

# **Bysanttilaiset kenraalit**

Antti Tani

Helsinki 29.10.2007

Tutkielma

HELSINGIN YLIOPISTO  
Tietojenkäsittelytieteen laitos

# Sisältö

<b>1</b>	<b>Johdanto</b>	<b>1</b>
<b>2</b>	<b>Bysanttilainen konsensusongelma</b>	<b>2</b>
2.1	Määritelmiä . . . . .	2
2.2	Bysanttilaisten kenraalien ongelma . . . . .	4
2.3	Kommunikaatiotarpeen rajoittaminen . . . . .	7
2.4	Muunnoksia ja laajennoksia . . . . .	8
	<b>Lähteet</b>	<b>10</b>

# 1 Johdanto

Tietokonejärjestelmä jossa on useita keskenään kommunikoivia komponentteja, voi olla alttiina monille virhetekijöille, jotka vaarantavat järjestelmän oikean suorituksen. Virhetekijä voi olla esimerkiksi häiriö kommunikaatioyhteydessä, järjestelmän osan suorituksen pysähtyminen lopullisesti tai joksikin aikaa, tai järjestelmän osan joutuminen mielivaltaiseen tilaan siihen kohdistuneen häiriön seurauksena. Useissa järjestelmän vikasietoisuutta kuvaavissa malleissa oletetaan, että järjestelmän *toiminnassa* olevat komponentit toimivat oikealla tavalla jonkin ajanhetken *jälkeen, sen tiedon puitteissa* mitä ne saavat muusta järjestelmästä. Tässä paperissa tarkastelemme sellaista vikasietoisuuden mallia, jossa jotkin järjestelmän toiminnassa olevat komponentit voivat toimia *mielivaltaisen ajan* virheellisellä tavalla, mikä näyttääytyy järjestelmän oikein toimiville komponenteille siten, että ne voivat saavuttaa virheellistä *informaatiota* näiltä väärin toimivilta komponenteilta. Tässä vikasietoisuusmallissa järjestelmä voi siis kuitenkin *kokonaisuutena* toimia oikein väärin toimivista komponenteistaan huolimatta.

Tämän kirjoitelman tarkoituksena kuvailla tarkemmin edellä luonnosteltua mallia, jota kutsutaan *bysanttilaiseksi vikasietoisuudeksi* (Byzantine fault tolerance), erityisesti pureutuen joihinkin vikasietoisuusratkaisuihin valikoiduilla mallin erikoistapauksilla. Määrittelemme *konsensusongelman* – jossa luotettavasti toimivien komponenttien on päästävä yksimielisyyteen – bysanttilaisessa virhemallissa. Termi *bysanttilainen* viittaa *bysanttilaisten kenraalien ongelmaan*, joka esiteltiin Lamportin, Shostakin ja Peasen 1982 julkaistussa artikkelissa [LSP82] kuvaamaan abstraktilla tavalla järjestelmää, jossa on virheellisesti toimivia osajärjestelmiä jotka levittävät toimivien osajärjestelmien keskuuteen virheellistä informaatiota. Seuraavassa luvussa käymme läpi tämän klassisen ongelman ja sen jälkeen perehdymme mallin laajennoksiin

Bysanttilaisten kenraalien ongelma sai innoitusta – ollen oikeastaan yksinkertaistus – eräästä lentokoneteollisuudessa esiin tulleesta käytännön ongelmasta [WLG<sup>+</sup>78]. Siinä ongelmassa usean prosessin, joista jokainen saa lukemia omasta korkeusmittaristaan, pitäisi päästä yksimielisyyteen lentokoneen lentokorkeudesta. Yksinkertainen keskiarvon laskeminen ei välttämättä ole toimiva ratkaisu ongelmaan, koska viallisesti toimivat prosessit voivat lähettää eri korkeuslukemia eri prosesseille, jolloin keskiarvo riippuu siitä miltä prosessilta kysytään. Toinen vastaavanlainen ongelma on hajautettu vikadiagnoosi, jossa prosessit voivat tehdä erillisiä vikadiagnoosiproseduureja jollekin järjestelmäkomponentille, ja niiden pitäisi omien tulostensa pohjalta tehdä yhteinen päätös siitä, onko komponentti vaihtamisen tarpeessa. Bysanttilaisesti vikasietoisia moniprosessijärjestelmiä on kehitetty turvallisuuskriittisiin sovelluksiin esimerkiksi miehittämättömissä sukellusveneissä, ydinsukellusveneissä ja ydinvoimaloiden hallintajärjestelmissä [Lyn96],[LHA91]. Bysanttilaiselle vikasietoisuudelle voi olla yhä enenevässä määrin tarvetta lähempänä tavallista käyttäjää olevissa sovelluksissa ja järjestelmissä. Esimerkiksi verkon yli tapahtuvat hyökkäykset, operaattorivirheet ja ohjelmavirheet ovat yleisiä aiheuttajia bysanttilaisille häiriöille. Lisääntynyt riippuvaisuus tietokonejärjestelmistä lisää myös motiivia niihin

kohdistuville vihamielisille hyökkäyksille. Ohjelmistojen koon ja monimutkaisuuden kasvaessa suurenee myös ohjelmistovirheiden riski.

