

## 4. Verkkokerros

- **sovelluskerros**
  - ‘asiakas’
- **kuljetuskerros**
  - ‘end-to-end’
- **verkkokerros**
  - ‘deliver packets given to it by its customers’
- **siirtoyhteyskerros**
- **peruskerros**

# Verkkokerroksen palvelut

- **tavoitteet**

- palvelut riippumattomia aliverkkojen tekniikasta
- kuljetuskerros eristettävä aliverkkojen ominaisuuksista
  - lukumäärä
  - tyypit
  - topologia
- kuljetuskerroksen käyttämät **verkko-osoitteet globaaleja**

# connection-oriented ~ connectionless

- **yhteydetön (Internet, 30 vuoden kokemus)**
  - aliverkot ovat luonnostaan epäluotettavia
    - tehtävä: bittien kuljetus
    - operaatiot: send packet, receive packet
    - virheen tarkistus, vuonvalvonta isäntäkoneille
- **yhteydellinen (puhelin 100 vuoden kokemus)**
  - muodostetaan yhteys, neuvotellaan parametrit (palvelunlaatu (QOS), kustannus)
  - kaksisuuntainen kuljetus, paketit järjestyksessä
  - vuonvalvonta, virhevalvonta

# Virtuaalipiiri (virtual circuit)

- **Pakettikytkentäinen verkko voidaan toteuttaa kahdella tavalla**
  - datasähkeverkkona
    - jokainen paketti käsitellään ja reititetään erikseen
    - pakettien järjestys voi muuttua
  - virtuaalipiiriverkkona
    - ~ piirikytkentäinen verkko
      - ensin yhteyden (virtuaalipiirin) muodostus
      - sitten pakettien lähettäminen yhteyttä pitkin
    - ATM, X.25

# Piirikytkentäinen verkko

- ensin yhteyden muodostus
- sitten datan siirto yhteyttä pitkin
- yhteyden purku



## 4.1. Verkkokerroksen tärkein tehtävä: reititys

- **(hajautettu) päätöksenteko reitistä**
  - yhteydellinen: alussa
  - yhteydetön: jatkuvasti
- **jatkuvaa muutosta verkossa**
  - rikkoutuvat komponentit, muuttuva topologia
- **ristiriitaisia vaatimuksia reititykselle**
  - optimaalisuus /reiluus (fairness)
- **reitityksen suorituskyky**
  - mean packet delay, network throughput

# Reititysalgoritmi

- **Päättää, mikä reitti valitaan**
  - Pyyrkii löytämään mahdollisimman hyvän reitin lähdekoneelta kohdekoneelle ('pienin kustannus')
- **dynaaminen verkkoympäristö => dynaaminen reititys ↔ staattinen reititys**
  - jatkuvaan verkon tarkkailuun perustuva
    - Internetin reititys
  - muuttumaton ympäristö => käytetään kerran laskettuja reittejä tai sovittua lähetystapaa
    - tulvitus (flooding)
    - Dijkstran algoritmilla lasketut lyhyimmät reitit

# Tulvitus

- jokainen saapunut paketti lähetetään kaikille muille ulosmenoille
  - => **verkko täyttyy pian paketeista**
- eri tapoja tulvituksen lopettamiseen
  - **käsitellään harjoituksissa**
- **käyttö**
  - **tietyissä erityistilanteissa tilanteissa hyödyllinen**
    - **käsitellään harjoituksissa**



# Dijkstran algoritmi

- **'lyhyin' reitti yhdestä solmusta muihin**
  - $A \rightarrow \{\text{muut solmut}\}$
- **kaariin liittyy kustannus**
  - kapasiteetti (bps)
  - viive: hyppyjä, aikaa
  - raha
  - virhetodennäköisyys

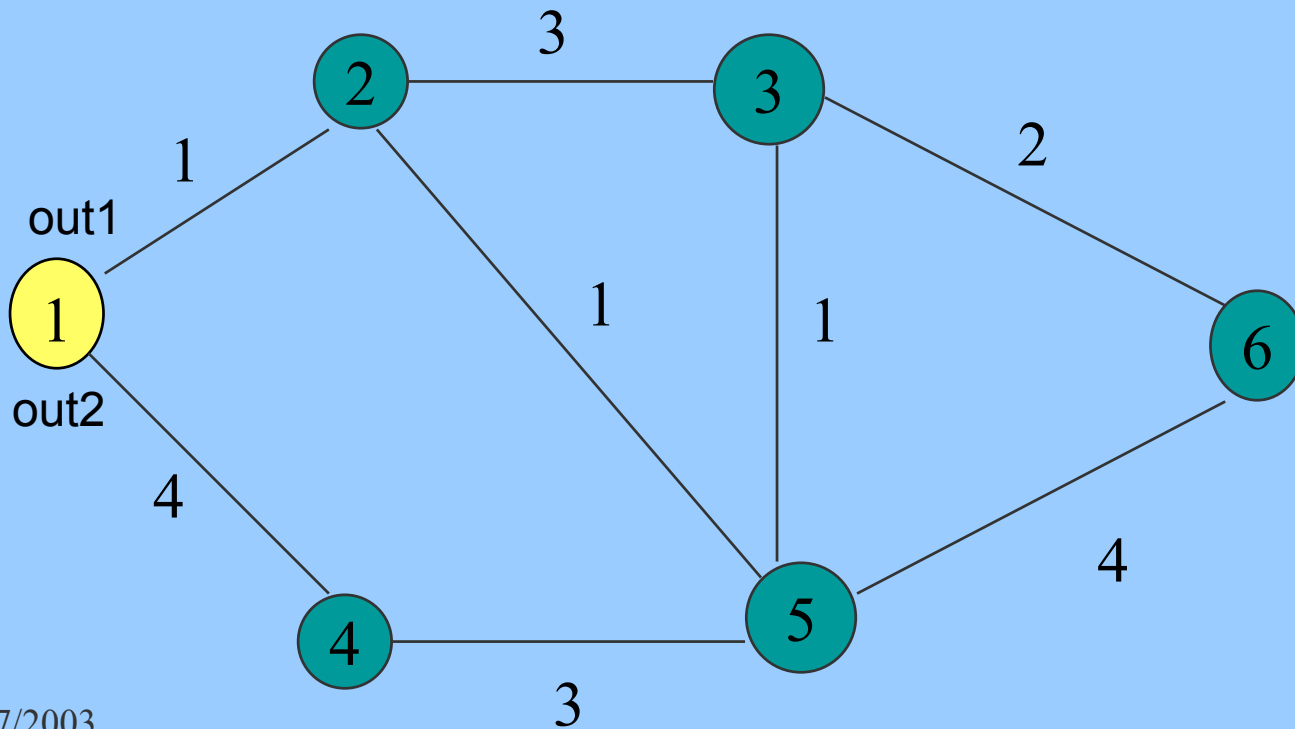
# • Algoritmi

- merkitään  $D(v)$  on tähän asti tutkituista reiteistä lähtösolmusta  $A$  solmuun  $v$  halvin kustannus eli lyhyin pituus
- verkko  $G = (V, E)$ ,  $V$  on solmujen joukko,  $E$  kaarten joukko
- olkoon  $c(i,j)$  on kaaren  $(i,j)$  kustannus ( $\geq 0$ ). Jos kaarta ei ole,  $d(i,j)$  on ääretön
  - algoritmossa oletetaan, että kaikki kustannukset ovat ei-negatiivisiä

1.  $N := \{1\}$ ;  $D(1) := 0$ ;  $D(j) := d(j,1)$  ( $j \neq 1$ );
2. while  $N \neq V$  do
3. etsi solmu  $w$ , joka ei vielä ole  
joukossa  $N$  ja jonka  $D(w)$  on pienin  
 $N$ :ään kuulumattomista solmuista
4.  $N := N \cup \{w\}$
5. kaikille muille  $N$ :ään kuulumattomille  
solmuille  $v$   $D(v) := \min\{D(v), D(w) + c(w,v)\}$
6. end while
7. end

# Esimerkki

- Tarkastellaan esimerkkinä verkkoa



1.  $N = \{1\}$ ;  $D(1) := 0$ ;  $D(2) := 1$ ;

$D(3) := \text{ääretön}$ ,  $D(4) := 4$ ;

$D(5) := \text{ääretön}$ ,  $D(6) := \text{ääretön}$

3. pienin  $D(v)$  on solmulla 2 (=1)

4.  $N = \{1, 2\}$

5.  $D(3) := 1+3=4$ ,  $D(4) = 4$ ,  $D(5) := 1+1=2$ ,

$D(6) = \text{ääretön}$

3. pienin  $D(v)$  on nyt solmulla 5 (=2)

4.  $N = \{1, 2, 5\}$

5.  $D(3) := 1 + 2 = 3$ ,  $D(4) := 4$ ,  $D(6) := 4 + 2 = 6$

3. pienin  $D(v)$  solmulla 3 (=3)

4.  $N = \{1, 2, 3, 5\}$

5.  $D(4) := 4$ ,  $D(6) := 2 + 3 = 5$ ;

3. Pienin  $D(v)$  solmulla 4 (=4)

4.  $N = \{1, 2, 3, 4, 5\}$

5.  $D(6) := 5$

4.  $N = \{1, 2, 3, 4, 5, 6\}$

# Löydetyt reitit ja kustannukset

- 1-> 2 :1
- 1-> 2->5->3: 3
- 1-> 4: 4
- 1->2->5: 2
- 1->2->5->3->6: 5

Solmu	linkki	kustann.
2	1	1
3	1	3
4	2	4
5	1	2
6	1	5

Solmulle 1

# Reititystaulu

- Kukin **reititin** pitää kirjaa reittitiedoista
  - minne paketti seuraavaksi lähetetään

Kohde	minne lähetetään
Abc	reititin D, ulosmeno 2
...	.....
Xyz	reititin T, ulosmeno 3

- **reitittimien tietojen hankinta ja ylläpito?**



# Reititystietojen keruu

- kukin reititin kerää ‘kustannustietoja’ omasta ympäristöstään
  - esim. viiveet naapurireitittimiin
- ja vaihtaa tietoja muiden reitittimien kanssa
  - tai lähettää tiedot reitittimelle, joka keskitetysti laskee parhaat reitit
- kukin laskee esim. Dijkstran algoritmilla parhaat reitit koko verkosta
  - tai saa tarvitsemansa reititystiedot ne laskeneelta

