

# **Konsensusongelma hajautetuissa järjestelmissä**

Niko Välimäki

Helsinki 29.10.2007

Seminaarityö

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

# Sisältö

<b>1 Johdanto</b>	<b>1</b>
<b>2 Konsensusongelma</b>	<b>2</b>
2.1 Ratkeamattomuustodistus . . . . .	3
<b>3 Satunnaistettu algoritmi</b>	<b>5</b>
3.1 Alaraja ristiriidan todennäköisyydelle . . . . .	11
<b>4 Yhteenveto</b>	<b>11</b>
<b>Lähteet</b>	<b>13</b>

# 1 Johdanto

*Konsensusongelmalla* [Gra78, Lyn96] tarkoitetaan yhteisen päätöksen muodostamista hajautetussa järjestelmässä. Päätöksen muodostaminen on helppoa, jos voidaan olettaa, että prosessien välinen viestinvälitys toimii aina moitteetta. Käytännössä, esimerkiksi hajautetuissa tietokantajärjestelmissä, tällaista oletusta ei voida tehdä, mutta prosessien muodostaman päätöksen tulisi silti olla yksimielinen. Tässä tutkielmassa tarkastellaan konsensusongelmaa järjestelmissä, joissa prosessien välinen viestinvälitys ei ole luotettava.

*Hajautettu järjestelmä* on suuntaamaton verkko  $G = (V, E)$ , jonka solmut  $V$  ovat *prosesseja*. Prosessit  $i \in V$  koostuvat tiloista  $tila_i \in tilat_i$  sekä alkutiloista  $alku_i$ . Prosessiin  $i \in V$  kaarilla  $E$  kytketyt prosessit merkitään naapureiksi  $naapurit_i$ . Naapuriprosessit voivat kommunikoida keskenään lähettämällä toisilleen *viestejä*. Viestien sisältö on rajoitettu aakkostoon  $\Sigma$  tai tyhjään viestiin  $\phi$ . Oletetaan lisäksi, että verkko on yhtenäinen.

Prosessin *tilasiirtymät* määritellään prosessin tilojen ja viestien avulla. Uusia viestejä lähettävät tilasiirtymät *lahetykset* määritellään funktioina, jotka kuvaavat jokaisen prosessin tilan  $tilat_i$  ja naapurin  $naapurit_i$  naapurille lähetettäväksi viestiksi  $\Sigma \cup \phi$ . Saapuvien viestien perusteella määritellään tilasiirtymät *saapuvat*, jotka kuvaavat saapuvan viestin  $\Sigma \cup \phi$  ja nykyisen tilan joukosta  $tilat_i$  prosessin uudeksi tilaksi  $tila_i$ .

Tilasiirtymien avulla prosessi voi viestiä naapureilleen omasta tilastaan sekä muuttaa omaa tilaansa saapuvien viestien perusteella. Järjestelmän oletetaan toimivan synkronisesti siten, että jokaisella *kierroksella* prosessit suorittavat seuraavat kaksi askelta ennen kuin seuraava kierros voi alkaa:

1. Lähetetään prosessin nykyisen tilan perusteella naapuriprosesseille viestit.
2. Vastanotetaan naapuriprosesseilta saapuvat viestit ja muutetaan niiden perusteella tarvittaessa omaa tilaa.

Suoritus etenee deterministisesti siten, että samoilla alkutiloilla päästään aina samaan *suoritukseen*, eli jonoon järjestelmän tiloja. Järjestelmän suoritusta kuvataan jonolla  $C_0, L_1, S_1, C_1, L_2, S_2, C_2, \dots$ , missä  $L_r$  on kaikki kierroksella  $r$  lähetetyt viestit ja  $S_r$  saapuneet viestit. Viestinvälitys prosessien välillä on virhealtis ja saattaa hukata viestejä, joten  $S_r$  ei sisällä välttämättä kaikkia  $L_r$ :n viestejä. Prosessien tilaa kierroksen  $r$  lopuksi merkitään  $C_r$ .

Kaksi erilaista suoritusta  $\alpha$  ja  $\alpha'$  voivat olla prosessin  $i \in V$  kannalta samat, jos prosessi  $i$  käy suorituksissa  $\alpha$  ja  $\alpha'$  läpi samat tilat sekä lähettää ja vastaanottaa samat viestit. Merkitään tällaisia suorituksia  $\alpha \stackrel{i}{\sim} \alpha'$ .

## 2 Konsensusongelma

Konsensusongelma [Gra78, Lyn96] voidaan ymmärtää koordinoitun hyökkäyksen järjestämisenä: kenraalit suunnittelevat hyökkäystä yhteistä vihollista vastaan, mutta taistelu on voitettavissa vain, jos kaikki kenraalit hyökkäävät yhtä aikaa. Kukin kenraaleista voi hyökätä vain, jos omat joukot ovat valmiina. Aluksi jokainen kenraali tietää vain omien joukkojensa tilanteen.

Kenraalien armeijat sijaitsevat toisistaan erillään siten, että kenraalien väliseen yhteydenpitoon on käytettävä viestinviejiä. Jos viestinvälitys kenraalien välillä olisi luotettava, koordinoitu hyökkäys olisi mahdollinen hyvin yksinkertaisesti tulvittamalla jokaisen kenraalin tilaa verkossa. Tilatieto saavuttaisi jokaisen kenraalin verkon läpimittaa vastaavassa määrässä askelia. Yhteinen päätös hyökkäyksestä voitaisiin muodostaa, jos kaikki kenraalit ilmoittaisivat olevansa valmiina. Jos yksikin kenraaleista ei voisi hyökätä, kaikki kenraalit peruisivat hyökkäyksen.

