

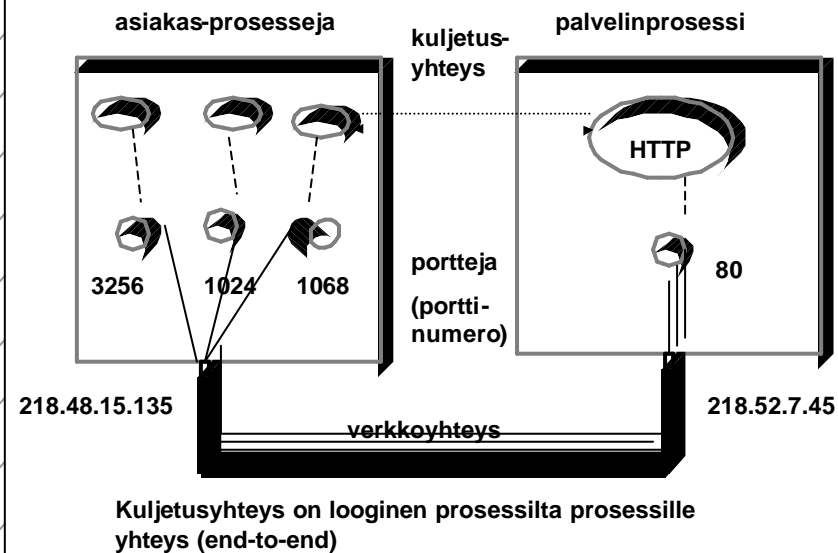
3. Kuljetuskerros

3.1. Kuljetuspalvelu

- 'End- to- end'
 - prosessilta prosessille looginen yhteys
 - portti
 - verkkokerros koneelta koneelle
 - IP-osoite
- peittää verkkokerroksen puutteet
 - jos verkkopalvelu ei ole riittävän hyvä, sitä voidaan parantaa kuljetuskerroksella
 - kuljetuskerros huomaa verkkokerroksen kadottamat paketit ja pyytää niiden uudelleenlähetyistä

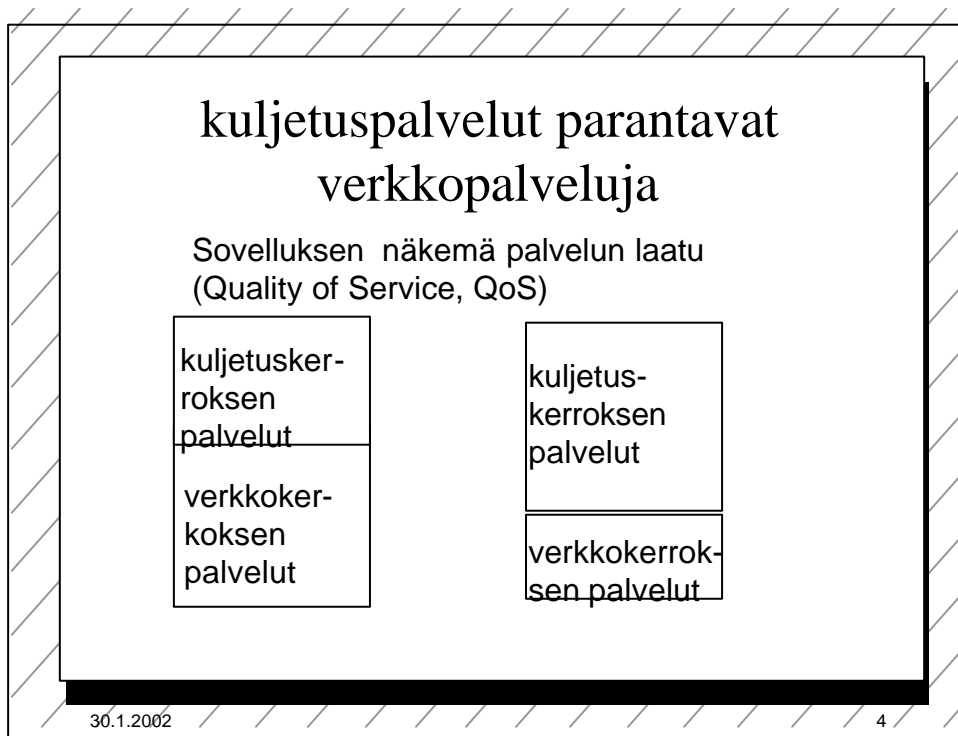
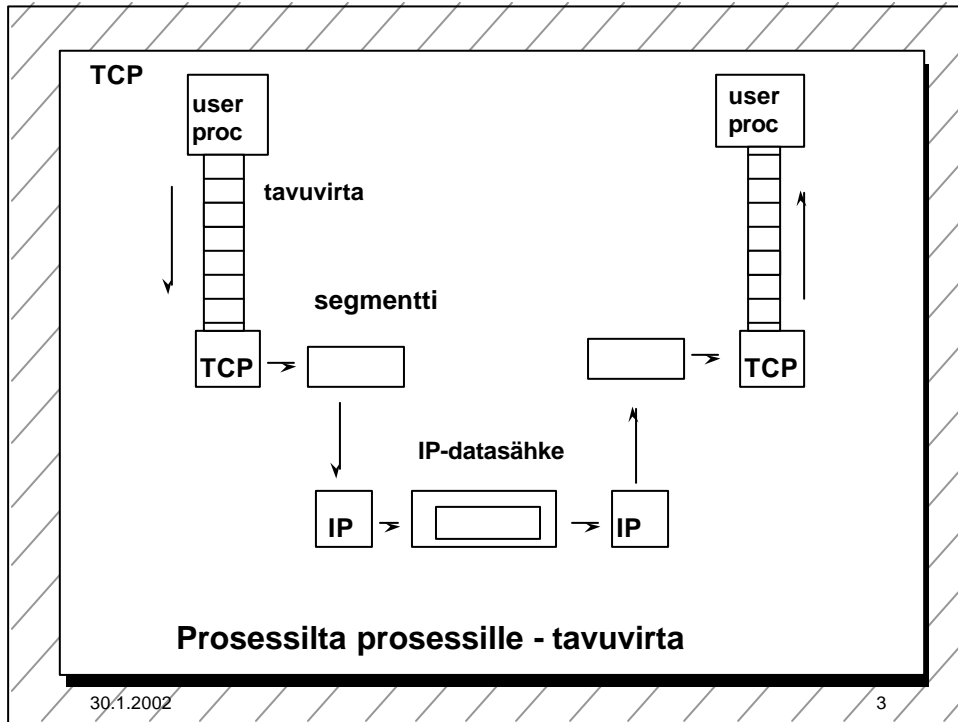
30.1.2002

