Mittuniversitetet
MID SWEDEN UNIVERSITY

Datornätverk
Transportlagret

Lennart Franked

13 maj 2026

Mittuniversitetet
MID SWEDEN UNIVERSITY

1. Transportlagret

2. User Datagram Protocol

3. Transmission Control Protocol

Mittuniversitetet
MID SWEDEN UNIVERSITY

Läsanvisningar

Denna föreläsning är baserad på

- [1, Chapter 9]

Denna föreläsning är ett komplement till läsanvisningarna.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Table of Contents

1. Transportlagret

2. User Datagram Protocol

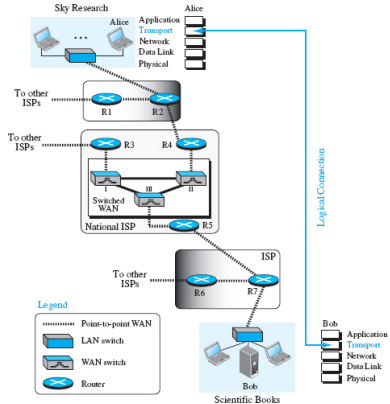
3. Transmission Control Protocol

Mittuniversitetet
MID SWEDEN UNIVERSITY

Transportlagret

- Transportlagret ligger mellan nätverkslagret och applikationslagret
- Om IP förbinder datorer, så förbinder transportlagret processer.
- Eftersom detta är ett protokoll som förbinder processer kan man här också styra hur datat flödar mellan processerna.

Kommunikation mellan processer

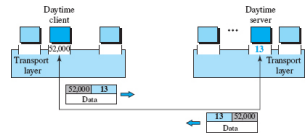


Figur: Transportlagret [1]

Portnummer

Portnummer

- IP-numret anger vilken dator paketet skall nå
- En dator kör oftast flera applikationer/processer
- Port-nummer används för att styra datat till rätt applikation/process.
- Port-nummer används också för att identifiera sändande applikation.



Figur: Användning av portar [1, fig9.3]

Portnummer

Well-known

- 0 — 1023
- Tilldelade av ICANN

Registered

- 1024 — 49151
- Tilldelas inte av ICANN, men går att få registrerade.

Dynamic

- 49152 — 65535
- Ej kontrollerade.
- Används av sändare för att säkerställa att svarspaket hamnar rätt

Mittuniversitetet
MID SWEDEN UNIVERSITY

Sockets

Socketadress

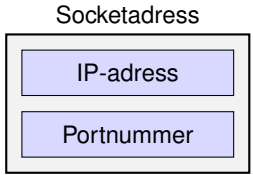
- Kombinationen av en IP-adress och ett portnummer
- IP-adressen identifierar värden (datorn) på nätverket
- Portnumret identifierar den specifika processen/applikationen på värden
- Skrivs ofta på formen `IP:port`, t.ex. `192.168.1.10:80`

Socket

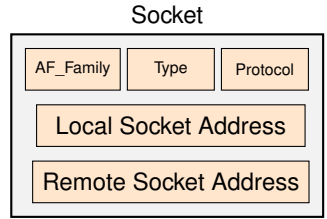
- Slutpunkten för kommunikation mellan två applikationer
- Används för att skicka och ta emot data över nätverket
- Identifieras unikt av en kombination av socketadresser (källa och destination) samt transportprotokoll
- Skapas och hanteras av operativsystemet via ett API (t.ex. Berkeley sockets)