# Etäisyysvektoreireitys (distance vector)

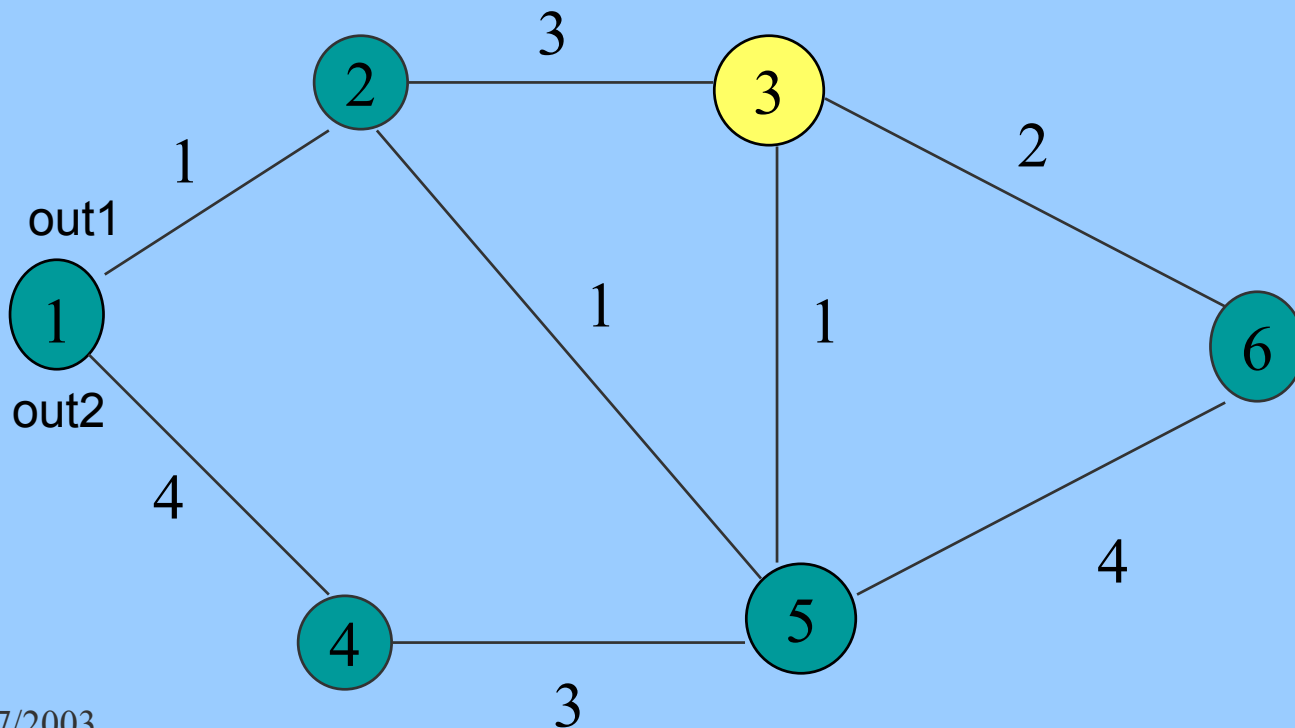
- **Arpanetin alkuperäinen reititys**
  - vieläkin (RIP) jonkin verran käytössä Internetissä
- **kullakin reitittimellä reititystaulu**
  - kullekin verkon reitittimelle
    - ulosmenolinja
    - aika/etäisyys kohteeseen
      - hyppyjen lkm
      - arvioitu viive
      - jononpituus
      - jokin mitattavissa oleva

## reititystaulun ylläpito

- **tietojen vaihto naapurireitittimien kanssa**
  - tietyin aikaväleihin
  - tilan vaihtuessa
- **lasketaan uudet reittitaulut ('etäisyystaulut')**
  - 'kustannus' naapuriin + naapurin ilmoittama 'kustannus' kohteeseen
  - kullekin solmulle valitaan pienimmän 'kustannuksen' reitti

# Esimerkki

- Tarkastellaan esimerkkinä verkkoa



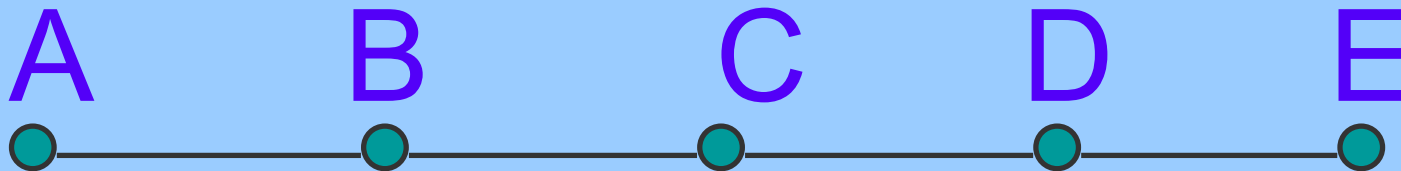
# Solmun 3 reititystaulu

	<b>3</b>	<b>2</b>	<b>5</b>	<b>6</b>	
<b>1</b>	-	1(2)			=> 4 (2)
<b>2</b>	<b>3</b>		1(5)		=> 2 (5)
<b>4</b>	-		3(5)		=> 4 (5)
<b>5</b>	<b>1</b>	1(2)		4(6)	=> 1(5)
<b>6</b>	<b>2</b>		4(5)		=> 2 (6)

# Ongelma: tietojen muuttumisnopeus

- **tietojen muuttamiseen kuluu aikaa**
- **reagoi melko nopeasti hyviin uutisiin**
  - uusi nopea reitti löytynyt/linkki jälleen pystyssä
  - tieto etenee joka vaihdossa yhden hypyn
- **reagoi hitaasti huonoihin uutisiin**
  - linkki nurin => etäisyys ääretön
  - joka vaihdossa ‘paras arvio’ huononee yhdellä
  - **count - to - infinity** -ongelma

# Hyvät uutiset etenevät nopeasti:

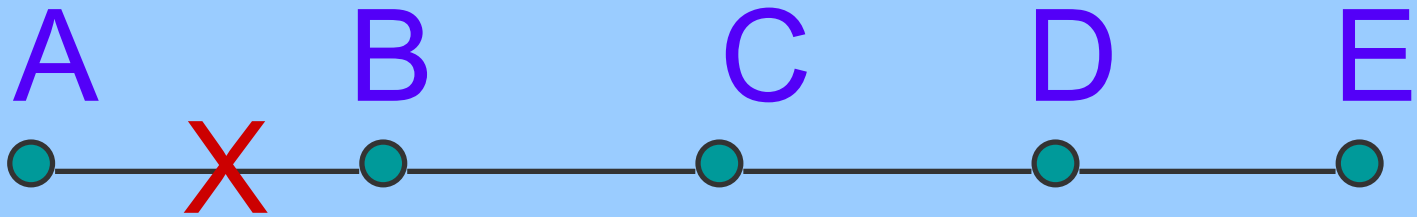


Aluksi yhteys A:han on poikki ja sitten linkki AB toimii taas:

**Etäisyys  
A:han**

	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>
ääretön	ääretön	ääretön	ääretön	ääretön
1	ääretön	ääretön	ääretön	ääretön
1	1	2	ääretön	ääretön
1	1	2	3	ääretön
1	1	2	3	4

# Huonot uutiset etenevät hitaasti:



Toimiva linkki katkeaa välillä AB:

**Etäisyys  
A:han**

	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>
1		2	3	4
3		2	3	4
3		4	3	4
5		4	5	4
5		6	5	6
7		6	7	6
7		8	7	8

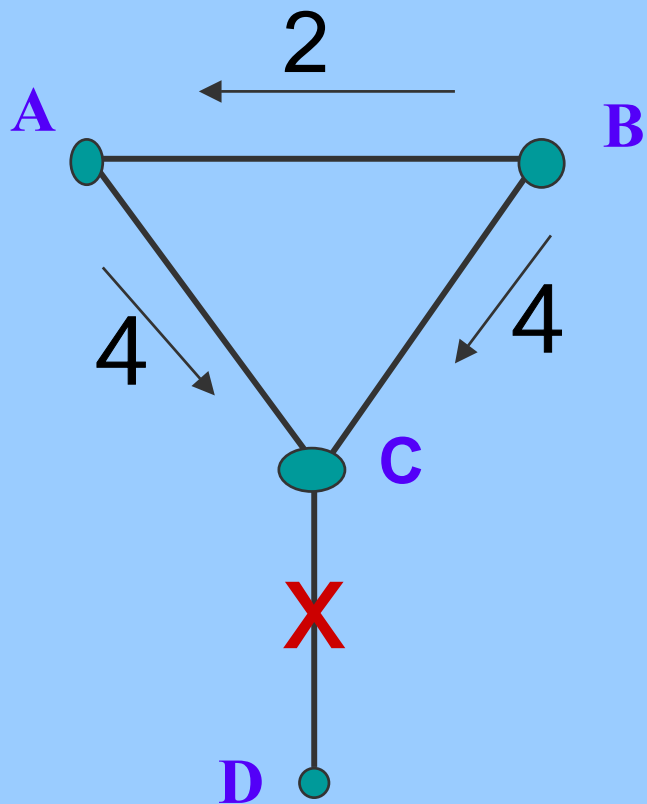


## ‘Split horizon with poisoned reverse’

- ratkaisu ‘count -to-infinity’-ongelmaan
  - reititystietoja vaihdettaessa
    - ilmoitetaan etäisyys reitittimeen X äärettömäksi sille naapurille, jonka kautta tämä reitti kulkee
    - muille kerrotaan oikea etäisyys
  - tieto etenee yhden hypyn joka vaihdolla!

