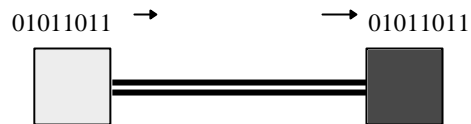


# 3. Siirtoyhteyskerros

linkkikerros (Data Link Layer)

- yhtenäinen linkki solmusta solmuun

- bitit sisään => bitit ulos



- ongelmia:

- siirtovirheet
  - havaitseminen
  - korjaaminen
- solmun kapasiteetti
  - vuonvalvonta

## Virheiden hallinta

- bittivirta ==> kehyksiä

- tarkistus
- korjaus/uudelleenlähetyks

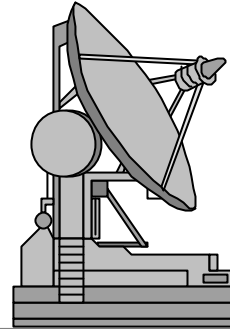
- kuittaus

- mitä lähetettävä uudelleen
- kuittausviive
  - odotettava kuittausta



# Kuittausviive

- hidastaa lähetystä
  - odotettava kuittaus ennenkuin voidaan lähettää lisää
    - esim. satelliiteilla viive hyvin pitkä
- tuhlaa kapasiteettia
  - linja käyttämättömänä odottelun ajan
- lähetetään ennakkoon
  - kuittaukset ja lähetykset 'limittyvät'
- korjaava koodi
  - vastaanottaja pystyy korjaamaan

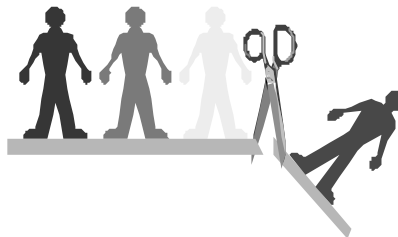


27.9.2000

3

# Vuonvalvonta

- lähetys ei saa ylittää vastaanottajan käsittelykapasiteettia
  - rajalliset puskurit
  - ylimäärä häviää
- kapasiteettiraja =  
(ennakko)lähetyksen määrän rajoitus



27.9.2000

4

## 3.1. Suunnitteluperiaatteita

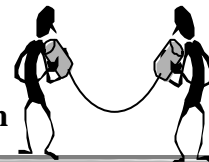
- **siirtoyhteyskerros tarjoaa palveluja verkkokerrokselle**
  - **datan siirto lähdekoneen verkkokerrokselta kohdekoneen verkkokerrokselle**
    - **kuittaamaton yhteydetön palvelu**
    - **kuittaava yhteydetön palvelu**
      - virheelliset havaitaan ja lähetetään uudestaan
    - **kuittaava yhteydellinen palvelu**
      - luotettava bittivirta

27.9.2000

5

## Verkkokerroksen tarpeet ja perustason laatu => linkkitason tehtävät

- **virheettömyys**
  - perustaso luotettava, linkkitaso ei tee mitään
  - perustaso epäluotettava, linkkitaso kuittaava
- **järjestys**
  - yhteydetön: järjestys voi muuttua
  - yhteydellinen: järjestys säilyy
- **kaksoiskappaleet**
  - kuittaava: voi syntyä
    - yhteydetön: ei havaita/havaitaan
    - yhteydellinen: kaksoiskappaleet tuhoataan



27.9.2000

6

## Missä virhe hoidetaan?

- **kuittaava linkkikerros havaitsee virheet ja korjaa ne**
- **yhteydetön, kuittaamaton & virhe**  
=> **kuljetuskerros havaitsee ja korjaa**
- **ja jos ei, niin sovelluskerros havaitsee ja korjaa**
- **ja jos ei, niin asiakas havaitsee ja korjaa**

### 3.1.2. Kehystys (framing)

- **tavoite**
  - **bittivirheiden hallinta**
    - **muuttuu**
    - **katoaa**
    - **monistuu**
- **bittivirta kehyksinä**
- **kehys tarkistettavissa**
  - **tarkistustietoa**

# Kehysrakenne

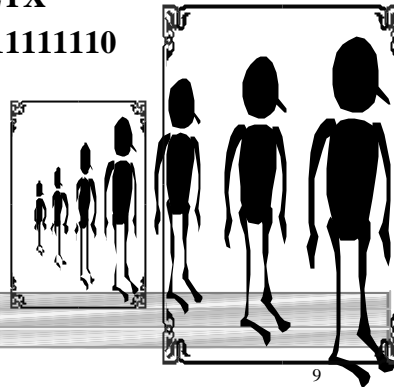
## ● bittivirta => kehysrakenne?