1



30.1.2002

2



Sovelluksen vaatimuksia kuljetuspalvelulle:

- **Virheetön, luotettava**
- **järjestyksen säilyttävä**
- **kaksoiskappaleet karsiva**
- **mielivaltaisen pitkiä sanomia salliva**
- **vuonvalvonnan mahdollistava**

Verkkokerros kuitenkin voi

- **kadottaa sanomia**
- **toimittaa sanomat epäjärjestyksessä**
- **viivyttää sanomia satunnaisen pitkän ajan**
- **luovuttaa useita kopioita samasta sanomasta**
- **rajoittaa sanomien kokoa**

Internetin kuljetuskerros

- **UDP (User Datagram Protocol)**
 - yhteydetön, epäluotettava palvelu
- **TCP (Transmission Control Protocol)**
 - yhteydellinen, luotettava palvelu
 - **virhevalvonta**
 - havaitsee ja korjaa siirrossa syntyneet virheet
 - **vuonvalvonta**
 - ei ylikuormita vastaanottajaa
 - **ruuhkanvalvonta**
 - huolehtii ettei verkko pääse ruuhkautumaan

Sovelluksien datavirtojen erottaminen

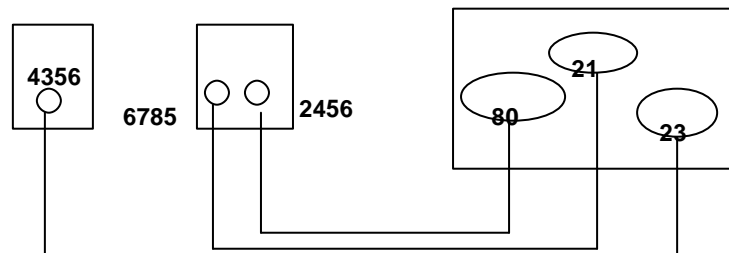
- IP-osoite
 - osoittaa koneen yksikäsitteisesti
- Sovellusprosessi tunnistetaan porttinumerosta (16 bittiä =>0-65535)
 - jokaisessa lähetetyssä segmentissä on
 - lähettäjän porttinumero
 - vastaanottajan porttinumero
- Yleisillä palvelimilla omat varatut porttinumerot (0-1023)
 - SMTP 25, HTTP 80, jne

30.1.2002

7

Asiakkaalle kuljetuskerros usein automaattisesti antaa käyttöön jonkin vapaan porttinumeron yhteyden ajaksi