Sockets och socketadresser

Socketadress



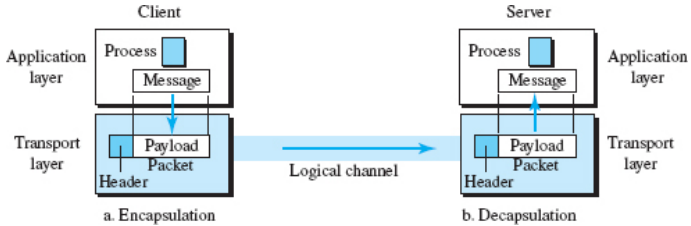
Socket




```
$ ss -tln
State  Recv-Q Send-Q Local Address:Port Peer Address:Port
LISTEN 0      128    0.0.0.0:22         0.0.0.0:*
LISTEN 0      100    127.0.0.1:631      0.0.0.0:*
LISTEN 0      128    [::]:80            [::]:*
```

```
$ ss -tn state established
State Recv-Q Send-Q Local Address:Port Peer Address:Port
ESTAB 0      0      192.168.1.10:54321 140.82.121.4:443
ESTAB 0      0      192.168.1.10:48210 93.184.216.34:80
```

Inkapsling



Figur: Inkapsling [1, fig 9.7]

Multiplexing

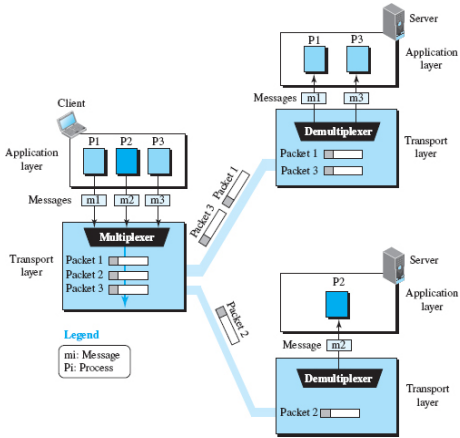
Multiplexing

- Tillåter flera processer att dela en resurs
- Flera processer/applikationer delar på en nätverksanslutning
- Många-till-en (many-to-one)

Demultiplexing

- Tillåter en mottagare att dela upp inkommet data till rätt destination.
- En nätverksanslutning kan leverera data åt flera processer/applikationer.
- En-till-många (one-to-many)

Multiplexing



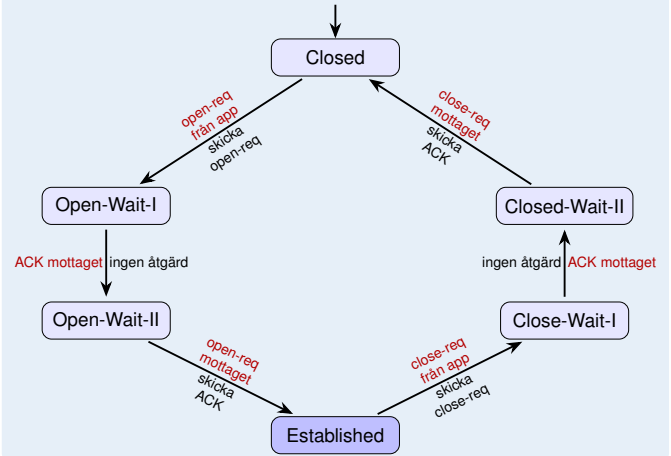
Figur: Multiplexing [1, fig 9.8]

FSM förbindelselöst och förbindelseorienterat

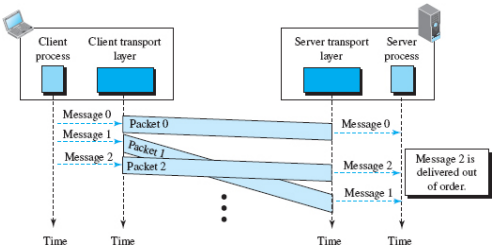
Förbindelselöst

Båda ändpunkterna är alltid i *Established*.

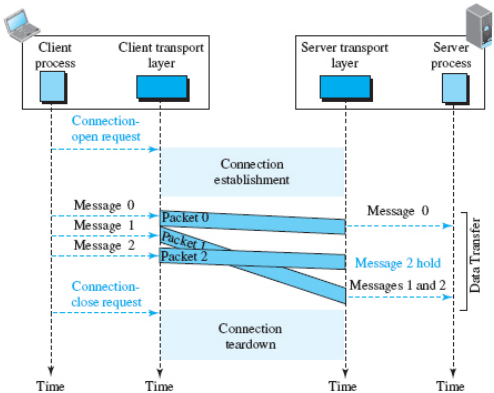
Förbindelseorienterat



Förbindelselöst och förbindelseorienterat



Figur: Förbindelselös anslutning [1, fig 9.13]



