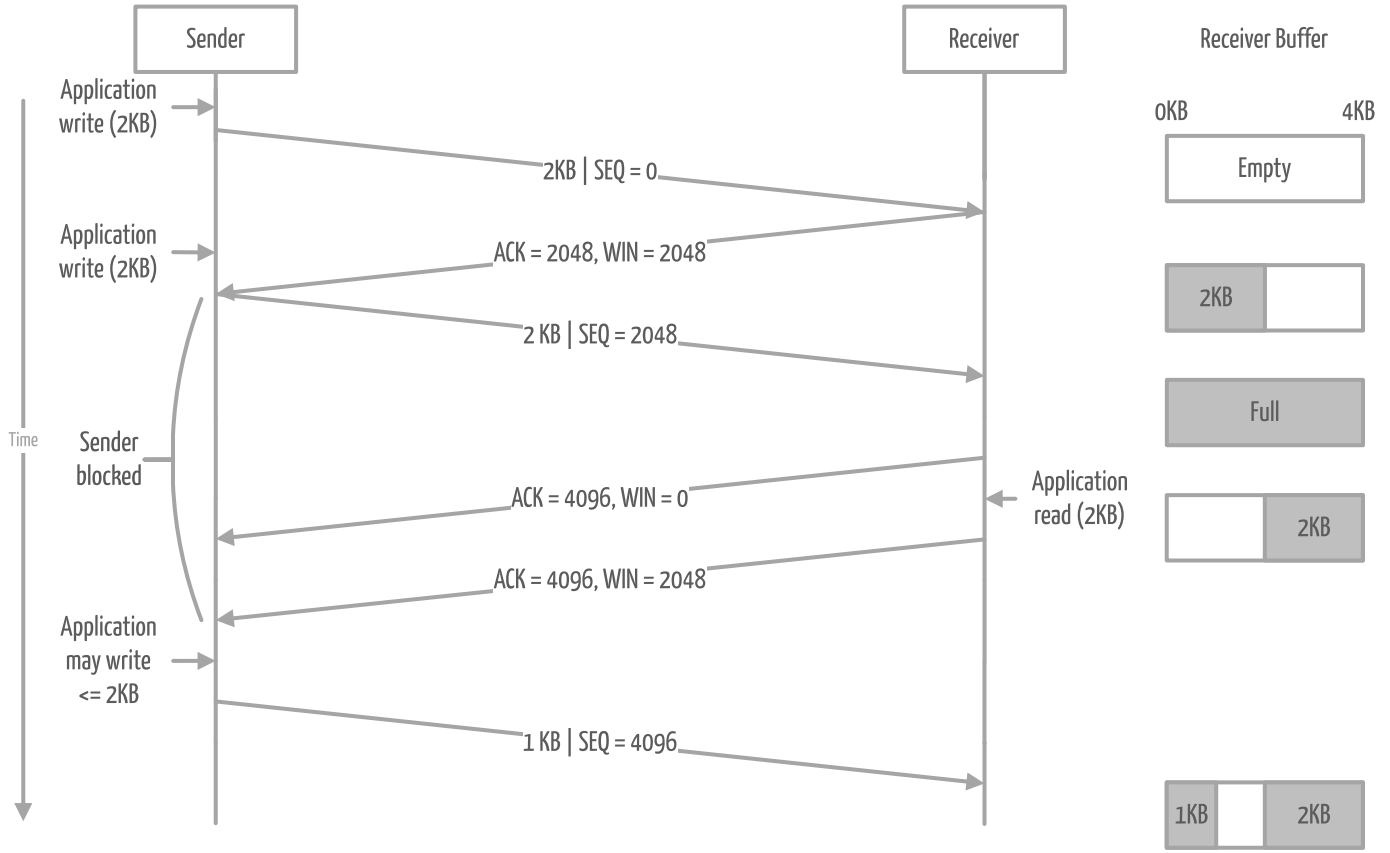
TCP | Window Size



Which window size is appropriate?

- Affected by flow and congestion control.
- *Sliding window* is given by receiver buffer (flow control).
- *Congestion window* is learned by sender (congestion control).
- Sender uses smallest of the two windows as window size.