Palvelimilla kiinteät numerot yhteydenottoa varten

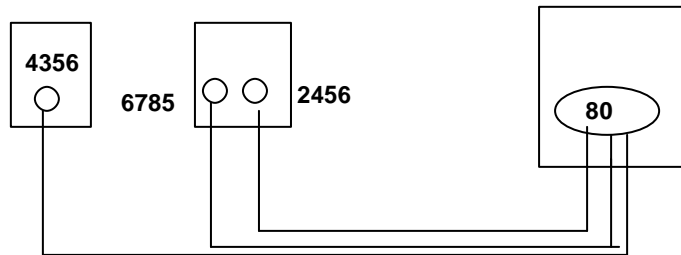


Kolme yhteyttä: 4356 <=> 23, 6785 <=> 21, 2456 <=> 80

30.1.2002

8

Tarvitaan sekä lähteen että kohteen porttinumerot

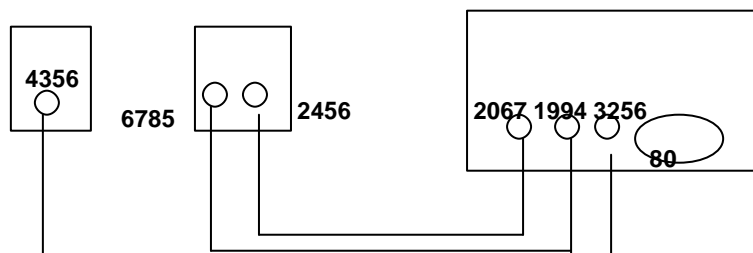


Kolme yhteyttä: 4356 \Leftrightarrow 80, 6785 \Leftrightarrow 80, 2456 \Leftrightarrow 80

30.1.2002

9

Palvelimessa yhteyksille uudet porttinumerot, jotta portti 80 voi ottaa vastaan uusia yhteyspyyntöjä

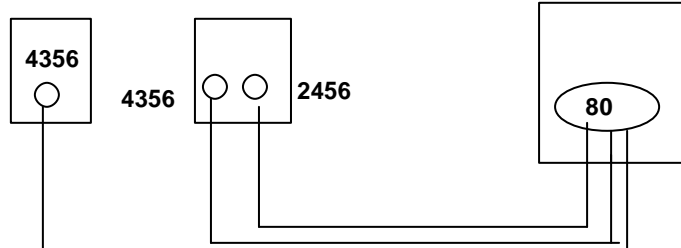


Kolme yhteyttä: 4356 \Leftrightarrow 3256, 6785 \Leftrightarrow 1994, 2456 \Leftrightarrow 2067

30.1.2002

10

Eri koneissa voidaan ottaa sama numero!



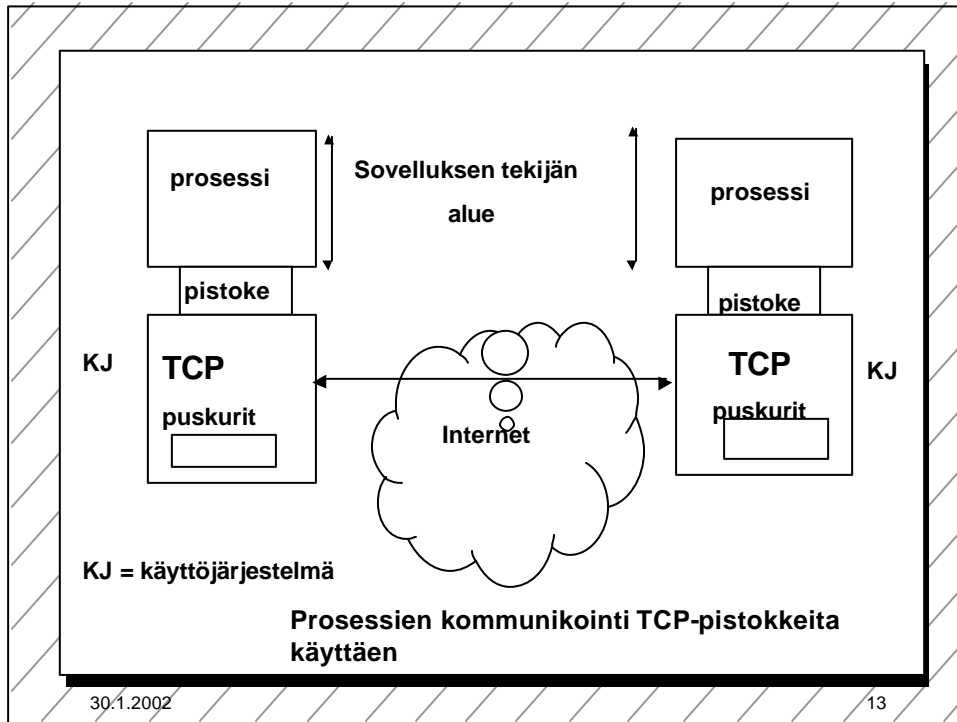
Kolme yhteyttä: 4356 \Leftrightarrow 80, 4356 \Leftrightarrow 80, 2456 \Leftrightarrow 80!