Figur: Förbindelseorienterad anslutning [1, fig 9.15]

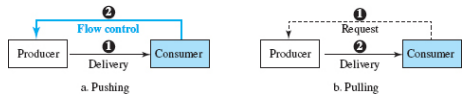
Flödeskontroll

Pushing

- Sändaren skickar utan att mottagaren begär data
- Riskerar att överbelasta mottagaren

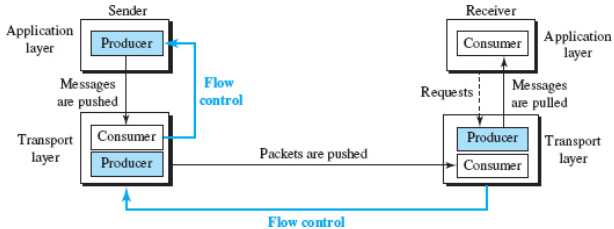
Pulling

- Mottagaren begär data
- Ingen risk för överbelastning



Figur: Push/Pull [1, fig 9.9]

Flödeskontroll

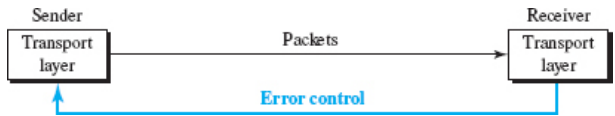


Figur: Flödeskontroll [1, fig 9.10]

Felkontroll

Felkontroll på transportlagret

- IP är sk Best-effort, dvs icke tillförlitligt.
- Tillförlitligheten hanteras därför av transportlagret
- Upptäcker korrupta eller förlorade segment.
- Hanterar dupletter.
- Kan säkerställa att data levereras i ordning.



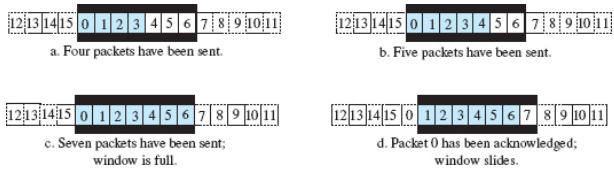
Figur: Felkontroll [1, fig 9.11]

Mittuniversitetet
MID SWEDEN UNIVERSITY

Sekvensnummer

- Varje segment ges ett sekvensnummer
- Hjälper transportlagret att hålla reda på vad som skickats och tagits emot.
- Möjliggör felkontroll, flödeskontroll och överbelastningskontroll.
- Av praktiska anledningar sätts ett tak på sekvensnumret.
- När taket är nått, startas det om från början.

Sliding window



Figur: Sliding window [1, fig 9.13]

Mittuniversitetet
MID SWEDEN UNIVERSITY

Table of Contents

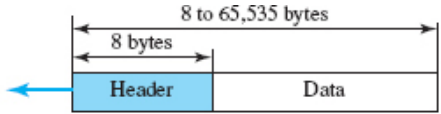
1. Transportlagret

2. User Datagram Protocol

3. Transmission Control Protocol

UDP

- Förbindelselöst.
- Ingen tillförlitlighet.
- 8 B stor header = låg overhead.
- Skickar direkt vidare inkomna datagram.



a. UDP user datagram

b. Header format

Figur: UDP-header [1, fig 9.18]

Mittuniversitetet
MID SWEDEN UNIVERSITY

Användningsområden

- Användbart då tillförlitlighet inte är nödvändig
- Låg overhead.
- Låg fördröjning.
- Tunnling.
- Exempel: DNS, DHCP, VoIP, videostreaming och QUIC.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Table of Contents

