



12. Liikenteen- ja ruuhkanhallinta Internetissä

Sisältö

- Johdanto
- IP-verkot
- Liikenteen- ja ruuhkanhallinta Internetissä
- TCP ruuhkanhallinta
- QoS arkkitehtuurit

Liikenteenhallinta

- Liikenteellisiä ongelmia:
 - Liikenne on luonteeltaan satunnaista (vaihdellen ennustamattomasti)
 - ajoittain verkossa syntyy ruuhkaa (congestion)
 - Liikennelähteet voivat käyttäytyä “huonosti” (pyrkien saamaan käyttöönsä enemmän kuin “reilun” osuuden verkon resursseista)
- Liikenteenhallintaa (traffic management) tarvitaan, jotta
 - verkko saavuttaisi halutut laatu- ja suorituskykytavoitteet
 - verkko pystyisi suojaamaan itsensä ja muut käyttäjät huonosti käyttäytyviä lähteitä vastaan
- Verkon suojaamiseen ruuhkatilanteilta on kaksi lähestymistapaa:
 - **ennakoivat** (proactive) menetelmät, joilla pyritään ennalta välttämään ruuhkan syntyminen
 - **reagoivat** (reactive) menetelmät, joilla pyritään lievittämään ruuhkasta aiheutuvia ongelmia ja poistamaan koko ruuhka

Internet

- Erittäin laajalle levinnyt
 - tulee työpöydälle asti
 - tietokoneisiin ja niiden sovellutuksiin on rakennettu tarvittava protokollatuki
- Juuret yhteydettömään pakettiverkko- ja reititintekniikkaan perustuvassa tietokoneiden välisessä kommunikaatiossa (ARPANET)
- Alunperin tarkoitettu ei-reaaliaikaisten viestien välittämiseen
 - tiedostojen siirto
 - tietokoneiden etäkäyttö
 - sähköposti
- Palvelun laatu on tässä maailmassa ollut perinteisesti tuntematon käsite
 - palvelua on tarjottu "best effort" -periaatteella
- Uudet palveluarkkitehtuurit (DiffServ, IntServ) kuitenkin pyrkivät eriytettyyn palvelun laatuun

Vuot

- Vuo on verkkokerroksessa (siis reitittimissä) havaittu approksimaatio ylempien kerrosten liikenteestä
- **Vuo** (flow) = sarja peräkkäisiä ja toisiinsa kuuluvia IP-paketteja, joilla sama lähde ja määränpää
 - Karkeassa luokittelussa huomioidaan vain lähde- ja määränpääosoite, hienommassa luokittelussa voidaan erotella esim. eri ylemmät protokollat (esim. TCP, UDP, HTTP, FTP, ...)
 - Peräkkäiset vuot erotetaan toisistaan ajastimella: peräkkäiset paketit kuuluvat samaan vuohon vain, jos niiden ajallinen etäisyys on tarpeeksi pieni
 - Huom. Vuon määritelmä on hyvin joustava. Se, miten vuot käytännössä pitäisi luokitella, on oma taiteenlajinsa:
 - karkeuden (granularity) valinta
 - ajastuksen (timeout) valinta