### – Miten kehys rajataan?

- aloitusmerkki DLE STX,
- lopetusmerkki DLE ETX
- erillinen lipputavu: 01111110

01111110	...01111110...	01111110
----------	----------------	----------

Entä jos haluaa lähettää kehyksessä dataa 01111110 tai DLE-merkin ?



27.9.2000

9

01111110	... 01111110 ...	01111110
----------	------------------	----------

01111110	...	01111110	...	01111110
----------	-----	----------	-----	----------

Ongelma ratkaistaan: Bitin lisääminen (Bit stuffing)

- lisäämällä lähetysvaiheessa automaattisesti dataan ylimääräinen 0-bitti aina viidennen peräkkäisen 1-bitin perään.

01111110 => lähetetään 011111010

- siis ensin linjalle lähetetään lippukuvio ja sitten data, johon on tarvittaessa lisätty ylimääräinen bitti
- vastaanotossa ylimääräiset 0-bitit poistetaan automaattisesti
- Samoin lisätään ylimääräinen DLE-merkki jokaisen datassa olevan DLE-merkin jälkeen (character stuffing)

## Kehystahdistus = löydetään kehysrajat

- **kehystahdistus onnistuu mielivaltaisen bittivirheen jälkeen**

- bittivirhe rikkoo lippukuvion  
01111110 => 01101110
- bittivirhe tekee dataan ylimääräisen lippukuvion  
... 01110110 => 01111110
- näistä huolimatta parin virheellisen kehyksen jälkeen löydetään 'oikea' kehyksen alku

## Kehysten kuljetus

- **tavoite**

- kaikki kehykset
- kukin kehys virheettömästi
- lähetysjärjetyksessä

- **vastaanottaja kertoo lähettäjälle**

- **ACK:** kehys vastaanotettu ok
  - tietty kehys
  - kaikki kehykset tähän asti
- **NAK:** kehyksessä vikaa => lähetettävä uudelleen
- **Saako lähettää lisää vai pitääkö keskeyttää**
  - vuonvalvonta

## Kehysten kuljetus (2)

- **Entä jos kehys katoaa kokonaan?**
  - Ei tule mitään kuittausta!
  - Joko sanoma kadonnut tai kuittaus kadonnut
- **lähettäjän ajastin**
  - laukeaa, jos kuittausta ei tule tietyssä aikana
  - ajastimen laukeaminen => uudelleenlähetys
- => vastaanottaja voi saada useita kopioita
  - kyettävä erottamaan duplikaatit
  - ==> kehysnumero

## Vuonvalvonta

- **lähettäjä ei saa lähettää enempää kuin vastaanottaja kykenee käsittelemään**
  - puskurit vuotavat yli => dataa katoaa
- **vastaanottaja säätelee lähettämistä**
  - lähetyslupa N:lle kehykselle
  - odota sitten uutta lupaa

## 3.2. Virheiden havaitseminen ja korjaaminen

Virheiden takia dataan lisäinformaatiota:

- **virheen korjaus** (error-correcting code, forward error control)
  - lisäinformaatiota niin paljon, että vastaanottaja sekä havaitsee että kykenee itse korjaamaan virheen
- **virheen havaitseminen** (error-detecting code, feedback/backward error control)
  - lisäinformaatiota, jotta vastaanottaja havaitsee virheen tapahtuneen => korjauksena uudelleenlähetyks