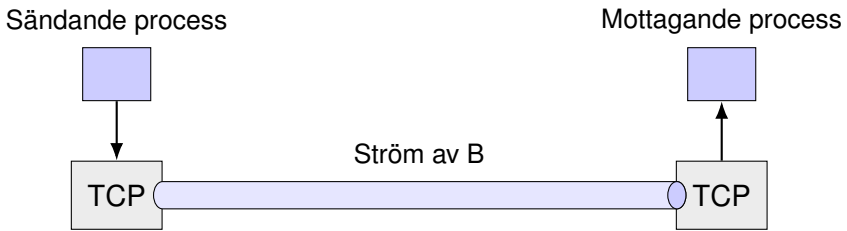
1. Transportlagret

2. User Datagram Protocol

3. Transmission Control Protocol

TCP

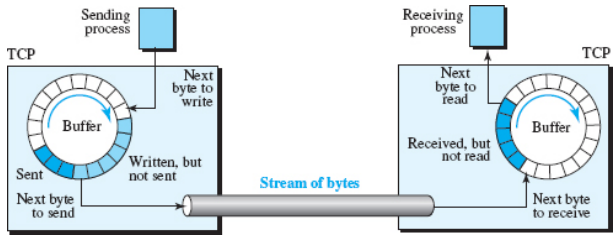
- Förbindelseorienterat.
- Tillförlitligt.
- Inbyggd fel-, flödes- och överbelastningskontroll.
- Levererar data i den ordning den skickades (ström).



Figur: TCP Ström

Buffrar

- Buffrar används för att säkerställa att data kan skickas som en ström mot applikationen.
- Behövs både för sändande och mottagande enhet.

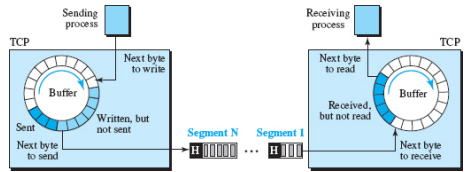


Figur: TCP Buffrar [1, fig 9.21]

Segmentering av datat

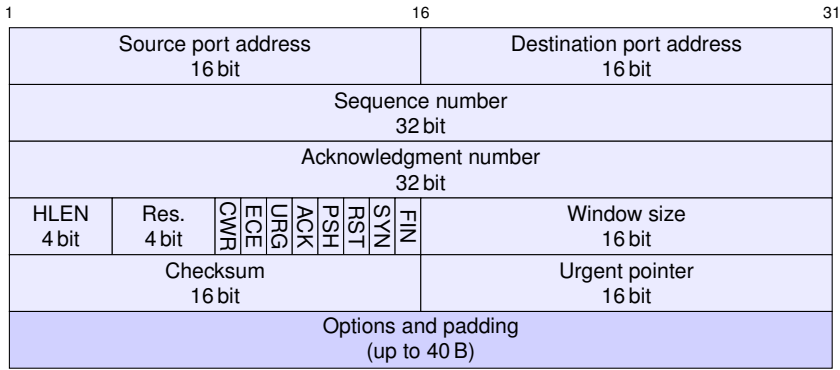
Segmentering

- Segmentstorleken (MSS) beräknas utifrån MTU och val av underliggande protokoll.
- Exempelvis Ethernet standard MTU på 1500 B
  - TCP-header minst 20 B.
  - IPv4-header minst 20 B.
  - MSS = 1500 - 40 = 1460 B (utan options).



Figur: TCP Segment [1, fig 9.22]

TCP-header



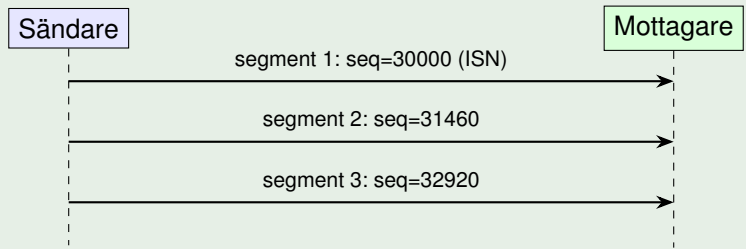
Figur: TCP header

Sekvens och bekräftelsevärden

Sekvensvärde

- Relativt värde som anger den första byten i strömmen som detta segments data ska placeras på.
- Initialt sekvensvärde (ISN) väljs i samband med sessionsupprättningen.

