Virheiden havaitseminen ja korjaus tiedonvaihdossa

LuK-tutkielma
Jarno Kaikkonen
2584317
Matemaattisten tieteiden laitos
Oulun yliopisto
Syksy 2020

Sisältö

Johdanto

Jo kauan ennen tietokoneita ihmisten välinen viestiminen niin lähi- kuin kaukoetäisyyksillä on ollut erittäin tärkeää. Viestien välittämisen tärkeyden lisäksi myös niiden ymmärtäminen ja oikein tulkitseminen on olennaista. Ihmisten käyttämä kieli jo itsessään tuottaa omia ongelmiaan viestien välittämisessä ja ymmärtämisessä mm. erilaisten kielten ja kulttuurin, sarkasmin, kontekstin ja tietynlaisten oletuksien myötä.

Kun tiedonvälitys siirtyi puheesta käsinkirjoitukseen, ongelmilta silti pääosin vältyttiin, sillä kirjoitusvirheet käsin kirjoittaessa ovat harvinaisempia. Kirjoitetun tekstin lukeminen voi kuitenkin tuottaa ongelmia ja johtaa väärinymmärryksiin, koska vastaanottava osapuoli ei saa käsialasta selvää tai tulkitsee tiettyjä kirjoitustapoja eri tavalla (Esim. pieni l ja iso I, numeron 1 ja 7, jne.).

Käsinkirjoituksesta konekirjoitukseen siirryttäessä ongelmat kuitenkin alkavat lisääntyä, sillä nyt yksinkertainen etusormen ohilyönti yhden senttimetrin verran johtaa viestissä kokonaisen kirjaimen, numeron tai välimerkin muutokseen. Erityisesti numeroiden kanssa tämä voi aiheuttaa suuriakin ongelmia, jos konekirjoituksen tehtävä on määrätä esimerkiksi rahansiirto tai jonkin omaisuuden määrällistä myyntiä tai ostoa. Konekirjoitusvirheestä johtuvia lipsahduksia on tapahtunut esimerkiksi vaalien äänenlaskussa.

Modernissa maailmassa tieto siirretään yleensä tietokoneita käyttäen. Ihmiset puhuvat toisilleen sähköpostin, pikaviestisovelluksien ja muiden sovelluksien avulla, mutta tärkeämpää tämän tutkielman kannalta on miten esimerkiksi nämä ihmisten päätelaitteet – tietokoneet, näppäimistöt, näytöt ja muut laitteet – vaihtavat tietoa toistensa kanssa.

Kun esimerkiksi näppäimistön painikkeita painetaan, ei tämän tiedon välittyminen tietokoneeseen ole suinkaan itsestäänselvyys. Jokaista näppäimistön painiketta ei voida kytkeä yksittäiseen sähköjohtoon, sillä perinteisessä näppäimistössä on yli 100 painiketta, ja jos jokaiselle painikkeelle olisi oma johdin, olisi näppäimistöstä tietokoneeseen tuleva johtokimppu erittäin suuri, painava ja kallis. Tilanne olisi tuhansia kertoja pahempi, jos esimerkiksi tietokonenäytön jokaisen pikselin punainen, vihreä ja sininen signaali siirrettäisiin omassa johdossaan – perinteinen Full-HD -näyttö vaatisi tällöin vähintään $1920 \cdot 1080 \cdot 3 = 6220800$ johdinta.

Ongelma on ratkaistu erilaisilla sarja- ja rinnakkaiskommunikaatioprotokollilla, joissa näppäinpainallukset, pikseliarvot ynnä muut välitetään vain muutamaa johdinta käyttäen, lähettäen tietynlaisia sarjoja korkeaa ja matalaa jännitettä johtimiin – niin kutsuttuja "ykkösiä ja nollia"

Sähkölaitteissa ja niiden ympäristössä on kuitenkin usein paljon sähkömagneettista aktiivisuutta, josta esimerkiksi johtimissa voi syntyä magneet-

tikenttiä, jotka vaikuttavat muihin johtimiin indusoimalla niihin ei-haluttuja jännitteitä. Lopputuloksena lähetetty viesti ei välttämättä "saavu perille"samanlaisena, kun se on lähetetty.