# ratkaisu ei toimi aina!

(etäisyys



**Linkki CD katkeaa, A ja B ilmoittavat C:lle, ettei D:hen pääse (etäisyys ääretön)**

**C päättelee (oikein), että D:tä ei voi saavuttaa**

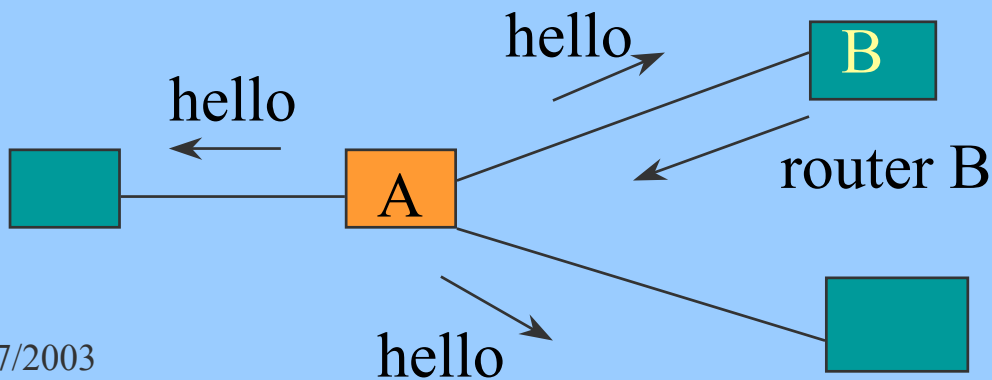
**Mutta A kuulee B:ltä, että sillä on etäisyys 2 D:hen => A:n oma etäisyys D:hen := 3 ja tämä reitti ei kulje C:n kautta! => kerrotaan C:lle.<sup>26</sup>**

# Linkkitilareititys (Link State Routing)

- **reitittimen tehtävät**
  - selvitettävä naapurit ja niiden osoitteet
  - mitattava etäisyys / kustannus naapureihin
  - koottava tietopaketti ko. tiedoista
  - lähetettävä tietopaketti kaikille reitittimille
  - laskettava lyhin reitti kaikkiin muihin reitittimiin esim. Dijkstran algoritmilla

# Naapurien löytäminen

- reititin lähettää jokaiseen kaksipisteyhteyteen **HELLO**-paketin
- linjan toisessa päässä oleva reititin vastaa ja lähettää nimensä
  - router ID
  - nimien oltava yksikäsitteisiä koko verkossa

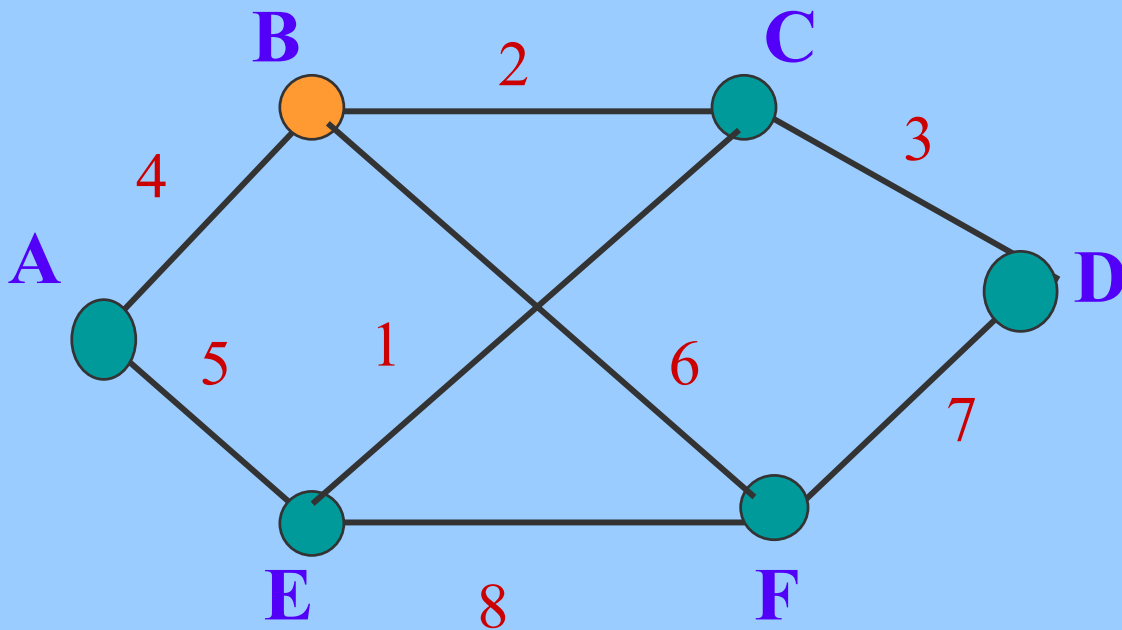


# Etäisyyden mittaaminen

- **kaikille naapureille ECHO-paketti**
  - vastaanottajan palautettava paketti välittömästi
- **=> kiertoviive (round-trip-time)**
  - dynaaminen etäisyysmitta
- **pitäisikö ottaa kuormitus huomioon?**
  - kello käynnistetään , kun paketti viedään jonoon
  - kello käynnistetään, kun paketti lähtee
  - kuormitus mukana kuvaa todellista tilannetta
  - jos kuormitus mukana => reititys muuttaa kuormitusta => reititys suosii huonoa reittiä

# Tietopakettien kokoaminen

- **muodostus**
  - tietyin aikaväleihin
  - kun muutoksia havaittu
- **sisältö**
  - reitittimen tunnus
  - paketin järjestysnumero
  - paketin ikä
  - ‘etäisyydet’ kuhunkin reitittimen naapuriin
    - Erilaisia etäisyysmittoja => eri reittejä eri liikenteelle



<b>B</b>	
<b>seq</b>	
<b>age</b>	
<b>A</b>	<b>4</b>
<b>C</b>	<b>2</b>
<b>F</b>	<b>6</b>

B:n generoima  
tietopaketti

# Tietopakettien jakelu

- **käytetään tulvitusta** (n. 10 minuutin välein)
  - **pidetään kirjaa jo nähdyistä paketeista**
    - reititin A, paketti 145
  - => paketti lähetetään korkeintaan kerran
  - **paketissa elinaikalaskuri** (age, time-to-live)
    - väärät ja vanhentuneet tiedot katoavat aikanaan, vaikka reititin itse olisikin vikaantunut
- **tietopaketit kuitataan**
  - linjavirheiden takia
- **autentikointi paketteja vaihdettaessa**



# Miksi elinaikalaskuri on tarpeen?

- **virheellinen järjestysnumero**
  - kaatunut reititin aloittaa väärästä numerosta
    - edennyt jo pakettiin 204 ja aloittaa uudestaan paketista 0 => kaikki seuraavat paketit hylätään duplikaatteina pakettiin 205 saakka
  - virhe tietopaketin seq-kentässä
    - 4 muuttuu virheellisesti 65540:ksi => seuraavat paketit hylätään pakettiin 65541 saakka

## elinaikalaskuri (TTL-laskuri)

- **laskuri vähenee ajan kuluessa**
  - vähenee yhdellä sekunnin välein
- **paketti tuhotaan, kun laskuri = 0**
  - vanhentunut (virheellinen) tieto poistetaan
  - pitkäikäinen elinaika  $\gg$  päivitysten väli
    - tuhotaan vain jos reititin kaatunut
    - usea (6) paketti on jäänyt saapumatta reitittimeltä
- **käytössä myös tulvituksessa**
  - kukin reititin vähentää yhdellä

# Lisäparannuksia

- **paketteja ei lähetetä välittömästi eteenpäin**
  - ne jätetään odottamaan
  - jos samalta reitittimeltä tulee muita paketteja, niistä valitaan vain yksi, tuorein edelleenlähetettäväksi

# Reittitaulun laskeminen

- kukin reititin laskee omat reittitaulunsa
- kaikki tarvittava tieto on saatu tietopakettien avulla
  - kukin linkki molempiin suuntiin
- laskeminen Dijkstran algoritmilla
  - lyhyin reitti kuhunkin muuhun reitittimeen
  - isoissa verkoissa voi olla muisti- ja laskenta-aikaongelmia

# ongelmia

- **väärin toimiva reititin**
  - kertoo väärää tietoa
  - ei välitä tietopaketteja
  - väärentää tietopaketteja
  - laskee reitit väärin
- **isossa verkossa aina joku toimii väärin**
  - tavoitteena rajata ongelmat pienelle alueelle

# Käyttö

- **paljon käytetty nykyisissä verkoissa**
  - Internetin **OSPF**-protokolla
  - ISO:n IS-IS -protokolla

# Hierarkkinen reititys

- **reitityksen skaalautuvuus**
  - isossa verkossa runsaasti reitittämiä (Internet: miljoonia)
    - reititystaulut suuria
    - reittien laskeminen raskasta
    - tietopaketit kuluttavat linjakapasiteettia
- **hallinta-autonomia => autonominen järjestelmä AS**
  - organisaatio päättää omista asioistaan
    - myös reitityksestä
      - oma sisäinen reititystapa