Exempel: ISN=30000, segmentstorlek=1460 B

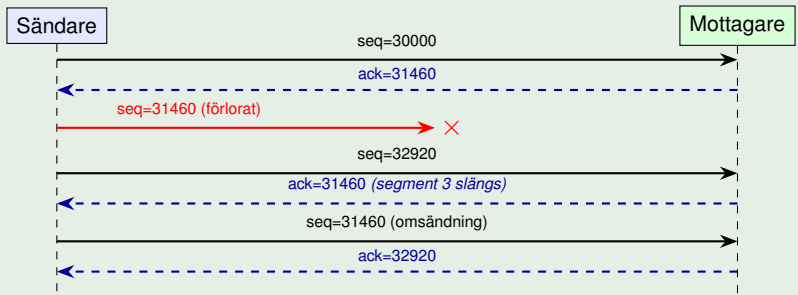


Sekvens och bekräftelsevärden

Bekräftelsevärde

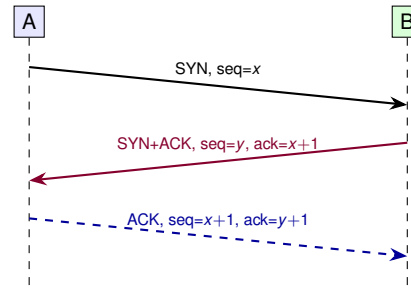
- Relativt värde som anger vilket sekvensvärde som förväntas tas emot.
- Om ett segment blivit förlorat på vägen, kommer samma bekräftelsevärde skickas om.

Exempel: ISN=30000, segmentstorlek=1460 B



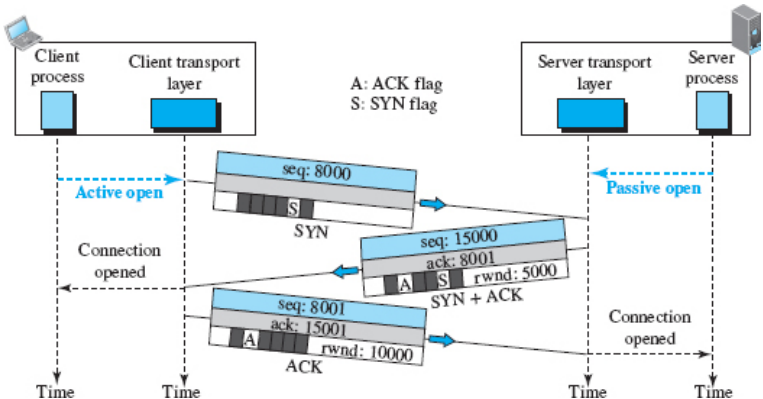
Trevägshandskakning

- Innan två datorer kan börja kommunicera över TCP behöver en session upprättas.
- Upprättas med en trevägshandskakning.
  - SYN: A $\rightarrow$ B
  - SYN + ACK: B $\rightarrow$ A
  - ACK: A $\rightarrow$ B
- Säkerställer att både sändare och mottagare är överens om att påbörja dataöverföring.
- SYN-segmenten innehåller inget data, men “förbrukar” ett sekvensnummer.[1]



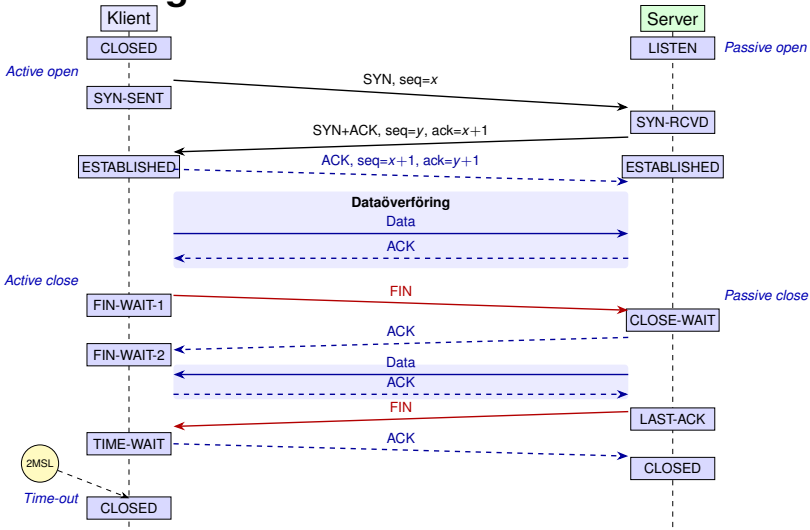
[1] Undantag: TCP Fast Open (RFC 7413) tillåter data i SYN-segmenten.

