Application & Transport layer

TA Recap session 2026 - Malte W.

Overblik

Application layer:

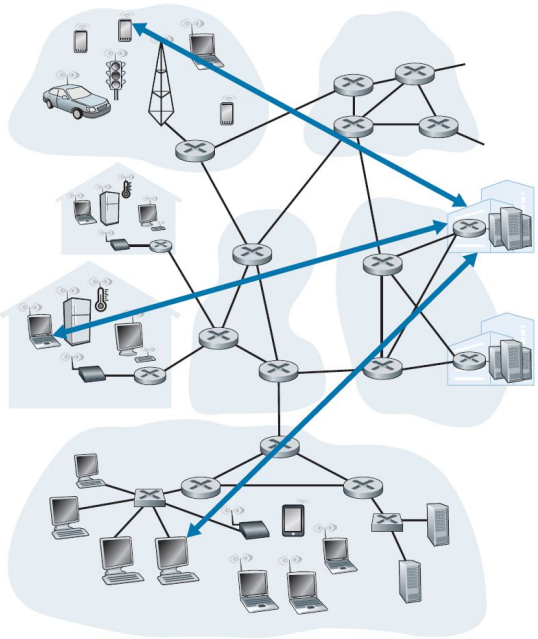
- **Client server & p2p model**
- **HTTP & DNS**

Transport layer:

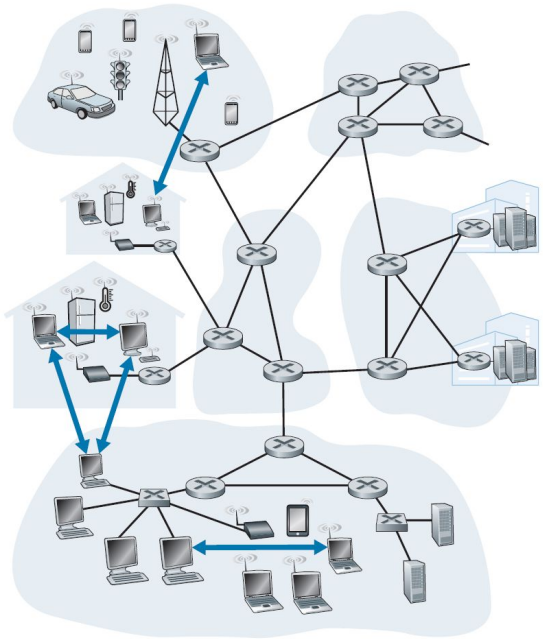
- **UDP & TCP**
- **Flow control**
- **Congestion control**

Application layer

Network Architectures



a. Client-server architecture



b. Peer-to-peer architecture

- P2P complexity (A3)
- Client-server single point of failure
- P2P for resource sharing applications (BitTorrent, Crypto currency, etc…)
- Client-server for centralized control and security (Games, Email, etc..).
-

Exam question related to network architecture

Question 3.1.2: Congratulations, you've just started a job working for a new startup company. The company is building drones to deliver parcels throughout Denmark, with potentially millions of drones in flight at once. We will assume that you have some top-notch colleagues who have designed some top-notch drones, that if a drone knows its own position, and the position of any other drone, then they will never crash into each other. Your boss has read about the client-server network architecture, and thinks it is a good idea to have all drones regularly tell their position to a server at the main office. Each time that server gets a message from a drone it can reply with all the positions of all the drones, so the drone has all the info it needs to avoid a crash.

Your job is to improve this system in whatever way you can. What improvements can you suggest? Justify your choices, and state what effects you think your improvements might make. You should also state any assumptions you make about the drones, the company or anything else relevant.

P2P Architecture: Avoid single point of failure, decentralize the drones!