## Korjaus/havaitseminen

- **virheen korjaava koodaus**
  - kallis koko ajan
    - paljon lisäinformaatiota
  - rajoitettu korjauskyky
    - esim. kokonaan kodonnut kehys
- **virheen havaitseva koodaus**
  - virheen sattuessa kallis
    - uudelleen lähettäminen maksaa
    - uudelleen lähettäminen on hidasta



# Virheet

- **Kahdenlaisia virheitä:**

- yhden bitin virheet
- usean peräkkäisen bitin vääristyminen (burst error)

- **Virheiden esiintymistiheys**

- **BER (bit error rate)**
- **mitä suurempi BER, sitä lyhyempiä kehyksiä kannattaa käyttää**

# Virheen korjaus

- **Käytetään esim.**

- CD- ja DVD-levyissä, digitaalitelevisiossa
- nopeissa modeemeissa, kannettavissa puhelimissa
- satelliittiyhteyksissä, avaruusluotaimissa

- **Esimerkkejä**

- Hamming-pariteettitarkistus (Tito-kurssilla)
  - pystyy korjaamaan yhden virheellisen bitin
  - virheryöpyä, jos se jaetaan yhden bitin virheiksi
- Reed-Solomon -koodit
  - lohkokoodit, jotka pystyvät korjaamaan virheryöppyjä

## Yhden bitin virheet korjaava koodi

- **Hamming: minimimäärä tarkistusbittejä, kaikkien yhden bitin virheiden korjaamiseksi**
- **00110010000**
  - bitit 1, 2, 4, 8, 16,.. eli kakkosen potenssit tarkistusbittejä (pariteettibittejä)
  - muut 3,5,6,7,9,10,11, ... databittejä
- **yhtä databittiä varmistaa usea pariteettibitti**
  - esim. paikassa 11 olevaa bittiä tarkistusbitit paikoissa  $1+2+8=11$
  - kun tarkistusbitit paikoissa 1, 2 ja 8 eivät täsmää => bitti paikassa 11 on virheellinen

## Virheryöpyn korjaaminen

- **Hamming-koodit korjaavat vain yksittäisiä virheitä**
- **virheryöpyn korjaamiseksi järjestetään k peräkkäistä koodisanaa matriisiksi**
  - yksisana per rivi
- **lähetetään matriisi sareke kerrallaan**
  - ei siis rivi kerrallaan
  - **k:n bitin mittainen virheryöppy hajoaa k:ksi yksittäiseksi virheeksi**
    - korkeintaan yksi virhe yhdessä koodisanassa
    - jonka Hamming-koodi osaa korjata

Esimerkiksi:

1001000	=>	1110111
1100001		0110111
1101101		0000000
0000000		1010110
1101001		0010011
1101110		0000011
1100111		0110101

virheryöppy

yksittäisiä virheitä

## Virheen havaitseminen

### ● Pariteettibitti

- parillinen pariteetti
- pariton pariteetti

### ● horisontaaliset ja vertikaaliset pariteetit

### ● CRC (Cyclic redundancy code (tai check))

- yleisimmin käytetty virheen paljastusmenetelmä
- perustuu polynomien aritmetiikkaan (modulo2-aritmetiikkaan, XOR)
- useita tarkistusbittejä => havaitaan usean bittivirheen ryöppy

## Pariteetti

- **esimerkki yksinkertaisesta virheen havaitsevasta koodista**
- **jokaiseen merkkiin lisätään yksi ylimääräinen ns. pariteettibitti**
  - lisäyksen jälkeen kaikissa merkeissä on parillinen (tai jos niin sovitaan pariton) määrä ykkösiä
- **paljastaa kaikki yhden bitin virheet**
  - kehyksen pituudesta riippumatta
- **ei paljasta kahden bitin virheitä**

## Pariteettibitin käyttö

- **erityisesti asynkronisessa tiedonsiirrossa merkkejä siirrettäessä**
- **käytännössä paljastaa noin puolet virheellisistä bittijonoista**
  - esim. modeemeissa syntyy useita virheitä
  - linjahäiriöt aiheuttavat usein pitkiä virheryöppyjä

## Horisontaaliset ja vertikaaliset pariteetit

- järjestetään bittijono kaksikulotteiseen taulukkoon
- lasketaan pariteetti jokaiselle vaaka- ja pystyriville