Oletetaan kuitenkin, etteivät reitit vierekkäin sijaitsevien armeijoiden välillä ole turvallisia. Kenraalien tulee saapuneiden viestien perusteella päättää, että hyökkääkö kenraalin armeija vai ei. Kaikkien kenraalien tulisi päätyä samaan lopputulokseen, koska muuten taistelu hävitään. Lisäksi kenraalien tulisi päättää hyökätä, jos taistelu on voitettavissa.

Tietojenkäsittelytieteessä konsensusongelma tulee esille esimerkiksi hajautetuissa tietokantajärjestelmissä. Kun tietokantaan tehdään muutoksia, tulee prosessien joko hyväksyä tai hylätä muutokset. Muutokset tulisi pyrkiä hyväksymään aina, kun mahdollista, ja hylätä virhetilanteissa, joissa jokin prosesseista ei hyväksy muutoksia. Tietokantojen sisältö pysyy yhtenäisenä vain, jos kaikki prosessit päätyvät yksimielisyyteen lopputuloksesta.

Formaalisti määriteltynä jokainen prosessi  $i \in V$  saa alkutilansa  $alku_i$  joukosta  $\{0, 1\}$ , missä 1 tarkoittaa hyväksyvää tilaa ja 0 hylkäävää tilaa. Määritellään seuraavat ehdot, joiden tulee täyttyä:

- (a) Jokaisen prosessin tulee tehdä lopullinen päätös,  $tulos_i \in \{0, 1\}$ , äärellisen monen kierroksen kuluessa.

- (b) Päätöksen tulee olla yksimielinen siten, että kaikki prosessit päätyvät samaan lopputulokseen  $tulos_i$ .
- (c) Kaikkien prosessien alkutilojen ollessa 0 myös kaikkien tulosten tulee olla 0.
- (d) Kaikkien prosessien alkutilojen ollessa 1 myös kaikkien tulosten tulee olla 1, jos kaikki viestit on toimitettu onnistuneesti.

Triviaali ratkaisu, jossa kaikki prosessit valitsevat tulokseksi aina 1, poissuljetaan ehdolla (c). Ratkaisu, jossa kaikki prosessit valitsevat tulokseksi aina 0, poissuljetaan ehdolla (d). Osoittautuu, että näinkin heikkoa variaatiota konsensusongelmasta on mahdotonta ratkaista edes kahden prosessin verkoille, jos viestinvälitys ei ole luotettava.

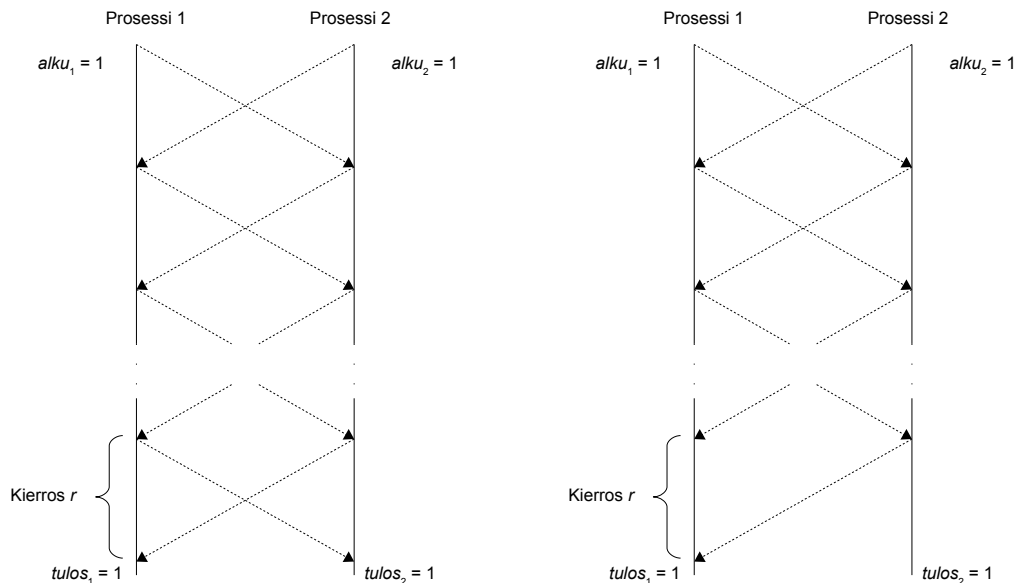
## 2.1 Ratkeamattomuustodistus

Konsensusongelma voidaan todistaa ratkeamattomaksi deterministisesti etenevässä suorituksessa [Gra78, Lyn96]. Ongelma osoittautuu ratkeamattomaksi jopa kahden solmun verkolle. Ratkeamattomuus voidaan edelleen yleistää kaikille  $n$  prosessin verkoille.