Application layer protocols

HTTP

- HTTP er stateless
- Består af requests & response.
- Eksempel på en client request

```
GET compSys/ HTTP/1.1
Host: absalon.ku.dk
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
```

DNS

- Oversættelse af hostnames til IP addresse (“google.com” -> 142.250.74.174)
- Opslag i DNS hierarkiet

- HTTP Gør brug af DNS

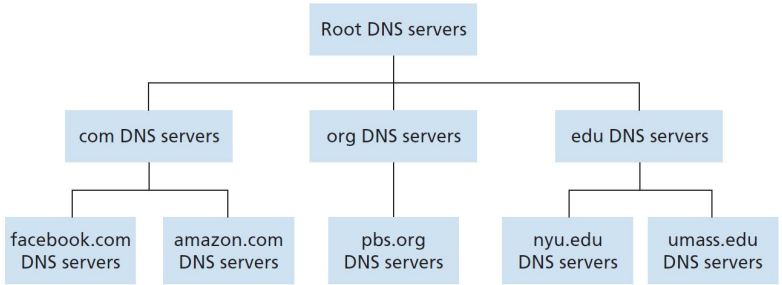


Figure 2.17 ♦ Portion of the hierarchy of DNS servers

Exam question related to HTTP & DNS

Question 3.1.1: What does it mean when we say that HTTP is a stateless protocol, and what are the advantages of this approach?

The server does not keep any information about the client, each request is treated as its own transaction.
Avoids a lot of server side logic of keeping track of clients, more concurrent requests possible, using very little buffer space on server side.

Question 3.1.1: Why is DNS not a centralised system run from a single server? Explain how DNS works in practice, and how a client request is dealt with by the DNS system. For this answer you can assume there is no DNS caching.

Single point of failure, high latency due to geographical distance, distributed approach spreads the workload across many levels.

First checks local DNS (not there because no DNS caching), local DNS will request from the root DNS which will determine which TLD can parse the request. It then sends the domain to the local DNS, which then requests the authoritative DNS for the IP, which is then send back to the local DNS and finally to the client.

Transport layer

TCP & UDP

TCP

- Reliability: TCP sørger for at data kommer korrekt (checksum, ACK, retransmissions)
- Connection: Før data kan blive sent, skal en connection etableres (3-way handshake)
- Flow control: Kontrollere mængden af data der sendes på en gang,.
- Congestion control: justere sende raten baseret på netværket.

UDP

- Unreliable: Garantere hverken rækkefølgen af pakker eller om de overhovedet kommer.
- Connectionless: UDP opsætter ikke en connection som TCP, hver pakke sendes uafhængigt.
- Low overhead/fast speed: UDP header er kun 8 bytes (TCP er 20). Dette plus at det ikke kræver handshake, gør UDP en del hurtigere end TCP.

TCP & UDP headers.

UDP segment; 8 byte header

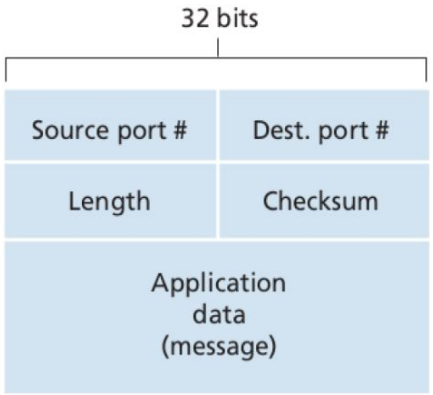


Figure 3.7 ♦ UDP segment structure

TCP segment; (mindst) 20 byte header

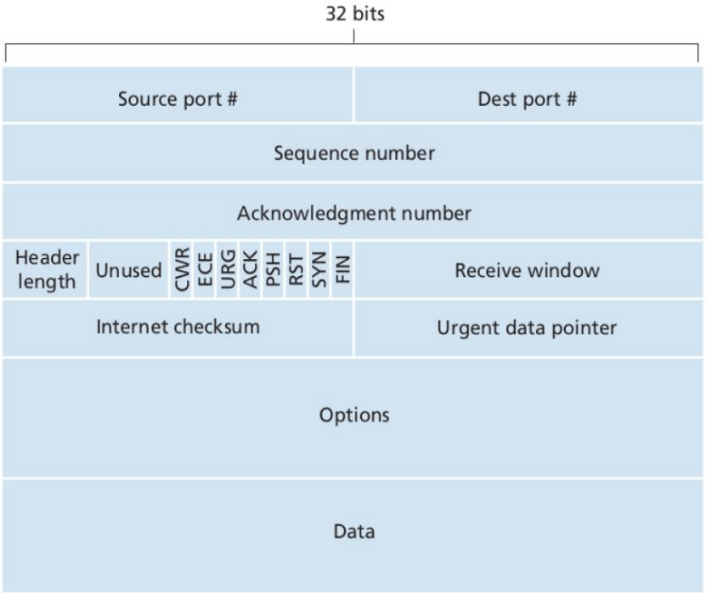


Figure 3.29 ♦ TCP segment structure

TCP Handshakes - Syn flooding?

