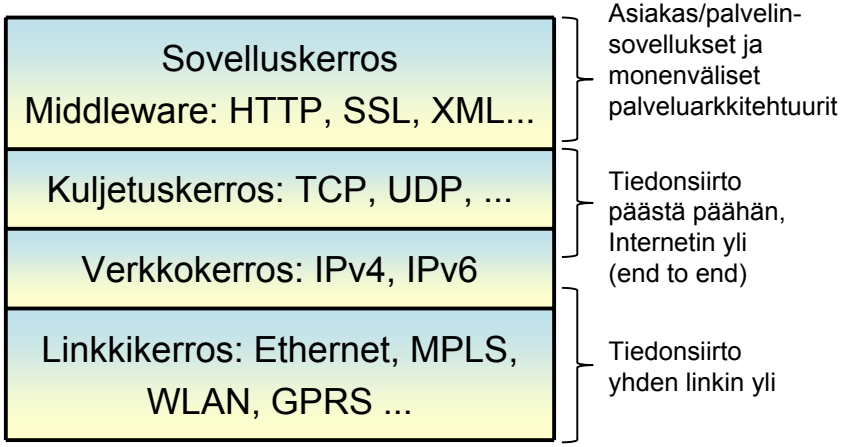
Aalto-yliopisto
Teknillinen korkeakoulu

Kuljetuskerros

Matti Siekkinen

T-110.2100 Johdatus tietoliikenteeseen
kevät 2010

TCP/IP-protokollapino



Sisältö

- Kuljetuskerroksen tehtävä ja ominaisuudet
- UDP (User Datagram Protocol)
- Luotettava tiedonsiirto
- TCP (Transmission Control Protocol)
- Ruuhkanhallinnan perusteet

Tämän luennon jälkeen…

- Ymmärrätte:
  - kuljetuskerroksen tehtävän ja toiminnan
  - luotettavan tiedonsiirron erityyppiset menetelmät
  - UDP:n ja TCP:n toimintaperiaatteet
- Tiedostatte:
  - Mitä on ruuhkanhallinta ja miksi sitä tarvitaan

Kuljetuskerroksen tehtävä

- Kuljetuskerros yhdistää sovelluksia
  - Viestejä päätelaitteen sovelluksesta toiseen (end-to-end)
  - Aktiivisia sovelluksia voi olla monia yhtäaikaa yhdessä päätelaitteessa
- Kuljetuskerros tarjoaa sovelluksille erilaisia palveluita
  - Luotettava/epäluotettava tiedonsiirto
  - Viestinvälitys (datagrammi) tai tavuvirta
- Kuljetuskerros toteutetaan eri protokollilla, jotka ovat vaihtoehtoisia
  - TCP tarjoaa luotettavan tavuvirran palveluna sovellukselle
  - UDP tarjoaa epäluotettavan viestinvälityksen palveluna
  - Myös muita, ei käsitellä tällä kurssilla

Kuljetuskerroksen ominaisuuksia

- Portti
  - Jokaisella päätelaitteella on osoite (IP)
  - Portti (16-bittinen numero) identifioi sovelluksen päätelaitteessa
    - Monta aktiivista yhtäaikaisesti
  - *Well-known port numbers:* 0-1023
    - Varattuja, esim. 80=HTTP, 53=DNS
    - Internet Assigned Numbers Authority: www.iana.org
- Socket rajapinta
  - Sovelluksen ja kuljetuskerroksen protokollan välissä
  - Porttinumero määräytyy sokettia luodessa
- Data välitetään segmentteinä
  - UDP viesti, TCP tavuvirran osa
  - Kapseloidaan pakettiin alemmalla kerroksella (IP)

UDP

- User Datagram Protocol
- Standardi RFC-768
- UDP tarjoaa epäluotettavan yhteydettömän kuljetuspalvelun
  - Kevyt, ei tilaa, ei yhteydenmuodostusta, helppo toteuttaa
- Datagrammien välitys päätelaitteessa
  - Kohdeosoitteen ja kohdeportin avulla

UDP

- UDP välittää datagrammeja (viesti)

Source port	Destination port
Length	UDP checksum
Data	

- Tarkistussummaan lasketaan sekä otsake että data
  - Ei ole välttämätön
- UDP-sovelluksia: DNS, Radius, NTP, SNMP, RTP (VoIP)

UDP-kaappaus: dig (DNS)

```
siekkine@b128-dell:~$ dig www.hs.fi

; <<>> DiG 9.4.2-P2 <<>> www.hs.fi
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50872
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;www.hs.fi.                     IN      A

;; ANSWER SECTION:
www.hs.fi.              600     IN      A       194.137.237.63

;; AUTHORITY SECTION:
hs.fi.                  600     IN      NS      ns4.sanoma.fi.
hs.fi.                  600     IN      NS      ns3.sanoma.fi.
hs.fi.                  600     IN      NS      ns2.sanoma.fi.
hs.fi.                  600     IN      NS      ns1.sanoma.fi.

;; ADDITIONAL SECTION:
ns1.sanoma.fi.          27395   IN      A       194.137.237.33
ns2.sanoma.fi.          27395   IN      A       194.137.237.34
ns3.sanoma.fi.          58894   IN      A       195.165.77.83
ns4.sanoma.fi.          58894   IN      A       195.165.77.84

;; Query time: 54 msec
;; SERVER: 130.233.192.1#53(130.233.192.1)
;; WHEN: Thu Feb 18 22:01:29 2010
;; MSG SIZE  rcvd: 186
```