TCP | Flow Control



TCP | Error Control / Reliability

- TCP uses a schema called Automated Repeat reQuest (ARQ).
  Redundant data is sent upon detecting a loss. Reactive scheme.
- Loss detection works via a timer.
  Upon timer expiry, packet considered lost. Although could just be overly delayed.
- Timer configuration is a complex topic of its own.
  Not dealt with here.
- TCP ensures full reliability.
  There are no means to tell the stack that we are no longer interested in a byte. As long as a single byte is not delivered to the application layer, no subsequent bytes will be delivered.
- TCP uses a combination of Go-Back-N and Selective Repeat.
  The result is conceptually similar to our rdt3.0 protocol with pipelining.

TCP Congestion Control (CC)

What is Congestion?

Common Speech

con·ges·tion
/kən'jesCH(ə)n/

noun

the state of being congested.
"the new bridge should ease congestion in the area"
synonyms: crowding, overcrowding; More

Transportation System

- The system is overloaded.
  More incoming *requests* as served.
- Units experience large delays.
- Units might not be served.

Source

BDP as a Natural Resource

BDP describes how many bytes can be in a link at the same time.

Know the Limits

- Links cannot transmit more packets as they can fit in.
- Additional packets must be stored somewhere (e.g. in buffers).
- Buffers are always limited. If full, additional packets are dropped.
- Amongst others, this is one reason why packets get lost.

Do not exhaust resources!

Be Fair

- Users contend to use a portion of the BDP to send their data.
- Best effort: The network does its best to serve all, by treating all packets equal.
  Also known as net neutrality.

Share resources in a fair way!

A Perfect Day on The (Data) Highway

How can we totally avoid congestion on a highway? While still allowing people to travel...

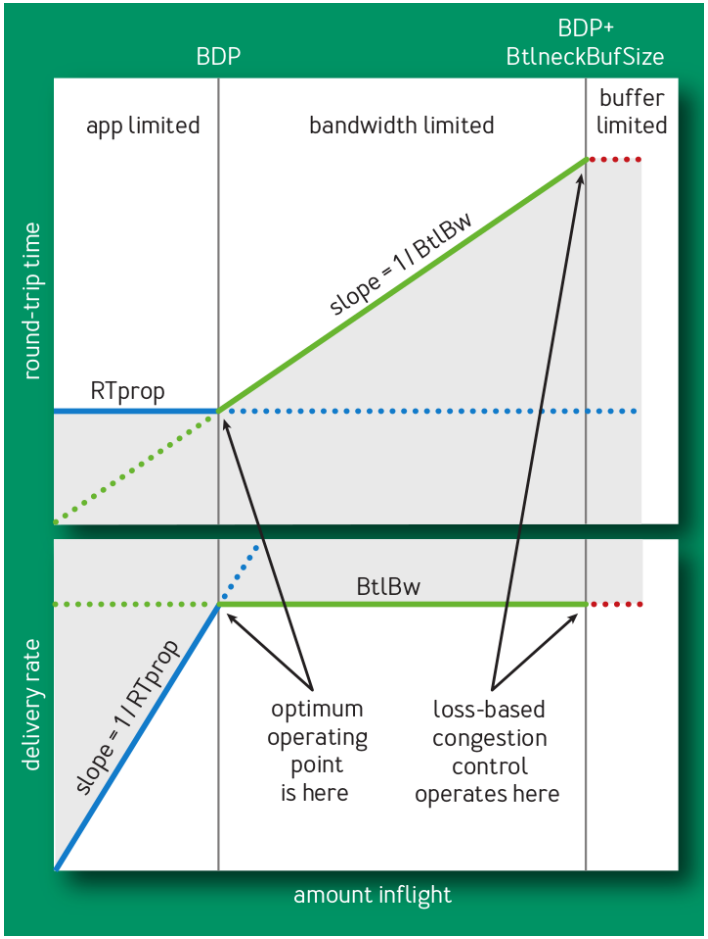
(Highway) Congestion-Avoidance Criteria:

1. Cars should only enter at the speed at which they can move forward.
2. There should not be more cars than the highway can fit.

Data Highway Congestion-Avoidance Criteria:

1. Packets should enter at a rate that is at max. the throughput (*pacing rate*).
2. There should not be more packets than the link can fit (*window size*).