1 0 0 1 0 1 0		1	
0 1 1 1 0 1 0		0	
1 1 1 0 0 0 1		0	horisontaaliset
1 0 0 0 1 1 1		0	pariteetit
0 0 1 1 0 0 1		1	
<hr/>			
1 0 1 1 1 1 1		0	taulukon pariteetti
			vertikaaliset
			pariteetit

## Virheiden havaitseminen

- Ei löydy lyhyitä virheryöppyjä, joissa neljä bittiä vaihtuu sopivasti

```
1 0 0 1 0 1 0
0 1 1 1 0 1 0
1 1 ① 0 0 ① 1
1 0 0 0 1 1 1
0 0 ① 1 0 ① 1
```

## CRC:n perusidea

- tarkistusavain (virittäjä, virittäjäpolynomi)

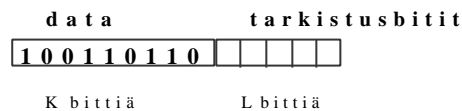
- L+1 bittiä
- lähettäjä ja vastaanottaja tuntevat

- lähettäjä

- laskee lähetettävälle datalle tarkistusbitit ja liittää ne kehykseen

- vastaanottaja

- tarkistaa, onko koko saapunut kehys (data + tarkistusbitit) pysynyt muuttumattomana

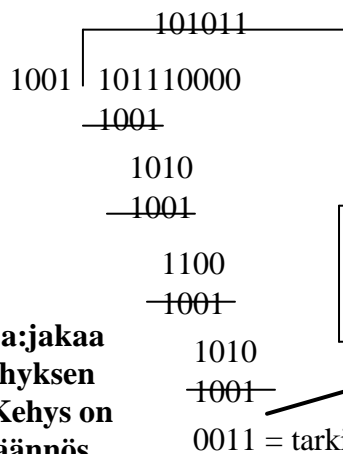


K bittiä                      L bittiä



**Esimerkki:** data = **101110**, virittäjä = **1001**, (polynomina  $X^3 + 1$ ), tarkistusbittejä 3

Lähetettävä data = **101110???** tarkistusbitit



**Modulo 2-  
aritmetiikka:**

**1+1 = 0 (XOR)**

Lähetetään: **101110 011**

**Vastaanottaja:** jakaa saamansa kehyksen virittäjällä. Kehys on ok, jos jakojäännös on 0!

### Standardoituja virittäjäpolynomeja

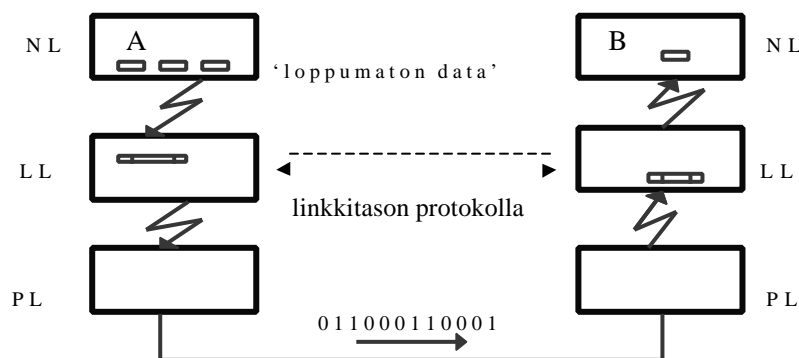
- CRC-12 =  $x^{12} + x^{11} + x^3 + x^2 + x + 1$
- CRC-16 =  $x^{16} + x^{15} + x^2 + 1$
- CRC-32 =  $x^{32} + x^{26} + x^{23} + \dots + x^4 + x^2 + x + 1$

### CRC: n virheiden havaitsemiskyky

- kaikki virheröpyt, joiden pituus < tai = virittäjän
  - useimmat virheröpyt, joiden pituus on suurempi
    - CRC-32:  $P\{\text{ryöppy} > 33 \text{ havaitaan}\} = 0.999999998$
- Huom
- » Arvioinneissa lähtökohtana ollut täysin satunnainen bittien jakautuminen, mutta todellisuudessa näin ei ole!
  - » Joten havaitsemattomien virheiden määrä on arvioitua suurempi.