UDP-kaappaus: DNS kysely

```
Internet Protocol, Src: 130.233.194.105 (130.233.194.105), Dst: 130.233.192.1 (130.233.192.1)
User Datagram Protocol, Src Port: 36287 (36287), Dst Port: domain (53)
    Source port: 36287 (36287)
    Destination port: domain (53)
    Length: 35
    Checksum: 0x8872 [incorrect, should be 0xc8f9 (maybe caused by "UDP checksum offload"?)]
        [Good Checksum: False]
        [Bad Checksum: True]
Domain Name System (query)
    [Response In: 209]
    Transaction ID: 0xc6b8
    Flags: 0x0100 (Standard query)
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
    Queries
        www.hs.fi: type A, class IN
```


UDP-kaappaus: DNS vastaus

```
Internet Protocol, Src: 130.233.192.1 (130.233.192.1), Dst: 130.233.194.105 (130.233.194.105)
User Datagram Protocol, Src Port: domain (53), Dst Port: 36287 (36287)
    Source port: domain (53)
    Destination port: 36287 (36287)
    Length: 194
    Checksum: 0xd768 [correct]
        [Good Checksum: True]
        [Bad Checksum: False]
Domain Name System (response)
    [Request In: 208]
    [Time: 0.053526000 seconds]
    Transaction ID: 0xc6b8
    Flags: 0x8180 (Standard query response, No error)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 4
    Additional RRs: 4
    Queries
        www.hs.fi: type A, class IN
    Answers
    Authoritative nameservers
    Additional records
```


Sisältö

- Kuljetuskerroksen tehtävä ja ominaisuudet
- UDP (User Datagram Protocol)
- → Luotettava tiedonsiirto
- TCP (Transmission Control Protocol)
- Ruuhkanhallinnan perusteet

Luotettava tiedonsiirto

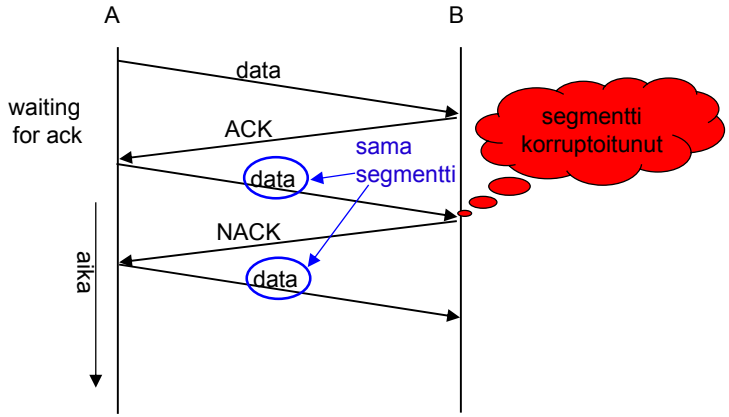
- Linkit eivät ole luotettavia
  - Bittivirheet korruptoivat paketteja
    - Esim. langattomat linkit
- Alempi verkkokerros (IP) ei ole luotettava
  - Reittittimet tietoisesti pudottavat paketteja
- Kuljetuskerros varmistaa että lähetetty tieto pääsee ehjänä perille
  - Lähettävältä sovelluksesta vastaanottavalle sovellukselle
- Miksi kuljetuskerros?
  - Sovelluskerros
    - Redundanttia samaa toiminnallisuuden toteuttamista
  - Alemmat kerrokset
    - ”Per-hop” (linkki) luotettavuus ei aina riitä
      - Paketit voivat korruptoitua reitittimen muistissa

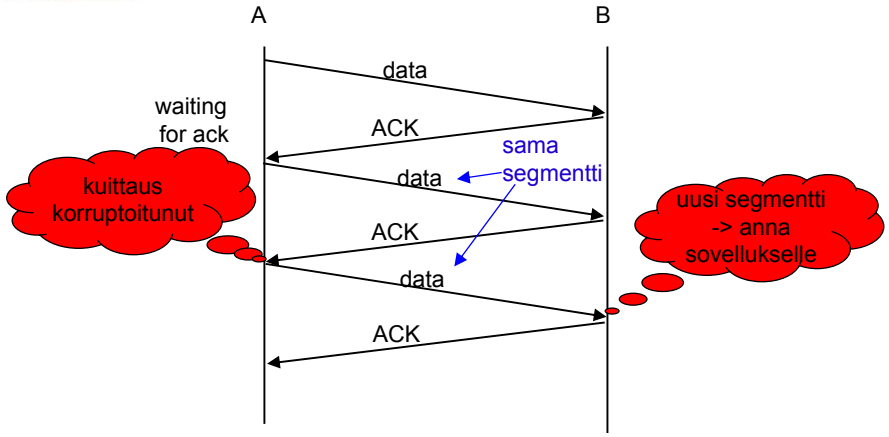
Luotettava tiedonsiirto

- ARQ: Automatic Repeat Request
  - Virheenkorjauksen konsepti
  - Oikeastaan joukko tekniikoita
  - mm. TCP hyödyntää tätä
  - Kuittaukset ja segmentin uudelleenlähetys
- Muitakin on..
  - Forward Error Correction (FEC)
  - Hybridit