Initialisering (3-way handshake).

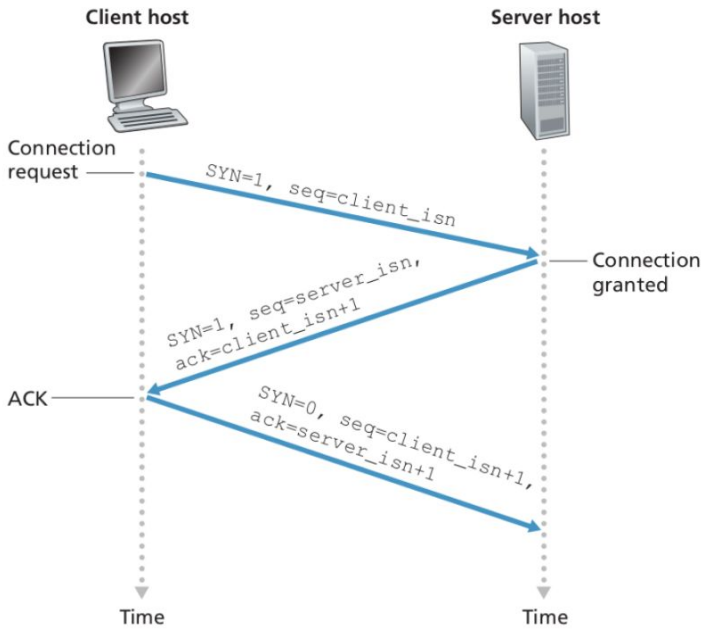


Figure 3.39 ♦ TCP three-way handshake: segment exchange

Terminering (4-way teardown).

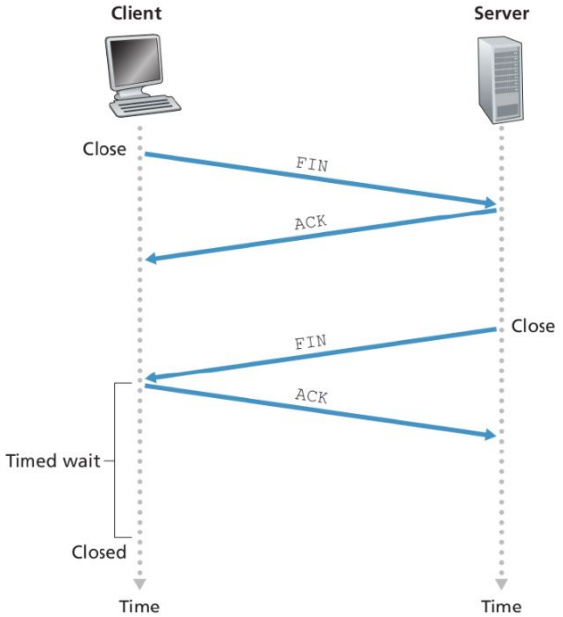


Figure 3.40 ♦ Closing a TCP connection

Flow control - GBN

GBN (Go back N)

- Senderen kan sende op til N pakker uden at have modtaget ACK.
- Discarder out-of-order pakker (Ingen buffer på modtagers side).
- Ved tripple ACK starter afsender forfra ved sidste seq
- Afhængigt af window size, kan det være rigtigt dyrt at skulle gensende