# Reitityshierarkia

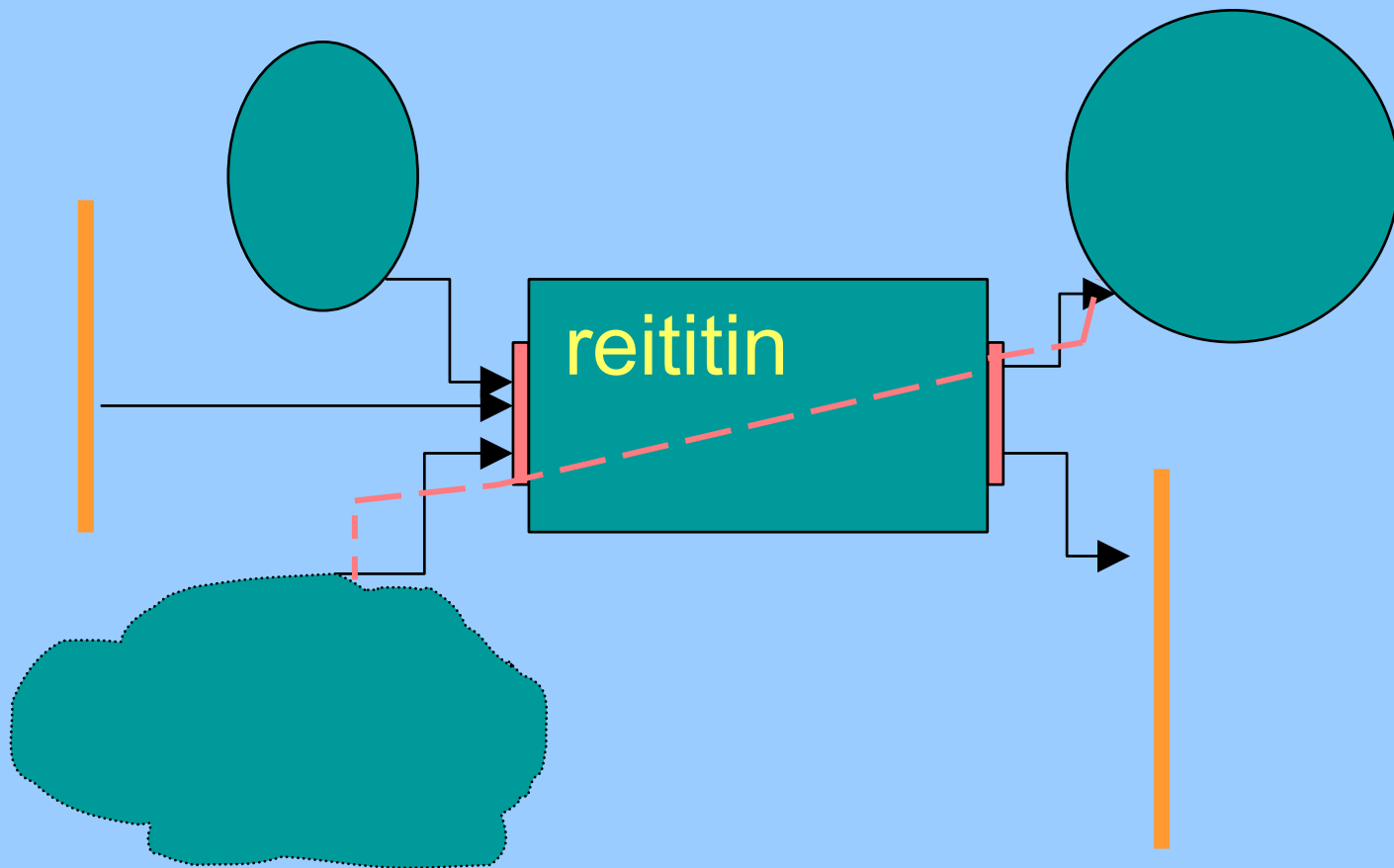
- **Ylimmällä tasolla AS**
  - sama reititys AS:n sisällä
    - tehokkuus tärkeää
  - reititys AS:ien välillä
    - ‘poliittinen asia’
- **AS:n sisällä alueita**
  - jaetaan reitittimet ryhmiin (alueet, regions)
  - kukin reititin tuntee kaikki alueensa sisällä
  - tietää mikä reititin hoitaa liikenteen muihin alueisiin



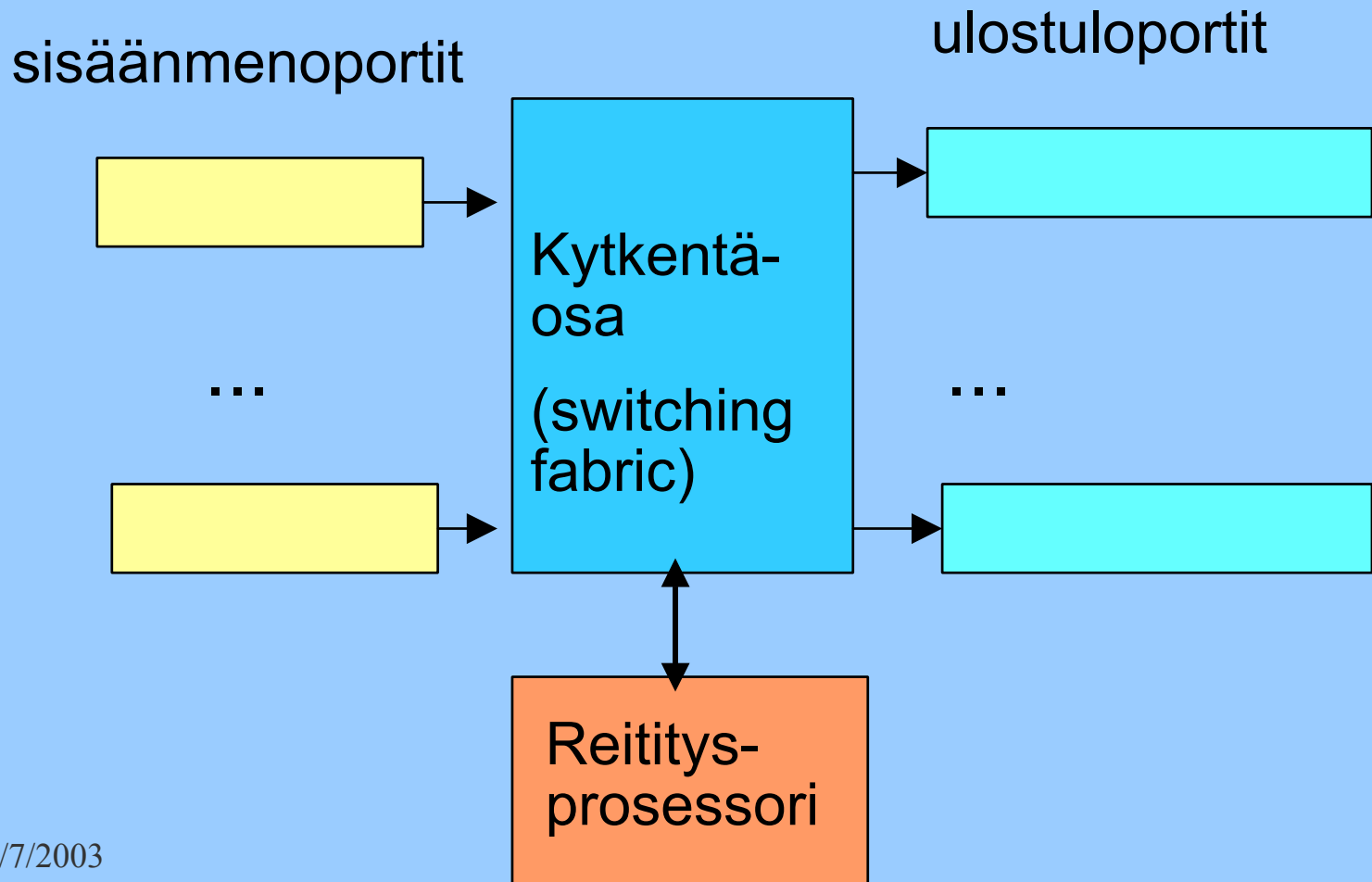
# Hierarkkisen reitityksen ongelmat

- **reitin pituus kasvaa**
  - aina ei voida käyttää optimaalista reittiä
  - yleensä siedettävä
- **hierarkiatasojen määrä**
  - suorituskyky
  - hallinto

## 4.2. Reititin (Router)



# Reitittimen rakenne



# • Portit

## – peruskerroksen toiminnot (PK)

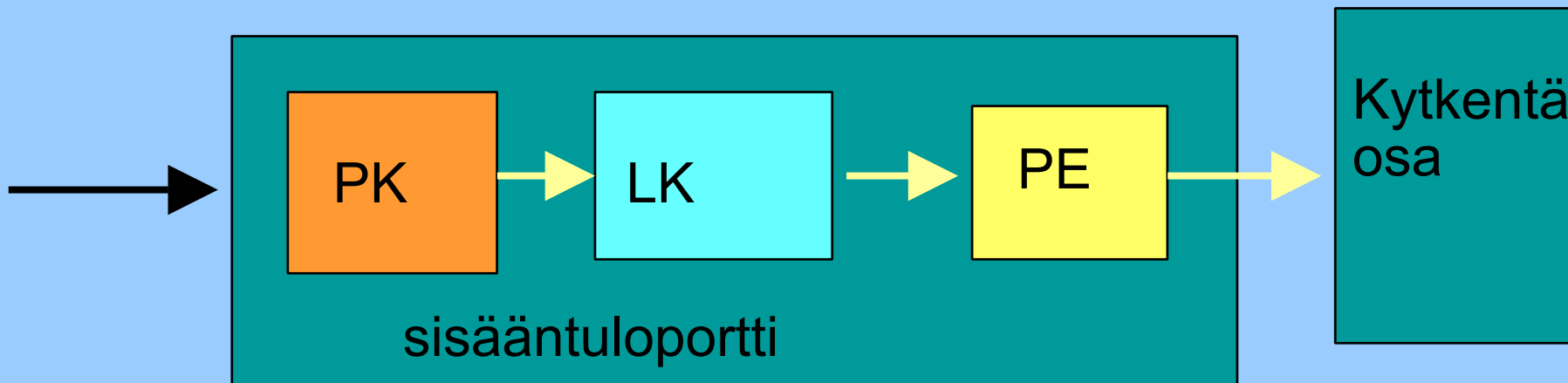
- fyysisen siirtoyhteyden pää

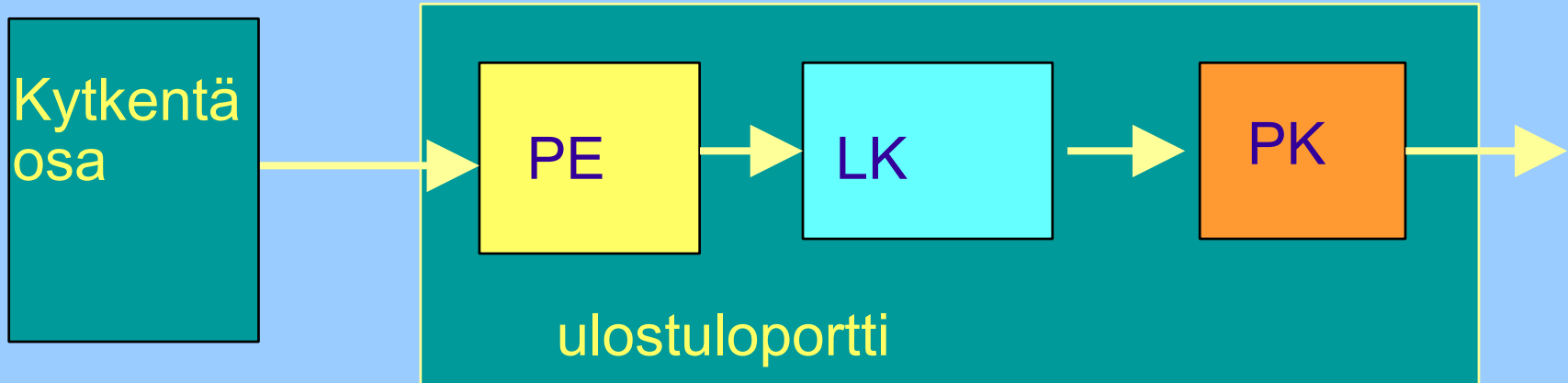
## – linkkikerroksen toiminnot (LK)

- virhetarkistukset, vuonvalvonta,
- MAC-kerroksen toiminnot

## – pakettien edelleenohjaaminen (PE)

- datapaketit kytkentäverkoston kautta oikeaan ulostuloporttiin
- valvontapakettit (RIP, OSPF, BGP) reititysprosessorille





Vastaavasti kukin ulostuloportti tallettaa sen kautta eteenpäin lähtevät paketit ja suorittaa niille linkkikerroksen ja peruskerroksen vaatimat toimenpiteet.