ARQ mekanismit

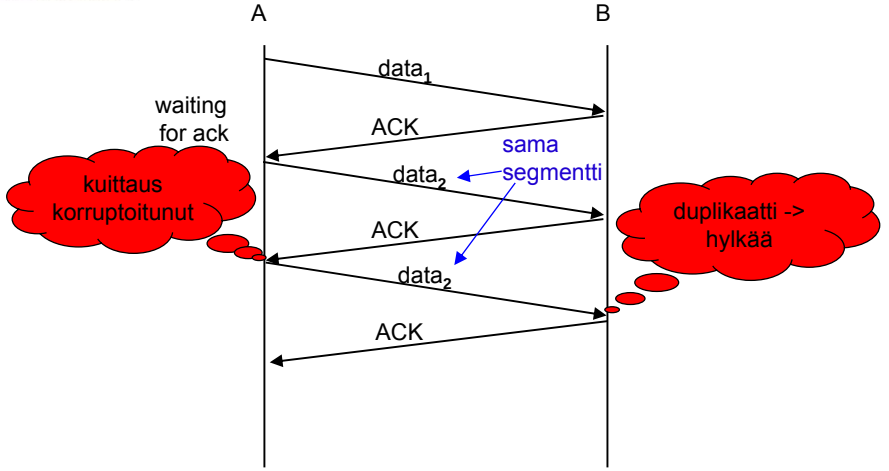
- Tarkistesumma
  - Korruptoituneen segmentin havaitseminen
- ACK: positiivinen kuittaus
  - ”Vastaanotin segmentin ok”
- NACK: negatiivinen kuittaus
  - ”Segmentti rikki, lähetä uudelleen”





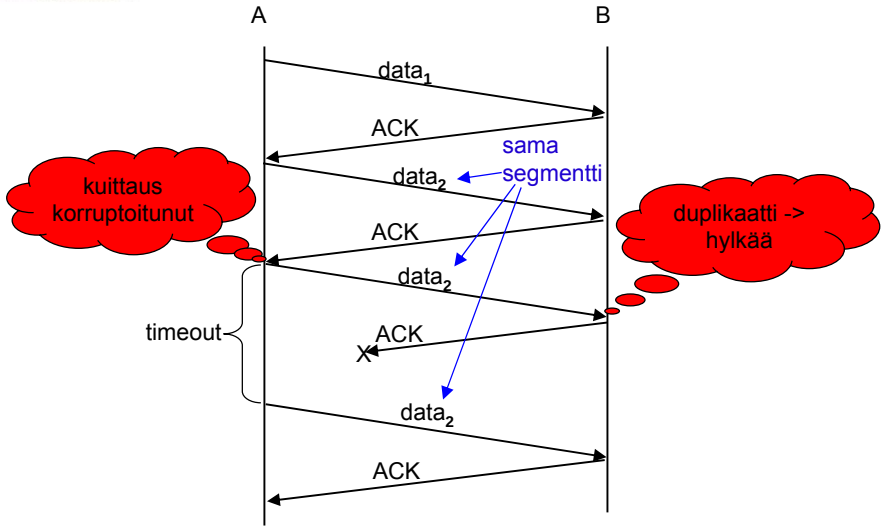
ARQ mekanismit

- Tarkistesumma
  - Korruptoituneen segmentin havaitseminen
- ACK: positiivinen kuittaus
  - ”Vastaanotin segmentin ok”
- NACK: negatiivinen kuittaus
  - ”Segmentti rikki, lähetä uudelleen”
- Sekvenssinumerot
  - Erottaa uuden datan uudelleenlähetetystä
  - Esim. korruptoitunut kuittaus



ARQ mekanismit

- Tarkistesumma
  - Korruptoituneen segmentin havaitseminen
- ACK: positiivinen kuittaus
  - ”Vastaanotin segmentin ok”
- NACK: negatiivinen kuittaus
  - ”Segmentti rikki, lähetä uudelleen”
- Sekvenssinumerot
  - Erottaa uuden datan uudelleenlähetetystä
  - Esim. korruptoitunut kuittaus
- Ajastimet
  - Lähetä uudelleen paketti jollei kuulu kuittausta
  - Kadonneiden pakettien aiheuttamat tilanteet

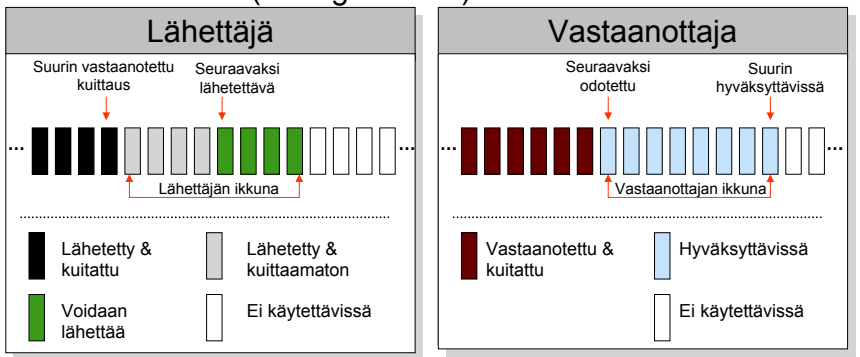


Stop-and-wait

- Vain yksi segmentti matkalla kerrallaan
  - Uusi lähetetään kuittauksen tai timeoutin jälkeen
- 1-bittinen sekvenssinumero riittää
  - Uusi segmentti, sekvenssinro vaihtuu
  - Uudelleenlähetys, sama sekvenssinro
  - Yksi bitti ei riitä, jos verkko saattaa kopioida kehyksiä
- Ei ole kovinkaan tehokas
  - Linkin käyttöaste jää matalaksi
  - Ratkaisu: segmenttien ”putkitus” (pipelining)

Putkitus

- Useita segmenttejä matkalla yhtäaikaisesti
- Tarvitaan riittävä numeroavaruus sekvenssinumeroille
- Liukuva ikkuna (sliding window)

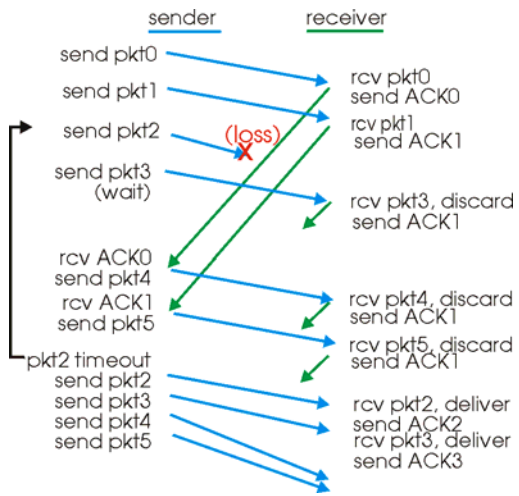


- Go-Back-N ja Selective Repeat protokollat
  - Huomattavasti parempi käyttöaste
  - Riippuu viiveestä