Elektroniikan komponenttien ja laitteiden välillä viestien perille pääsy virheettömänä on kuitenkin erityisen tärkeää, joten näitä sarja- ja rinnakkaiskommunikaatiomenetelmiä on pyritty tekemään mahdollisimman virhekestäviksi alusta alkaen.

Tämä tutkielma käsittelee matematiikan kannalta tärkeitä virheentunnistus- ja korjausmenetelmiä näissä tiedonsiirtomuodoissa, käsitellen nimenomaan ykkösien ja nollien sarjoja. Näitä esimerkkiviestejä tutkitaan, ja selvitetään mitä ominaisuuksia niissä voidaan käyttää hyväksi virheiden tunnistamisessa ja korjaamisessa.

1 Viestit

Määritelmä 1.1. Määritellään nyt *viesti*, eli lukujono¸

$$M = \{m_1, m_2, ..., m_i\}, m_n \in \{0, 1\}, n, i \in \mathbb{N} \tag{1}$$

Joka ilmaistaan muodossa $m_1m_2m_3m_4...m_n$ asettamalla peräkkäiset arvot peräjälkeen, muodostaen ikään kuin 2-kantaisen luvun, jossa on i:n verran numeroita, lisäten tarvittaessa etunollia.

Esimerkki 1.2. Lukujono

$$M = \{1, 0, 1, 0, 1, 1, 0, 0\} \tag{2}$$

Ilmaistaan muodossa:
10101100

Esimerkki 1.3. Lukujono

$$M = \{0, 0, 1, 0, 1, 0, 0, 1\} \tag{3}$$

Ilmaistaan muodossa:
00101001

Määritelmä 1.4. Viestin M *pituus* on siihen sisältyvien lukujen määrä, ja se ilmaistaan $/M/$.

Esimerkki 1.5. Viestin $M = 101101$ pituus $/M/ = 6$.

1.1 Virheet

Oletetaan tiedonsiirtotapahtuma, jossa A lähettää viestin M ja B vastaanottaa tämän viestin.
A:n lähettämä viesti on nyt M^A, ja B:n vastaanottama viesti on M^B.
Viestin välitykseen käytetään virhealtista kanavaa, joten ei voida olettaa $M^A = M^B$.

Määritelmä 1.6. Vastaanotettu viesti M^B on *virheetön* jos ja vain jos:

1. $M_i^B = M_i^A \ \forall \ i \in \mathbb{N}$
2. $|M^B| = |M^A|$

Virheetön vastaanotettu viesti on siis täydellinen kopio lähetetystä viestistä, $M^B = M^A$.

Määritelmä 1.7. Vastaanotettu viesti M^B on *virheellinen*, jos ja vain jos se ei ole virheetön.

Seuraus 1.8. *Kaikki vastaanotetut viestit ovat joko virheettömiä tai virheellisiä*

Tutkitaan määritelmän 1.1 mukaista viestiä.
Huomataan, että viestiä voidaan muuttaa kahdella tavalla:

- Muuttamalla yhtä tai useampaa yksittäistä numeroa
- Muuttamalla viestin pituutta lisäämällä tai poistamalla numeroita

Huomautus 1.9. Koska viesti esitetään 2-kantaisena, voi yksittäistä numeroa muuttaa vain yhdellä tapaa.

Seuraus 1.10. *Jos viestin* M *numeroa* m_n *muutetaan kaksi kertaa, palautuu se ennalleen.*

Todistus. Olkoon $m_n = 0$. Selvästi m_n:llä on vain kaksi mahdollista arvoa: 0 ja 1. Muutetaan m_n, jolloin sen arvo on 1. Selvästi jos m_n muutetaan nyt toisen kerran, on taas $m_n = 0$. Sama pätee, jos $m_n = 1$. □

Seuraus 1.11. *Jos viestin* M *numeroa* m_n *muutetaan* ***parillinen määrä kertoja****, palautuu se ennalleen.*

Todistus. Kuten 1.10: Alku- ja loppupisteet ovat samat, joten kaksi muutosta voi tehdä peräkkäin loputtomiin muuttamatta arvoa. □

2 Virheen tunnistus

Jotta viestissä voidaan tunnistaa virheitä ilman kontekstin tapaista ulkopuolista tietoa, tulee kaikki tieto virheentunnistusta varten välittää viestin mukana. Samalla, kun haluttu informaatio välitetään viestin avulla, lähetetään myös tietoa itse viestistä, jonka avulla voidaan havaita, saapuiko viesti perille ehjänä.