Delay and Bandwidth



- Size of sending window determines delay and delivery rate.
- Delay can never be below *propagation delay* $RTprop$.
- Delivery rate can never be above *bottleneck bandwidth* $BtlBw$.
- Optimal operating point is BDP, as we have minimal delay and maximum throughput [Kleinrock1979].
- There is no distributed algorithm that can converge at this optimal point [Jaffe1981].

Source: [Cardwell2016]

Buffering

- Buffers increase latency as they store the packets.
- If we would not use buffers, we would not face additional delay, but losses.
- For avoiding these losses, networks and sending processes would need to be perfectly synchronized.

This is not feasible, as it is **complex**, **expensive** and **restrictive**.

Control **sending behaviour** and provide **large enough buffers** for stability.

TCP CC | General Process

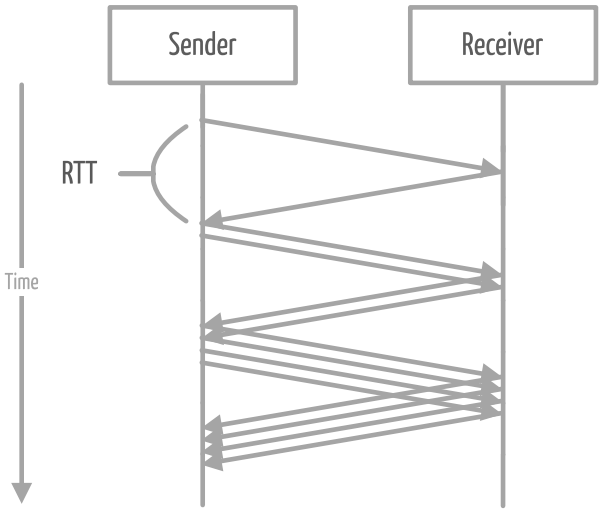
- Congestion window (`cwnd`) and slow start threshold (`ssthresh`) variables maintained at sender.
- Window changes using the following phases:
- Slow Start
  Initial probing to find fair share of throughput without congestion. Mode also sometimes used to recover from loss events.
- Congestion Avoidance
  Try to approach a maximal throughput, without causing (additional) congestion on network.
- Fast Recovery / Fast Retransmit
  Mild congestion detected. Back off, but reduce throughput only a bit.

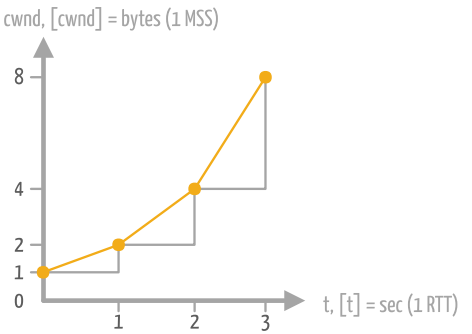
More details on this can be found in RFC5681.

TCP CC | Slow Start

- Initially `cwnd` = 1 MSS (maximum segment size) and `ssthresh` = ∞.
- While there is no loss event: Increment `cwnd` per ACK received.
- Initial rate is slow but increases exponentially.
- Effectively the sending rate is *doubled every RTT*.

RTT-Dependence
We increase every RTT, so for long RTTs it takes long to send fast.

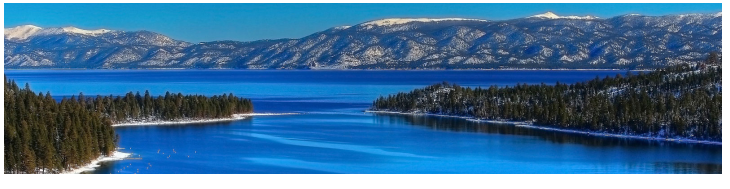




TCP CC | Congestion Avoidance

Loss Handling

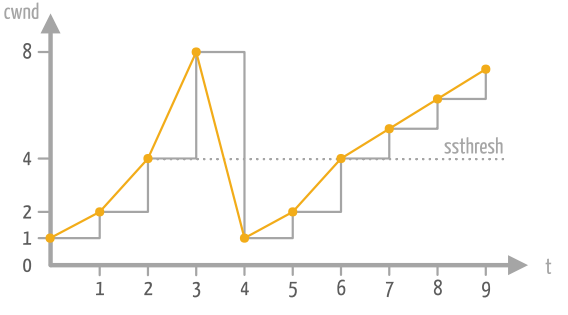
- Loss event indicated by timeout.
- Threshold adaptation: `ssthresh` = `cwnd` / 2.
- Sending window reduction: `cwnd` reset to 1 MSS.
- Continue with *slow start* until `cwnd` reaches `ssthresh`.
- In case of another loss, repeat this process.