Go-Back-N

- Segmentin kadotessa se ja kaikki sen jälkeen jo lähetetyt uudelleenlähetetään (eli go-back-n)
  - Lähettäjä havaitsee kehyksen katoamisen aikakatkaisulla (timeout)
  - Lähettäjällä ikkunan suuruinen puskuri
- Kumulatiiviset kuittaukset
  - Vastaanottaja kuittaa vain oikeassa järjestyksessä saapuvat segmentit
  - Ehjänä mutta väärässä järjestyksessä vastaanotetut hylätään
  - Kuittauksen katoaminen ei vaarallista
    - Myöhempi kuittaus korvaa sen
- TCP samankaltainen kuin Go-Back-N

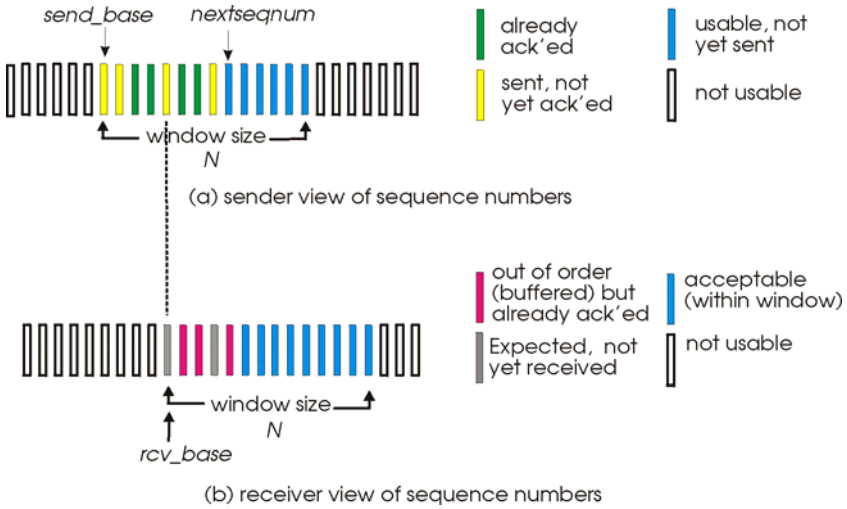
Go-Back-N



Selective Repeat

- Go-Back-N tehokkuus kärsii jos pitkä viive ja iso kaistanleveys
  - Yksi kadonnut segmentti aiheuttaa paljon turhia uudelleenlähetyksiä
- Selective Repeat
  - Vastaanottaja kuittaa erikseen jokaisen segmentin
  - Lähettäjä uudelleenlähettää vain ei kuitatut segmentit
  - Vastaanottajalla on oltava riittävän suuri puskuri

Selective Repeat



TCP

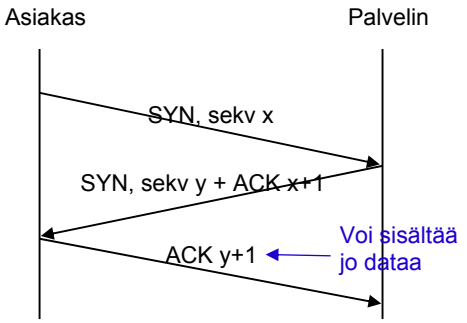
- Transmission Control Protocol
- Standardi RFC-793
- Yhteydellinen protokolla
- Full duplex
  - Sovellusdataa molempiin suuntiin samanaikaisesti
- Luotettava tavuvirta
  - Jakaa sovelluksen lähettämän tavuvirta segmentteihin
  - Nämä kapseloidaan IP-paketeiksi
- Ominaisuuksia
  - Kolmivaiheinen yhteyden muodostus
  - ARQ virheenkorjaus
  - Vuonhallinta
  - Ruuhkanhallinta
- TCP on useimpien sovellusten käyttämä: SMTP, HTTP (WWW)...

TCP yhteys

- Yksiselitteisesti identifioidaan neljällä parametrilla
  - Lähettäjän ja vastaanottajan osoitteet ja porttinumerot
- Segmenttien ohjaus (demux) päätelaitteessa
  - Kaikki neljä parametria tarkistetaan
  - Eroaa UDP:sta
- TCP soketti
  - Vastaanottava soketti palvelun portissa
    - Esim. portti 80 Web-palvelimessa
  - Uusi soketti luodaan välittömästi uutta yhteyttä muodostettaessa

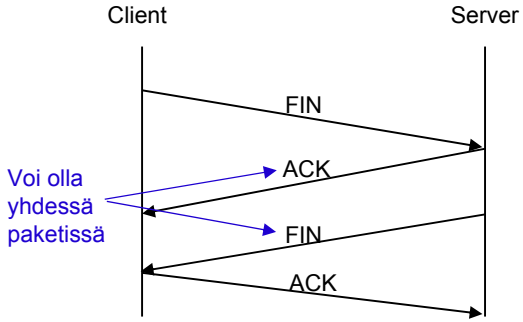
TCP-yhteyden muodostaminen

- "three-way handshake"



- SYN-paketit alustavat sekvenssinumerot satunnaisluvuilla x ja y
  - Mahdollisia vanhoja vielä matkalla olevia paketit eivät sotke yhteyttä
- Kolmas paketti varmistaa ettei palvelin jää turhaan odottelemaan asiakasta
  - Esim. SYN+ACK häviää