Trevägshandskakning - Tillstånd



Figur: Aktiv och Passiv [1, fig 9.26]

Dataöverföring



Flödeskontroll

Sändfönster

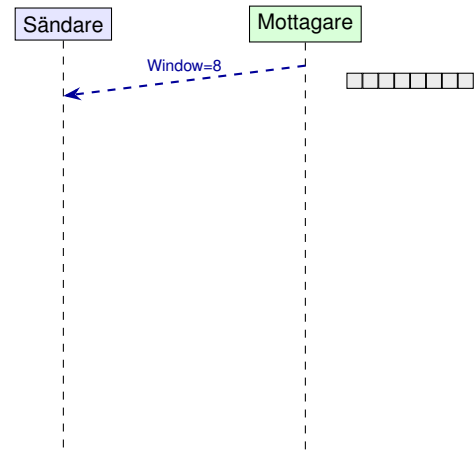
- Hur mycket data som får vara obekräftad (in-flight) samtidigt.
- I detta skede: sändfönster $=$ `rwnd`.

Mottagarfönster (`rwnd`)

- Mottagarens lokala variabel för ledigt buffertutrymme.
- Annonseras i TCP-headerns *Window*-fält vid varje ACK.

Fönsterstorlek

- Sändfönstret sätts baserat på senast mottagna *Window*-värde (dvs. `rwnd`).



Flödeskontroll

Sändfönster

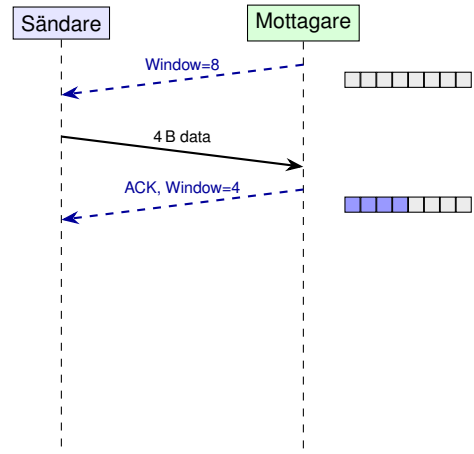
- Hur mycket data som får vara obekräftad (in-flight) samtidigt.
- I detta skede: sändfönster $=$ `rwnd`.

Mottagarfönster (`rwnd`)

- Mottagarens lokala variabel för ledigt buffertutrymme.
- Annonseras i TCP-headerns *Window*-fält vid varje ACK.

Fönsterstorlek

- Sändfönstret sätts baserat på senast mottagna *Window*-värde (dvs. `rwnd`).



Flödeskontroll

Sändfönster

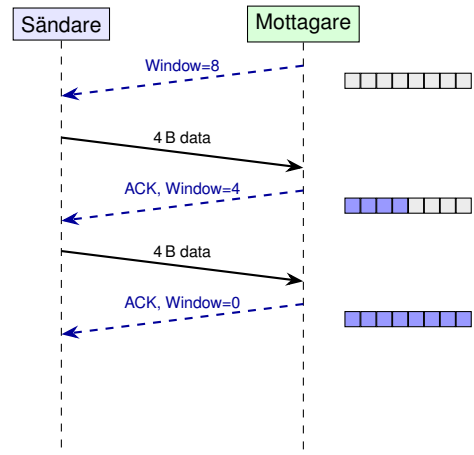
- Hur mycket data som får vara obekräftad (in-flight) samtidigt.
- I detta skede: sändfönster $=$ `rwnd`.