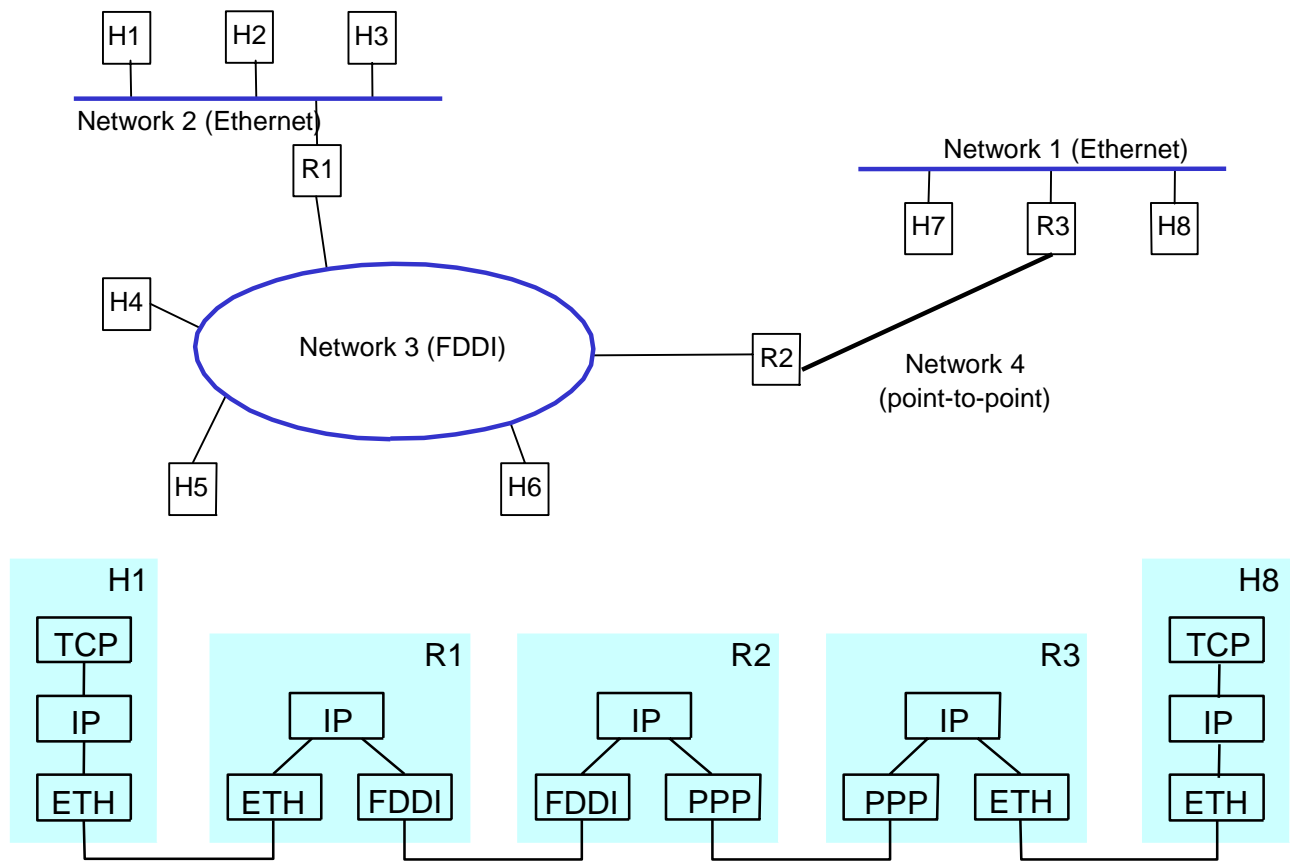
Palvelun laatu

- Palvelun laatua voidaan tarkastella sekä vuo- että pakettitasolla
- **Vuotasolla** voidaan toisaalta ottaa
 - **kokonaisvaltainen näkökulma**, jolloin
 - tehokkuuden kannalta tärkeää on verkon **kokonaisläpäisy** (throughput), mutta
 - toisaalta myös resurssien jaon **oikeudenmukaisuus** (fairness), tai
 - **vuokohtainen näkökulma**, jolloin
 - elastiselle liikenteelle tärkein laatutekijä on **läpäisy** (throughput) tai oikeammin **hyödyllinen läpäisy** (goodput)
- **Pakettitasolla**
 - virtaavalle reaaliaikaiselle liikenteelle tärkein laatutekijä on pakettien **päästä-päähän viive** ja **vaihtelut** tässä viiveessä, sen sijaan yksittäisten pakettien katoaminen tai korruptoituminen ei ole niin vaarallista
 - elastiselle liikenteelle ja transaktioille (ts. yksittäisille paketeille) oleellisinta on **pakettien menetyssuhteen** pysyminen vähäisenä

Sisältö

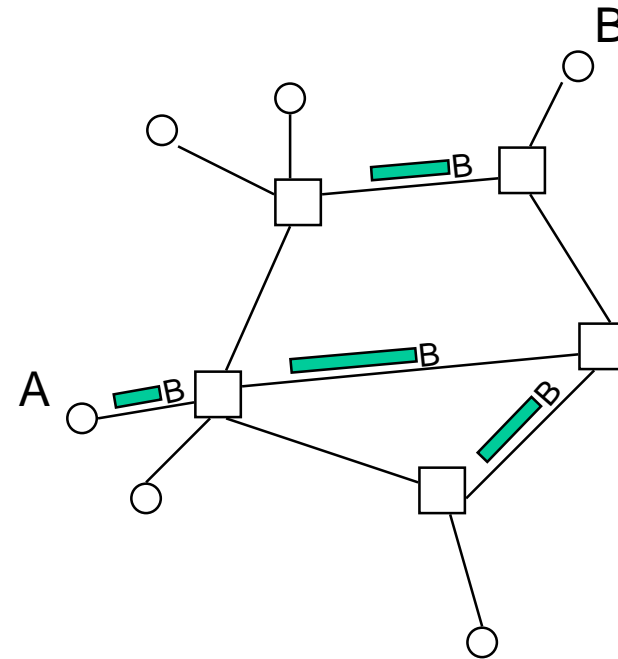
- Johdanto
- IP-verkot
- Liikenteen- ja ruuhkanhallinta Internetissä
- TCP ruuhkanhallinta
- QoS arkkitehtuurit

IP-protokollapino



IP

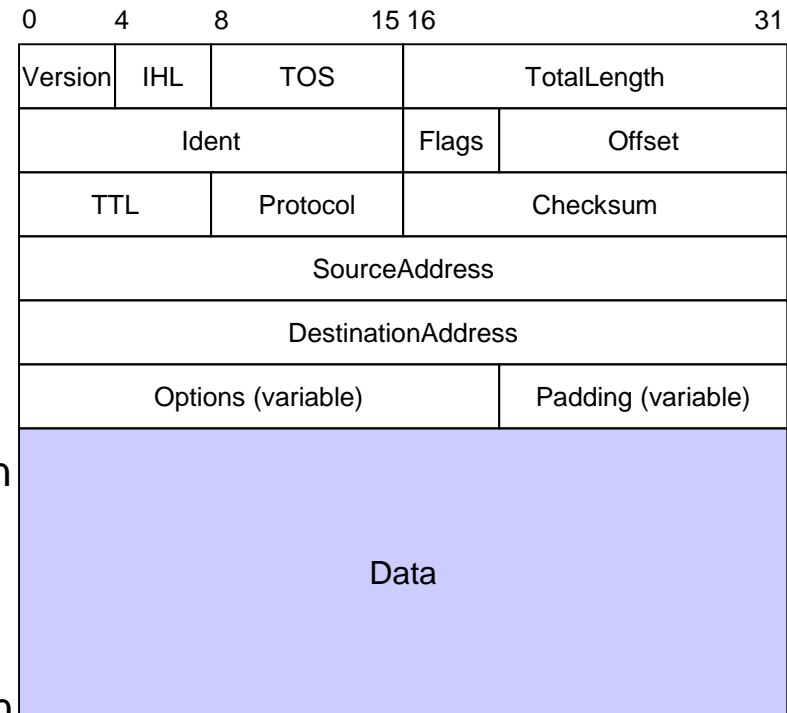
- **IP** = Internet Protocol
- **Yhteydetön:**
 - ei yhteydenmuodostusta
 - ei resurssien varausta
- Informaation siirto **diskreetteinä paketteina**
 - vaihtelevanmittaisia
 - sisältää otsikon, jossa mm. kohteen globaali osoite
- **Best Effort** –palvelumalli
 - verkon solmut, reitittimet, toimittavat paketteja eteenpäin “parhaan kykynsä mukaan”
 - paketteja saattaa kadota, ne voivat viivästyä, tai niiden järjestys saattaa muuttua