Tietty osa viestistä täyttyy nyt ns. *redundanssitiedolla (engl. redundancy)* R).

Määritelmä 2.1. Redundanssitiedon ja viestin sisällön välistä suhdetta $\frac{|M|}{|R|+|M|}$ kutsutaan viestin *tehokkuudeksi*.

Lisäksi virheentunnistuksen tehokkuutta voidaan mitata esimerkiksi kysymyksillä:

1. Kuinka monta virhettä voidaan tunnistaa?
2. Kuinka monta virhettä tarvitaan virheelliseen tulokseen?
3. Kuinka raskaita tunnistuslaskut ovat suorittaa?

2.1 Triviaaliratkaisuja

Hyvin perustavanlaatuisia ratkaisuja viestin redundanssitiedolle on monia, mutta nämä ratkaisut kärsivät mm. huonosta viestin tehokkuudesta, pienestä virheentunnistusmäärästä ja hitaudesta.

2.1.1 Toisto

Yksinkertainen tapa saada viesti perille sellaisenaan on lähettää se useita kertoja. Tätä tapaa käytetään paljon myös ihmisten välissä kommunikoinnissa: Esimerkiksi lentoliikenteessä välittömän hengenvaaran "Mayday-hätämerkki ja sen lievempi ja vähemmän tunnettu vastine "Pan pan"tavataan sanoa kolme kertaa peräkkäin, jotta kaikki merkin kuulijat voivat olla varmoja, että kyseessä on aito hätätilanne eikä esimerkiksi väärin kuultu sana tai lause.

"Mayday, mayday, mayday"

Matematiikassa toistoa käytetään niin, että vastaanotetuista viesteistä tarkkaillaan jokaista yksittäistä numeroa ja verrataan niitä keskenään. Jos eroavaisuuksia löytyy, korjataan ne enemmistöperiaatteella, eli esimerkiksi

kolminkertaisen viestin tapauksessa oikeaksi numeroksi valitaan se, jota on kaksi tai enemmän.

Toistetulla viestillä on muihin tapoihin verrattuna kovin huono määritelmän 2.1 mukainen *tehokkuus*: $\frac{1}{n+1}$, missä n on toistojen määrä.

Esimerkki 2.2. Lähetetään viesti 11010 kolme kertaa: 110101101011010, vastaanottaja tietää jakaa viestin kolmeen osaan ja verrata niitä keskenään.

$$\begin{matrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{matrix}$$

Kaikki sarakkeet vastaavat toisiaan, viesti on virheetön.

Esimerkki 2.3. Lähetetään viesti 11010 kolme kertaa: 110101101011010. Viestin välityksessä ilmenee virheitä.

$$\begin{matrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{matrix}$$

Ensimmäisessä sarakkeessa viestien välillä on eroavaisuuksia, mutta ykkösiä on enemmän, joten oletetaan $m_1 = 1$. Toinen rivi on virheetön, kolmannessa eroavaisuus, nollia eniten joten oletetaan $m_3 = 0$. Korjattu viesti on M = 11010, eli viesti on korjattu onnistuneesti.

Huomautus 2.4. Pahimmassa tapauksessa virheitä tarvitsee esiintyä vain kaksi, jotta alkuperäinen viesti menetetään. Jos virhe ilmenee samassa kohti toistojen kesken enemmän kuin yhden kerran, ei virheenkorjaus onnistu. Tapa tunnistaa siis vähintään kaksi virhettä (kolmella virheellä muutos saadaan näkymättömäksi), ja korjaa vähintään yhden mutta mahdollisesti jopa koko viestin.

Esimerkki 2.5. Lähetetään viesti 11010 kolme kertaa: 110101101011010. Viestin välityksessä ilmenee virheitä.

$$\begin{matrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{matrix}$$

Ensimmäisessä sarakkeessa viestien välillä on eroavaisuuksia, ja valitaan $m_1 = 0$. Nyt vastaanotettu viesti ei vastaa lähetettyä viestiä, joten virheenkorjaus epäonnistuu.