Tehdään vastaoletus, että on olemassa algoritmi  $A$ , joka ratkaisee konsensusongelman kahdesta prosessista koostuvassa verkossa. Olkoon molempien prosessien alkutila  $alku_1 = alku_2 = 1$  ja suoritus  $\alpha$  sellainen, että kaikki viestit prosessien välillä toimitetaan onnistuneesti. Edellä kuvatun ehdon (d) mukaisesti molempien prosessien tulee lopulta valita  $tulos_1 = tulos_2 = 1$ . Valitaan  $r$  siten, että molemmat prosessit tekevät päätöksensä  $r$  kierroksen kuluessa — ehdon (a) mukaan  $r$  on olemassa, koska prosessit valitsevat päätöksensä äärellisen monen kierroksen kuluessa.

Olkoon suoritus  $\alpha_1$  muuten sama kuin  $\alpha$ , mutta kierroksen  $r$  jälkeen prosessien välinen viestinvälitys lopettaa toimintansa. Koska molemmat prosessit valitsevat tuloksensa  $r$  kierroksen kuluessa, tämä ei vaikuta lopputulokseen: suorituksen  $\alpha_1$  tulos  $tulos_1 = tulos_2 = 1$  on sama kuin suorituksen  $\alpha$ .

Tarkastellaan kierrosta  $r$  ja prosessien tilaa kierroksen  $r$  lopuksi. Olkoon suoritus  $\alpha_2$  muuten sama kuin  $\alpha_1$ , mutta prosessin 1 lähettämä viesti kierroksella  $r$  hukkuu. Koska prosessi 1 ei koskaan saa tietää viestinsä kadonneen, päättyy se samaan tulokseen kuin suorituksessa  $\alpha_1$ . Prosessi 2 ei koskaan vastaanota kierroksen  $r$  viestiä, mutta sen on ehdon (b) mukaan päädyttävä samaan lopputulokseen kuin prosessin 1. Toisin sanoen molemmat prosessit päätyvät suorituksessa  $\alpha_2$  tulokseen 1.



Kuva 1: Vasemmalla esimerkki suorituksesta  $\alpha_1$  ja oikealla suorituksesta  $\alpha_2$ . Onnistuneesti toimitetut viestit on merkitty katkoviivalla ja nuolella.

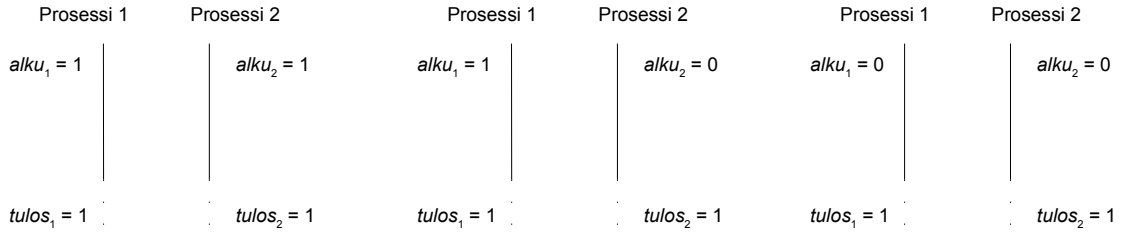
Kuvassa 1 on esimerkki suorituksista  $\alpha_1$  ja  $\alpha_2$ . Suoritus etenee ylhäältä alaspäin siten, että prosessien välinen viestinvaihto on merkitty katkoviivalla ja nuolella. Suoritukset  $\alpha_1$  ja  $\alpha_2$  ovat prosessin 1 kannalta samat  $\alpha_1 \stackrel{1}{\sim} \alpha_2$ , koska prosessi 1 käy suorituksissa  $\alpha_1$  ja  $\alpha_2$  läpi samat tilat sekä lähettää ja vastaanottaa samat viestit.

Tarkastellaan suoritusta  $\alpha_3$ , joka eroaa suorituksesta  $\alpha_2$  vain siten, että prosessin 2 kierroksella  $r$  lähettämä viesti hukkuu. Prosessi 2 ei tiedä viestin hukkuneen, joten suoritukset  $\alpha_2$  ja  $\alpha_3$  ovat prosessin 2 kannalta samat  $\alpha_2 \stackrel{2}{\sim} \alpha_3$ . Koska  $\alpha_2 \stackrel{2}{\sim} \alpha_3$ , prosessi 2 päättyy suorituksessa  $\alpha_3$  samaan lopputulokseen  $tulos_2 = 1$  kuin suorituksessa  $\alpha_2$ . Prosessin 1 kannalta tämä tarkoittaa sitä, että ehdon (b) nojalla myös prosessi 1 päättyy tulokseen 1 suorituksessa  $\alpha_3$ .

Tähän mennessä olemme siis osoittaneet, kuinka prosessien välisten viestien hukkaaminen kierroksella  $r$  ei näytä vaikuttavan lopputulokseen  $tulos_1 = tulos_2 = 1$ , jos on olemassa vastaoletuksen mukainen algoritmi  $A$ . Suoritusten  $\alpha_1$  ja  $\alpha_3$  ainut ero on se, etteivät kierroksella  $r$  lähetetyt viestit saavu vastaanottajalle.

Jatkamalla edellä kuvatulla tavalla voidaan viestinvälitys lopettaa jokaisella kierroksella  $r-1, r-2, \dots$  ilman, että lopputulos muuttuu. Lopulta päädytään suoritukseen  $\alpha'$ , jossa prosessit eivät vastaanota yhtään viestiä toisiltaan, mutta lopputuloksena on silti  $tulos_1 = tulos_2 = 1$ .