⇒ “älykkyyys” implementoitava verkon reunalle, päätelaitteisiin

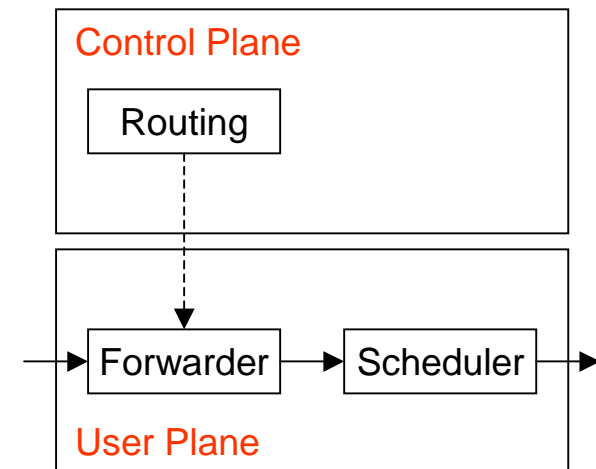
IP-paketti

- IP-paketti = **tietosähke** (datagram)
 - vaihtelevanpituisen (4 tavun monikerta)
 - **Otsikko** (väh. 20 tavua) [RFC791]:
 - **Version**: nykyinen versio 4
 - **IHL**: otsikon pituus
 - **TOS**: pakettien priorisointiin
 - **TotalLength**: kokonaispituus
 - **Ident, Flags, Offset**: pirstoutumiseen
 - **TTL**: jäljelläoleva elinikä hyppyinä
 - **Protocol**: kertoo ylemmän tason protokollan, esim. TCP (6), UDP (17)
 - **Checksum**: otsikon virheettömyyteen
 - **SourceAddress, DestinationAddress**: lähteen ja määränpään globaalit Internet-osoitteet



IP-pakettien käsittelymekanismit BE-reitittimissä

- Pakettien **toimitus eteenpäin** (forwarding):
 - reitittimen prosessi, joka
 - valitsee paketin sisääntulopuskurista,
 - tutkii sen määränpääosoiteen,
 - katsoo reititystaulusta ko. osoitetta vastaavan ulosmenon,
 - ja vie ko. paketin ko. ulosmenopuskuriin
- Pakettien **skedulointi** (scheduling)
 - reitittimen prosessi, joka
 - valitsee paketin ulosmenopuskurista ja
 - käynnistää sen lähetyksen ulosmenolinkillä
 - tavallisin skedulointialgoritmi: **FIFO**
- Vrt. **reititys** (routing):
 - hajautettu prosessi, jolla luodaan reitittimien reititystaulut (routing table)



Reititys

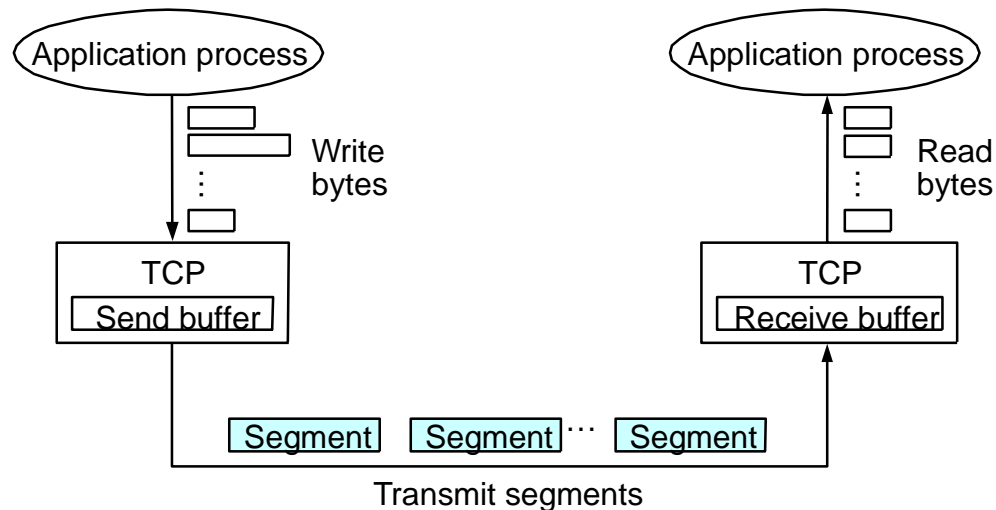
- Internet jakaantuu **hallintapiireihin** (routing domain)
- Reititysongelma voidaan jakaa hierarkkisesti
 - yhden piirin sisällä (intradomain) pyritään löytämään **optimaaliset reitit**
 - etsi edullisin reitti minkä tahansa kahden saman piirin solmun välillä, kun ko. piirin topologia ja linkkien kustannukset tunnetaan
 - tähän ongelmaan on esitetty erilaisia algoritmeja
 - **Bellman-Ford**-algoritmissa solmut kertovat naapureilleen etäisyydet kaikkiin muihin piirin solmuihin, esim. **RIP**
 - **Dijkstra**-algoritmissa solmut kertovat kaikille muille piirin solmuille etäisyytensä naapurisolmuihin, esim. **OSPF**
 - yleensä linkin kustannukseksi (so. kahden naapurisolmun etäisyydeksi) valitaan 1, jolloin optimitreitit minimoivat tarvittavien hyppyjen lkm:n
 - eri piirien välillä (interdomain) tärkeintä on selvittää **saavutettavuus**
 - etsi jokin reitti minkä tahansa kahden piirin välillä, kun piirien välinen topologia tunnetaan
 - esim. **BGP**

Kuljetuskerroksen päästä-päähän protokollat

- Verkkokerroksen (IP) päällä on kuljetuskerros, joka hoitaa IP-pakettien käsittelyn päätelaitteissa. Tähän on käytössä kaksi eri protokollaa.
- **TCP** = Transmission Control Protocol
 - **yhteydellinen**
 - tarkoitettu elastiselle, ei-reaaliaikaiselle liikenteelle
 - ei absoluuttisia QoS takeita, mutta takaa luotettavan tavupohjaisen tiedonsiirron
 - liikenteen valvontaan kehitetty oma vuonohjaus/ruuhkanhallintamenettely, joka perustuu adaptiivisen liukuvan ikkunan käyttöön
- **UDP** = User Datagram Protocol
 - **yhteydetön**
 - soveltuu erityisesti transaktioille (interaktiiviselle liikenteelle, jossa siirrettävät sanomat tyypillisesti lyhyitä)
 - soveltuu myös (paremman puutteessa) virtaavalle, reaaliaikaiselle liikenteelle, jolloin UDP:n päällä tarvitaan vielä muita protokollia, esim. RTP
 - ei minkäänlaisia QoS takeita