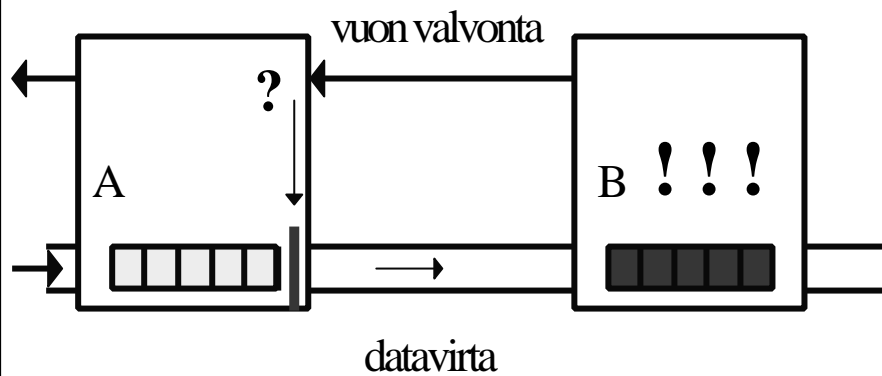
## 3.3. Linkkitason protokollat

### Peruskäsitteet



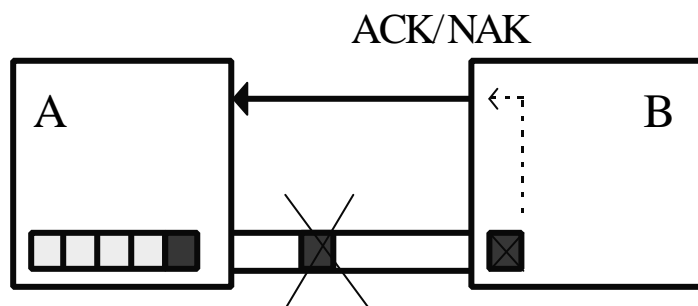
- NL: A => B “ datavirtaa luotettavasti ja yhteydellisesti”
- LL: siirtää dataa, suorittaa tarkistukset, virheestä toipumiset, huolehtii vuon valvonnasta

## Vuon valvonta



- X-ON / X-OFF : GO! | STOP!

## Kohinainen kanava



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



## Yksinkertainen **Stop and wait** -protokolla

### ● Oletus

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

### ● lähettäjä

- lähettää kehyksen
- odottaa lupaa lähettää seuraava kehys

### ● vastaanottaja

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

## **ARQ-protokolla** (Automatic Repeat Request)

### ● normaali tilanne

- kohinainen kanava => virheitä
- sanomat ja kuittaukset voivat kadota

### ● ajastin lähettäjälle

- jos kuittausta ei kuulu, kehys lähetetään automaattisesti uudelleen
- kuittaus: ACK = 'ok, lähetä seuraava'
- uudelleenlähetykset synnyttävät kaksoiskappaleita!

### ● kehysnumerointi

- jotta vastaanottaja tunnistaa kaksoiskappaleet
- Miten paljon numeroita tarvitaan?
  - Numero vie tilaa kehyksessä!

## Stop and wait -protokollan suorituskyky

- Esim. satelliittiyhteydellä
  - 50 kbps, kiertoviive ~520 ms, kehys 1000 bittiä
  - kanavan käyttöaste < 4%
- => lähetetään useita kehyksiä ja sitten vasta odotetaan kuittauksia
  - **ideaali: lähetykset liukuhihnalla (pipeline)**
    - lähetykset ja kuittaukset limittyvät
    - ei mitään odottelua
    - lähetyiskanava koko ajan käytössä
  - suorituskyky kasvaa

## Liukuvan ikkunan protokolla (Sliding Window)

- **Lähetyksikkuna**
  - **ikkunan koko**
    - montako kehystä saa korkeintaan olla kuittaamatta
    - järkevä koko riippuu yhteyden tyypistä ja vastaanottajan kapasiteetista
  - **sisältö = mitkä kehykset saa lähettää**
    - kehyksellä järjestysnumero
      - rajallinen, N bittiä =>  $2^N$  arvoa
      - numerot käytettävä järjestyksessä