Prosessien alkutilat  $alku_1 = alku_2 = 1$  ovat olleet tähän asti samat. Muodostetaan



Kuva 2: Esimerkkejä suorituksista, kun viestinvälitys ei toimi. Suoritukset lueteltuna vasemmalta oikealle:  $\alpha'$ ,  $\alpha''$  ja  $\alpha'''$ .

suoritus  $\alpha''$ , joka vastaa suoritusta  $\alpha'$ , jossa prosessin 2 alkutila on  $alku_2 = 0$ . Koska edelleen  $alku_1 = 1$  eikä yksikään viesti saavuta prosessia 1, on  $\alpha' \stackrel{1}{\sim} \alpha''$  ja prosessin 1 valitsema lopputulos  $tulos_1 = 1$ . Ehdon (b) mukaan myös prosessi 2 valitsee  $tulos_2 = 1$ .

Olkoon suoritus  $\alpha'''$  sama kuin  $\alpha''$ , mutta nyt myös prosessin 1 alkutila on  $alku_1 = 0$ . Koska yksikään viesti ei saavuta prosessia 2, muutos ei vaikuta prosessin 2 suoritukseen, joten  $\alpha'' \stackrel{2}{\sim} \alpha'''$  ja näin ollen  $tulos_2 = 1$ . Jälleen ehdon (b) perusteella prosessin 1 täytyy valita  $tulos_1 = 1$ .

Kuvassa 2 on esimerkki suorituksista  $\alpha'$ ,  $\alpha''$  ja  $\alpha'''$ . Kumpikaan prosesseista ei vastaanota viestejä, mutta molemmat valitsevat lopputuloksensa  $r$  kierroksen kuluessa.

Suorituksessa  $\alpha'''$  päädytään vihdoin ristiriitaan: prosessien 1 ja 2 alkutilat olivat  $alku_1 = alku_2 = 0$ , mutta lopputulokset  $tulos_1 = tulos_2 = 1$ . Ristiriita seuraa ehdosta (c), joka ei täyty suorituksessa  $\alpha'''$ .

### 3 Satunnaistettu algoritmi

Konsensusongelma  $n$  prosessin verkolle  $G = (V, E)$  voidaan ratkaista *satunnaistetulla algoritmilla*, kun sallitaan *virhetilanne* todennäköisyydellä  $\epsilon$  [VL92, Lyn96]. Virhetilanteessa osa prosesseista valitsee tuloksen 0 ja loput tuloksen 1. Oletetaan, että jokainen prosessi tekee lopullisen päätöksensä,  $tulos_i \in \{0, 1\}$ ,  $r \geq 1$  kierroksen kuluessa. Satunnaistettu algoritmi ratkaisee konsensusongelman  $r$  kierroksen kuluessa todennäköisyydellä  $1 - \epsilon$  siten, että seuraavat ehdot ovat voimassa:

(a)  $Pr^B[\exists i, j \in [1, n] : tulos_i = 0 \text{ ja } tulos_j = 1] \leq \epsilon$ .

(b) Kaikkien prosessien alkutilojen ollessa 0 myös kaikkien tulosten tulee olla 0.

- (c) Kaikkien prosessien alkutilojen ollessa 1 myös kaikkien tulosten tulee olla 1, jos kaikki viestit on toimitettu onnistuneesti.

Todennäköisyys  $Pr^B[\exists i, j \in [1, n] : tulos_i = 0 \text{ ja } tulos_j = 1]$  vastaa todennäköisyyttä tilanteelle, jossa lopputulos prosessien välillä on ristiriitainen *vastustajalla*  $B$ . Vastustaja (adversary) pyrkii hankaloittamaan ongelmanratkaisua määräämällä haluamallaan tavalla prosessien alkutilat  $alku_i$  sekä prosessien välillä onnistuneesti välitetyt viestit. Tarkemmin sanoen vastustajaan  $B$  liittyy *viestinvälitystä*  $\gamma$  kuvaava joukko

$$\{(i, j, k) | (i, j) \in E \text{ ja } k \geq 1\},$$

joka sisältää alkion  $(i, j, k)$ , jos ja vain jos prosessi  $j$  vastaanotti onnistuneesti prosessin  $i$  kierroksella  $k$  lähettämän viestin.

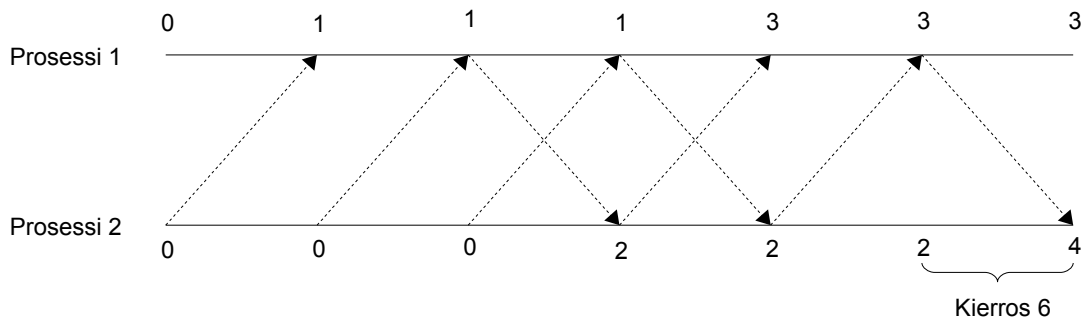
*Validi viestinvälitys* on joukko  $\gamma = \{(i, j, k) | (i, j) \in E \text{ ja } 1 \leq k \leq r\}$ , jossa kaikki viestinvälitys tapahtuu  $r$  kierroksen kuluessa. Määritellään *viestinvälitysjärjestys*  $\leq_\gamma$  pareille  $(i, k)$ , missä  $i \in [1, n]$  on prosessi ja  $k \geq 0$  on kierros, siten, että pätee:

1.  $(i, k) \leq_\gamma (i, k')$  kaikille  $i \in [1, n]$  ja kaikille  $0 \leq k \leq k'$ .
2. Jos  $(i, j, k) \in \gamma$ , niin  $(i, k - 1) \leq_\gamma (j, k)$ .
3. Jos  $(i, k) \leq_\gamma (i', k')$  ja  $(i', k') \leq_\gamma (i'', k'')$ , niin myös  $(i, k) \leq_\gamma (i'', k'')$ .

Ehdoista ensimmäinen pitää huolen siitä, että järjestys on refleksiivinen, ja viimeinen määrittelee järjestyksen olevan transitiiivinen. Keskimäinen ehto kuvaa informaation kulkua lähettäjältä vastaanottajalle silloin, kun prosessin  $i$  kierroksen  $k$  viesti saapuu onnistuneesti prosessille  $j$ .

Viestinvälityksen  $\gamma$  ja sen järjestyksen  $\leq_\gamma$  avulla voidaan nyt määritellä prosessin  $i$  *informaatiomäärä*  $taso_\gamma(i, k)$  kierroksella  $0 \leq k \leq r$ . Informaatiomäärä  $taso_\gamma(i, k)$  on:

1. Jos  $k = 0$ , niin  $taso_\gamma(i, k) = 0$ .
2. Jos  $k > 0$  ja on olemassa  $j \neq i$  siten, että  $(j, 0) \not\leq_\gamma (i, k)$ , niin  $taso_\gamma(i, k) = 0$ .
3. Jos  $k > 0$  ja  $(j, 0) \leq_\gamma (i, k)$  kaikille  $j \neq i$ , niin  $taso_\gamma(i, k) = 1 + \min\{\ell_j | j \neq i\}$ , missä  $\ell_j = \max\{taso_\gamma(j, k') | (j, k') \leq_\gamma (i, k)\}$ .



Kuva 3: Esimerkki informaatiomääristä kahden prosessin verkolle. Onnistuneesti toimitettua viestiä kuvataan nuolella.

Ensimmäinen ehto määrittelee informaatiomäärän  $taso_\gamma(i, k) = 0$  kaikille prosesseille  $i$  kierroksella  $k = 0$ . Toisen ehdon mukaan informaatiomäärä on edelleen  $taso_\gamma(i, k) = 0$ , jos on olemassa yksikin prosessi  $j \neq i$ , jolta ei ole vastaanotettu yhtäkään viestiä  $k$  kierroksen kuluessa. Viimeisen ehdon  $\ell_j = \max\{taso_\gamma(j, k') \mid (j, k') \leq_\gamma (i, k)\}$  vastaa suurinta informaatiomäärää, jonka prosessi  $i$  tietää prosessin  $j$  saavuttaneen kuluneiden  $k$  kierroksen aikana. Koska prosessien välisiä viestejä voi hävitä, prosessi  $i$  ei välttämättä tiedä prosessin  $j$  kierroksen  $k$  todellista informaatiomäärää  $taso_\gamma(j, k)$ , joten  $\ell_j \leq taso_\gamma(j, k)$ .

Tarkastellaan verkon kaikkien prosessien informaatiomäärää, joka on aluksi nolla. Prosessin informaatiomäärä on nolla, kunnes se saa viestin kaikilta muilta prosesseilta ja siirtyy tasolle 1. Kun prosessi saa tietää, että kaikki muutkin prosessit ovat vähintään tasolla 1, prosessi siirtyy tasolle 2. Kuvassa 3 on esimerkki informaatiomääristä kahden prosessin verkolle ensimmäisen  $r = 6$  kierroksen aikana. Esimerkin viestinvälitys  $\gamma$  on sama kuin joukko:

$$\{(2, 1, 1), (2, 1, 2), (1, 2, 3), (2, 1, 3), (1, 2, 4), (2, 1, 4), (2, 1, 5), (1, 2, 6)\}.$$

Informaatiomäärän määritelmästä voidaan nähdä, että informaatiomäärän ero verkon prosessien välillä on rajoitettu:

**Lemma 1** ([VL92, Lyn96]) *Kaikilla prosesseilla  $i$  ja  $j$  sekä kierroksella  $0 \leq k \leq r$  pätee  $|taso_\gamma(i, k) - taso_\gamma(j, k)| \leq 1$ , missä  $\gamma$  on validi viestinvälitys.*

Tämä seuraa selvästi siitä, että joka kierroksella  $0 \leq k \leq r$  prosessin  $i \in [1, n]$  informaatiomäärä joko pysyy ennallaan tai kasvaa. Jos prosessin  $i$  informaatiomäärä kasvaa kierroksella  $k$ , se kasvaa korkeintaan arvoon  $\ell_{j'} + 1$ , missä  $\ell_{j'}$  on pienin prosessin  $i$  tiedossa olevista informaatiomääristä  $\ell_j$  prosesseille  $j \neq i$  kierroksella  $k$ .