Lake Tahoe Source

Congestion Avoidance

- Increment window size by MSS/`cwnd` per received ACK.
- Rate increases linearly.
- Effectively the sending rate is *increased by 1 every RTT*.
- On timeout: See left.



TCP CC | Fast Recovery

Different Types of Losses

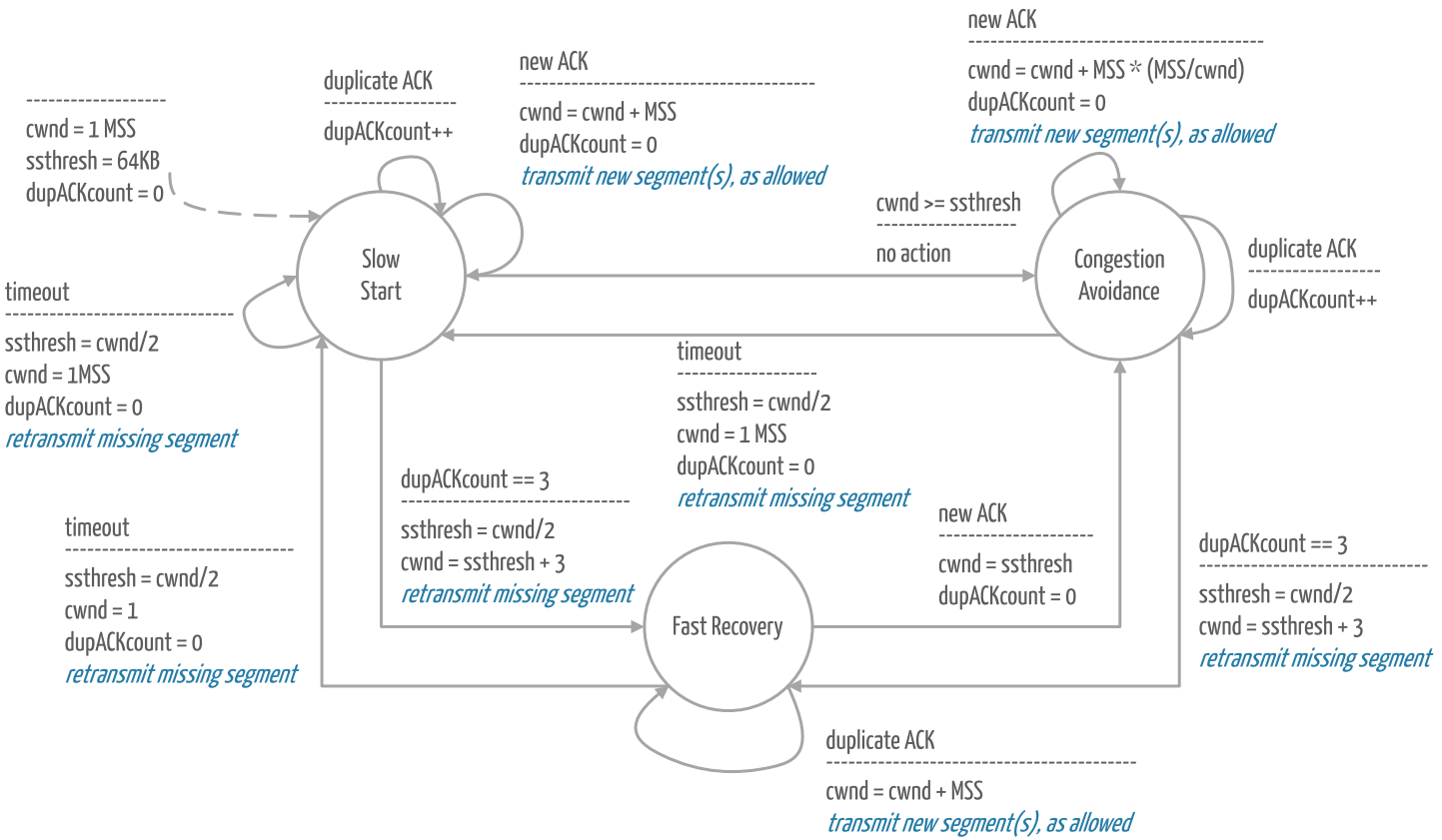
- Sometimes a single packet gets lost without congestion (e.g. physical causes).
- TCP Tahoe will consider this as loss event and return to slow start.
- Performance penalty, as throughput is significantly reduced.
- Duplicate ACKs (ACK for packet already ACKed) are considered:
  - Duplicate ACKs indicate no full congestion.
  - Don't reduce to minimal window.