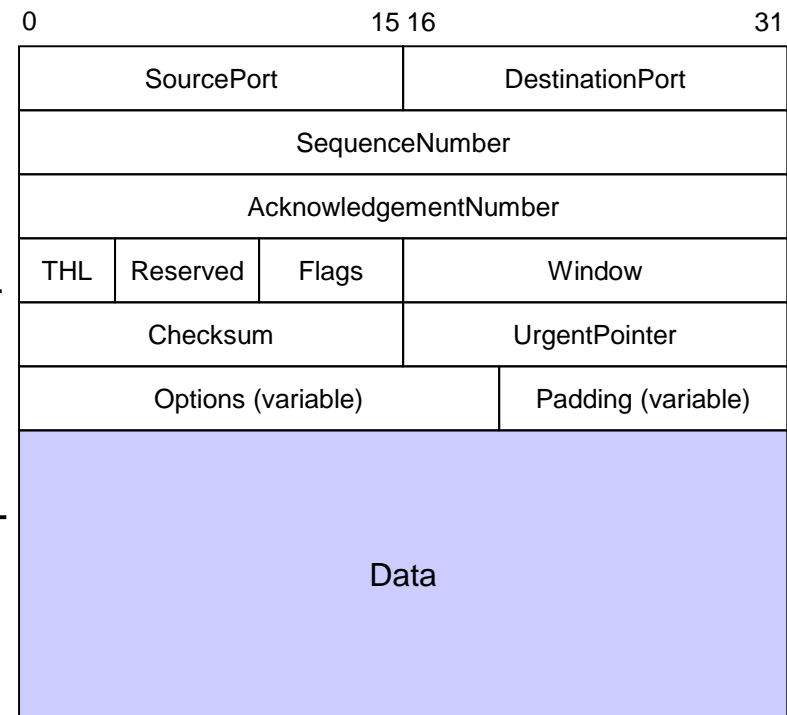
TCP

- **TCP** = Transmission Control Protocol
 - **yhteydellinen** kuljetuskerroksen päästä-päähän protokolla
 - IP:n päällä luotettava tavuvirran (reliable byte stream) siirto
 - pakettien perillemeno oikeassa järjestyksessä varmistetaan kuittauksin ja uudelleenlähetyksin
 - **vuonohjaus** (flow control): estää ylikuormittamasta vastaanottajan
 - **ruuhkanhallinta** (congestion control): estää ylikuormittamasta verkon



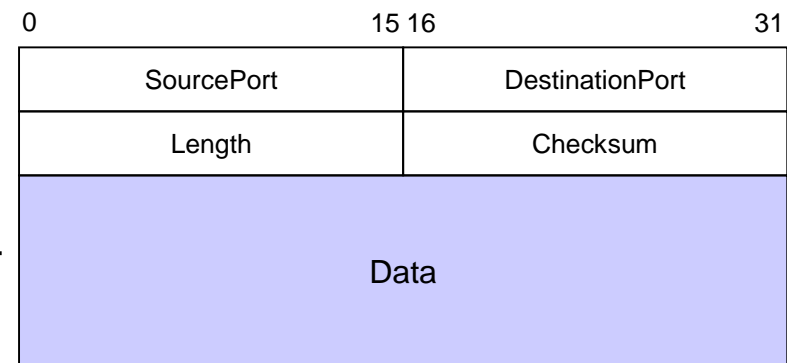
TCP-paketti

- TCP-paketti = **lohko** (segment)
 - vaihtelevanpituisen (4 tavun monikerta)
 - kuljetetaan läpi verkon IP-paketissa
 - **Otsikko** (väh. 20 tavua) [RFC793]:
 - **SourcePort, DestinationPort**: lähettäjän ja vastaanottajan identifiointiin
 - **SequenceNumber**: lähetettävän lohkon ensimmäisen tavun indeksi
 - **AcknowledgementNumber**: seuraavaksi vastaanotettavan tavun indeksi
 - **THL**: otsikon pituus
 - **Flags**: lipuilla kerrotaan esim. yhteyden avauksesta ja lopetuksesta
 - **Window**: seuraavaksi vastaanotettavien tavujen kokonaismäärä
 - **Checksum**: pakollinen, sisältäen datan, TCP-otsikon ja IP-osoitteet



UDP

- **UDP = User Datagram Protocol**
 - **yhteydetön** kuljetuskerroksen päästä-päähän protokolla
 - IP:n päällä vain multipleksointipalvelu
 - ei takeita pakettien perillemenosta (siis epäluotettava)
 - ei vuonohjausta (flow control): voi ylikuormittaa vastaanottajan
 - ei ruuhkanhallintaa (congestion control): voi ylikuormittaa verkon
- UDP-paketti = **tietosähke** (datagram)
 - vaihtelevanpituisen (4 tavun monikerta)
 - kuljetetaan läpi verkon IP-paketissa
 - **Otsikko** (8 tavua) [RFC768]:
 - **SourcePort, DestinationPort**: lähettäjän ja vastaanottajan identifiointiin
 - **Length**: kokonaispituus
 - **Checksum**: optionaalinen, sisältäen datan, UDP-otsikon ja IP-osoitteet



Sisältö

- Johdanto
- IP-verkot
- Liikenteen- ja ruuhkanhallinta Internetissä
- TCP ruuhkanhallinta
- QoS arkkitehtuurit