Kuljetusyhteydellä käytetään apuna myös IP-osoitetta:

=> koneilla eri IP-osoitteet, joten yhteydet pystytään erottamaan

Pistokerajapinta (Socket interface)

- Verkkopalvelun ja sitä käyttävän sovelluksen rajapinta
 - yleensä käyttöjärjestelmän tarjoama palvelu
 - pistokerajapinta alunperin Berkeley Unixin mukana, nyt lähes kaikissa käyttöjärjestelmissä
 - miten verkkoprotokollan tarjoamiin palveluihin päästään käsiksi sovelluksesta



■ pistoke (socket)

- TCP-yhteyden päätepiste sovellukselle
 - lähettäjällä ja vastaanottajalla oma pistoke
- pistokenumero 48 bittiä
 - koneen 32 bitin IP-osoite
 - 16 bitin porttinumero

TCP-yhteys

- kaksisuuntainen (full-duplex) kaksipisteyhteys
- tunnistetaan päätepisteinä olevien pistokkeiden tunnuksista (pistoke1, pistoke2)



30.1.2002

15

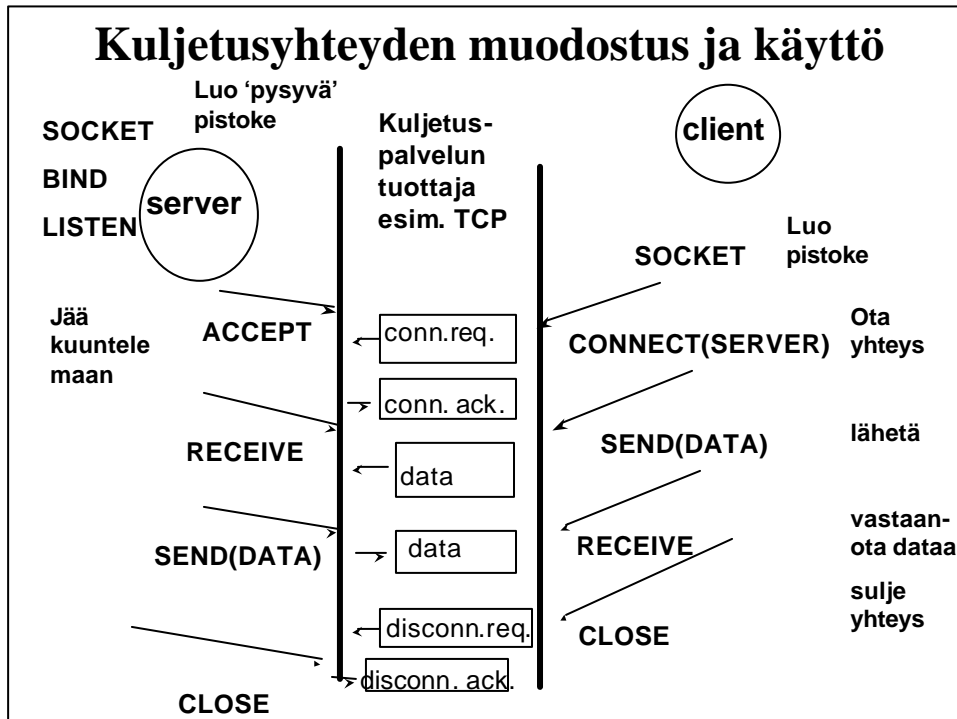
TCP:n pistokeprimitiivit

- SOCKET luo uusi yhteyden päätepistepistoke
- BIND anna pistokkeelle osoite
- LISTEN halukas vastaanottamaan yhteyksiä
- ACCEPT jää odottamaan yhteyksyrityksiä
- CONNECT yritä muodostaa yhteys
- SEND lähetä dataa yhteyttä pitkin
- RECEIVE vastaanota dataa yhteydeltä
- CLOSE pura yhteys (symmetrinen)