Fast Recovery

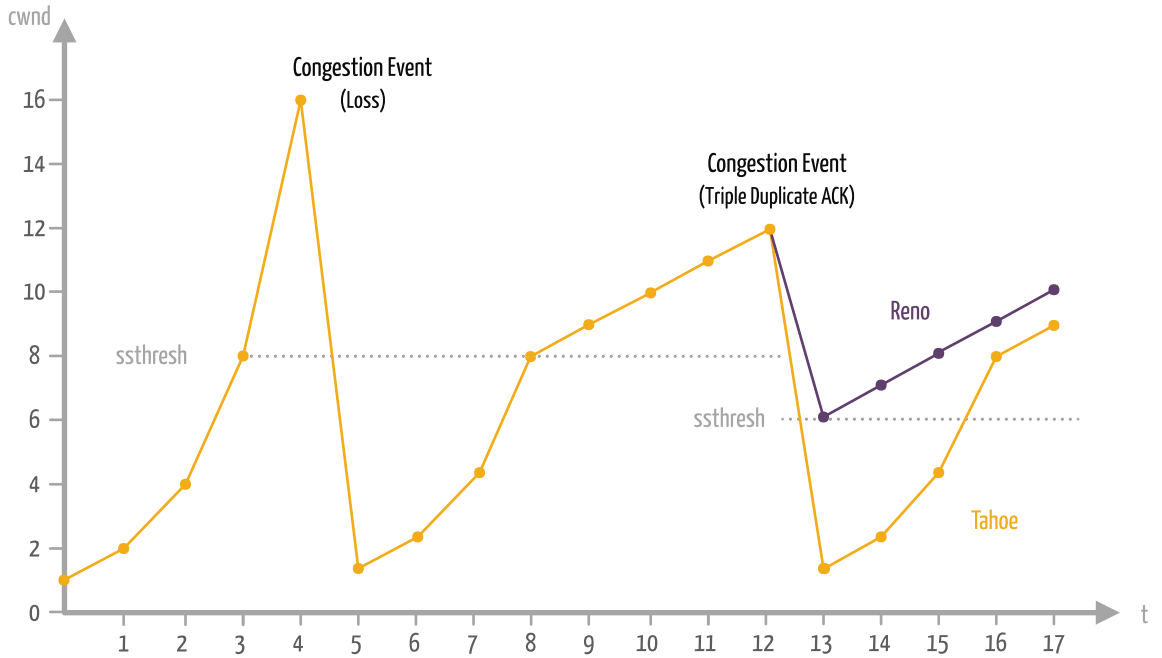
- When receiving three duplicate ACKs in sequence before timeout:
  - `ssthresh` = `cwnd` / 2
  - `cwnd` = `ssthresh`
  - Enter congestion avoidance phase *immediately*.

Reno Source

TCP CC | State Graph

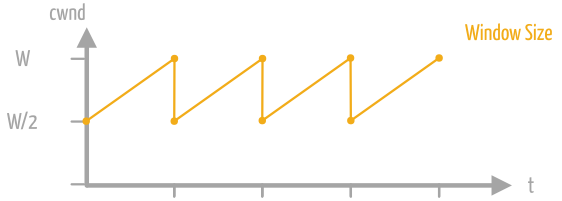


TCP CC | Reno vs. Tahoe



TCP | Throughput

Average Throughput



- Function of window size W (indicates the window size where losses occur) and RTT: $\frac{3}{4} \cdot \frac{W}{RTT}$
- Assume no slow start and there is always data to be sent.