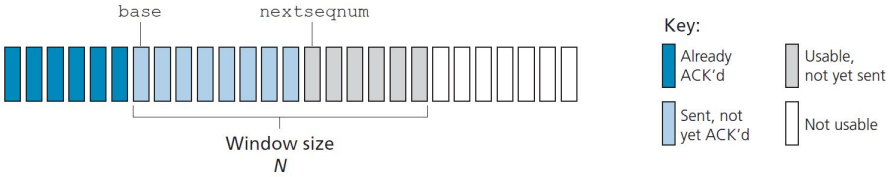


Figure 3.19 ♦ Sender's view of sequence numbers in Go-Back-N

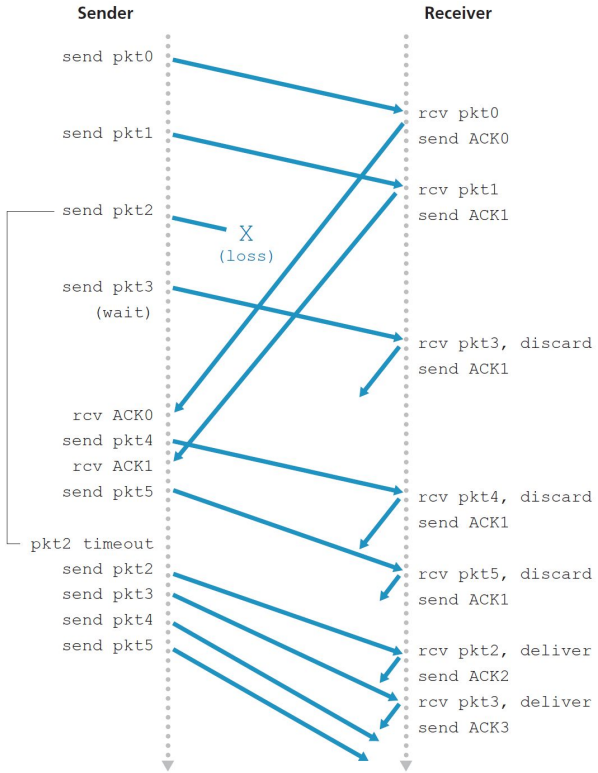


Figure 3.22 ♦ Go-Back-N in operation

Flow control - Selective repeat

Selective repeat - SR

- Bruger buffer til out-of-order pakker
- Retransmitter kun tabte eller korrupte pakker (ikke Base til Base+N som GBN)
- Sliding windows hos både afsender og modtager.
- Kræver buffer på modtager siden.
- Bruges af TCP