Käytännössä useita portteja on yhdistetty yhdeksi linjakortiksi (line card) reitittimen sisällä.

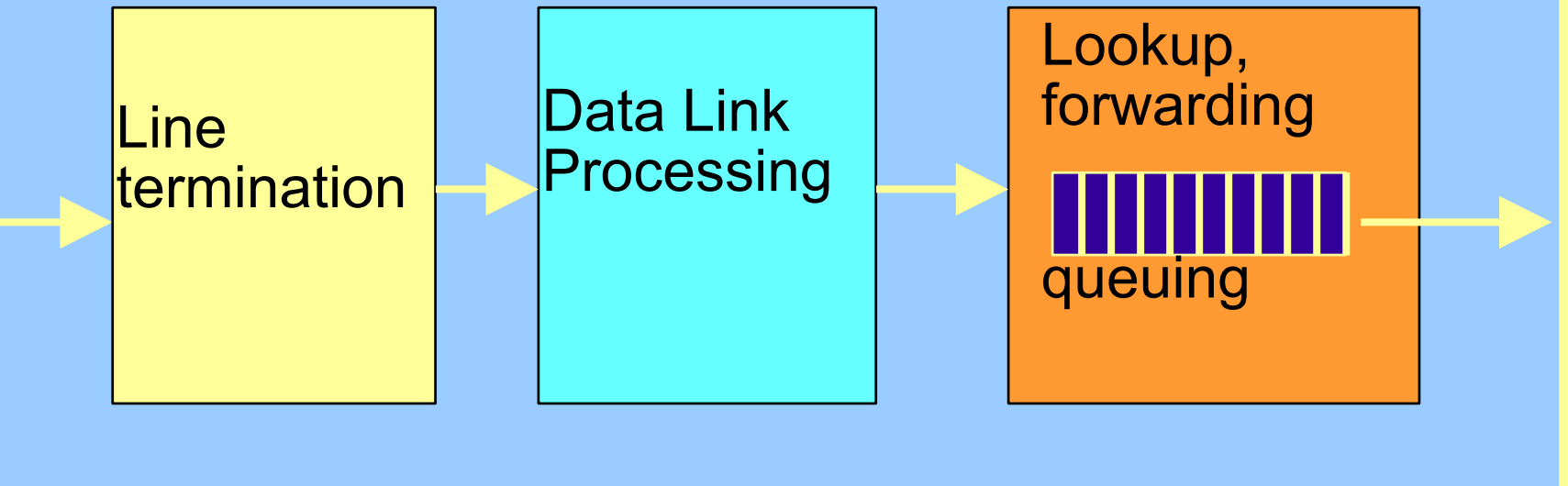
## " **Reititysprosessori**

- " suorittaa reititysprotokollaa
  - " RIP, OSPF, BGP, ..
- " päivittää reititystauluja
- " hallinta- ja ylläpitotoimintoja

## " **Kytkenäosa**

- " yhdistää paketin sisääntuloportit ulostuloportteihin
- " paketti siirtyy oikeaan verkkoon

2/7/2003 täysin reitittimen sisällä



## Sisääntuloportin toiminta

Etsitään reititystaulusta kohdeosoitetta vastaava ulosmenoportti.

Yleensä kopio reitistystaulusta talletettu porttiin ja reititysprosessori päivittää sitä. Näin kukin portti pystyy itse etsimään oikean ulosmenoportin.

Muuten paketti ohjataan reititysprosessorille, joka etsii reititystaulusta oikean portin (portti on pelkkä verkkokortti).

- **Runkolinjareitittimiltä vaaditaan hyvin suuria nopeuksia**

- miljoonia hakuja sekunnissa
- pitäisi pystyä toimimaan linjan nopeudella
  - OC48-linkki => 2.5 Gbps
  - jos paketin koko 256 tavua => noin miljoona hakua sekunnissa

- **erilaisia tekniikoita**

- talletetaan reittitaulun alkiot puurakenteina



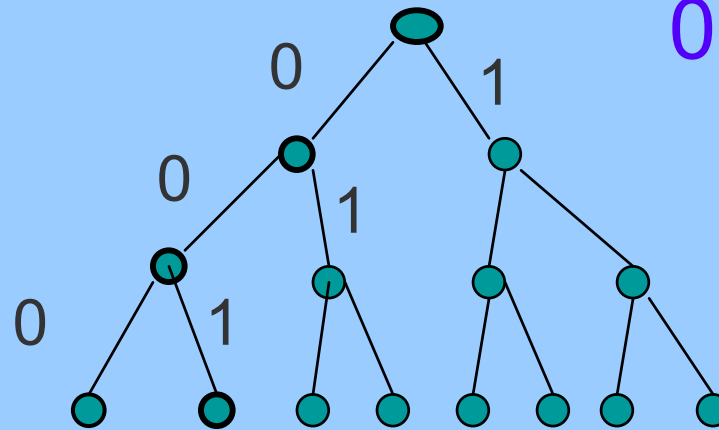
# Osoitteen

1. bitti

2. bitti

3. bitti

jne



001.....

Kun  $n = 32$  ei ole tarpeeksi nopea nykyisiin runkoreitittimiin!

- content addressable memory (CAM)
- välimuistin käyttö

# Kytkeäosa

- **Kytkeä muistin kautta**
  - portit tavallisia käyttöjärjestelmän I/O-laitteita
  - keskeytys ilmoittaa paketin saapumisesta
  - CPU kopioi paketin sisääntuloportista muistiin
  - CPU tutkii osoitteen ja reitistystaulusta etsii vastaavan ulosmenoportin
  - CPU kopioi paketin muistista tähän ulosmenoporttiin
  - muistin saant nopeus rajoittaa toimintaa
- **nykyiset reitittimet**
  - käyttävät linjakortin omia prosessoreita

- **KytKentä väylän kautta**

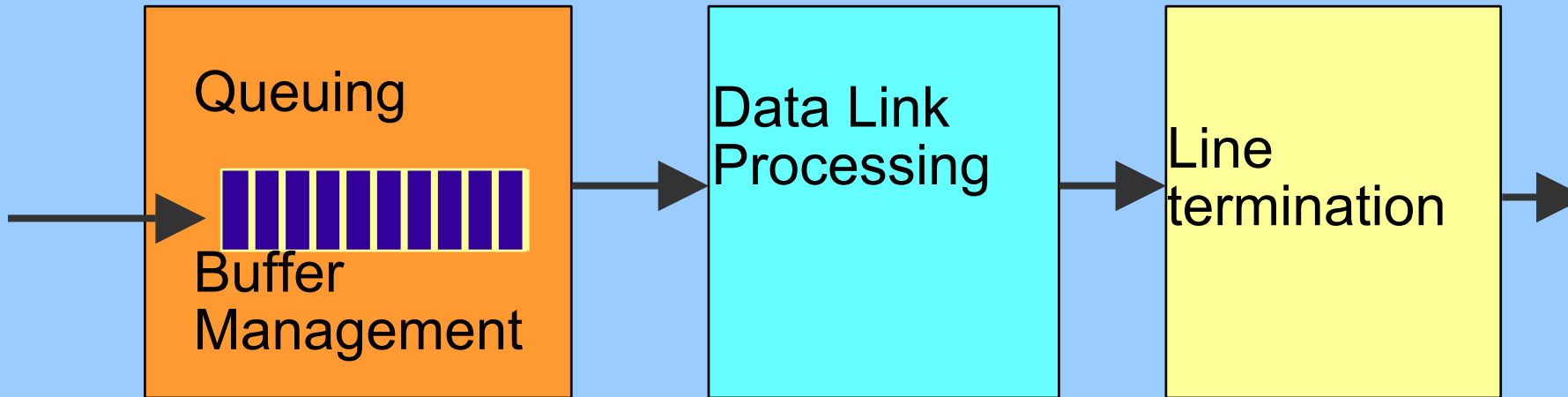
- sisääntuloportit siirtävät paketin väylän kautta suoraan oikeaan ulosmenoporttiin
- vain yksi paketti kerrallaan voi kulkea väylässä
- jos väylä on varattu, paketti joutuu odottamaan
- väylän nopeus rajoittaa kytkentänopeutta
  - Gbps nopeudet riittävät LANeille ja yritysverkoilla

- **KytKentä kytkentäverkon kautta**

- ristikkäinkytkin (crossbar switch)
- $2N$  väylää, jotka yhdistävät  $N$  sisääntuloporttia  $N$ :ään ulosmenoporttiin
- voivat tukkeutua => odotusta sisäänmenoportissa
  - Cisco 12000: 64 Gbps