**Lemma 2 ([VL92, Lyn96])** *Jos  $\gamma$  on virheetön viestinvälitys, joka sisältää jokaiselle  $1 \leq k \leq r$  kaikki mahdolliset kolmikot  $(i, j, k)$ , niin kaikilla  $i \in [1, n]$  pätee kierroksella  $k$  taso $_{\gamma}(i, k) = k$ .*

Lemma 2 seuraa siitä, että prosessi  $i \in [1, n]$  tietää jokaisella kierroksella muiden prosessien  $j \neq i$  tarkat informaatiomäärät:  $\ell_j$  on kierroksella  $k \geq 1$  prosessin  $j$  informaatiomäärä edellisen kierroksen lopuksi taso $_{\gamma}(j, k - 1)$ . Induktiolla nähdään, että aluksi on  $\ell_j = 0$ , joten taso $_{\gamma}(i, 1) = 1$ , ja tämän jälkeen aina taso $_{\gamma}(i, k) = \text{taso}_{\gamma}(j, k - 1) + 1$ , missä  $j \neq i$ .

Informaatiomäärän avulla voidaan vihdoin määritellä satunnaistettu algoritmi konsensusongelmaan. Kiinnitetään kierrosten lukumäärä  $r$ , jonka kuluessa kaikkien prosessien tulee tehdä lopullinen päätös. Jokainen prosessi pitää kirjaa omasta informaatiomäärästään  $r$  kierroksen ajan. Kierroksella  $r$  jokainen prosessi tarkistaa ylittääkö prosessin oma informaatiomäärä satunnaisesti valitun raja-arvon väliltä  $[1, r]$ . Raja-arvon arpoo prosessi numero 1 heti ensimmäisellä kierroksella, jonka jälkeen arvoa tulvitetaan verkossa.

Prosessi  $i \in [1, n]$  pitää kirjaa kaikkien prosessien alkutiloista taulukossa  $alku_i[j]$ . Aluksi tiedossa on vain oma alkutila  $alku_i[i]$ . Raja-arvo pidetään muuttujassa  $raja_i$ , jonka arvon tietää aluksi vain prosessi  $i = 1$ . Lisäksi seurataan prosessien informaatiomäärää taulukolla  $taso_i[j]$ . Alustetaan jokaiselle prosessille  $i \in [1, n]$  nämä arvot seuraavasti:

- Alkutilat  $alku_i[j] \in \{\text{tuntematon}, 0, 1\}$  asetetaan  $alku_i[j] = \text{tuntematon}$  kaikille  $j \neq i$  ja  $alku_i[i] = alku_i$ .
- Informaatiomäärä  $[-1, r]$  on  $taso_i[j] = -1$  kaikille  $j \neq i$  ja  $taso_i[i] = 0$ .
- Raja-arvo  $raja_i \in [1, r] \cup \text{tuntematon}$  on  $raja_i = \text{tuntematon}$  kaikille  $i \neq 1$ , ja  $raja_1$  on satunnainen luku väliltä  $[1, r]$ .
- Tulos  $tulos_i \in \{\text{tuntematon}, 0, 1\}$  on  $tulos_i = \text{tuntematon}$ .

Prosessit lähettävät jokaisella kierroksella  $1 \leq k \leq r$  viestin kaikille muille verkon prosesseille. Oletetaan siis yksinkertaisuuden vuoksi, että verkko on täydellinen. Sama algoritmi toimisi triviaalisti myös tapauksessa, jossa verkko ei ole täydellinen: tilanne vastaisi viestinvälitystä  $\gamma$  täydellisessä verkossa  $G = (V, E)$ , jossa vastustaja on estänyt joidenkin kaarten  $(i, j) \in E$  viestinvälityksen kokonaan.

---

**Algoritmi**  $saapuvat_i(L_j, V_j, k_j)$ :

```

1  if  $k_j \neq tuntematon$  then  $raja_i \leftarrow k_j$ .
2  for all  $j' \in [1, n]$  and  $j' \neq i$  do
3      if  $V_j[j'] \neq tuntematon$  then  $alku_i[j'] \leftarrow V_j[j']$ .
4      if  $L_j[j'] > taso_i[j']$  then  $taso_i[j'] \leftarrow L_j[j']$ .
5  end for
6   $taso_i[i] \leftarrow 1 + \min\{taso_i[j'] | j' \neq i\}$ .
7  if  $kierros = r$  then
8      if  $raja_i \neq tuntematon$  and  $taso_i[i] \geq raja_i$  and  $alku_i[j'] = 1$  kaikille  $j'$  then
9           $tulos_i \leftarrow 1$ .
10     else  $tulos_i \leftarrow 0$ .
11  end if

```

---

Kuva 4: Prosessi  $i$  vastaanottaa viestin  $(L_j, V_j, k_j)$  prosessilta  $j$  kierroksella  $kierros$ .

Prosessin  $i \in [1, n]$  lähettämät viestit ovat kolmikkoja  $(L, V, k)$ , missä vektori  $L[j]$  pitää sisällään arvot  $taso_i[j]$  ja vektori  $V[j]$  arvot  $alku_i[j]$  kaikille  $j \in [1, n]$ . Arvo  $k$  on aina  $raja_i$ . Toisin sanoen prosessit viestittävät kaikille muille prosesseille sekä omaa tilaansa että aiemmilla kierroksilla vastaanottamiaan tietoja muista prosesseista.