Liikenteen- ja ruuhkanhallintamekanismit eri aikaskaaloissa

Response time:	Proactive methods:	Reactive methods:
Long term (hours - days)	- Routing algorithm - MPLS Traffic Engineering	
Connection duration (secs - mins)		
RTT (ms - secs)	- TCP Vegas congestion avoidance	- TCP flow control - TCP congestion control
Packet handling time (μ s - ms)	- Scheduling - Active queue management	- Queue management

source based methods

router based methods

Lähteissä toteutetut menetelmät

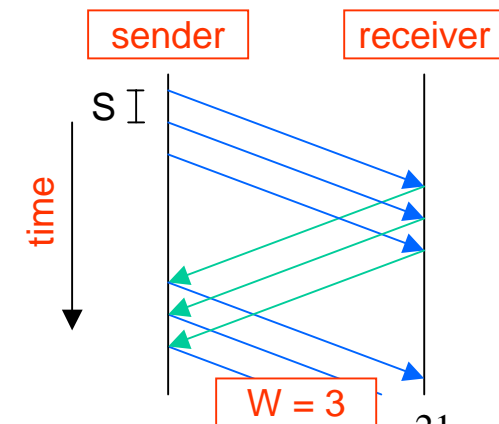
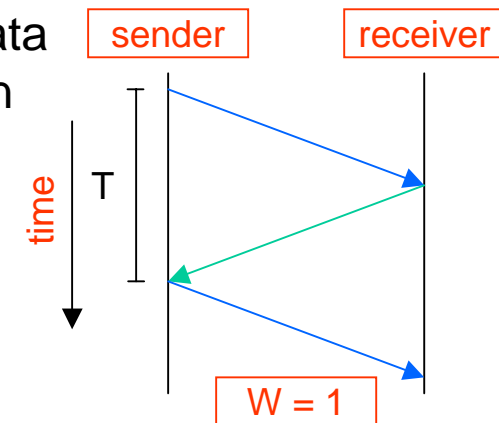
- **TCP vuonohjaus** (flow control)
 - estää lähdettä ylikuormittamasta vastaanottajan
 - menetelmänä adaptiivinen liukuva ikkuna
 - vastaanottaja kertoo, montako tavua hän on valmis vastaanottamaan
- **TCP ruuhkanhallinta** (congestion control)
 - estää lähdettä ylikuormittamasta verkon solmuja
 - menetelmänä adaptiivinen liukuva ikkuna
 - verkosta ei tule suoraan tietoa, montako tavua lähde voi lähettää
 - sen sijaan lähteen on itse pääteltävä, milloin verkko on ruuhkainen
 - tällaisen signaalin antaa paketin katoaminen
 - paketin kadotessa ikkunaa pienennetään
 - muutoin sitä kasvatetaan (verkon tilan tunnustelemiseksi)
- **TCP Vegas ruuhkanvälttäminen** (congestion avoidance)
 - tavoitteena voidaan pakettien lukumäärän rajoittaminen pullonkaulalinkeissä (ja sitä kautta ruuhkan ennaltaehkäiseminen)
 - ikkunan kokoa hallitaan vertaamalla toteutunutta ja lyhyintä kiertoaikaa

Reitittimissä toteutetut menetelmät

- **Pakettien skedulointi (scheduling)**
 - seuraavaksi lähetettävän paketin valinta
 - **FIFO** (First In First Out): paketit lähetykseen saapumisjärjestyksessä
 - **vuopohjaiset menetelmät** (kullekin aktiiviselle vuolle oma jono)
 - **reilu jonotus** (Fair Queueing, FQ): eri voiden paketit lähetykseen tasapuolisesti
 - **painotettu reilu jonotus** (Weighted Fair Queueing)
 - **prioriteettipohjaiset menetelmät**
- **Jononhallinta (queue management)**
 - poistettavan paketin valinta jonon täytyessä
 - **häntäpudotus** (Tail Drop): saapuva paketti hylätään
 - **satunnaispudotus** (Random Drop)
- **Aktiivinen jononhallinta (active queue management)**
 - pakettien satunnainen poistaminen jo ennen jonon täyttymistä
 - ehkäisee ruuhkan syntymistä ja TCP-lähteiden synkronoitumista
 - esim. **RED** (Random Early Detection)

Liukuva ikkuna

- Luotettavan tiedonsiirron takaamiseksi lähetetty data pitää **kuitata** (acknowledge). Pakettien katoamisen varalta tarvitaan myös **aikavalvontaa** (timeout) ja **uudelleenlähetyksiä** (retransmission).
- Yksinkertainen menetelmä: **stop-and-wait**
 - uusi paketti lähetetään, kun edellinen on kuitattu
 - ongelma: linkin matala käyttöaste
- Tehokkaampi: **liukuva ikkuna** (sliding window)
 - ikkunan koko W kertoo, montako kuitaamatonta pakettia voi korkeintaan olla lähetettynä
 - stop-and-wait: $W = 1$
 - ikkunan koko W yhdessä kiertoajan T ja paketin lähetysajan S kanssa määrää lähetysnopeuden R :
 - $R = \min\{1/S, W/T\}$
 - ikkunalla on siis efektiivinen yläraja: $W \leq T/S$ (eli kiertoajan T ja kaistan $1/S$ tulo)

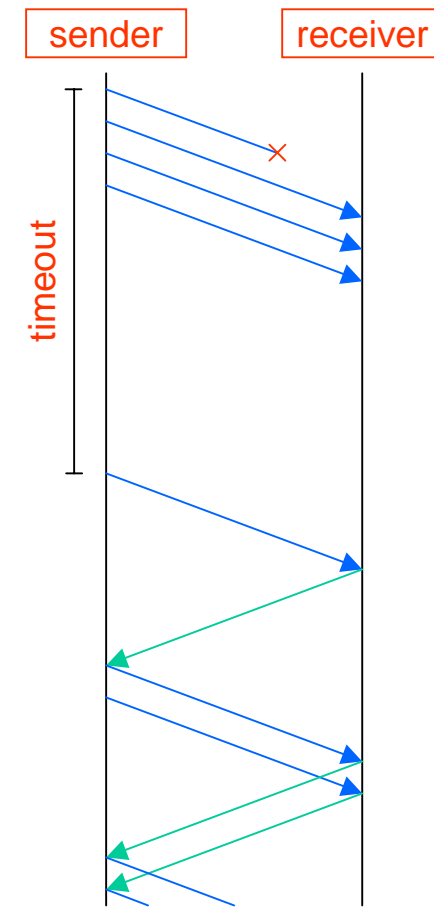


Adaptiivisen liukuvan ikkunan käyttö vuonohjaukseen

- Liukuvaa ikkunaa voidaan käyttää vuonohjaukseen.
- Jos vastaanottaja tyhjentää puskuriaan vakionopeudella C , niin lähetysikkunalla on yläraja $W \leq CT$, joka estää lähettä ylikuormittamasta vastaanottajan.
- Jos vastaanottaja tyhjentää puskuriaan epäsäännöllisemmin, niin vastaanottajan on eksplisiittisesti kerrottava, kuinka monta pakettia se on valmis vastaanottamaan kullakin hetkellä, ts. kuinka paljon sen puskuriin juuri sillä hetkellä mahtuu paketteja.
 - TCP vuonohjauksessa tämä tieto kulkee (tosin tavuina) kuittaavan TCP-paketin Window-kentässä