2.1.2 Pituus

Sovitaan viestin pituudesta etukäteen, jolloin vastaanottaja tietää jo valmiiksi, minkä pituista viestiä odottaa. Koska tietokoneet käsittelevät informaatiota tyypillisesti tavuina, joiden pituus on 8 bittiä, käytetään tätä virheentunnistusta käytännössä miltei aina. Tietoyhteyteen voidaan sopia esimerkiksi tietty aikakatkaisuaika, jonka vastaanottaja odottaa, ja jos kokonaista tavua ei ole siirretty ennen aikakatkaisua, tieto hylätään ja vastaanottaja odottaa seuraavaa kokonaista tavua.

Tyypillistä esimerkiksi internetin UDP -paketeissa on myös, että lähetetyn viestin alkupäässä jotkut ensimmäisistä tavuista kertovat, kuinka monta tavua on vielä tulossa. Näin vastaanottava laite tai ohjelma osaa odottaa tiettyä määrää dataa, ja jos sitä tulee enemmän tai vähemmän kuin aiemmin määrätty, tieto hylätään. Huomattavaa on kuitenkin, että tällainen tiedon hylkääminen johtaa *pakettihäviöön*, joten se ei sovi tilanteisiin, joissa jokaisen paketin on ehdottoman tärkeää päästä perille.

Pakettihäviötä voidaan vähentää esimerkiksi kaksisuuntaisella tiedonvaihdolla, jolloin vastaanottaja voi tarvittaessa kertoa lähettäjälle, ettei äskeinen paketti tullut perille. Tällöin käytetään myös tyypillisesti jotain pakettien numerointijärjestelmää, jotta lähettäjä voi uudelleenlähettää juuri oikean paketin.[2]

2.1.3 Parillisuusbitti

Kuten huomautuksessa 1.9 kerrottiin, yhtä numeroa voi viestissä vaihtaa vain yhdellä tavalla. Voidaan siis laskea viestin M summa:

Määritelmä 2.6. Viestin M *summa* on viestin M alkioiden, eli yksittäisten *bittien* summa:

$$\sum_{i=1}^{|M|} m_i \tag{4}$$

Tämä summa on joko parillinen tai pariton. Selvästi muuttamalla yhtä bittiä viestissä sen summa joko kasvaa tai nousee yhdellä, näin ollen muuttaen viestin parillisuuden joko parillisesta parittomaksi tai toisin päin.

Otetaan käyttöön *parillisuusbitti*, ja määritellään se viestin ensimmäiseksi bitiksi. Viestin lähettäjä asettaa tämän parillisuusbitin siten, että viestin M summa on sovitun mukaan joko aina pariton tai parillinen.

Huomautus 2.7. Yleinen käytäntö on, että parillisuusbitti ylläpitää parillisuutta, ei parittomuutta.

Esimerkki 2.8. Välitetään 7 bittiä pitkä viesti M = 1010001 käyttäen parillisuusbittiä ensimmäisenä bittinä.

Lasketaan viestin M summa:

$$\sum M = 1 + 0 + 1 + 0 + 0 + 0 + 1 = 1 + 1 + 1 = 3$$

Viesti on sellaisenaan pariton, joten määritetään viestin parillisuusbitti arvoon 1, jolloin summa $3 + 1 = 4$ on parillinen.

Parillisuusbittiä käyttävä viesti on nyt:

$$M^P = 11010001$$

Ja sen summa on

$$\sum_{i=1}^{8} m_i = 1 + 1 + 0 + 1 + 0 + 0 + 0 + 1 = 4$$

Joka on parillinen luku.

Viestin vastaanottaja voi nyt tarkistaa viestin eheyden laskemalla vastaanotetun viestin summan. Jos summa on parillinen, viesti on ehyt.

Parillisuusbitti on erityisen tärkeä ja olennainen osa virheentunnistusta, mutta se ei kerro lainkaan, missä kohtaa virhe on tapahtunut taikka miten sen voi korjata. Parillisuusbitin avulla voidaan myös tunnistaa varmasti vain yksi virhe, sillä kaikki parilliset virheiden määrät tekevät vastaanotetusta viestistä parillisen.