TCP Inefficiencies

- Example: 1500 byte segments. 100ms RTT.
- Throughput incorporating loss probability L ([Mathis 1997]):
$$\frac{1.22 \cdot MSS}{RTT \cdot \sqrt{L}}$$
- Goal: 10 Gbps throughput. Requires $L \approx 10^{-10}$.
- TCP's congestion control algorithms have to be efficient.

TCP | State of the Art and Research

CUBIC [Rhee 2008] is part of the Linux kernel since version 2.6.19.

- Uses a cubic function to increase fast but stay at presumed channel capacity for long time, before probing again for higher throughput.
- Default algorithm in many distributions.

BBR [Google 2016] is part of Linux kernel 4.9.

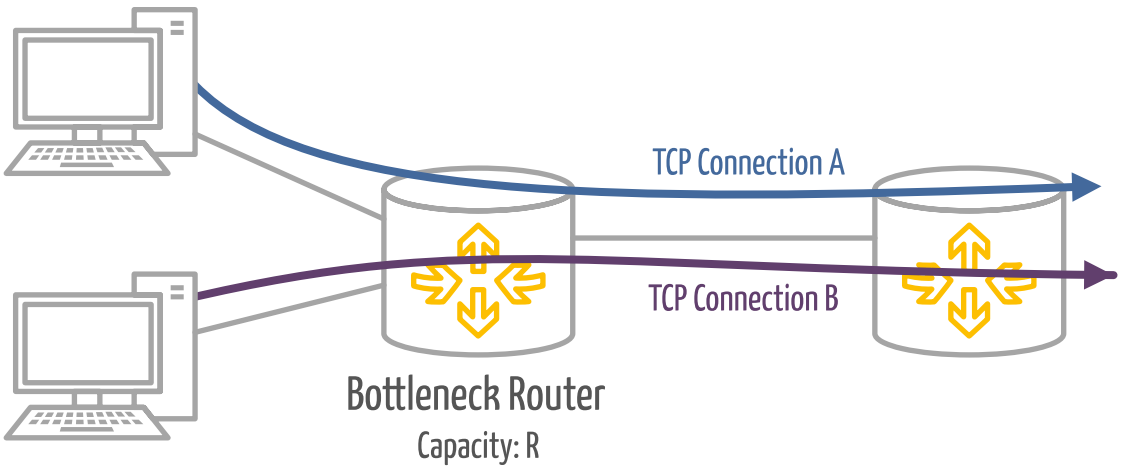
- Uses congestion-based instead of loss-based detection.
  Constantly measures the BDP.
- Tests by Google showed that it is better than CUBIC most of the time and regarding most of the metrics.
  Might be the replacement for CUBIC in the near future.

Certain **Congestion Control Algorithms** perform better, when the RTT is reduced.

- We [Schmidt, Herfet 2016] are investigating mechanisms to reduce effective RTTs and make `cwnd` adapt faster to what is possible on the channel.
- How BBR and our approach interact has to be investigated soon.

TCP | Fairness

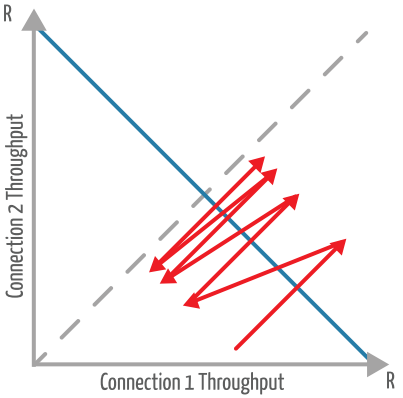
◎ **Goal:** With K sessions sharing the same link with bandwidth R, each session should get average rate of R/K.



TCP | AIMD

Algorithms are called "Additive-Increase-Multiplicative-Decrease", because:

- Data rate increase with slope of 1.
- Data rate decrease is proportional to throughput.