Adaptiivisen liukuvan ikkunan käyttö ruuhkanhallintaan

- Liukuvaa ikkunaa voidaan käyttää myös ruuhkanhallintamenetelmänä
 - Tavoitteena on sovittaa ikkunan koko W vastaamaan kyseiselle vuolle (juuri sillä hetkellä) verkossa käytettävissä olevaa kaistaa C ja kiertoaikaa T , ts. $W = CT$
 - Tehtävää vaikeuttaa tietysti se, että sekä C että T vaihtelevat satunnaisesti, eikä niitä mikään tahoo ilmoita lähteelle (saati sitten täsmällistä ylärajaa ikkunalle)
 - Lähde voi kuitenkin epäsuorasti saada tietoa ruuhkasta tekemällä havaintoja pakettien katoamisesta, esim. uudelleenlähetyksajastimen laukeaminen viittaisi tällaiseen tilanteeseen
 - Paketin kadotessa pitää lähetyssikkunan kokoa tietysti pienentää (eli pudottaa lähetyssnopeutta)
 - Toisaalta taas paketin mennessä läpi voidaan ikkunaa kasvattaa (eli kasvattaa lähetyssnopeutta)
 - TCP ruuhkanhallinta käyttää näitä ideoita



Sisältö

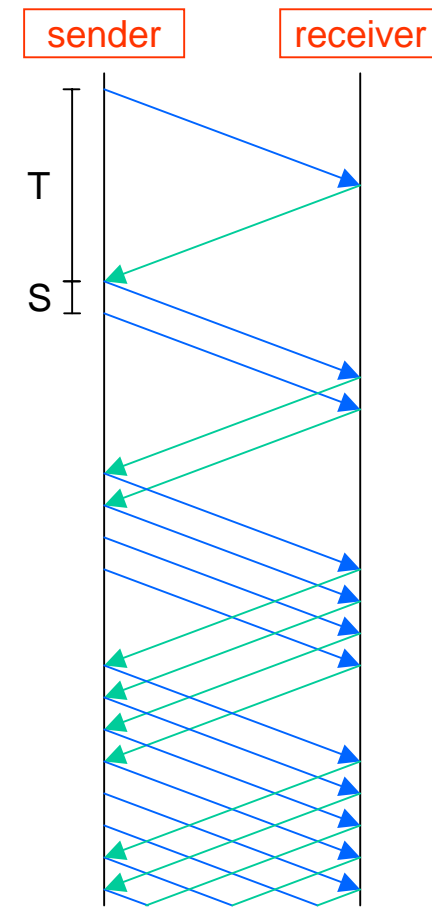
- Johdanto
- IP-verkot
- Liikenteen- ja ruuhkanhallinta Internetissä
- TCP ruuhkanhallinta
- QoS arkkitehtuurit

TCP ruuhkanhallinta

- Alkuperäinen TCP [RFC793] (1981) käytti adaptiivista liukuvan ikkunan menetelmää (adaptive sliding window) vain vuonohjaukseen
 - Tämän seuraksena Internet kaatui aika-ajoin (congestion collapse): uudelleenlähetykset tukkivat verkon
- Jacobsonin (1988) ehdotuksesta adaptiivista liukuvan ikkunan menetelmää ruvettiin käyttämään myös ruuhkanhallintaan.
 - Tällöin TCP:hen lisättiin seuraavat ruuhkanhallintamekanismit:
 - **hidas aloitus** (slow start)
 - **ruuhkan välttäminen** (congestion avoidance)
 - **nopea uudelleenlähetyks** (fast retransmit)
 - Tätä versiota kutsutaan **TCP Tahoeksi**
- Myöhemmin Jacobson (1990) ehdotti vielä joitakin lisämuutoksia.
 - Tällöin TCP-ruuhkanhallintaan lisättiin vielä
 - **nopea toipuminen** (fast recovery)
 - Tätä versiota kutsutaan **TCP Renoksi**
- TCP-ruuhkanhallintamekanismit on koottu RFC2581:een (1999)

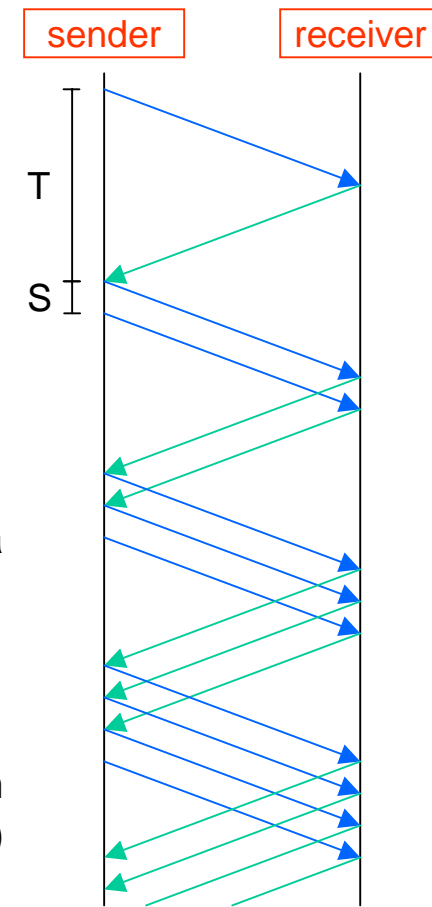
Slow Start

- Koska alkuperäisessä TCP:ssä liukuvaa ikkunaa käytettiin vain vuonohjaukseen, saattoi yhteyden alussa lähteä verkkoon suurikin purske paketteja.
 - Tällöin tietysti oli suuri vaara pakettien katoamiselle (reitittimien puskureiden vuotaessa yli).
 - Aikavalvontarajan (timeout) ylittyessä tehdyt uudelleenlähetykset vain pahensivat ruuhkaa.
- Jacobson (1988) havaitsi, että saavuttaakseen tasapainon lähteen pitää tunnustella käytettävissä olevaa kaistaa sulavasti
- Menetelmänä on aloittaa lähetys varovaisesti (cwnd = 1). Tästä nimi **hidas aloitus** (slow start). Ruuhkaikkunan (cwnd) kasvatus tämän jälkeen tapahtuu kuitenkin eksponentiaalisti: jokainen kuittaus kasvattaa ikkunaa yhdellä, eli kiertoajan kuluessa saatu ikkunallinen kuittauksia tuplaa ikkunan koon.