3 Virheiden paikannus

Edellisessa kappaleessa on nyt käyty läpi yksinkertaisia ratkaisuja virheiden havaitsemiselle. Jos tiedonsiirrossa on mahdollisuus pyytää data uudelleen virheen ilmetessä, tämä voi jo riittää:

1. A lähettää B:lle viestin
2. B vastaanottaa viestin, mutta bittien summa on pariton. Viestissä on virhe.
3. B pyytää A:ta lähettämään viestin uudelleen. Viestillä voi olla esimerkiksi tietty nimi tai numero, jonka perusteella A tietää, mistä viestistä on kyse.
4. A vastaanottaa pyynnön uudesta viestistä, ja hakee jo lähetetyn viestin tietokannasta sen tunnisteen avulla.

5. A lähettää edellisen viestin uudestaan

B:n ei tarvitse vahvistaa viestin vastaanottoa, sillä A tietää, että jos B ei kerro virheestä, viesti on vastaanotettu onnistuneesti.

Tällaiselle tiedon uudelleenlähettämiselle ei kuitenkaan usein ole mahdollisuutta, esimerkiksi jos tiedonsiirto on vain yhdensuuntaista, liian nopeaa tai jos tiedon on ehdottoman tärkeää ola reaaliaikaista, eli vastaanotettu data vanhenee, ennen kuin sitä ehditään uudelleenlähettää. (Triviaaliesimerkinä viestinvälitys, jossa A kertoo B:lle tämänhetkisen ajan mikrosekunnin tarkkuudella joka mikrosekunti)

Tällaisiin tilanteisiin tarvitaan tapa paikantaa, missä virhe on tapahtunut. Jos saadaan selville, mikä bitti viestissä on vaihtunut, tiedämme seurauksen 1.10 mukaan, että bitti voidaan korjata oikeaksi vaihtamalla se kerran.

3.1 Paikannus parillisuusbittien avulla

Käyttäen aliluvun 2.1.3 ratkaisua virheen tunnistukselle, *parillisuusbittiä*, voidaan virhettä myös paikantaa asettamalla lisää parillisuusbittejä.

Ajatellaan esim. 4-bittistä viestiä: viestistä voidaan tunnistaa yksi virhe asettamalla viides bitti parillisuusbitiksi koko viestin suhteen. Jos kuitenkin asetetaan kaksi parillisuusbittiä, voidaan ne asettaa esimerkiksi vain puolelle biteistä, jolloin parillisuusbitti 1 asetetaan ensimmäisen 2 viestibitin mukaan, ja parillisuusbitti 2 asetetaan 2 jälkimmäisen parillisuusbitin mukaan. Tällöin jos viestissä ilmenee virhe, tiedetään heti kummalla puolella virhe on tapahtunut. Myös molempiin puoliin voi sattua 1 virhe, ja molemmat havaitaan. Mitään virhettä ei kuitenkaan voida vielä paikantaa tasan yhteen bittiin, mutta tämä voidaan saada aikaan kolmannella parillisuusbitillä:

Asetetaan parillisuusbitti 1 vastaamaan koko viestistä, mukaan lukien muut parillisuusbitit. Asetetaan sitten parillisuusbitti 2 vastaamaan ensimmäisestä kahdesta viestibitistä, ja parillisuusbitti 3 vastaamaan 2. ja 4. viestibitistä, eli parillisen indeksinumeron viestibiteistä. Esimerkki tällaisesta viestistä olisi:

$$0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0$$

1. parillisuusbitti on nyt 0, koska muiden bittien summa on jo parillinen. 2. parillisuusbitti on 1, koska ensimmäinen puolisko olisi muutoin pariton:

$$0 \quad \mathbf{1} \quad 1 \quad \mathbf{0} \quad \mathbf{1} \quad 1 \quad 0$$

3. parillisuusbitti on 1, koska parillisien indeksien viestibittien summa olisi muutoin pariton:

0 1 **1** 0 **1** 1 **0**

Näin ollen vain kolmella redundanssibitillä saadaan paikannettua ja korjattua 1 bitin virhe. Tällaisen 4-bittisen viestin *tehokkuus* on tällöin kaavan 2.1 mukaan $\frac{4}{7} \approx 57\%$
Yllä oleva virheentunnistustapa on *Richard Hammingin* 1940-luvulla kehittämä *Hamming -koodi*ksi kutsuttu virheenkorjausmenetelmä.