TCP-yhteyden sulkeminen



- Kumpi tahansa osapuoli voi aloittaa sulkemisen
- Molemmat "simplex-yhteydet" suljetaan erikseen
- FIN paketin lähettämisen jälkeen ajastin käynnistyy
  - Yhteys ei jää roikkumaan auki kuittausten hävitessä

TCP virheenkorjaus

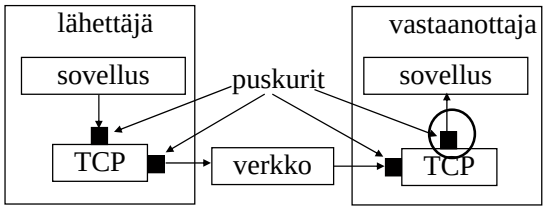
- Go-Back-N tyyppinen ARQ
  - Ajastimet, tarkistussummat, uudelleenlähetykset
  - Suurin ero: TCP uudelleenlähettää vain kadonneet segmentit
    - Vastaanottaja puskuroi myös epäjärjestyksessä tulleita segmenttejä
- Kumulatiiviset (positiiviset) kuittaukset
  - Indikoi mitä sekvenssinumeroa vastaanottaja odottaa seuraavaksi
  - Lasketaan tavuina aloitussekvenssinrosta
  - Delayed ACKs menetelmä: 1 kuittaus per 2 täyttä segmenttiä
- Lisäksi on mahdollista käyttää Selective Repeatia
  - TCP SACK (selective acknowledgments) optio
  - Virheenkorjauksesta tulee GBN+SR hybridi

TCP virheenkorjauksen ajastin

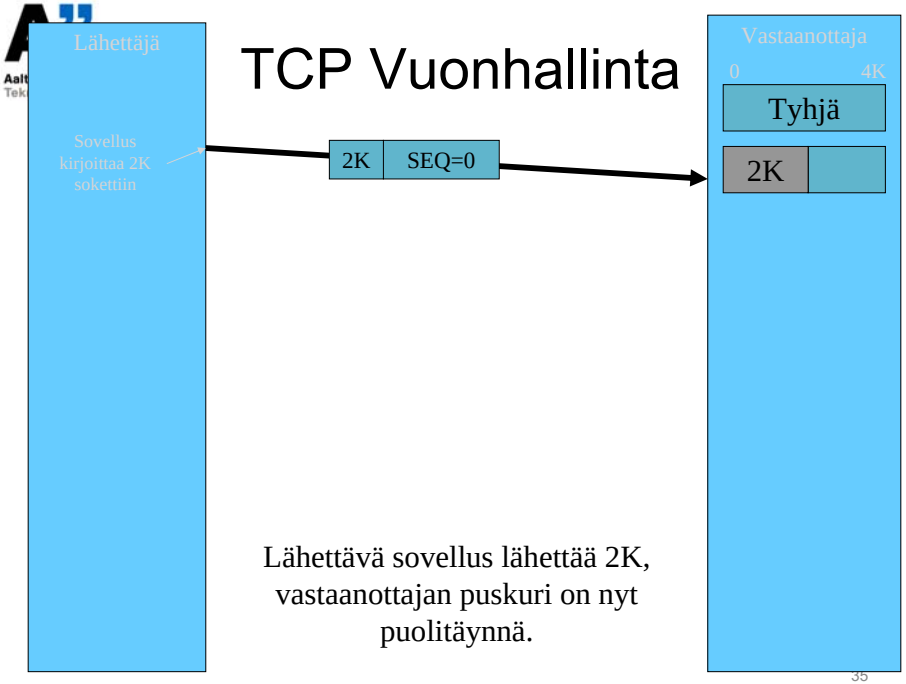
- Uudelleenlähetyksen ajastin
  - Eli Retransmission timeout (RTO)
  - Jokaisella segmentillä oma
- Ajastimen pituuden määritys
  - Pitää olla:
    - pidempi kuin edestakainen viive (RTT: Round trip time)
    - mahdollisimman lyhyt jotta reagoidaan nopeasti virheisiin
  - Säädellään koko ajan koska RTT vaihtelee myös
    - RTT lasketaan painotettuna liikkuvana keskiarvona viiveestä
    - RTT = (α*oldRTT)+((1-α)*newRTTsample) (suositeltu α=0,9)
    - RTO = β*RTT, β>1 (suositeltu β=2)
  - Viiveen jatkuva mittaus
    - Kulunut aika paketin lähetyksen ja sen kuittauksen vastaanottamisen välillä

TCP vuonhallinta

- Eli flow control
- Vastaanottava sovellus kuluttaa dataa tietyllä nopeudella
- TCP yhteyden yli voidaan joskus lähettää tätä nopeammin
- Vuonhallinta varmistaa ettei näin tapahdu
- Menetelmä perustuu liukuvan ikkunan koon vaihteluun