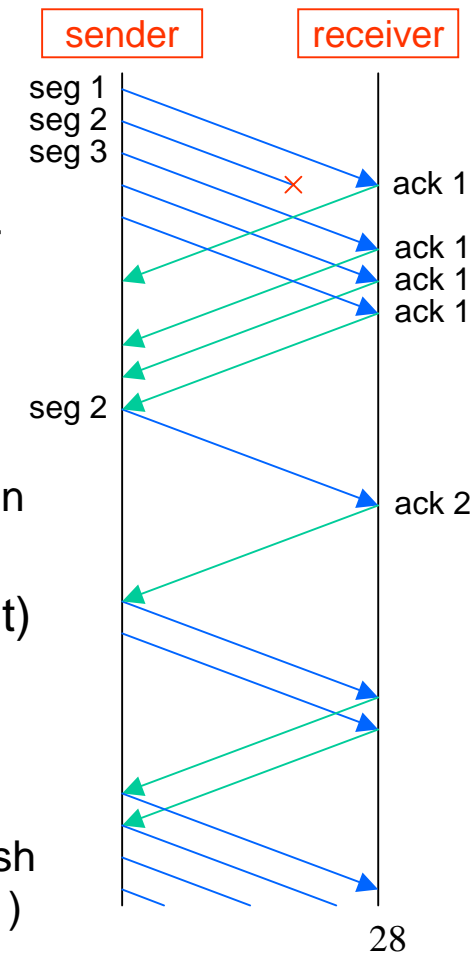
Congestion Avoidance

- Toinen Jacobsonin (1988) tekemä havainto liittyy tasapainotilanteen ylläpitämiseen
- Menetelmänä on siirtyminen hitaasta aloituksesta **ruuhkan välttämiseen** (congestion avoidance), kun ruuhkaikkunan (cwnd) koko on kasvanut ”tarpeeksi”
 - tätä kynnystäkin (sssthresh) vaihdellaan dynaamisesti
- Ruuhkan välttämisvaiheessa
 - ikkunaa kasvatetaan lineaarisesti: kiertoajan kuluessa saatu ikkunallinen kuittauksia kasvattaa ikkunaa yhdellä paketilla
- Mutta uudelleenlähetysajastimen lauetessa
 - kadonneeksi tulkittu paketti lähetetään uudelleen ja
 - kynnyks asetetaan ruuhkaikkunan puolikkaaksi ($sssthresh = cwnd/2$) ja siirrytään hitaaseen aloitukseen ($cwnd = 1$)
- Tätä menetelmää kutsutaan myös **AIMD**:ksi (additive increase, multiplicative decrease)



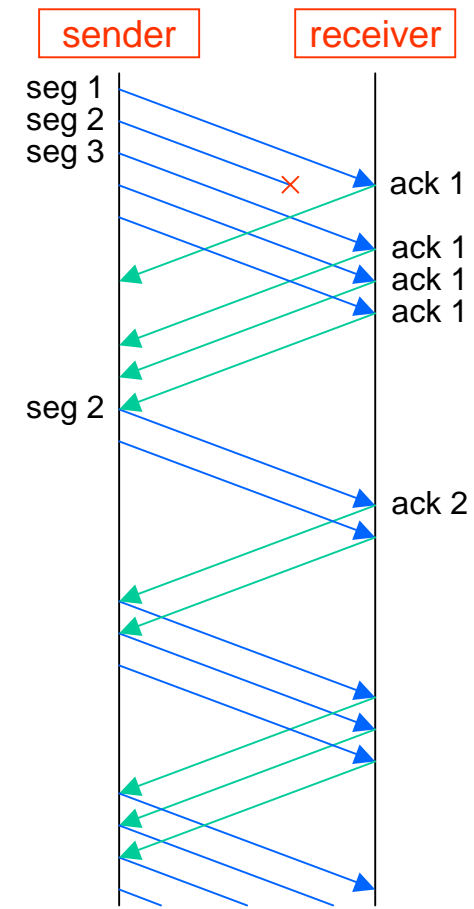
Fast Retransmit

- Kolmas Jacobsonin (1988) havainto liittyi pakettien katoamisten (jotka siis tulkitaan indikaatioksi ruuhkasta) havainnointiin:
 - paitsi ajastimen laukeaminen, myös useampi peräkkäinen saman lähetetyn paketin kuittaus (ts. peräkkäisten kuittaavien TCP-pakettien AcknowledgmentNumber-kenttien pysyminen samana) voidaan kohtalaisella varmuudella tulkita paketin katoamiseksi
 - kahdentunut kuittaus saattaa tosin liittyä myös pakettien järjestyksen vaihtumiseen (ilman katoamisia)
- **Nopeassa uudelleenlähetyksessä** (Fast Retransmit) kolmas peräkkäinen kahdentunut kuittaus tulkitaan paketin katoamiseksi, jolloin
 - kadonneeksi tulkittu paketti lähetetään uudelleen ja
 - kynnyks asetetaan ruuhka-ikkunan puolikkaaksi ($ssthresh = cwnd/2$) ja siirrytään hitaaseen aloitukseen ($cwnd = 1$) [TCP Tahoe]



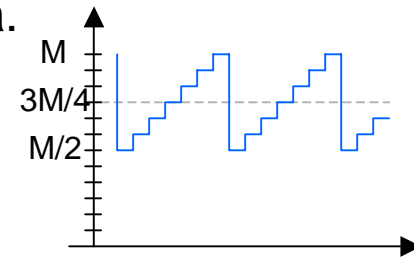
Fast Recovery

- Myöhemmin Jacobson (1990) vielä havaitsi, että
 - paketin kadotessa on turhaa tehdä hidas aloitus, jos lähde on jo päässyt lähelle tasapainoa
 - kolmannen tuplakuittauksen tullessa, saattaa muitakin kuittauksia olla vielä tulossa, jotka ilmaisevat tasapainoisen lähetysnopeuden
- **Nopeassa toipumisessa (Fast Recovery)** kolmannen tuplakuittauksen jälkeen ei siirrytäkään hitaaseen aloitukseen, vaan pelkästään
 - puolitetaan ruuhkaikkuna ($cwnd = ssthresh = cwnd/2$) [TCP Reno]



