***Pariteettitarkistus***

* menetelmä perustuu siihen, että edellytämme tietyltä bittijoukolta parillisen tai parittoman määrän 1-bittejä. jotta 1-bittien lukumäärä saadaan halutuksi, varataan yksi ylimääräinen bitti pariteettia varten ja tästä bitistä käytämme nimitystä pariteettibitti. pariteettitarkistus voidaan tehdä sanomalle joko pitkittäin tai poikittain. pitkittäispariteetissa joukko muodostuu kaikista sanoman biteistä b1, joiden sijainti merkin sisällä on sama. näin ollen sanoman kaikkien merkkien bitit b2 lasketaan mukaan, kun muodostetaan pariteettitavun bitin 2 arvo. poikittaispariteetissa joukko muodostuu yhden merkin kaikista biteistä ja pariteettia varten varataan oma erillinen bitti.
* Parieteetin muodostaminen on teknisesti yksinkertaista, ja siksi myös erittäin nopeata. Useimmat sarjaliikenneohjaimet osaavat muodostaa pystypariteetin halutulla tavalla, ja monet prosessorit sisältävät tilasanoissa lippuja kertomassa, onko jokin rekisterin ykkösbittien lukumäärä parillinen, vakio vai pariton. Pitkittäispariteetin muodostetaan yksinkertaisesti siten, että alustamme pariteettia varten varatun muuttujan bitin ykkösiksi (x’ff’), jos haluamme muodostaa parittoman pitkittäispariteetin, ja parillista pitkittäispariteettia varten muuttujan kaikki bitit asetetaan nollaksi (x’00’). Sitten lisäämme modulo 2 –yhteenlaskulla (XOR-käskyllä) kaikki siirrettävän sanoman merkit tähän laskuriin. Lopuksi laskuri siirretään erillisenä pariteettitavuna sanoman perässä. Merkkipohjaisessa siirrossa on tapana muodostaa pitkittäispariteetti siten, ettei se sisällä sanoman alussa olevaa STX -merkkiä, mutta sisältää sanoman lopussa olevan ETX- tai ETB -merkin. Jos sanoma alkaa SOH/tieto/STX -sekvenssillä, pitkittäispariteetti kattaa kaikki merkit lukuun ottamatta SOH -merkkiä.
* Pariteettitarkistuksen antama suoja perustuu todennäköisyyteen, että tarkistuksen kohteena olevasta bittijoukosta vääristyy pariton määrä bittejä. Jos bittijoukosta vääristyy parillinen määrä bittejä, jäävät virheet huomaamatta.
* Pariteettitarkistus, jossa on yhdistetty sekä pitkittäis- että poikittaispariteetti, on kaikista heikkouksistaan huolimatta ollut melko yleinen. Merkkipohjaisilla ASCII –protokollilla, joita käytettiin tietoliikenteessä 1970- ja 1980 –luvuilla, tämä oli yleisin tapa, jolla virheet havaittiin, ja yksinkertaisissa sovelluksissa tämä on edelleen käyttökelpoinen menetelmä. Tällaisia sovelluksia ovat omatekoiset protokollat, joissa tietokoneen sarjaporttia käyttäen liikennöidään jonkun toisen laitteen kanssa.

***Polynominen tarkistus***

* Polynomisessa tarkistuksessa eli CRC -menetelmässä (Cyclic Redundancy Check) sanoman muodostava binääriluku jaetaan ennalta sovitulla jakajalla syntyvän jakojäännöksen muodostaessa tarkistusluvun.
* Pariteettitarkistuksen antamaa suojaa pidetään nykypäivänä liian heikkona, ja siksi se on saanut väistyä tiedonsiirrossa ja datan taltioinnissa. tilalle ovat tuleet polynomiset CRC-menetelmät. crc-menetelmällä muodostetusta tarkistusluvusta käytämme nimitystä FCS. CRC- menetelmät perustuu seuraavaan ajatukseen  
  - käsitellään sanoma suurena binäärilukuna, joka jaetaan ennalta sovitulla jakajalla. syntynyt jakojäännös muodostaa tarkistusluvun.  
  - laskennassa käytetään modula 2- artimetiikka, jonka säännöt ovat  
  1+1=0  
  1+0=1  
  0+1=1  
  0+0=0

***TCP/IP tarkistusluvut  
 (1-komplementti)***

* Yksinkertaisin menetelmä tiedon oikeellisuuden tarkistamiseksi on pariteettitarkistus, mutta heikon suojansa takia nykyään käytetään polynomisia CRC- menetelmiä. Tässä dataan liitetään tarkistusluku, joka pitää ohjelmallisesti tuottaa. Koska CRC –menetelmä on aikaa vievä, TCP/IP -protokollissa käytetään 1-komplementti-laskentaa, jonka etuna on laskennan helppous, vaikka luotettavuus kärsiikin.
* TCP/IP protokollissa käytetään tarkistusluvun muodostamisessa 1-komplementti-laskentaa.
* tämä menetelmä ei ole aivan yhtä luotettava kuin CRC – menetelmä, mutta sen etuna on laskennan helppous.
* jos CRC –menetelmällä tuotetaan tarkistusluku ohjelmallisesti, kuuluu sen vaatimiin bittioperaatioihin melkoisesti aikaa verrattuna tähän menetelmään, jossa riittää 16-bittisen sanojen yhteenlasku.
* Laskennalla saavutetaan virheenpalastustarkkuus, jossa   
  - havaitaan kaikki yhden bitin virheet  
  - havaitaan kaikki kahden bitin virheet  
  - 0,000019% alle 16 bitin virheryöpyistä jää havaitsematta
* 1. Alimmalla tasolla on liittymäkerros, joka määrittelee tavan, jolla iäntäkone liittyy internet-verkkoon. Kerrosta voidaan pitää hieman epämääräisenä, koska sitä ei olla TCP/IP-arkkitehtuurin puolelta jäsennelty tarkemmin, vaan ollaan käytännössä sisällytetty OSI\_arkkitehtuurin kaksi alinta kerrosta kuvaamaan liittymäkerroksen toimintoja.
* 2. Internet-kerroksen tehtävänä on siirtää tietosähkeitä Internet-verkon yli lähettäjältä vastaanottajalle. Jokainen tietosähke reititetään itsenäisesti parasta mahdollista reittiä pitkin, eikä osapuolten välillä ole olemassa erillistä sopimusta yhteyden olemassaolosta, vaan puhumme yhteydettömästä tiedonsiirrosta.
* 3. TCP/IP:n kuljetuskerros muistuttaa tehtäviltään OSI-viitemallin kuljetuskerrosta. TCP/IP-viitemallissa meillä on kaksi protokollaa: yhteydellinen TCP ja yhteydetön UDP. Käsitteet yhteydellinen ja yhteydetön käsitellään tarkemmin siirtoyhteyskerrosta koskevassa luvussa. TCP-protokollassa avataan yhteys, siirretään tietoa yhteyden ollessa auki ja päätetään siirto osapuolten keskinäisellä sopimuksella. UPD-protokolla ei edellytä erillistä yhteyden muodostamista ja ylläpitämistä, vaan sanomat siirretään tarvittaessa lähettäjältä vastaanottajalle.
* 4. Sovelluskerroksen yleisesti tunnettuja palveluja ovat esim. tiedonsiirto (FTP), päätekäyttö (Telnet) ja sähköposti (SMTP).