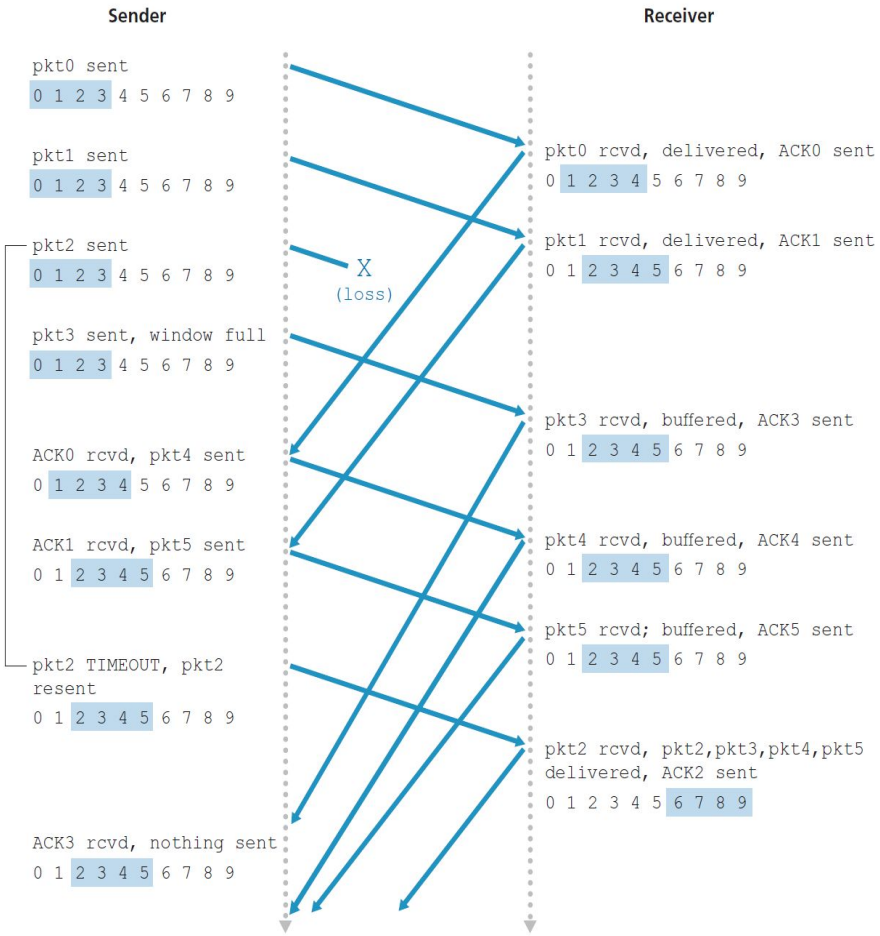
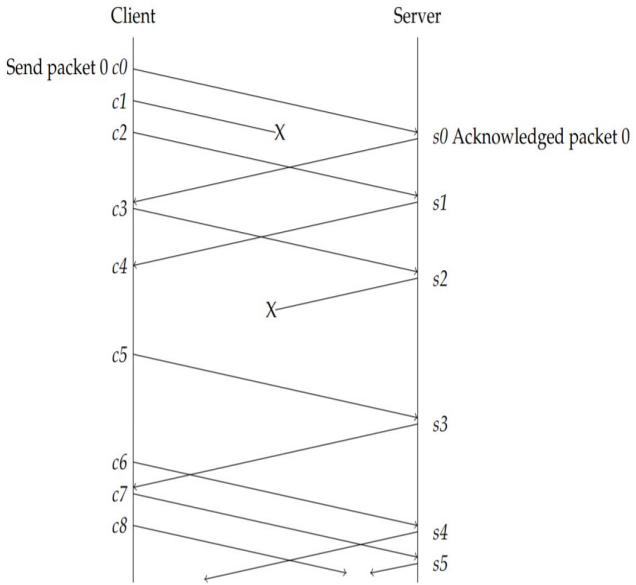


Figure 3.26 ♦ SR operation

Exam question - SR graph

Antagelser? windowsize?

Consider the following TCP communication diagram, showing a Selective Repeat Protocol, with package losses denoted by an X.



c0	Sent packet 0
c1	Send p1
c2	Send p2
c3	rec ack0 og send p3
c4	rec ack2
c5	p1 timeout og resend p1
c6	p3 timeout og resend p3
c7	rec ack1 og send p4
c8	Send p5