Mottagarfönster (`rwnd`)

- Mottagarens lokala variabel för ledigt buffertutrymme.
- Annonseras i TCP-headerns *Window*-fält vid varje ACK.

Fönsterstorlek

- Sändfönstret sätts baserat på senast mottagna *Window*-värde (dvs. `rwnd`).



Flödeskontroll

Sändfönster

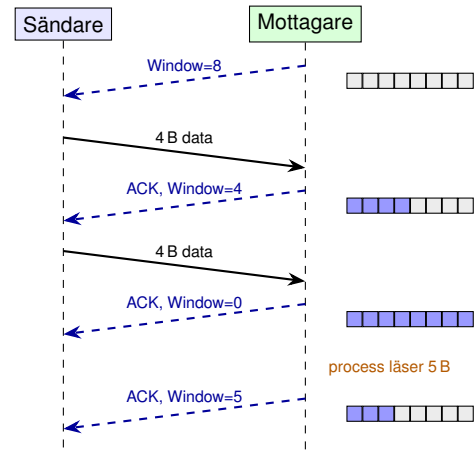
- Hur mycket data som får vara obekräftad (in-flight) samtidigt.
- I detta skede: sändfönster = `rwnd`.

Mottagarfönster (`rwnd`)

- Mottagarens lokala variabel för ledigt buffertutrymme.
- Annonseras i TCP-headerns *Window*-fält vid varje ACK.

Fönsterstorlek

- Sändfönstret sätts baserat på senast mottagna *Window*-värde (dvs. `rwnd`).



Sändfönster

Sändfönster

Effektivt sändfönster = fönsterstorlek − (LastByteSent − LastByteAcked).

Mottagarfönster

Effektivt Mottagarfönster = buffertstorlek − (Data som ännu inte hämtats från applikationslagret).

Felkontroll i TCP

- TCP-specifik felkontroll, byggd på de generella mekanismerna från transportlagret.
- Checksummor
- Sekvensnummer och möjlighet att begära omsändning
  - ACK — Allt mottaget data upp till detta sekvensnummer har blivit mottaget
  - SACK — Kan informera om att en delmängd har blivit mottaget.

Överbelastningskontroll

Congestion Window

- Variabel som sätts baserat på upplevda symptom.
  - Ej mottagna bekräftelsemeddelanden.
  - Sent mottagna bekräftelsemeddelanden.
  - Mottagit samma bekräftelsemeddelande flera gånger.

Överbelastningskontroll II

Congestion window

- Beroende på vilken variant av TCP som används, så beräknas denna olika
- Målet är att den ska vara så stor som möjligt.
- Då denna baseras på symptom på nätverket, tar den hänsyn till delade länkar
- Om flera TCP-anslutningar delar länk kommer detta resultera i en mer jämlik fördelning av bandbredden.

Sändfönster

Effektivt sändfönster = min (rwnd, cwnd) *rwnd* i detta fall definieras som fönsterstorlek − (LastByteSent − LastByteAcked)