3.2 Hamming -koodi

Hamming -koodi on koodattu viesti, joka sisältää edellisessä osiossa määriteltyjä parillisuusbittejä jokaisessa bitissä, jonka indeksi on jokin kahden potenssi. Nämä parillisuusbitit käsittävät viestin eri osia systemaattisesti, jonka lopputuloksena viestissä ilmennyt virhe voidaan paikantaa tarkasti yhteen bittiin.

Hamming -koodit nimetään niiden viestibittien ja parillisuusbittien perusteella *Hamming(t,m)* -koodeiksi, missä m on viestibittien määrä ja t kaikkien bittien yhteismäärä. Näin ollen esimerkiksi edellisessä osiossa kehitetty koodi on Hamming(7,4) -koodi.

Tässä osiossa määrittelemme Hamming(15,11) -koodin tutkittavaksi.

Määritelmä 3.1. Jatkoa varten määritellään nyt viesti M matriisiksi, jonka koko on $n \times m,\ n, m \in \mathbb{N}$:

$$\begin{matrix} m_0 & m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 & m_7 \\ m_8 & m_9 & m_{10} & m_{11} \\ m_{12} & m_{13} & m_{14} & m_{15} \end{matrix}$$

Asetetaan nyt viestin sisältö alkioihin $m_i,\ i \in \mathbb{N}\backslash\{0, 2^k\},\ k \in \mathbb{N},\ i \leq n \times m$. Viestit, joiden kokonaispituus on suurempi kuin $n \times m$, voidaan jakaa useampaan matriisiin.

Esimerkki 3.2. Viesti M = 10100101011 yllä määritellyssä 4x4 -matriisissa:

$$\begin{matrix} m_0 & m_1 & m_2 & 1 \\ m_4 & 0 & 1 & 0 \\ m_8 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{matrix}$$

Määrätään nyt esimerkin 3.2 matriisiin Hamming -koodin parillisuusbitit:

Määritelmä 3.3. Asetetaan m_1 käsittämään parittoman indeksin bittejä. Siis jos parittoman indeksin bittien summa on pariton, asetetaan m_1 arvoon 1, jolloin niiden summa muuttuu parilliseksi:

$$\begin{array}{cccc} m_0 & \mathbf{0} & m_2 & 1 \\ m_4 & 0 & 1 & 0 \\ m_8 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{array}$$

m_2 käsittää "oikean puolen"bitit, eli kahdeksan oikeanpuolimmaista bittiä:

$$\begin{array}{cccc} m_0 & 0 & \mathbf{1} & 1 \\ m_4 & 0 & 1 & 0 \\ m_8 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{array}$$

m_4 puolestaan käsittää toisen ja neljännen rivin bitit:

$$\begin{array}{cccc} m_0 & 0 & 1 & 1 \\ \mathbf{0} & 0 & 1 & 0 \\ m_8 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{array}$$

Ja viimeinen parillisuusbitti, m_8 käsittää alemman puolikkaan bitit, eli kolmannen ja neljännen rivin:

$$\begin{array}{cccc} m_0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ \mathbf{0} & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{array}$$

Kun m_0 jätetään huomiotta, ollaan nyt määritelty *Hamming(15,11)*-koodi.

Lause 3.4. *Hamming(15,11)-koodilla voidaan havaita ja paikantaa yhden bitin muuttuminen missä tahansa bitissä.*

Todistus. Olkoon bitti m_i, $i \in \mathbb{N}\backslash\{0,1,2,4,8\}$ Hamming(11,15) -koodissa virheellinen. Bitin sarake voidaan löytää parillisuusbitteillä m_1 ja m_2:

Jos viestin parittomien bittien summa on parillinen, on virheen oltava parillisessa indeksissä, eli sarakkeessa 1 tai 3. Jos nyt viestin oikea puolen bittien summa on parillinen, on virheen oltava sarakkeessa 1. Muutoin virhe on sarakkeessa 3.

Jos viestin parittomien bittien summa on pariton, on virhe jossain näistä parittomista biteistä, eli sarakkeessa 2 tai 4. Nyt taas jos oikean puolen bittien summa on parillinen, on virheen oltava sarakkeessa 2. Muutoin virhe on sarakkeessa 4.