30.1.2002

16

Kuljetusyhteyden muodostus ja käyttö

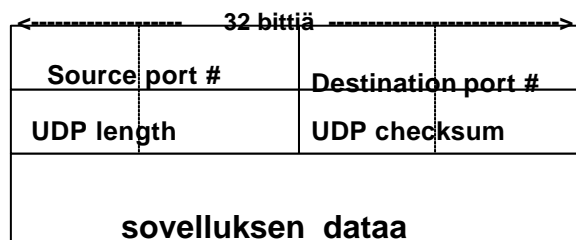


3.3 UDP

■ UDP (User Data Protocol)

- voidaan lähettää sanomia ilman yhteyden muodostusta

UDP-otsake



UDP-tarkistussumma

- Virheen havaitsemista varten otsakkeeseen liitetään tarkistussumma
 - kaikki segmentin 16 bitin sanat lasketaan yhteen ja summasta otetaan yhden komplementti
 - = muutetaan ykköset nolliksi ja nollat ykkösiksi
 - vastaanottaja laskee taas kaikkien segmentin sanojen (mukana myös tarkistussumma) summan
 - jos tulokseksi saadaan 16 ykköstä, niin ok!

30.1.2002

19

Esimerkki

- Lasketaan yhteen kolme 8 bitin mittaista sanaa:

- Lähettäjä vastaanottaja

1011 0100	1011 0100
0111 0101	1111 0101
1000 1101	1000 1101
=====	0100 1001
1011 0110	=====
	0111 1111
0100 1001	

Yhden komplementti

30.1.2002

20

- Miksi tarvitaan tarkistussumma?
 - Kaikki siirtoyhteyskerrokset eivät suorita tarkistuksia
- UDP-tarkistussumma ei ole kovin tehokas havaitsemaan virheitä!
- Se ei myöskään yritä toipua virheistä!
 - Jotkut toteutukset voivat tuhota virheellisen segmentin
 - jotkut antavat se sovellukselle varoituksen kera

UDP:n etuja:

- Yhteydetön
 - aikaa ei kulu yhteyden muodostamiseen ja purkamiseen
 - ei tarvita resursseja yhteyden tilatietojen ylläpitoon
- Otsake (= 8 tavua) pieni => pieni yleisrasite => lisää tehokkuutta
- Ruuhkanvalvonta ei säännöstele liikennettä

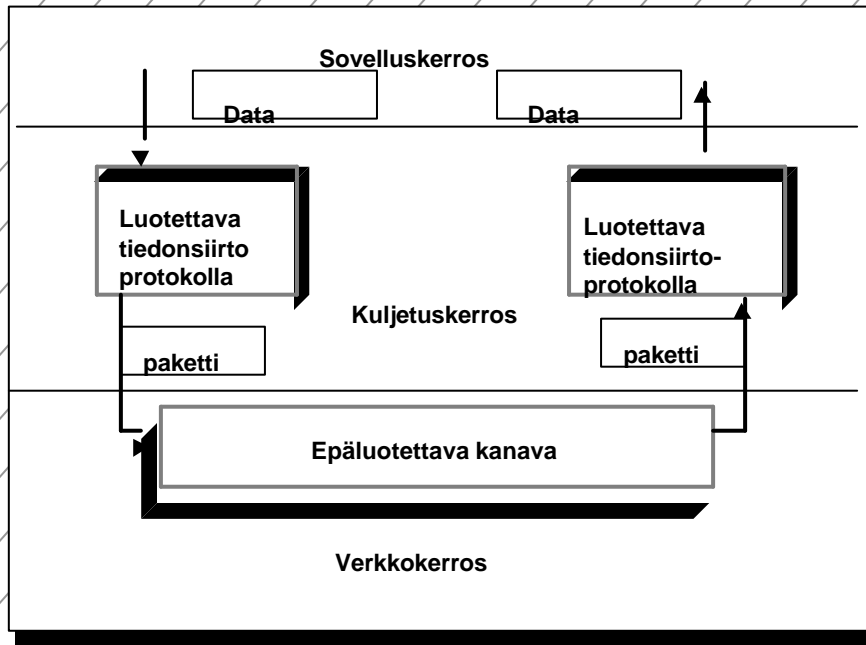
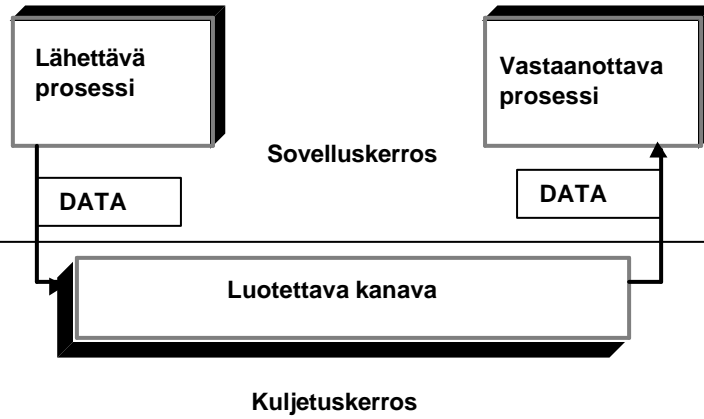
Tehtäviä:

- Lähetetään 10 tavun viesti UDP:llä.
 - Miten kauan kestää lähettäminen, jos lähetyksenopeus on 56 kbps?
 - $10 \text{ tavua} + 8 \text{ tavua} = 18 * 8 \text{ b} = 144 \text{ bittiä}$
 - $144 \text{ b} / 56\,000 \text{ b/s} = 2.57 \text{ ms}$
 - Miten suuri on etenemisviive, jos etäisyys lähettäjältä vastaanottajalle on 1000 km?
 - $1000\text{km} / 200\,000 \text{ km/s} = 5 \text{ ms}$
 - Miten suuri on UDP-otsakkeen aiheuttama yleisrasite (overhead)?
 - $8/18 = 0.44$ eli 44 %

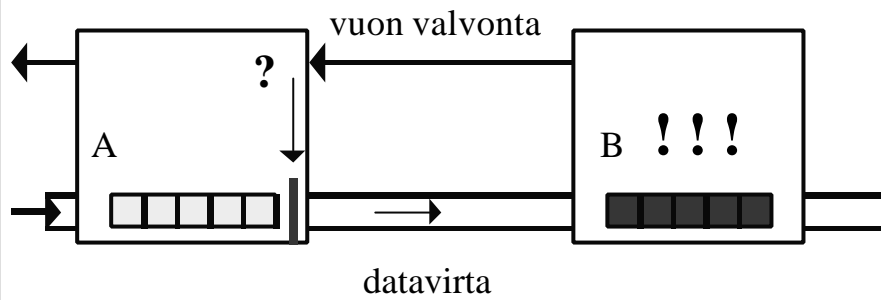
UDP:n käyttö

- Vaikka UDP on epäluotettava, se sopii monien sovellusten tarpeisiin:
 - Remote file server (NFS)
 - multimedia
 - Internet-puhelin
 - verkon hallinta (SNMP)
 - reititys (RIP)
 - nimipalvelu (DNS)
- Miksi nämä sovellukset suosivat UDP:tä?

3.4 Luotettava tiedonsiirto

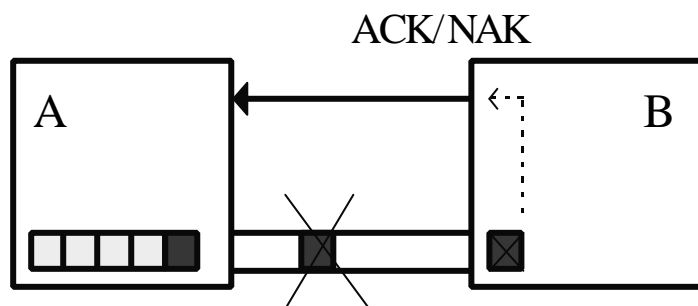


Vuon valvonta



■ X-ON / X-OFF : GO! | STOP!

Kohinainen kanava



- sanoma vääristyy => virhetarkistus
- sanoma katoaa => ajastin ja uudelleenlähetys
 - duplikaattien havaitseminen

Yksinkertainen **Stop and wait** -protokolla

■ Oletus

- virheetön siirto => ei huolta virheistä, mutta vuonvalvontaa tarvitaan

■ lähettäjä

- lähettää sanoman
- odottaa lupaa lähettää seuraava sanoma

■ vastaanottaja

- käsittelee sanoman
- lähettää tiedon (=antaa luvan) lähettäjälle

Entä jos virheitä?