# Ulosmenoportit

Ulosmenoportti lähettää paketin taas seuraavaan verkkoon

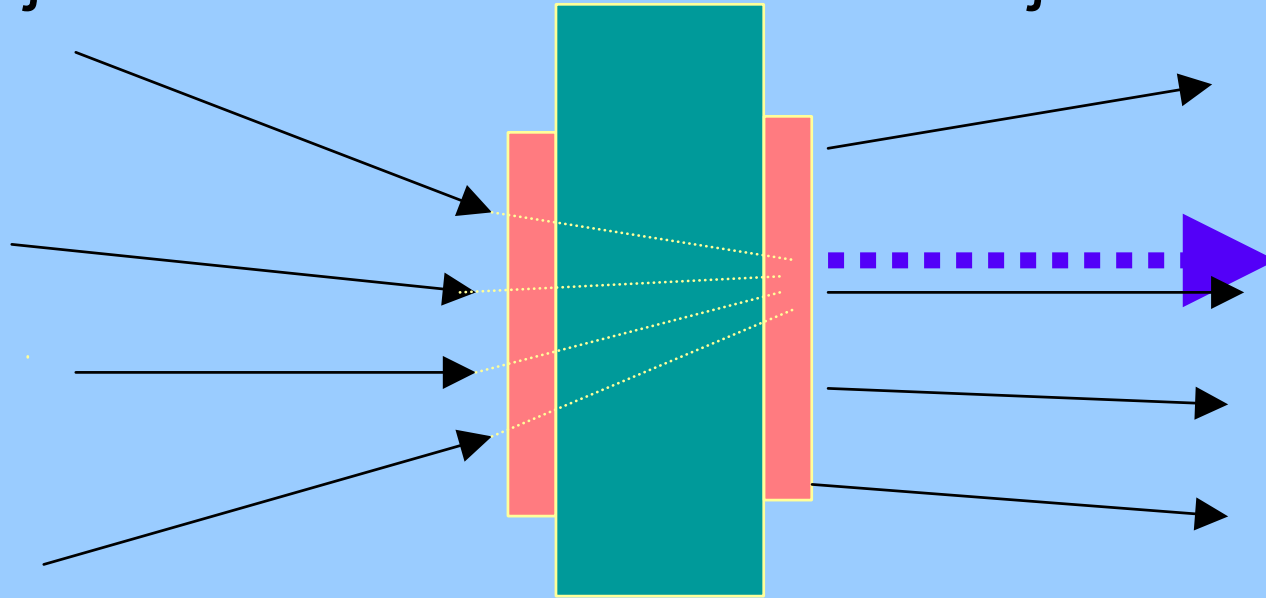


# Jonotus reitittimessä

- **Sekä sisäänmeno- että ulostuloporttiin voi syntyä jonoa**
  - näissä jonoissa reititin voi kadottaa paketteja, kun puskuritila ei enää riitä
  - se kummassa jonossa paketit katoavat, riippuu kytkimen ja linjan nopeuden suhteista
  - jonoa voi syntyä myös, koska useasta lähteestä pyritään samaan kohteeseen

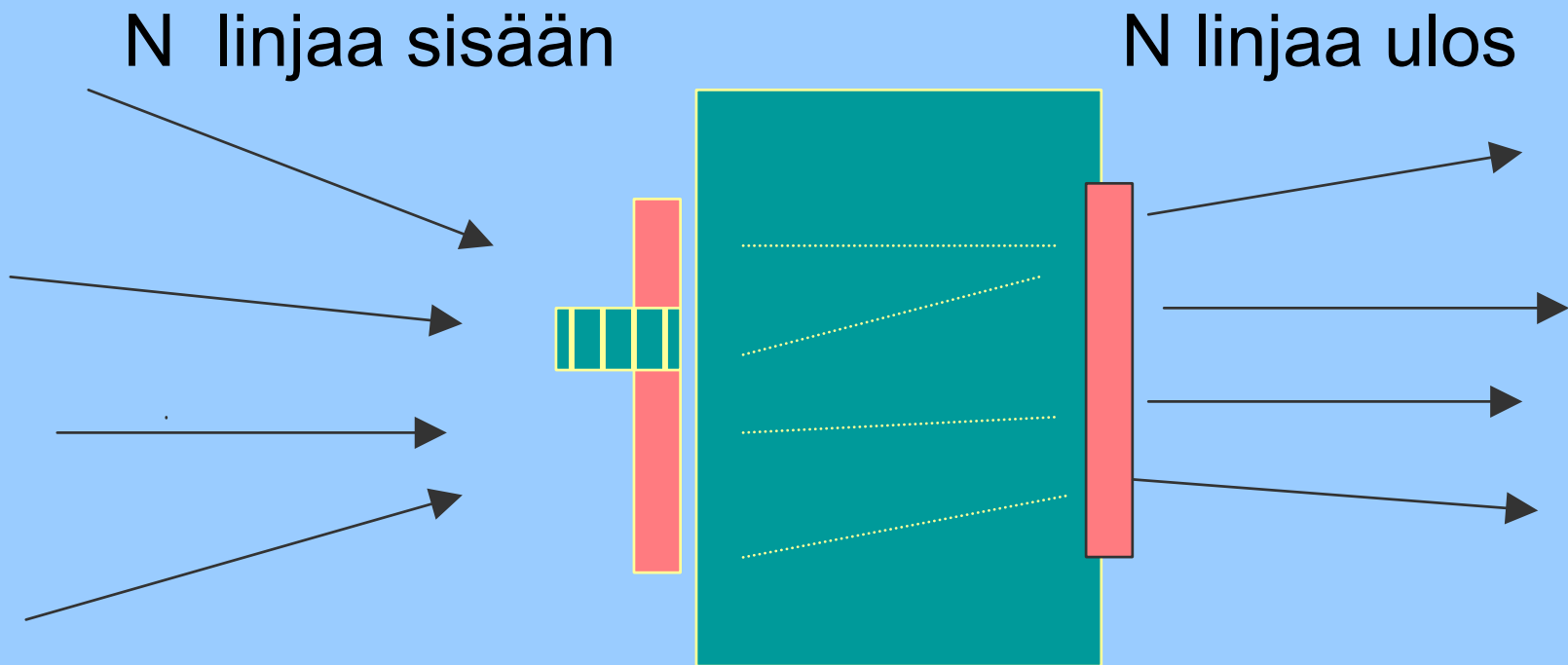
N linjaa sisään

N linjaa ulos



Kytkin toimii riittävällä nopeudella, joten sisääntulossa ei tarvitse jonottaa.

Yhdelle linjalle liian paljon liikennettä => ulosmenoportin puskuritila täyttyy ja paketteja katoaa!



Jos kytkin ei toimi tarpeeksi nopeasti,  
sisääntuloportteihin syntyy jonoja.

Esim. Ristikkäinkytkimessä paketti joutuu odottamaan, jos samaan kohteeseen on menossa useita paketteja. Jonottava paketti voi tukkia tien myös muilta saman portin paketeilta, jotka muuten voisivat edetä kytkimessä.

(head-of-the-line-blocking)

## 4.3. Internetworking

- **verkot erilaisia: nyt (ja aina?)**
  - palvelu: yhteydellinen / yhteydetön
  - osoittaminen: yksitasoinen /hierarkkinen
  - monilähetys/yleislähetys
  - paketin koko
  - toiminnot :
    - palvelulaatu (qos) , virheiden käsittely, vuonvalvonta, ruuhkanvalvonta, turvaus ja laskutus
  - protokolla



- **ongelmana on erilaisten toiminnallisuuksien yhteensopivuus**
  - luotettavuus
  - ruuhkan valvonta
  - kuittaukset
  - toimitusaikatakuut

# Yhteydettömien verkkojen yhdistäminen

- verkkokerroksen protokollien oltava (lähes) samoja
- osoittaminen
  - IP: 32-bittinen osoite
  - OSI: puhelinnumeron kaltainen osoite
    - osoitteiden yhteensovittaminen?
    - globaaliosoitteavaruus? standardi?

# Pakettien paloittelu (fragmentation)

- **kaikissa verkoissa paketilla jokin maksimikoko**
  - laitteisto (TDM-viipaleen pituus)
  - käyttöjärjestelmä (käytetty puskurinkoko)
  - protokolla (pituuskentän bittien lukumäärä)
  - standardinmukaisuus
  - virheistä johtuvan uudelleenlähetyksen vähentäminen
  - tasapuolisuuden tavoite
- **48 tavua (atm) => 65515 tavua (IP)**

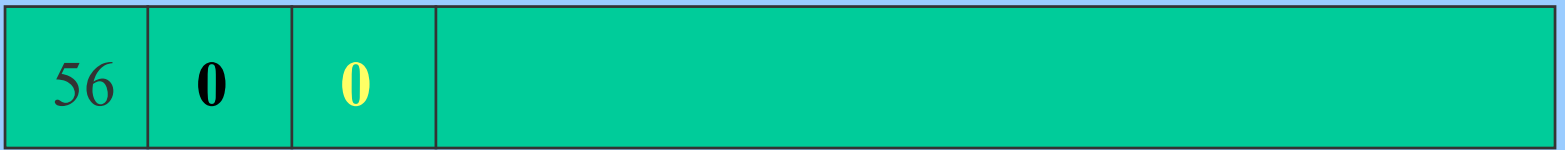
# Liian iso paketti verkkoon

- liian iso paketti paloitellaan yhdyskäytävässä
- missä paketti kootaan?
  - samassa verkossa, missä paloiteltiin
    - kaikki paketit ohjattava samaan yhdyskäytävään
    - jatkuvaa pilkkomista ja kokoamista!
  - vasta määränpäässä
    - pieni pakettikoko => lisää yleisrasitetta
    - kaikkien solmujen kyettävä kokoamaan paketteja

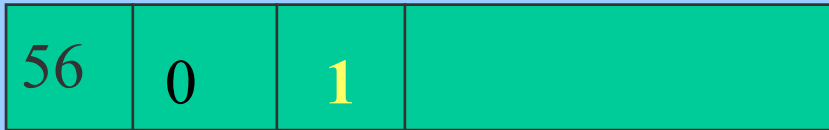
# Pakettien kokoaminen

- **edellyttää palojen ‘numerointia’**
  - on tiedettävä, mikä paketin mikä osa on kyseessä
- **kaikissa paloissa alkuperäisen paketin tunniste + sijainti paketissa**
  - sijainti: pakettiin kuuluvan ensimmäisen tavun sijainti alkuperäisessä paketissa
- **lisäksi tieto, onko pala paketin viimeinen**

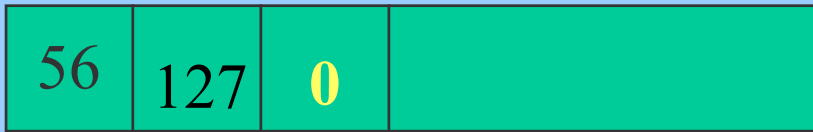
2/7/2003 • tai tiedettävä paketin pituus



alkuperäinen paketti



paketin alkuosa



paketin loppuosa

paketin  
sijainti-  
numero



sijainti-  
kohta  
eli

osan  
numero

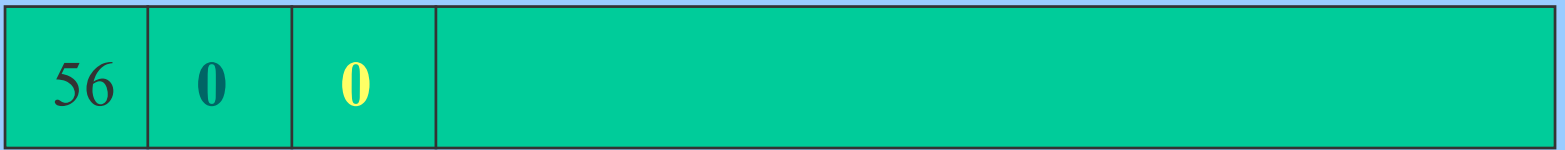


viimeinen  
paketin  
osa?