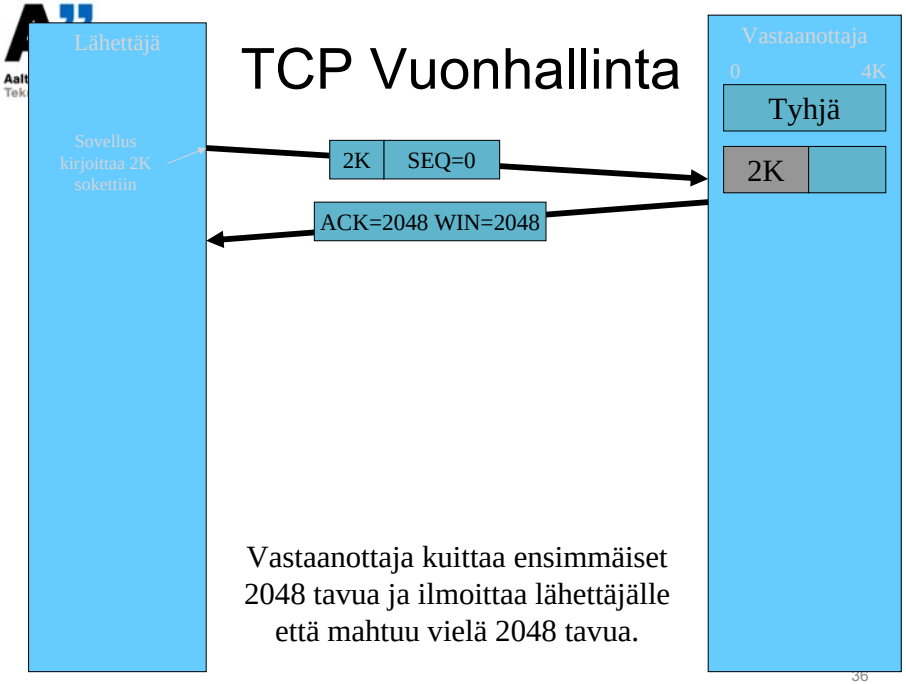


TCP Vuonhallinta



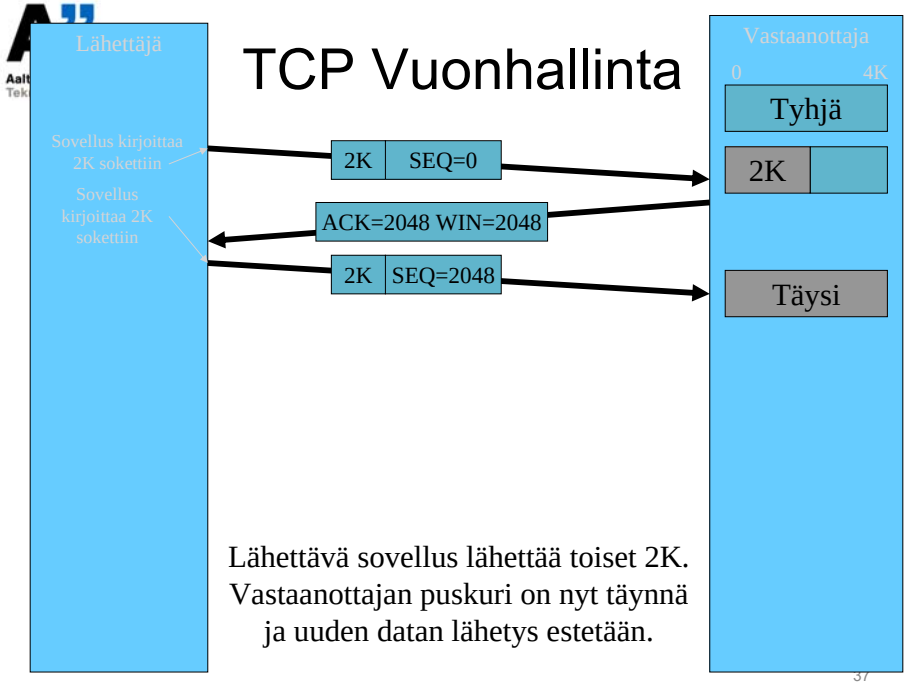
Lähettävä sovellus lähettää 2K, vastaanottajan puskuri on nyt puolitäynnä.

TCP Vuonhallinta



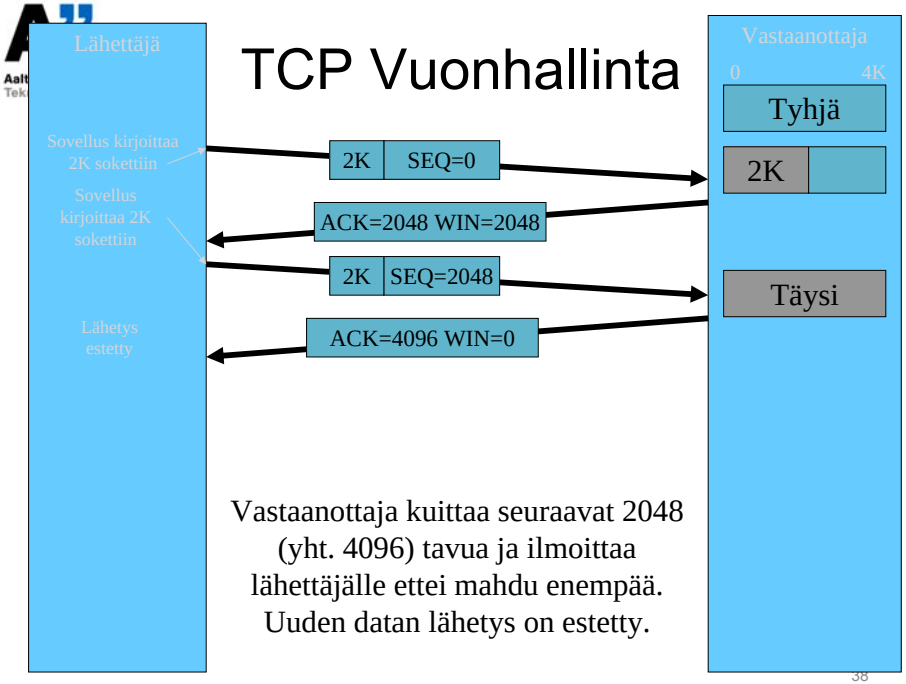
Vastaanottaja kuittaa ensimmäiset 2048 tavua ja ilmoittaa lähettäjälle että mahtuu vielä 2048 tavua.