s0	Acknowledged packet 0
s1	ack p2
s2	ack p3
s3	ack p1
s4	ack p3
s5	ack p4

Congestion control

Congestion control stadier

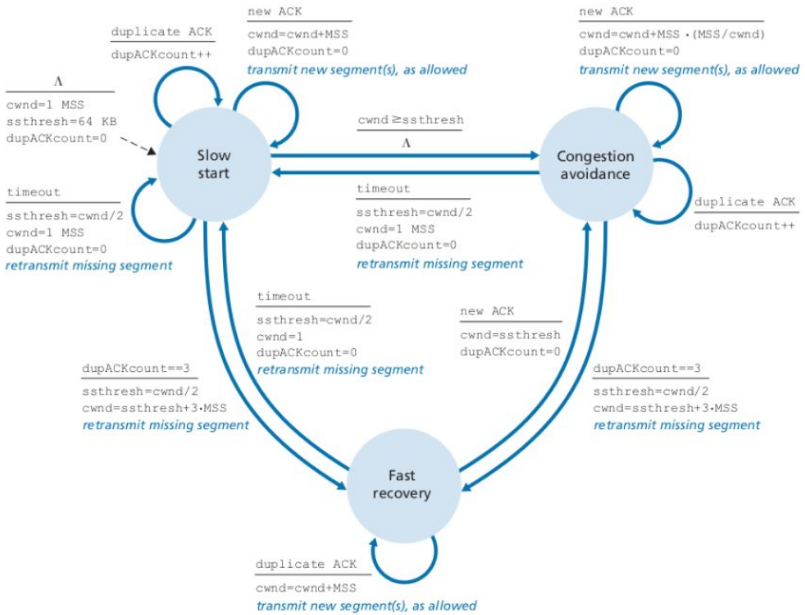
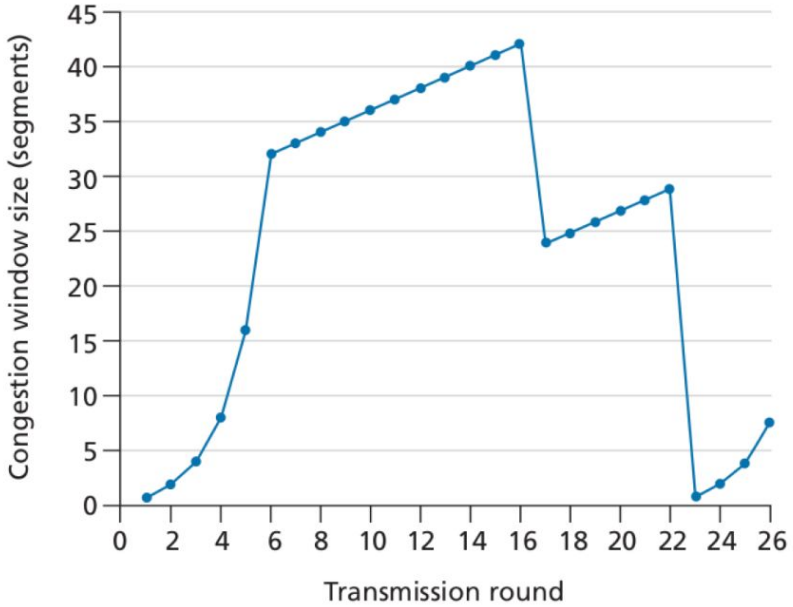


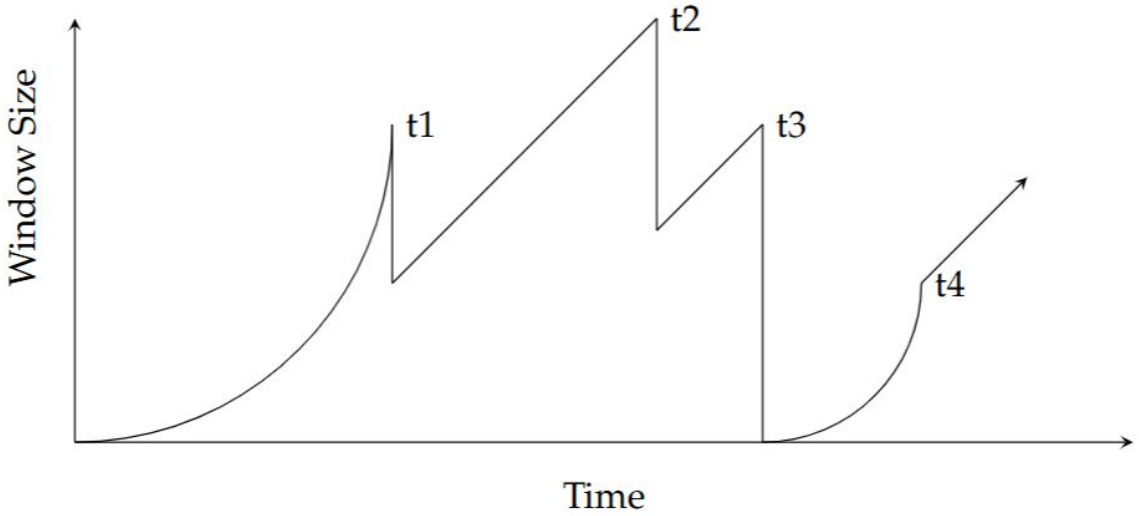
Figure 3.51 ♦ FSM description of TCP congestion control

Udvikling af congestion window



Exam question - Congestion control graph

Question 3.2.1:



The above diagram show packet loss occurring at 3 different points (t1, t2, and t3). No packet loss has occurred at t4, but what has happened instead? What can you deduce about the nature of the packet loss in either t1, t2 or t3 from this?