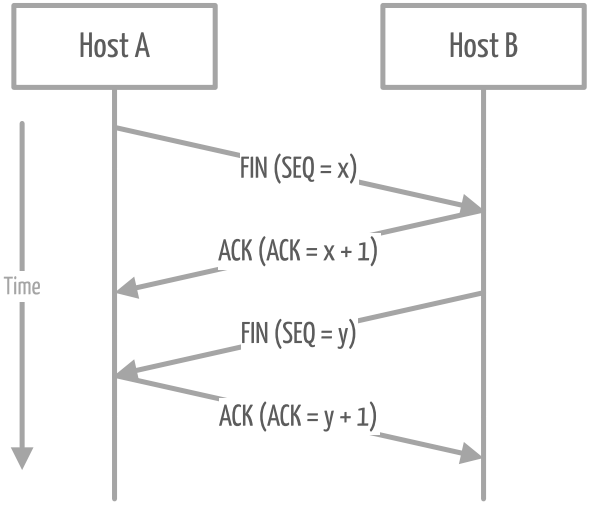


TCP | Connection Teardown

Four-Way Close

- Each sides closes connection (segment with `FIN=1`) and waits for ACK (two steps per side).
- The middle two steps can be combined into one using a `FIN,ACK`.
- **Result:** Both sides have properly closed their connection and can release resources.

Process



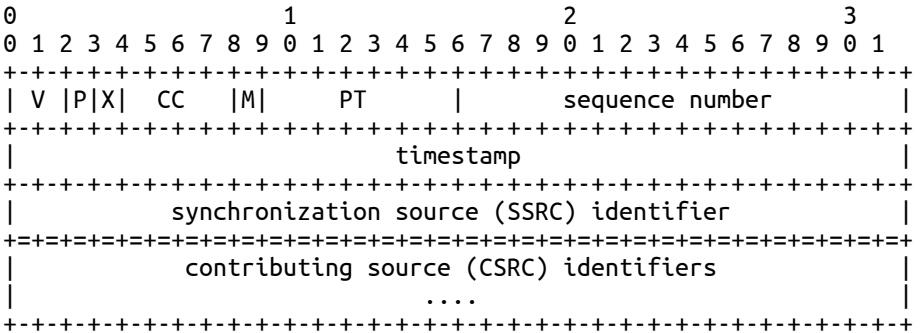
You could also end a connection by setting `RST=1`. That would as impolite as ending the phone conversation by slamming the phone on the cradle. This should be left to error cases, where one hosts thinks the other misbehaved.

Other Transport Layer Protocols

Real-time Transport Protocol (RTP)

- Provides common fields required by multimedia applications:
  - Payload Type (multiple profiles supported).
  - Sequence Number (identify packets).
  - Timestamp (relative time when the payload was sampled).
  - Source Identifiers (video, audio streams combined).
- Details specified in RFC3550.

Header



Predictably Reliable RT Transport (PRRT)

- UDP-based multimedia transport protocol developed at the TC chair.
  Check it out at http://prrt.larn.systems
- Error correction scheme:
  - Combines retransmissions (ARQ) of TCP with Forward Error Coding (FEC) approaches (see later).
  - Parameters chosen adaptively, while the channel changes.
  - Incorporates application requirements to optimize coding.
- Features:
  - Better approach channel capacity using FEC and ARQ.
  - Incorporates BBR for congestion control.
  - Multicast-enabled (no 1:1 connections as TCP).
  - C and Python APIs to be used in network projects.

At the Bar

A UDP packet walks into a bar, no one acknowledges him.

A TCP packet walks into a bar twice
because no one acknowledged him the first time.

Source

HON

Wrap-Up

Questions?

Take-Home Messages

- **UDP** should be used to implement **datagram-oriented**, **connection-less services**.
- **TCP** should be used to implement **reliable byte-streams** with **fairness**.
  - Receiver buffers never overflow with TCP.
  - **Congestion control algorithms** steer throughput (think of Reno and Tahoe).
  - Connections are properly **setup and torn down**.
- **Special transport protocols** for multimedia exist and provide certain benefits.

Further Reading

- Kurose-Ross "Computer Networking"
  - Sec. 3.3 (UDP)
  - Sec. 3.5 - 3.7 (TCP)