- Sanomissa virheitä tai sanomat voivat puuttua kokonaan

- Myös kuittaukset voivat kadota

■ Tarvitaan

- virheen havaitseminen ja korjaaminen
 - tarkistussumma
 - kuittaus
 - uudelleenlähetys
- sanomien numerointi
- uudelleenlähetysajastin

Monimutkaisempi stop and wait - protokolla

■ ajastin lähettäjälle

- jos kuittausta ei kuulu, sanoma lähetetään automaattisesti uudelleen
- **kuittaus: ACK = 'ok, lähetä seuraava'**
- **uudelleenlähetys synnyttää kaksoiskappaleita!**

■ Sanomanumerointi

- jotta vastaanottaja tunnistaa kaksoiskappaleet
- Miten paljon numeroita tarvitaan?
 - » **Numero vie tilaa sanomassa!**

Stop and wait -protokollan suorituskyky

■ Esim. satelliittiyhteydellä

- 50 kbps, kiertoviive ~520 ms, sanoma 1000 bittiä
- kanavan käyttöaste < **4%**

■ => lähetetään useita sanomia ja sitten vasta odotetaan kiitauksia

- **ideaali: lähetykset liukuhihnalla (pipeline)**
 - lähetykset ja kiitaukset limittyvät
 - ei mitään odottelua
 - lähetyiskanava koko ajan käytössä
- suorituskyky kasvaa

Liukuvan ikkunan protokolla

(Sliding Window)

■ Lähetysikkuna

– ikkunan koko

- montako sanomaa saa korkeintaan olla kuittaamatta
- järkevä koko riippuu yhteyden tyypistä ja vastaanottajan kapasiteetista

– sisältö = mitkä sanomat saa lähettää

- sanomalla järjestysnumero
 - rajallinen, N bittiä => $2^{**}N$ arvoa
 - numerot käytettävä järjestyksessä

30.1.2002

33

■ Lähettäjä joutuu odottamaan vasta, kun kaikki ikkunan sanomat on lähetetty

- eli numerot käytetty

■ Kun kuittaus saapuu => ikkuna liukuu

- seuraavat numerot tulevat luvallisiksi

■ eli

- lähettäjä: tietyllä hetkellä sallittujen numeroiden joukko = lähettäjän ikkuna

- mitkä sanomat saa lähettää “etukäteen” odottamatta kuittausta

30.1.2002

34

■ Vastaanottajan ikkuna

- kullakin hetkellä sallittujen numeroiden joukko
 - mitä sanomia suostuu vastaanottamaan
- kuittaus muuttaa myös vastaanottajan ikkunan

■ ikkuna pysäyttää sanomien lähetyksen

- seuraava sanomanumero ei ole lähetyksikkunassa

■ ikkuna estää sanoman vastaanoton

- saadun sanoman numero ei ole vastaanottoikkunassa

Kun ikkunan koko on 1

■ Aina vain yksi sanoma kuittaamattomana

- => One Bit Sliding Window -protokolla
- ~ stop and wait -protokolla

■ sanomanumerot 0 ja 1 riittävät

■ ACK-sanoma identifioi viimeksi vastaanotetun virheettömän sanoman

- jotta kuittausduplikaatti ei voi kuitata väärää sanomaa
- ACK ilmoittaa joko
 - » seuraavaksi odotetun sanoman numeron
 - » viimeksi vastaanotetun sanoman numeron

■ Entä kun tapahtuu virhe?

- kaksi eri tapaa hoitaa
- toisto virheestä lähtien (go back n) (tai paluu n:ään)
- valikoiva toisto (selective repeat)

Toisto virheestä eli Paluu n:ään ('Go back n')

■ virheellisen sanoman havaittuaan

- vastaanottaja hylkää kaikkia sen jälkeiset sanomat eikä lähetä niistä kuittauksia
- => sanomat hyväksytään vain oikeassa järjestyksessä

■ kun lähettäjä ei saa kuittauksia,

- sen lähetysikkuna 'täyttyy'
- eikä se voi enää lähettää

■ lähettäjä ajastimet laukeavat aikanaan ja

- virheellinen sanoma
- sekä kaikki sen jälkeen lähetetyt sanomat lähetetään uudelleen

■ tehoton, jos paljon virheitä ja iso ikkuna

Valikoiva toisto

■ vastaanottaja hyväksyy kaikki kelvolliset sanomat

- se kuittaa sanomat
- puskuroi ne ja toimittaa eteenpäin oikeassa järjestyksessä
 - » tarvitaan puskuritilaa

■ lähettäjä ei saa kuittausta virheellisestä sanomasta

- ajastin laukeaa ja sanoma lähetetään uudelleen
- lähettää uudelleen vain virheellisen sanoman
- ikkuna liikuu nytkin tasaisesti
 - » yksi puuttuva kuittaus voi pysäyttää lähetyksen

Kuittaukset

■ ACK

- kumulatiivinen ACK
 - tähän saakka kaikki ok!
 - Go-Back N
- yksittäinen ACK
 - vain tämä ok!
 - Valikoiva toisto