Kierrosten lukumäärästä pidetään kirjaa muuttujalla  $kierros$ , jonka arvoa kasvatetaan yhdellä aina kierroksen aluksi. Kuvassa 4 on esimerkki siitä, mitä prosessi  $i$  tekee vastaanottaessaan viestin  $(L_j, V_j, k_j)$  prosessilta  $j$  kierroksella  $kierros$ . Ensimmäisellä rivillä tarkistetaan sisältääkö viesti raja-arvon  $k_j$ . Raja-arvo on sama kaikissa viesteissä, joissa  $k_j \neq tuntematon$ , joten ei haittaa vaikka arvo  $raja_i$  asetetaan useamman kerran. Kunnes raja-arvo leviää myös muiden prosessien tietoon,  $k_j \neq tuntematon$  vain prosessin  $j = 1$  viesteissä.

Kolmannella rivillä ylläpidetään alkutilojen taulukkoa  $alku_i[j']$  kaikille  $j' \neq i$  siten, että kaikki vektorin arvot  $V_j[j']$ , joilla  $V_j[j'] \neq tuntematon$ , asetetaan arvoiksi  $alku_i[j']$ . Näin tieto prosessien alkutiloista  $alku_j$ ,  $j \neq i$ , leviää verkon prosessilta toiselle.

Neljännellä rivillä päivitetään informaatiomäärät taulukkoon  $taso_i[j']$  kaikille  $j' \neq i$ . Ehto  $L_j[j'] > taso_i[j']$  pitää huolen siitä, että arvo  $taso_i[j']$  saa suurimman arvonsa kaikista prosessille  $i$  saapuvista arvoista  $L_{j''}[j']$ , missä  $j'' \neq i$ . Kun rivien 2–5 silmukka on käyty loppuun, pätee  $taso_i[j'] = \ell_{j'}$ , missä  $\ell_{j'} = \max\{taso_\gamma(j', k') | (j', k') \leq_\gamma(i, k)\}$  kierroksella  $k = kierros$ .

Rivillä 6 päivitetään prosessin  $i$  omaa informaatiomäärää  $taso_i[i]$ . Informaatiomäärän määritelmän mukaan uusi arvo on  $\ell_{j''} + 1$ , missä  $\ell_{j''}$  on pienin arvoista  $taso_i[j']$ . Viestinvälityksellä  $\gamma$  ja kierroksella  $1 \leq k \leq r$  pätee siis  $taso_i[i] = taso_\gamma(i, k)$ . Jos  $taso_i[i] \geq 1$ , ovat sekä raja-arvo  $raja_i$  että kaikki  $alku_i(j')$ ,  $j' \in [1, n]$ , määriteltyjä, eli erisuuria kuin *tuntematon*.

Kierroksen  $kierros = r$  päätteeksi jokainen prosessi valitsee lopullisen tuloksensa  $tulos_i \in \{0, 1\}$ . Prosessi valitsee  $tulos_i = 1$ , jos se tietää raja-arvon  $raja_i$  ja prosessin oma informaatiomäärä on suurempi tai yhtäsuuri kuin raja-arvo. Lisäksi kaikkien prosessien alkutila tulee olla tiedossa ja  $alku_{j'} = 1$  kaikille  $j'$ . Jos nämä ehdot eivät täyty prosessi valitsee  $tulos_i = 0$ .

Satunnaistettu algoritmi ratkaisee konsensusongelman  $r$  kierroksen kuluessa siten, että ehdot (b) ja (c) ovat voimassa: Kaikkien prosessien alkutilan ollessa 0 myös kaikki prosessit asettavat  $tulos_i \leftarrow 0$ , koska rivin 8 ehdot eivät koskaan täyty. Vastaavasti, jos kaikkien prosessien alkutila on 1 ja  $\gamma$  on virheetön viestinvälitys, niin lemmän 2 mukaan kaikkien prosessien informaatiomäärä  $taso_i[i] = r$ . Näin ollen kaikki prosessit asettavat  $tulos_i \leftarrow 1$ , koska triviaalisti  $taso_i[i] \geq raja_1$  kaikille  $raja_1 \in [1, r]$ . Lisäksi pätee selvästi, että  $raja_i \neq tuntematon$  sekä  $alku_i[j'] = 1$  kaikille  $j'$ .

Satunnaistettu algoritmi ratkaisee konsensusongelman  $r$  kierroksen kuluessa siten, että arvolle  $\epsilon = \frac{1}{r}$  on voimassa ehto (a):

$$Pr^B[\exists i, j \in [1, n] : tulos_i = 0 \text{ ja } tulos_j = 1] \leq \epsilon$$

Tarkastellaan tilannetta kierroksen  $r$  päättyessä. Olkoon  $\ell_i^r = taso_i[i]$  kierroksella  $r$  kaikille prosesseille  $i \in [1, n]$ . Lemman 1 mukaan kaikki arvot  $\ell_i^r$  eroavat toisistaan korkeintaan yhden yksikön verran. Jos satunnaisesti valittu raja-arvo  $raja_1 \in [1, r]$  on suurempi kuin  $\max\{\ell_i^r\}$  tai yksikin prosesseista saa alkutilan 0, päättävät kaikki prosessit  $tulos_i = 0$ . Vastaavasti, jos raja-arvo  $raja_1 \leq \min\{\ell_i^r\}$  ja kaikkien prosessien alkutila on 1, niin prosessit valitsevat  $tulos_i = 1$ .