Vastaavasti voidaan määrittää virheen rivi: Jos viestin 2. ja 4. rivien bittien summa on parillinen, on virheen oltava rivillä 1 tai 3. Jos nyt viestin

alemman puoliskon bittien summa on parillinen, on virheen oltava rivillä 1. Muutoin virhe on rivillä 3.

Jos viestin 2. ja 4. rivien bittien summa on pariton, on virhe jollain näistä riveistä. Jos viestin alemman puoliskon bittien summa on parillinen, on virheen oltava rivillä 2. Muutoin virhe on rivillä 4.

Koska virheen sarake ja rivi saadaan näillä tiedoilla selville, on virheellinen bitti nyt yksiselitteisesti määritetty. □

Esimerkki 3.5. Paikannetaan virhe seuraavasta viestistä: 11101110011

$$\begin{array}{cccc} m_0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{array}$$

Parillisuustarkastus 1: Onko 2. ja 4. sarakkeiden bittien summa parillinen?

$$\begin{array}{cccc} m_0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{array}$$

$$1+1+1+0+1+1+0+1 = 6 \in 2\mathbb{Z}$$

Parillisuustarkastus 2: Onko 3. ja 4. sarakkeiden bittien summa parillinen?

$$\begin{array}{cccc} m_0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{array}$$

$$0+1+1+0+1+1+0+1 = 5 \notin 2\mathbb{Z}$$

Parillisuustarkastus 3: Onko 2. ja 4. rivien bittien summa parillinen?

$$\begin{array}{cccc} m_0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{array}$$

$$0+1+1+0+0+0+0+1 = 3 \notin 2\mathbb{Z}$$

Parillisuustarkastus 4: Onko 3. ja 4. rivien bittien summa parillinen?

$$\begin{array}{cccc} m_0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{array}$$

$$1+1+1+1+0+0+0+1=5\notin 2\mathbb{Z}$$

Lauseen 3.4 todistuksen tapaan näemme nyt, että:

1. Virhe ei ole sarakkeessa 2 tai 4
2. Virhe on sarakkeessa 3 tai 4

   Virhe on siis sarakkeessa 3
3. Virhe on rivillä 2 tai 4
4. Virhe on rivillä 3 tai 4

   Virhe on siis rivillä 4

Virhe on siis sarakkeessa 3, rivillä 4. Viestin alkio m_{14} on vaihtunut:

m_0	1	0	1
0	1	1	0
1	1	1	1
0	0	**0**	1

Korjattu viesti on tällöin:

m_0	1	0	1
0	1	1	0
1	1	1	1
0	0	1	1

Eli 11101110011

Määritelmä 3.6. Olkoon parillisuustarkastuksen totuusarvo 1, jos summa ei ole parillinen. Olkoot se muutoin 0.

Huomautus 3.7. Kun parillisuusbittien vaikutusalat valitaan näin, saadaan parillisuustarkastuksien ns. *totuusarvoista* suoraan virheen sijainti 2-kantaisena numerona.

Esimerkin 3.5 totuusarvot olisivat tällöin 0 1 1 1.

$$0\times 2^0+1\times 2^1+1\times 2^2+1\times 2^3=0+2+4+8=14$$

Virhe oli alkiossa m_{14}.

3.3 0. indeksin käyttöönotto

Kun 4x4 -matriisi määritellään osion 3.2 mukaan, jää "nollas"bitti m_0 käyttämättä. Hamming -koodia tutkiessa huomataan äkkiä, että koodi kykenee 100% varmuudella paikantamaan vain yhden virheen. Asetetaan 0-bitti määrittämään koko viestin parillisuuden.

Määritetään siis 4x4-matriisi, jossa bitit m_1:stä m_{15}:aan määritellään Hamming(15,11) -koodina, ja m_0 asetetaan siten, että koko matriisin alkioiden summa on parillinen.