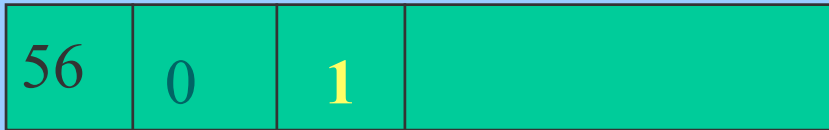


paketin data

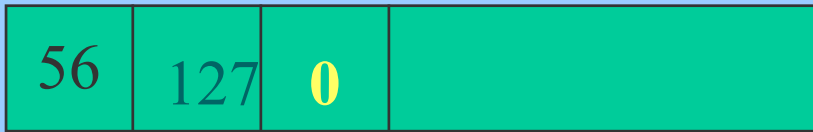




alkuperäinen paketti



paketin alkuosa



paketin loppuosa

paketin  
sijainti-



kohta  
eli  
osan  
numero

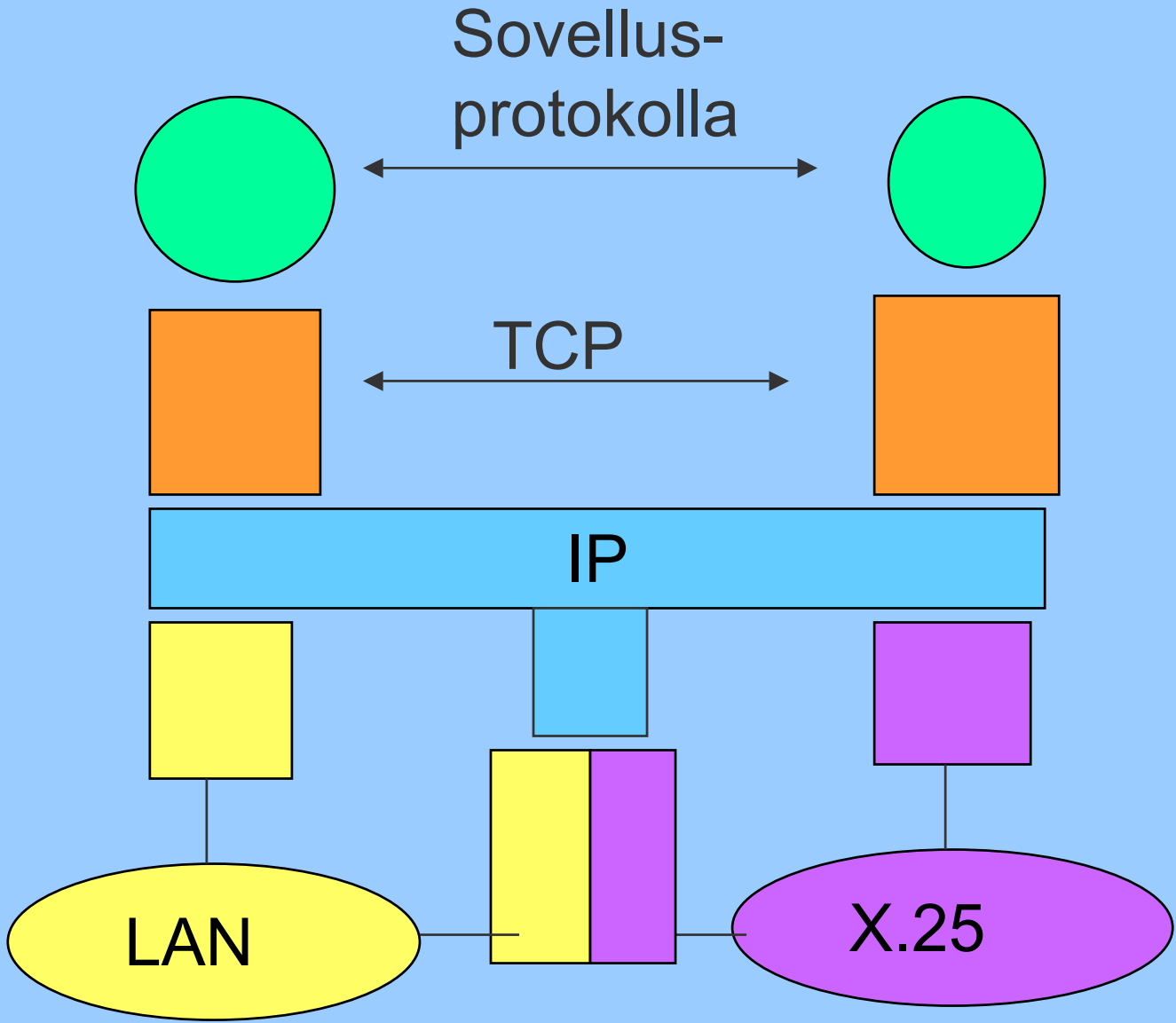
paketin data

Jatkuuko  
vielä?

## 4.4. Internetin verkkokerros

- **Internet**
  - on kokoelma ‘itsenäisiä’ aliverkkoja eli autonomisia järjestelmiä (AS, Autonomous Subsystem)
  - joita yhdistää runkolinjat
- **IP-protokolla**
  - verkkotason protokolla, joka pitää Internetin koossa
  - tavoite: kuljettaa paketti (datasähke, **datagram**) lähteestä kohteeseen yli kaikkien välissä olevien erilaisten verkkojen



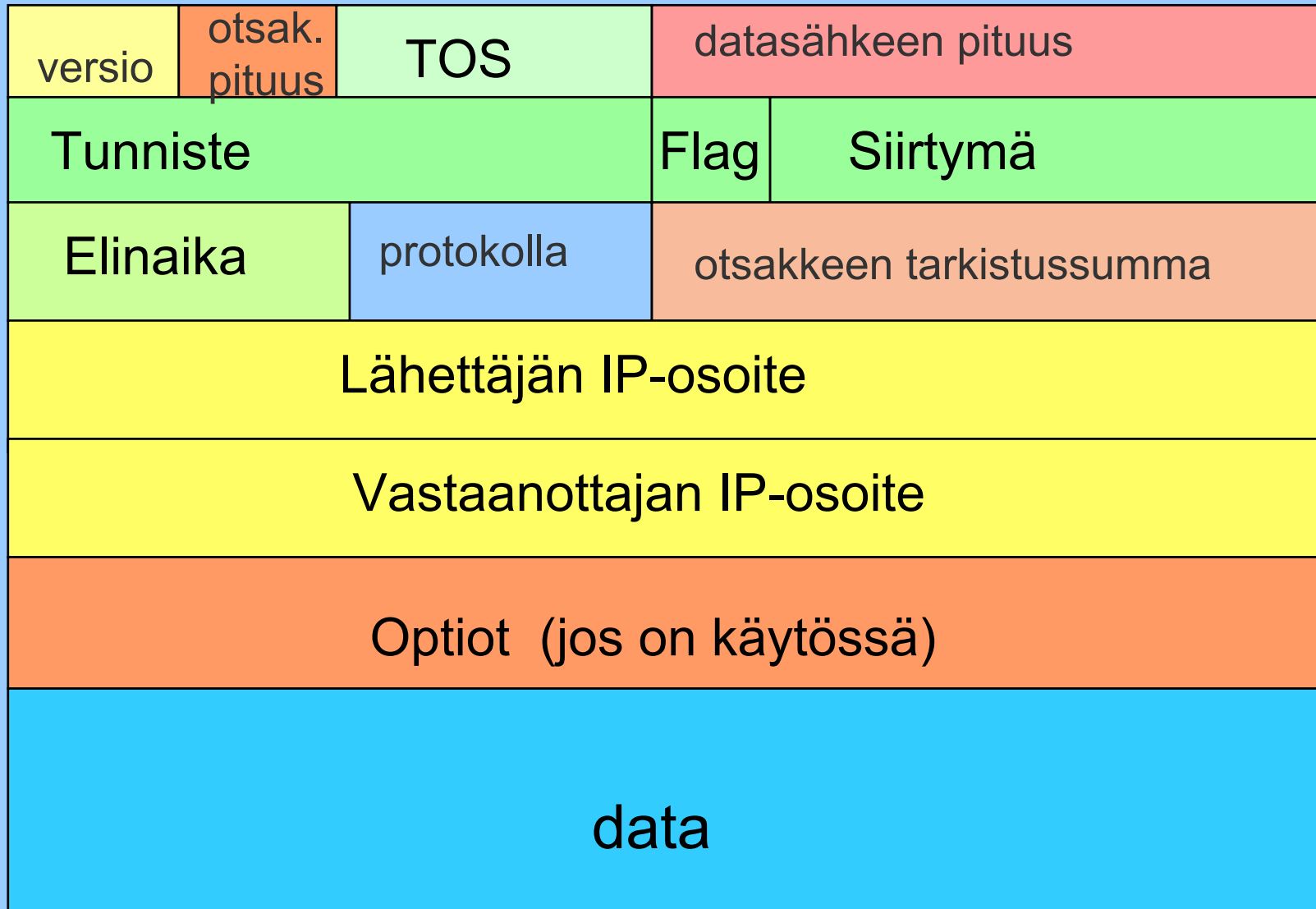


# IP kuljettaa lähdekoneelta kohdekoneelle

- **Tässä tehtävässä tarpeen:**
  - Osoitteet (lähettäjä, vastaanottaja)
  - tieto kuljetuskerroksen protokollasta
  - liian ison datasähkeen paloittelu
  - ‘eksyneiden’ pakettien hävittäminen (time-to-live)
  - tarkistukset (checksum)
- **Hyviä (?) lisäominaisuuksia**
  - kuljetuspalvelun eriyttäminen (type of service)
  - lisäpiirteitä: lähdereititys, tieto kuljetusta reitistä,

# IP-protokolla

- **IP-datasähke**
  - otsake
  - dataosa
- **otsake**
  - 20 tavun kiinteä osa
    - tunnistetiedot, pituustiedot, tarkistusbitit (-summa)
    - osoitteet, minkä kuljetusprotokollan sanoma
    - liian pitkän paketin paloittelu ja kokoaminen
    - erilaisen palvelun tarjoaminen eri sovelluksille
  - vaihtelevan mittainen valinnainen osuus
    - lisäoptioita