Ristiriita ehdossa (a) tapahtuu siis vain silloin, kun  $raja_1 = \max\{\ell_i^r\}$ . Todennäköisyys tälle on  $\frac{1}{r} = \epsilon$ , koska  $\max\{\ell_i^r\}$  määräytyy aina deterministisesti vastustajalle  $B$  ja  $raja_1$  valitaan satunnaisesti väliltä  $[1, r]$ . Toisin sanoen  $\max\{\ell_i^r\}$  on vakio, joka riippuu vastustajasta  $B$ .

Otetaan esimerkiksi kuvan 3 viestinvälitys sekä arvo  $r = 6$ , jolloin  $\epsilon = \frac{1}{6}$ . Ristiriidan todennäköisyys on siis korkeintaan  $\frac{1}{6}$ . Olkoon vastustajalla  $B$  alkutilat  $alku_1 = alku_2 = 1$ . Jos valitaan raja-arvo  $raja_1 = 4$ , niin prosessi 1 päättää  $tulos_1 = 0$  ja prosessi 2  $tulos_2 = 1$ , eli tulokset ovat ristiriidassa. Jos taas  $raja_1 \leq 3$ , niin

molemmat prosessit valitsevat  $tulos_1 = tulos_2 = 1$ . Vastaavasti, jos  $raja_1 \geq 5$ , niin valitaan tulokset  $tulos_1 = tulos_2 = 0$ . Ristiriita syntyy siis juuri todennäköisyydellä  $\frac{1}{6}$ .

Jos vastustaja valitsee mitkä tahansa muut alkutilat  $alku_1$  ja  $alku_2$  kuin edellä vastustaja  $B$ , niin prosessit valitsevat triviaalisti aina  $tulos_1 = tulos_2 = 0$ . Ristiriidan todennäköisyys on näissä tapauksissa siis 0.

### 3.1 Alaraja ristiriidan todennäköisyydelle

Mikä tahansa satunnaistettu algoritmi, joka toimii  $r$  kierrosta, rikkoo yksimielisyyden ehtoa (a) vähintään todennäköisyydellä  $\frac{1}{r+1}$  [VL92, Lyn96]. Alaraja voidaan todistaa seuraavan lemmän avulla:

**Lemma 3** ([VL92, Lyn96]) *Olkoon  $B$  mikä tahansa vastustaja, jolle kaikkien prosessien alkutilat ovat  $alku_i = 1$ , ja  $i$  mikä tahansa prosessi. Tällöin pätee*

$$Pr^B[i \text{ valitsee } tulos_i = 1] \leq \epsilon(taso_\gamma(i, r) + 1).$$

Olkoon  $B$  vastustaja, jolle kaikkien prosessien alkutilat ovat  $alku_i = 1$ , ja  $\gamma$  virheetön viestinvälitys. Todennäköisyys, jolla kaikki prosessit valitsevat  $tulos_i = 1$ , on pienempi tai yhtäsuuri kuin todennäköisyys, jolla jokin prosesseista valitsee  $tulos_i = 1$ . Lemman 3 perusteella tämä on korkeintaan  $\epsilon(taso_\gamma(i, r) + 1)$ . Koska lemmän 2 mukaan täydelliselle viestinvälitykselle  $\gamma$  pätee  $taso_\gamma(i, r) = r$ , supistuu todennäköisyys muotoon  $\epsilon(r + 1)$ .

Toisaalta ehto (c) määrittelee, että täydellisellä viestinvälityksellä  $\gamma$  ja kaikkien prosessien alkutilojen ollessa 1 myös kaikkien tulosten tulee olla 1. Todennäköisyys sille, että kaikki prosessit valitsevat  $tulos_i = 1$ , on siis tasan 1. Tästä saadaan  $1 \leq \epsilon(r + 1)$ , eli alaraja  $\epsilon \geq \frac{1}{r+1}$ .

## 4 Yhteenveto

Tämä tutkielma tarkasteli konsensusongelmaa hajautetuissa järjestelmissä, joissa prosessien välisiä viestejä voi kadota. Konsensusongelma osoittautui ratkeamattomaksi jopa kahden prosessin verkossa. Satunnaistetulla algoritmilla konsensusongelman voi kuitenkin ratkaista todennäköisyydellä  $\frac{r-1}{r}$ , missä  $r$  on algoritmin suoritus-

kierrosten lukumäärä. Lisäksi näytettiin, ettei mikään satunnaistettu algoritmi voi toimia virhetodennäköisyyden alarajaa  $\frac{1}{r+1}$  paremmin.

## Lähteet

- Gra78      Gray, J., Notes on data base operating systems. *Operating Systems, An Advanced Course*, London, UK, 1978, Springer-Verlag, sivut 393–481.
- Lyn96      Lynch, N. A., *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.
- VL92      Varghese, G. ja Lynch, N. A., A tradeoff between safety and liveness for randomized coordinated attack protocols. *PODC '92: Proceedings of the eleventh annual ACM symposium on Principles of distributed computing*, New York, NY, USA, 1992, ACM Press, sivut 241–250.