Esimerkki 3.8. Asetetaan esimerkin 3.5 korjattuun matriisiin m_0 siten, että

$$\sum_{i=0}^{15} m_i = 2k,\ k \in \mathbb{Z}$$

Summataan kaikki alkiot:

$$1+0+1+1+1+1+0+1+1+1+1+0+0+1+1 = 11$$

Täytyy siis määrittää $m_0 = 1$, jotta summa on parillinen. Valmis, kokonaan täytetty matriisi on nyt:

$$\begin{matrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{matrix}$$

Lause 3.9. *Hamming -koodin kaksibittinen virhe johtaa vähintään yhteen parillisuustarkistuksen epäonnistumiseen*

Todistus. Olkoot Hamming(15,11) -koodissa kaksi virhettä.
Vastaoletus: Kaksibittinen virhe voi säilyttää parillisuudet ennallaan
Tutkitaan, mihin tarkistusbittiin mikäkin viestibitti vaikuttaa. Saadaan seuraavanlainen taulukko:

	m_1	m_2	$1,2$
m_4	$1,4$	$2,4$	$1,2,4$
m_8	$1,8$	$2,8$	$1,2,8$
$4,8$	$1,4,8$	$2,4,8$	$1,2,4,8$

Huomataan ristiriita: Viestissä ei ole kahta bittiä, jotka vaikuttaisivat tismalleen samoihin tarkistusbitteihin. Tämä tarkoittaa, ettei kahdella bitillä voida saada aikaan tulosta, jossa viesti vaikuttaa virheettömältä. □

Kahden bitin virhe voidaan kuitenkin edelleen tulkita väärin yhden bitin virheenä, ja kahden bitin virhe antaa virheellisen sijainnin virheelle. Tässä tilanteessa voidaan tarkistaa koko viestin tarkillisuusbitti m_0. Jos viesti on parillinen, mutta ainakin yksi parillisuustarkastuksista epäonnistuu, viestissä on parillinen määrä virheitä. Näitä virheitä ei voida paikantaa, mutta kaksi virhettä voidaan havaita 100% varmuudella.

Seuraus 3.10. *Hamming(15,11)-koodi ylimääräisellä parillisuusbitillä kykenee paikantamaan minkä tahansa yhden bitin virheen, ja tunnistamaan minkä tahansa kahden bitin virheen*

4 Lopputulos ja pohdintaa

Binääritiedonsiirtoon on nyt määritetty Hamming(15,11)-koodi ylimääräisellä parillisuusbitillä, joka kykenee paikantamaan yhden bitin virheet ja havaitsemaan kahden bitin virheet. Viestit voidaan nyt lähettää 16 bitin paketeissa, joka on tietokoneille erityisen kätevää, sillä tietokoneissa käytetään "tavu-tietoyksikköä, jonka koko on 8 bittiä. Yksi paketti on siis tasan kaksi tavua. Yli 11 bitin viestit voidaan välittää useammassa peräkkäisessä paketissa. Näin ollen esimerkiksi 110 bitissä dataa voi olla jopa 10 virhettä jotka kaikki vastaanottaja voi korjata.

Koodin tehokkuus kasvaa, kun sen kokoa kasvatetaan. Havainto- ja korjausominaisuudet pysyvät ennallaan, mutta viestibittien suhde redundanssibitteihin kasvaa. Koodimenetelmä on kuitenkin vanha, ja tehokkaampia menetelmiä on kehitetty myöhemmin. Esimerkiksi CD-levyt ja ADSL -modeemit käyttävät niin kutsuttuja Reed-Solomon -virheenkorjauskoodeja. Tyypillinen CD -levy voi sisältää jopa miljoona virhettä, mutta sen voi soittaa silti virheettömästi virheenkorjauksen avulla.[4]

Lähdeluettelo

[1] T. Hiramatsu, G. Köhler: *Coding Theory and Number Theory.* Springer-Science+Business Media, Dordrecht, 2003.

[2] H. Sawashima, Y. Hori, H. Sunahara: *Characteristics of UDP Packet loss: Effect of TCP Traffic.* http://www.isoc.org/INET97/proceedings/F3/F3_1.HTM (Viitattu 28.11.2020)

[3] 3Blue1Brown: *Hamming codes and error correction.* https://www.youtube.com/watch?v=X8jsijhllIA (Viitattu 28.11.2020)

[4] The Irish Times: *High fidelity: how the sound of CDs stays error-free.* https://www.irishtimes.com/news/science/high-fidelity-how-the-sound-of-cds-stays-error-free-1.2362987