**IPv4 - datasähke**

# IP-otsakkeen kentät

- **Versio IPv4 ( IPv6)**
- **IHL**
  - otsakkeen pituus vähintään viisi 32 bitin sanaa (20-60 tavua)
- **type of service (8 bittiä)**
  - kertoo halutun palvelun
    - nopeus, luotettavuus, kapasiteetti
    - ääni <-> tiedostonsiirto
  - yleensä ei käytössä (käytössä uusissa Cisco-reitittimissä)

## Type of service -bitit:

- **presedence-kenttä (3 bittiä)**
  - sanoman **prioriteetti 0-7**
    - 0 normaali
    - 7 verkon valvontapaketti
- **D-bitti, T-bitti, R-bitti**
  - mikä on tärkeää yhteydessä
    - D: viive (Delay),
    - T: läpimeno (Throughput)
    - R: luotettavuus (Reliability)
- lisäksi vielä **2 käyttämätöntä bittiä**

# IP-otsakkeen kentät jatkuvat

- **Datagram length**

- koko datasähkeen pituus
- maksimi 65535 tavua
  - maksimipituus vielä riittävä, mutta tulevaisuuden nopeille verkoille jo ongelma
- yleensä koko 576 -1500 tavua

- **Identification**

- datasähkeen numero
- kaikissa saman datasähkeen osissa sama tunnus

# IP-otsakkeen kentät jatkuvat: liput

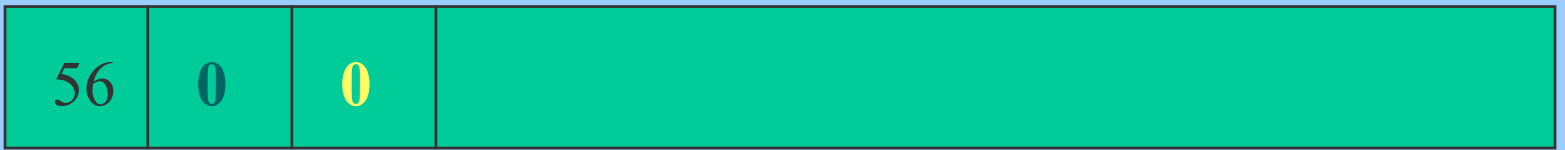
- **DF- bitti (Don't fragment)**
  - kieltää paloittelun
  - esim. jos vastaanottaja ei kykene kokoamaan datasähkettä
- **MF-bitti (More fragments)**
  - ilmoittaa, onko datasähkeen viimeinen osio vai tulee vielä lisää
- **Lisäksi yksi käyttämätön bitti**



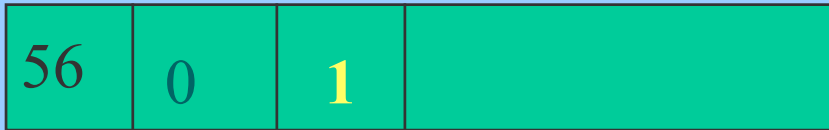
# IP-otsakkeen kentät jatkuvat

- **Fragment offset**

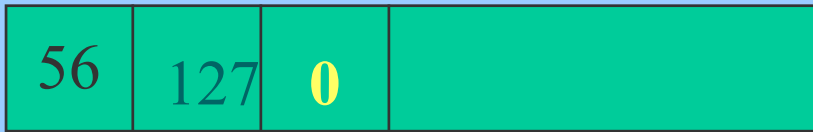
- osion paikka datasähkeessä
- osioiden oltava 8 tavun monikertoja (paitsi viimeisen)
- 13 bittiä => korkeintaan 8192 osiota yhdessä datasähkeessä



alkuperäinen paketti



paketin alkuosa



paketin loppuosa

paketin  
sijainti-  
numero



sijainti-  
kohta  
eli

osan  
numero



jatkuuko  
vielä?



paketin data



# IP-otsakkeen kentät jatkuvat

- **Time to live**

- rajoittaa paketin elinaikaa
- maksimi 255 sekuntia
- vähenee
  - joka hypyllä reitittimestä toiseen
  - myös odottaessaan reitittimessä (ei yleensä)
  - paketti hävitetään, kun laskuri menee nolille

- **Protocol**

- mille kuljetuskerrokselle kuuluu

- esim. TCP- tai UDP-siirtoon kuuluva

# IP-otsakkeen kentät jatkuvat

- **Header checksum**
  - tarkistussumma lasketaan vain otsakkeelle
  - 16-bitin sanat lasketaan yhteen yhden komplementin aritmetiikalla
  - laskettava uudestaan joka reitittimessä
- **Source address, Destination address**
  - kohteen ja lähettäjän osoitteet muodossa
    - verkon numero ja isäntäkoneen numero
  - = IP-osoite

# IP-otsakkeen kentät jatkuvat

- **Options**

- vaihtelevan mittaisia

- 1. tavu kertoo option koodin
- voi seurata pituuskenttä
- datakenttiä
- täytettä jotta 4 tavun monikertoja

- käytössä 5 optiota

- mutta reitittimet eivät välttämättä ymmärrä

# Optiot

- **Security**
  - datasähkeen luottamuksellisuus ja salassapidettävyys
- **Strict source routing**
  - datasähkeen kuljettava tarkalleen annettua reittiä
- **Loose source routing**
  - kuljettava ainakin annettujen reitittimien kautta
- **Record route**
  - reitin varrella olevat reitittimet liittävätkä tunnuksensa
- **Timestamp**
  - tunnuksen lisäksi liitettävä myös aikaleima

## 4.5. IP-osoitteet

- jokaisella verkon isäntäkoneella ja reitittimellä on oma yksikäsitteinen osoite muotoa
  - verkon numero
  - isäntäkoneen (liitäntäkortin) numero
- **osoite on 32-bittinen**
  - osoitteen luokasta riippuen bitit jaetaan verkon numeroon ja isäntäkoneen numeroon eri tavoin
- **osoitteet palvelun tarjoajille jakaa ICANN**  
(The Internet Corporation for Assigned Names and Numbers)
  - nämä puolestaan jakavat muille

- **osoitteet merkitään yleensä desimaalimuodossa**

- kukin osoitteen neljästä tavusta kirjoitetaan desimaalilukuna (0-255)
- luvut erotetaan pisteellä

– esim.

- heksadesimaaliosoite C0 29 06 14 on 192.41.6.20  
eli C0 => 192, 29 => 41, 06 => 6, 14 => 20

– pienin osoite on 0.0.0.0 ja suurin  
255.255.255.255





**IP-osoitteiden muodot**  
 (alkuperäinen luokallinen osoitus)

# IP-osoitteiden luokat

- **A-luokka hyvin isoille verkoille**
  - **7 bittiä** verkko-osoitteeseen, **24 bittiä** isäntäkoneille
  - **126 verkkoa, 16 miljoonaa konetta/verkko**
- **B-luokka keskikokoisille verkoille**
  - **14 bittiä** verkoille, **16 bittiä** koneille
  - **16382 verkkoa, 65528 konetta/verkko**C-luokka
- **C-luokka pienille verkoille**
  - **21 bittiä** verkoille, **8 bittiä** verkon koneille
  - **noin 2 miljoonaa verkkoa, 254 konetta/verkko**

# Osoiteluokkien ongelmia

- **verkon kasvu => ongelmia**
  - C-luokan verkossa max 256 osoitetta
    - liian vähän useimmille yrityksille => tarvitsevat B-luokan osoitteen tai monta C-luokan verkko-osoitetta
  - B-luokan verkkoja liian vähän (max 16382) ja niissä liian paljon osoitteita (max 65536)
    - 100000 verkkoa jo 1996!
    - useassa B-verkossa alle 50 konetta
- **=> B-luokan osoitteita tuhlaantuu ja osoitteista pulaa**

# CIDR (Classless InterDomain Routing)

- verkko-osa voi olla minkä tahansa kokoinen (ei vain 8,16,24 bittiä)
  - a.b.c.d/x, jossa x ilmoittaa verkko-osan bittien lukumäärän
  - esim. yritykselle, jolla 2000 konetta varataan  $2048 = 2^{11}$  koneosoitetta, jolloin verkko-osaa varten jää 21 bittiä
    - C-luokan verkkoja
  - yritys voi itse vielä jakaa koneosoitteen 11 bittiä aliverkko-osoitteeksi ja koneosoitteeksi

# CIDR-idea jatkuu

- jaetaan osoitteet neljään osaan, kukin osa varataan yhdelle maanosalle  
(Eurooppa, Pohjois-Amerikka, Etelä-Amerikka, Aasia+Pasific)
  - kullekin noin 32 miljoonaa osoitetta
  - 320 miljoona jää vielä varastoon
- reititetään myös maanosien mukaan
  - osoitteet: 194.0.0.0 - 195.255.255.255 Eurooppaan
- => pienemmät reititystaulut

# Muita Internet-protokollia

- **ICMP** (Internet Control Message Protocol)
  - verkon koneiden (reitittimien ja isäntäkoneiden) kommunikointiin esim. virhetilanteissa
- **ARP** (Address Resolution Protocol)
  - protokolla lähiverkon koneen verkko-osoitteen selvittämiseksi
- **Reititysprotokollat:**
  - **RIP (Routing Information Protocol):** etäisyysvektoreititys
  - **OSPF (Open Shortest Path First):** linkkitilareititys
  - **BGP (Border Gateway Protocol):** eri AS:ien välinen reititysprotokolla
- **IPv6**
  - uudempi versio IP-protokollasta
- **Näitä käsitellään Tietoliikenne II -kurssilla**

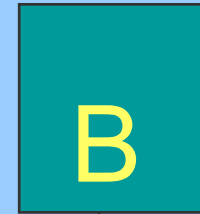
..128.214.4.29 ..

IP-paketissa  
on vain  
vastaan-  
ottajan IP-  
osoite



B:n  
verkko-  
osoite

..128.214.4.29 ..



**Pitää saada selville  
IP-osoitetta  
vastaava verkko-  
osoite.**

Yleislähetyksenä  
kysely: "Kenen IP-  
osoite?"



**Jokaisella koneella  
oma ethernet-osoite  
(48 bittiä), jota  
käytetään MAC-  
kehyksessä**