■ NAK-kuittaus

- sanoma virheellinen tai puuttuu

Negatiiviset kuittaukset

■ NAK-kuittauksilla voidaan nopeuttaa uudelleenlähettämistä

- vastaanottaja ilmoittaa heti virheellisestä tai puuttuvasta kehyksestä
- ei ole tarpeen odottaa ajastimen laukeamista

■ hyödyllinen, jos kuittausten saapumisaika vaihtelee paljon

- ajastinta vaikea asettaa oikein

■ NAK-kuittaukset voivat aiheuttaa turhia uudelleenlähetyksiä

- lähetys ja kuittaus menevät ristiin

■ NAK-kuittauksen katoaminen ei haittaa

■ implisiittinen uudelleenlähetys

- ei NAK-kuittauksia

■ explisiittinen uudelleenlähetys

- käytetään NAK-kuittauksia

Ikkunankoko

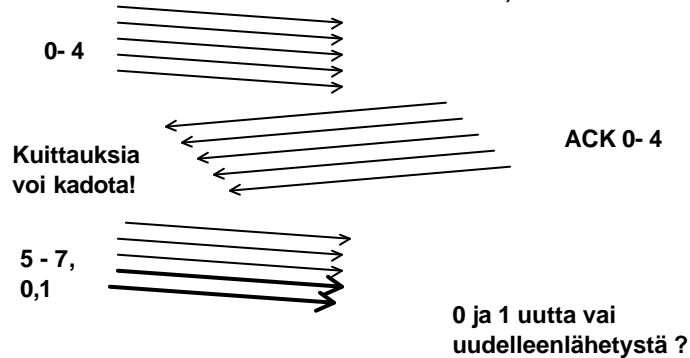
- Kun käytetty numeroavaruus on $0, 1, \dots, n$ ja eri numeroita siis käytettävissä $n+1$
 - yleensä jokin kakkosen potenssi
 - » koska numerokentän koko k bittiä => käytössä 2^{**k} numeroa
- ikkunan koko 'go back n':ssä voi olla korkeintaan n
 - eli ainakin yhtä pienempi kuin numeroavaruus
- ikkunan koko valikoivassa toistossa voi olla korkeintaan $(n+1)/2$
 - korkeintaan puolet numeroavaruudesta

30.1.2002

43

Miksi?

- Valikoiva toisto: ikkuna 5, numeroavaruus 8

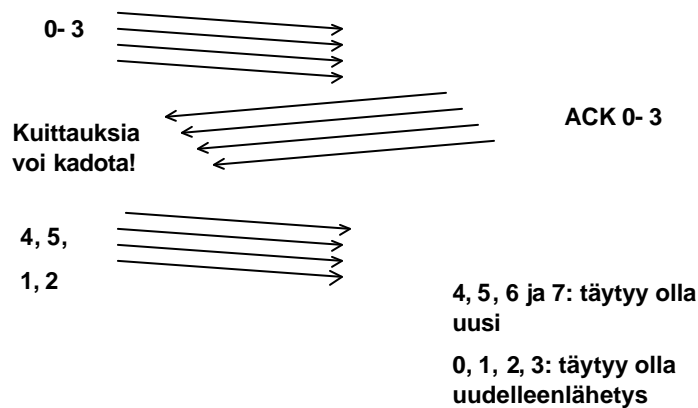


30.1.2002

44

Miksi?

- Valikoiva toisto: ikkuna 4, numeroavaruus 8



30.1.2002

45

Kaksisuuntainen liikenne

- datakehys ja kuittauskehys
- kehyksessä sekä data että kuittaus
 - ‘piggybacking’
 - tehostaa lähetystä
- ongelma: kauanko kuittaja odottaa dataa ennen pelkän kuittauksen lähettämistä?

30.1.2002

46

3.5. TCP-protokolla

- yhteyden muodostus ja purku
- luotettavan tavuvirran toteuttaminen
- vuonvalvonta
- siirron optimointi
- TCP-segmentti
- ruuhkan valvonta
- TCP-palvelun käyttö

6.2.2. Yhteyden muodostus ja purku TCP:ssä

- TCP käyttää yhteyden muodostamiseen ja purkuun ns. kolminkertaista kättelyä (three-way handshake)
 - välissä oleva verkko tekee yhteyden muodostamisen ja purun hankalaksi
 - viivästyneet sanomat => sanomille elinaika
 - sanomien numeroinnista sopiminen
 - Bysanttilainen ongelma (two-army problem)
 - “hyökkään, jos olen varma, että sinäkin hyökkää”
 - symmetrinen yhteyden purku = molemmat osapuolet tietävät, että toinenkin on varmasti purkanut yhteyden