Slow Start och Congestion Avoidance

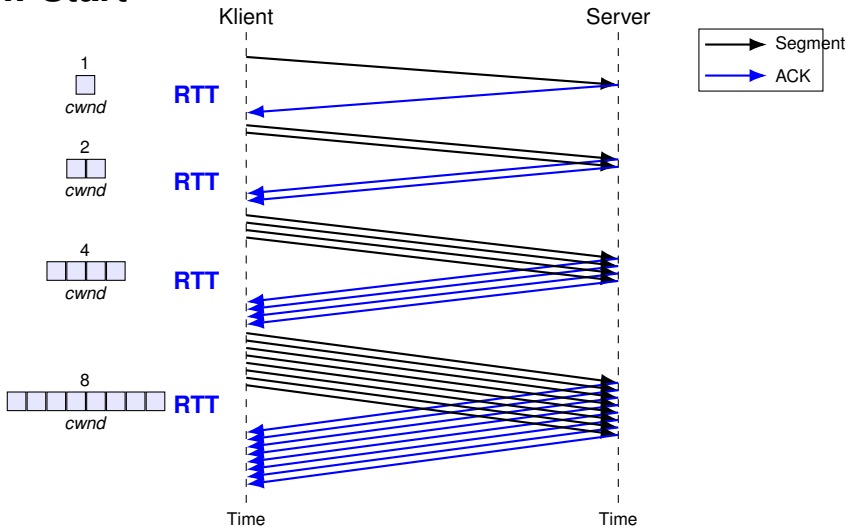
Slow Start

- Startar med `cwnd` = 1 MSS.
- Dubblerar `cwnd` per RTT (exponentiell tillväxt).
- Avbryts vid tröskelvärdet `ssthresh` eller vid paketförlust.

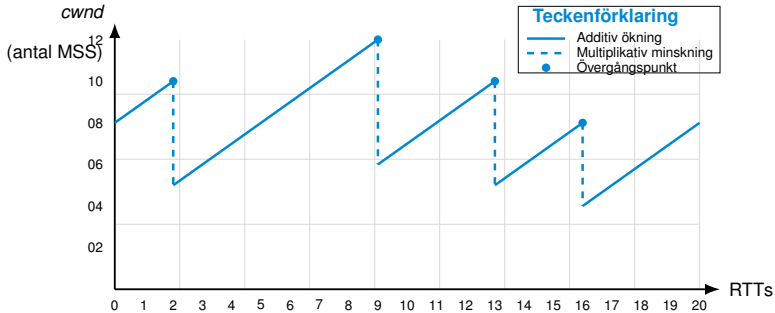
Congestion Avoidance

- Ökar `cwnd` med 1 MSS per RTT (linjär tillväxt).
- Aktiveras när `cwnd` $\geq$ `ssthresh`.
- Vid paketförlust halveras `cwnd` och processen börjar om.

Slow Start


Klient
Server
Segment
ACK
1
cwnd
RTT
2
cwnd
RTT
4
cwnd
RTT
8
cwnd
RTT
Time
Time

TCP Genomströmning

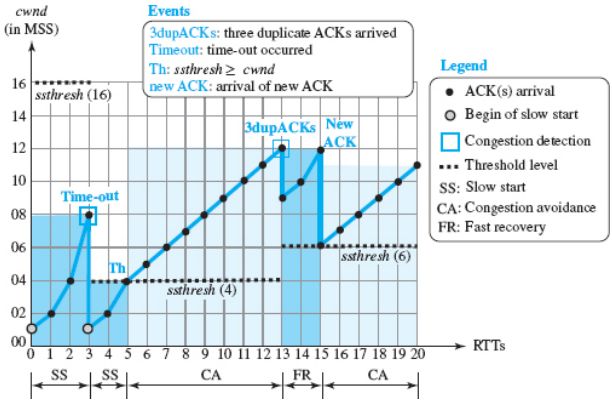


Genomströmning

Genomströmning $= 0.75W_{max}/RTT$

Där W_{max} är den genomsnittliga högsta fönsterstorleken när det upptäcks att det sker en överbelastning.

TCP RENO



Figur: TCP RENO [1, fig 9.50]

Beräkningsexempel TCP genomströmning

Om MSS = 10 kB och RTT = 100 ms.

Beräkningsexempel 9.12 (Fel i boken!)

$$W_{\max} = \frac{10 + 12 + 10 + 8 + 8}{5} \text{MSS} = 9.6 \,\text{MSS} = 96 \,\text{kB}$$

$$\text{Throughput} = \frac{0.75 \cdot W_{\max}}{\text{RTT}} = \frac{0.75 \cdot 96 \,\text{kB}}{100 \,\text{ms}} = 720 \,\text{kB/s} = 5.76 \,\text{Mbit/s}$$

Mittuniversitetet
MID SWEDEN UNIVERSITY