- Lähettäjä joutuu odottamaan vasta, kun kaikki ikkunan kehykset on lähetetty
  - eli kehysnumerot käytetty
- Kun kuittaus saapuu => ikkuna liikuu
  - seuraavat kehysnumerot tulevat luvallisiksi
- eli
  - lähettäjä: tietyllä hetkellä sallittujen numeroiden joukko = lähettäjän ikkuna
    - mitkä kehykset saa lähettää “etukäteen” odottamatta kuittausta

- **Vastaanottajan ikkuna**
  - kullakin hetkellä sallittujen numeroiden joukko
    - mitä kehyksiä suostuu vastaanottamaan
  - kuittaus muuttaa myös vastaanottajan ikkunan
- ikkuna pysäyttää kehysten lähetyksen
  - seuraava kehysnumero ei ole lähetysikkunassa
- ikkuna estää kehyksen vastaanoton
  - saadun kehyksen kehysnumero ei ole vastaanottoikkunassa

## Kun ikkunan koko on 1

- **Aina vain yksi kehys kuittaamattomana**
  - => One Bit Sliding Window -protokolla
  - ~ stop and wait -protokolla
- **kehysnumerot 0 ja 1 riittävät**
- **ACK-kehyksessä viimeksi vastaanotetun virheettömän kehyksen kehysnumero**
  - jotta kuittausduplicaatti ei voi kuitata väärää kehystä
- **samanaikainen aloitus tai liian lyhyt lähettäjän ajastimen aika voi aiheuttaa paljon turhia lähetyksiä**

- **entä kun tapahtuu virhe?**
  - kaksi eri tapaa hoitaa
  - toisto virheestä lähtien (go back n)
  - valikoiva toisto (selective repeat)

## **Paluu n:ään ('Go back n')**

- **virheellisen kehysten havaittuaan**
  - vastaanottaja hylkää kaikkia sen jälkeiset kehukset
  - eikä lähetä niistä kuittauksia
- **kun lähettäjä ei saa kuittauksia,**
  - sen lähetysikkuna 'täyttyy'
  - eikä se voi enää lähettää
- **lähettäjän ajastimet laukeavat aikanaan ja**
  - virheellinen kehys
  - sekä kaikki sen jälkeen lähetetyt kehukset lähetetään uudelleen
- **tehoton, jos paljon virheitä ja iso ikkuna**

## **Valikoiva toisto**

- **vastaanottaja hyväksyy kaikki kelvolliset kehukset**
  - se kuittaa kehukset
  - puskuroi ne ja toimittaa eteenpäin oikeassa järjestyksessä
    - tarvitaan puskuritilaa
- **lähettäjä ei saa kuittausta virheellisestä kehyksestä**
  - ajastin laukeaa ja kehys lähetetään uudelleen
  - lähettää uudelleen vain virheellisen kehysten
  - ikkuna liukuu nytkin tasaisesti
    - yksi puuttuva kuittaus voi pysäyttää lähetyksen

## Ikkunankoko

- Kun käytetty numeroavaruus on  $0, 1, .. n$  ja eri numeroita siis käytettävissä  $n+1$ 
  - yleensä jokin kakkosen potenssi
- ikkunan koko 'go back n':ssä voi olla korkeintaan  $n$
- ikkunan koko valikoivassa toistossa voi olla korkeintaan  $(n+1)/2$

## Kaksisuuntainen liikenne

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

## 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

## 3.6. Linkkitason protokollia

- Linkkitason protokollia on useita
  - **HDLC** (High-level Data Link Control)
    - useita, enemmän tai vähemmän toisistaan poikkeavia yhteensopimattomia versioita
    - käsitellään jonkin verran harjoituksissa
  - **PPP** (Point-to-Point Protocol)
    - soittoyhteys modeemin tai ISDN:n kautta tietokoneeseen
    - yleisimmin käytettyjä linkkiprotokollia