TCP Vuonhallinta



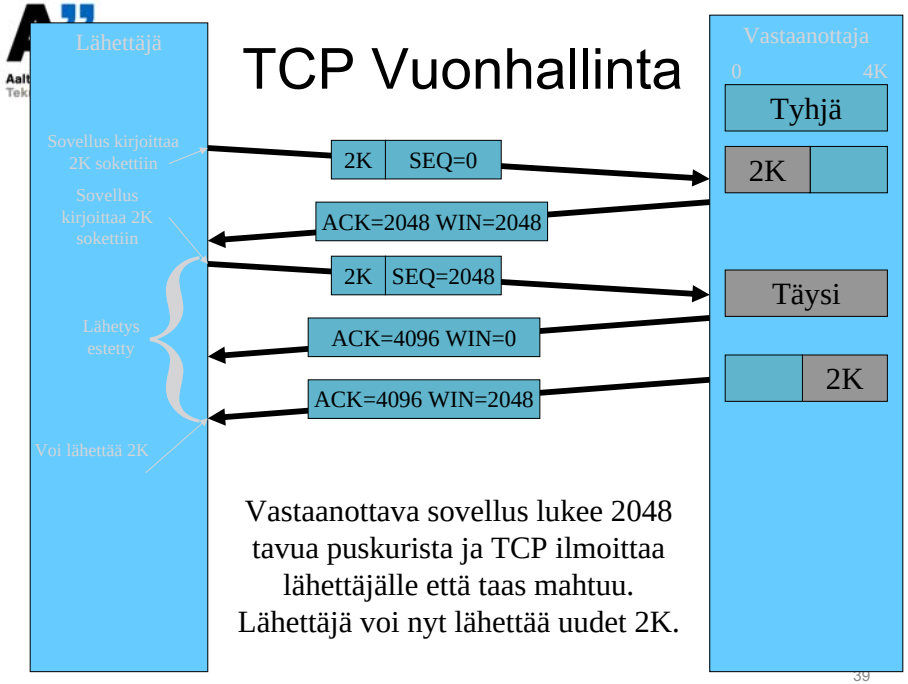
Lähettävä sovellus lähettää toiset 2K. Vastaanottajan puskuri on nyt täynnä ja uuden datan lähetys estetään.

TCP Vuonhallinta



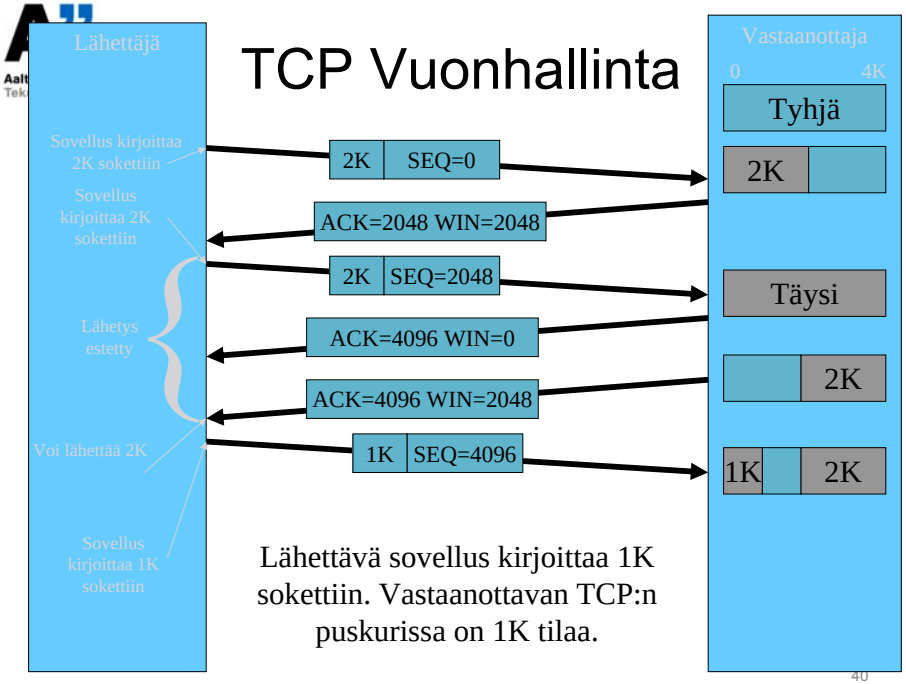
Vastaanottaja kuittaa seuraavat 2048 (yht. 4096) tavua ja ilmoittaa lähettäjälle ettei mahdu enempää. Uuden datan lähetys on estetty.

TCP Vuonhallinta



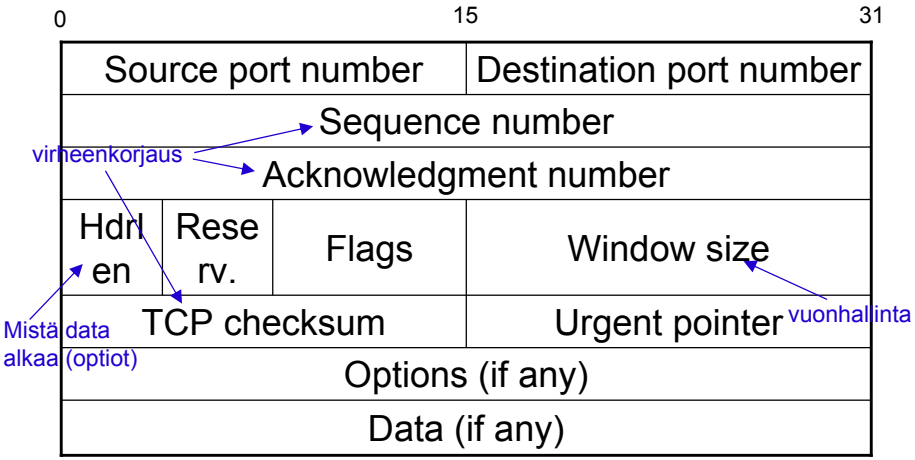
Vastaanottava sovellus lukee 2048 tavua puskurista ja TCP ilmoittaa lähettäjälle että taas mahtuu. Lähettäjä voi nyt lähettää uudet 2K.