Lähetysnopeuden riippuvuus pakettikadosta tasapainossa

- Tarkastellaan TCP Renon ruuhkan välttämisvaihetta.
 - jos paketti ei katoa, ikkunaa kasvatetaan lineaarisesti
 - $W = W + 1$ kiertoajan T välein
 - jos paketti katoaa, niin ikkuna puolitetaan
 - $W = W/2$
- Yksinkertaisuuden vuoksi oletetaan kiertoaika T vakioksi. Tavoitena on laskea lähetysnopeus R tasapainotilanteessa.
 - Oletetaan (jälleen yksinkertaisuuden vuoksi), että yksi paketti katoaa aina, kun ikkuna kasvaa “liian suureksi”, tarkemmin sanottuna kun $W = M$
 - Tällöin ikkunan W koko vaihtelee jaksollisesti
 - $W = M, M/2, M/2 + 1, M/2 + 2, \dots, M, M/2, M/2 + 1, M/2 + 2, \dots$
 - Katoamisjakso kestää $M/2 + 1$ kiertoaikaa ja sen aikana lähetetään keskimäärin $3M/4$ pakettia kiertoajassa
 - Keskimääräiseksi lähetysnopeudeksi tulee $R = (3M)/(4T)$
 - Pakettikadoksi tulee (likimain) $p = 8/(3M^2)$
 - Siis $R = \sqrt{3}/(\sqrt{2} \cdot \sqrt{p} \cdot T)$ pakettia aikayksikössä



Sisältö

- Johdanto
- IP-verkot
- Liikenteen- ja ruuhkanhallinta Internetissä
- TCP ruuhkanhallinta
- QoS arkkitehtuurit

QoS arkkitehtuurit

- **Integroidut palvelut** (Integrated Services, **IntServ**)
 - määrittely IETF:ssä 1995-1997
 - **hienojakoinen** (fine grained)
 - vuokohtainen palvelun laatu, vrt. ATM-yhteydet
 - **absoluuttiset päästä-päähän laatutakeet** mahdollisia (pääsynvalvonta & resurssien varaus)
 - skaalautuvuusongelmia vuokohtaisen palvelun/kirjanpidon vuoksi
- **Eriytetyt palvelut** (Differentiated services, **DiffServ**)
 - määrittely IETF:ssä 1998-2000
 - yrittää ratkaista IntServin skaalautuvuusongelman niputtamalla voita runkoverkossa
 - **karkeajakoinen** (coarse grained)
 - staattiset palvelutasosopimukset (Service Level Agreement, SLA)
 - palvelun laatu määritellään aggregoidulle liikenteelle (ei siis yksittäisille voille vaan vuokimpuille)
 - vain **suhteelliset hyppykohtaiset laatutakuut**

IntServ

- **Palveluluokat (Service Class)**
 - **Taattu-palvelu (Guaranteed Service)**, vrt. CBR
 - reaaliaikaisille lähteille, joilla ehdoton viivevaatimus
 - edellyttää pääsynvalvontaa, resurssien varausta ja pakettien priorisointia
 - **Kontrolloitu kuorma -palvelu (Controlled Load Service)**, vrt. VBR
 - sopeutuville lähteille, jotka toimivat hyvin kunhan kuorma on rajoitettu
 - edellyttää pääsynvalvontaa ja palveluluokkien erottelua reitittimissä
- **Vuotason mekanismit**
 - **Voiden määrittely (Flowspec)**
 - liikenteen kuvaus (TSpec) ja palveluvaatimuksen kuvaus (RSpec)
 - **Pääsynvalvonta (Admission control)**
 - **Resurssien varaus** signaloimalla (Resource reservation protocol, RSVP)
- **Pakettitason mekanismit**
 - Pakettien **luokittelu** (classification) ja **priorisointi**
 - **Palveluokkien erottelu** skeduloimalla paketit vuokohtaisella WFQ:lla

DiffServ

- Ei palveluluokkia eikä absoluuttisia palvelun laatutakuuta, vaan pakettien käsittelyluokat ja –mekanismit sekä suhteelliset laatutakuut
- **Pakettien käsittelyluokat** (Per Hop Behaviour, PHB)
 - **Joudutettu toimitus** (Expedited Forwarding, EF)
 - tavoittena lyhyet viiveet ja viiveenvaihtelut sekä alhainen pakettikato
 - toteutus antamalla ko. luokan paketeille ylin prioriteetti
 - **Varmistettu toimitus** (Assured Forwarding, AF)
 - kullekin AF-luokalle (max. 4) allokoidaan oma minimikaista
 - AF-luokan sisällä 3-tasoinen pakettien arvojärjestys (drop precedence)
- Monimutkaisemmat vuokohtaiset mekanismit **reunareitittimissä** (edge)
 - **Pakettien luokittelu** (classification) ja **merkkkaus** (marking)
 - **Voiden valvonta: mittaus** (metering) ja **muotoilu** (shaping)
- Yksinkertaiset vuonippukohtaiset mekanismit **runkoreitittimissä** (core)
 - **Käsittelyluokkien erottelu** skeduloimalla paketit luokkakohtaisella WFQ:lla
 - Luokka- ja arvokohtaiset **jononhallintamekanismit**

Kirjallisuutta

- 1 W. Stallings (1998)
 - “High-Speed Networks: TCP/IP and ATM design principles”
 - Prentice-Hall, New Jersey
- 2 L.L. Peterson and B.S. Davie (2000)
 - “Computer Networks – A Systems Approach”, 2nd edition
 - Morgan Kaufmann, San Francisco
- 3 V. Jacobson (1988)
 - “Congestion Avoidance and Control”
 - ACM SIGCOMM '88, Stanford, CA, August 1988

THE END