## PPP (Point-to-Point Protocol)

- IETF:n vaatimuksia
  - hyvin toimiva kehystys
  - kehysten virhetarkistus (virheellinen kehys tuhotaan!)
  - havaitsee, jos yhteys ei toimi ja ilmoittaa tästä verkkokerrokselle
  - useat verkkokerroksen protokollat voivat käyttää
  - verkko-osoitteista sopiminen: mm. IP-osoitteet neuvoteltavissa yhteyden muodostuksen aikana
  - autentisointi mahdollista
  - ei vuonvalvontaa



## PPP-kehys

Tavuja	1	1	1	1-2	vaihtelee	2-4	1
01111110	osoite	kontrolli	protokolla	data	CRC	01111110	

- **lipputavu 01111110,**
  - character stuffing, DLE = 01111101
- **osoitekenttä aina 11111111 (=yleislähetys)**
- **kontrollikenttä aina 00000011**
  - osoite- ja kontrollikenttä voidaan jättää kokonaan pois
- **protokolla: mille protokollalle data on tarkoitettu**
  - esim. IP, IP:n Control Protocol, PPP:n Link Control Protocol
- **data: sisältää ylemmälle protokollalle tarkoitettua dataa**
  - maksimi sovitaan, oletusmaksimi 1500 tavua
- **CRC: tarkistusbitit;**

- **LCP (Link Control Protocol)**

- » muodostaa ja testaa linjayhteyksiä
- » neuvottelee yhdeyden ominaisuuksista
- » purkaa yhteyden, kun sitä ei enää tarvita

- **NCP (Network Control Protocol)**

- » neuvottelee verkkokerroksen optioista
- » oma NCP kullekin verkkoprotokollalle
- » TCP/IP: tärkein tehtävä IP-osoitteen antaminen päätteelle dynaamisesti

## Yhteydenotto PPP:llä

- soitto modeemilla reitittimeen
  - » fyysinen yhteys
- PPP-parametrien valinta
  - » LCP-paketteja vaihtamalla
- verkkokerroksen konfigurointi
  - » TCP/IP: IP-osoitteen antaminen PC:lle
  - » PC => tilapäinen Internet isäntäkone
- PC voi lähettää ja vastaanottaa tavallisen isäntäkoneen tapaan

## Yhteyden purku

- NCP purkaa verkkoyhteyden ja vapauttaa IP-osoitteen
- LCP purkaa siirtoyhteyserroksen

## Linjayhteyden muodostus

- **Dead**
  - » ei kantoaaltoa, ei peruskerroksen yhteyttä
- **Established**
  - » peruskerroksen yhteys muodostettu
  - » sovitaan LPC-optioista
- **Authenticate**
  - » osapuolet varmistuvat toistensa identiteetistä
- **Network**
  - » NCP konfiguroi verkkokerroksen

- **Open**
  - » tiedonsiirto voi alkaa
- **Terminate**
  - » kun tiedonsiirto suoritettu => lopetustilaan
  - » tästä palataan alkutilaan lopettamalla kantoaalto

## LPC-pakettityypit

- optioista ja niiden arvoista sopiminen

- Configure-

- » request ehdotettuja optioita ja arvoja
    - » ack kaikki hyväksytään
    - » nak optioita, joita ei voida hyväksyä
    - » reject optioita, joista ei voida neuvotella

- linjan sulkeminen

- Terminate-

- » request linjan sulkemispyyntö
    - » ack OK, linja suljetaan

- tuntemattomat sanomat

- Code-reject tuntematon pyyntö
  - Protocol-reject tuntematon protokolla

- linjan testaus

- Echo-request palauta tämä kehys
  - Echo-reply tässä kehys takaisin
  - Discard-request hylkää tämä testisanoma

# Yhteenveto

- tehtävät
- kehystys
- virhetarkistukset
  - havaitseva /korjaava
  - pariteetti, Hamming, CRC
- linkkitason protokollien peruskäsitteitä
  - stop and wait
  - liukuva ikkuna
  - go-back n
  - valikoiva toisto
- esimerkki: PPP