TCP Vuonhallinta



Lähettävä sovellus kirjoittaa 1K sokettiin. Vastaanottavan TCP:n puskurissa on 1K tilaa.

TCP-otsake



TCP-otsake: liput (flags)

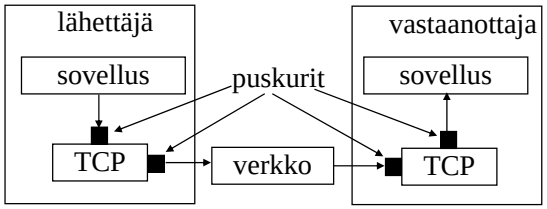
- Liput ovat bittejä
- URG viestin urgent pointer osoittaa dataan, joka on aiheellista lukea ohi jonossa olevan datan
  - Esim. käyttäjä näppäilee interrupt-komennon kesken telnet-istunnon
- ACK: kuittausnumero on aktiivinen
- PSH: tämän jälkeen ei toistaiseksi uutta dataa, välitä heti eteenpäin
  - ei odoteta segmentin täyttymistä
- RST: resetoi yhteys
- SYN: yhteyden avaus
- FIN: yhteyden sulkeminen

TCP-otsake: optiot

- MSS: suurimman sallitun segmentin koko
- Window scaling: sovitaan kerroin jolla vastaanottajan ikkunan koko kerrotaan
  - Tarvitaan koska window kenttä on usein liian pieni (max ~65KB) nykyaikaisille kaistanleveyksille
  - Saadaan käyttöaste korkeammaksi
- SACK: Selective Repeat –tyyliset kuittaukset
- Myös monia muita...

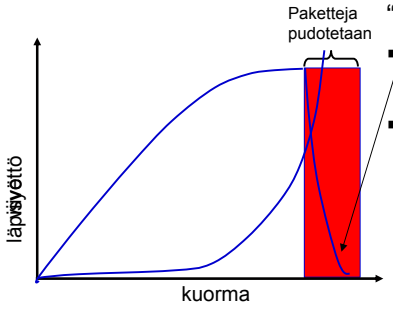
Ruuhkanhallinta: Miksi?

- Verkon hetkellinen jäljellä oleva vapaa kaista vaihtelee
  - Voi olla vähemmän kuin lähettävän ja vastaanottavan sovellusten kapasiteetti -> pelkkä vuonhallinta ei riitä
  - Monta lähettäjää jakaa samoja verkkoresursseja
  - Verkko voi siis olla pullonkaula
- Ruuhkanhallinta varmistaa että verkkoa ei ylikuormiteta



Ruuhkanhallinta: Miksi?

- Verkkoelementeissä (reitittimet) on puskurit
  - FIFO+drop tail
  - Puskurin täyttyessä paketit joutuvat jonottamaan -> viive kasvaa
  - Puskurien ollessa täynnä uudet paketit pudotetaan



"Congestion collapse":

- Pudotettujen pakettien uudelleenlähetys
  - Kasvattaa lisää kuormaa
- Uudelleenlähetetään tarpeettomasti vielä matkalla olevia paketteja
  - Viive kasvaa nopeasti -> RTO ei pysy perässä
  - Viive kasvaa yli maksimi RTOn
  - Lisää edelleen kuormaa!
    - Vrt. tulen sammuttaminen bensalla…
- Reitittimet tekevät enenevästi turhaa työtä
  - Esim. paketin pudottaa loppusuoralla oleva reititin

TCP Ruuhkanhallinta

- Periaate:
  - Kontrolloi jatkuvasti TCP lähetysnopeutta
  - Kasvata nopeutta kun kaikki menee hyvin
  - Pienennä nopeutta kun havaitaan ruuhkaa
    - Esim. segmenttejä katoaa
- Miten?
  - *Ruuhkaikkunan* (eli congestion window cwnd) avulla:

    $$\text{lähetetyt kuittaamattomat tavut} \leq min(cwnd, rwnd)$$

  - Lähetysnopeutta säädellään muuttamalla ruuhkaikkunan kokoa

vuonhallinta

Yhteenveto

- Kuljetuskerros tarvitaan yhdistämään sovelluksia
  - Verkkokerros välittää viestejä koneelta koneelle
  - Monia aktiivisia sovelluksia yhtäaikaa päätelaitteen sisällä
- Erityyppisiä palveluita
  - UDP: epäluotettavan viestinvälitys
  - TCP: luotettavan tavuvirta
- TCP:n ominaisuuksia
  - ARQ virheenkorjaus
  - Vuonhallinta vastaanottavan sovelluksen suojaksi
  - Ruuhkanhallinta tarvitaan verkon ylikuormittumisen estämiseksi

Ensi luennolla

- Verkkokerros eli IP-kerros
  - Miten segmentit kulkevat lähettäjältä vastaanottajalle?
  - Osoitteet, reititys ja forwardointi
- Myös
  - ICMP, NAT(Network Address Translation), ARP, DHCP

Aalto-yliopisto
Teknillinen korkeakoulu

Jatkokursseilla…

- Lisää TCP:tä
  - Miten TCP:n ruuhkanhallinta toimii?
  - Tuhat ja yksi TCP versiota…
- Mitä haasteita langattomat verkot luovat kuljetuskerrokselle?
- SCTP ja DCCP