## 2 Bysanttilainen konsensusongelma

### 2.1 Määritelmiä

Yksinkertaisuuden vuoksi tästä lähtien tässä kirjoitelmassa kutsutaan itsenäisesti toimivia komponentteja tai osajärjestelmiä prosesseiksi. Tässä nimennässä tehdään kuitenkin poikkeus bysanttilaisten kenraalien ongelmaa kuvaavan luvun aikana, jossa näitä komponentteja kutsutaan kenraaleiksi.

Määritellään prosessien muodostama järjestelmä *verkkona*, jossa prosessit ovat *solmuja* ja prosessien väliset viestiyhteydet *kaaria*. Oletetaan toistaiseksi että verkko on *täydellinen*; luvussa 2.4 tätä oletusta tullaan lieventämään.

Ajastusmallina käytetään *synkronista* mallia. Siinä prosessit tekevät askeleita samanaikaisesti, yhdellä *kierroksella* yhden askeleen, joka puolestaan voi koostua useasta atomisesta suorituksesta. Yhdellä kierroksella prosessit ensin vastaanottavat niille lähetetyt viestit yhtä aikaa, tekevät laskentaa saamansa informaation perusteelle yhtä aikaa, ja lopuksi lähettävät viestejä muille prosesseille yhtä aikaa. Mallia ei useinkaan voi suoraan soveltaa käytännön hajautettuihin järjestelmiin, mutta mallilla saatuja tuloksia voi toisinaan käyttää laajemmassakin kontekstissa; joko hieman muokaten, tai sitten ymmärtämisen apuna.

Kirjoitelmassa tehdään myös oletuksia väärin toimivien prosessien lukumäärän suhteen. Eräissä prosessivikoja kuvaavissa malleissa viallisia prosesseja esiintyy jonkin todennäköisyysjakauman mukaan. Kaikissa tässä kirjoitelmassa esitellyissä tapauksissa viallisten prosessien lukumäärälle vaaditaan jokin kiinteä yläraja. Tämä yksinkertaistaa analyysiä huomattavasti. Yksinkertaistus on sikäli perusteltua, kun sitä ajatellaan merkityksessä, että on epätodennäköistä tapahtuvan enemmän kuin  $f$  virhettä. Toisaalta monissa käytännön tilanteissa suuri viallisten prosessien lukumäärä kasvattaa lisävirheiden todennäköisyyttä, eikä mielekästä ylärajaa virheille näin voida antaa. Olettamalla ylärajan viallisten prosessien määrälle, oletetaan samalla että viat *korreloivat negatiivisesti*, kun käytännössä ne pikemminkin ovat riippumattomia toisistaan, tai korreloivat positiivisesti.

Esittelen *konsensusongelman* (agreement problem) määrittelyn sekä *pysähtymisvirheille* (stopping failure) että bysanttilaisille häiriöille. Olkoon  $V$  arvojoukko. Jokaisen prosessin syöte alkutilassa on jokin joukon  $V$  arvo. Prosessien päämääränä on lopulta tulostaa päätös  $d$  joukosta  $V$ . Prosessin  $j$  vastaanottamaa prosessin  $i$  arvoa merkitään  $v_i$ , tätä merkintää voidaan käyttää myös prosessin  $i$  alkuarvolle.

Pysähtymisvirhemallissa prosessin suoritus voi pysähtyä kokonaan milloin tahansa. Erityisesti suoritus voi pysähtyä keskellä viestinelähetysaskelta, jolloin sillä kierroksella jolla prosessi pysähtyy, vain osa lähettää tarkoituksena olevista viesteistä todella lähetetään. Prosessi voi myös pysähtyä lähetettyään kierroksen viestit, mutta

ennen siirtymistä seuraavaan tilaansa.

### Määritelmä 2.1 (Konsensusongelma pysähtymisvirhemallissa)

**päätös** *Jokainen ei-virheellinen prosessi lopulta päättää arvon  $d_i \in V$ .*

**konsensus** *Jokainen prosessi päättää saman arvon  $d \in V$ .*

**validisuus** *Jos kaikkien prosessorien alkuarvot  $v_i$  ovat identtiset, niin  $d_i = v_i$  kaikilla prosesseilla  $i$ .*

Bysanttilaisessa virhemallissa prosessi voi paitsi pysähtyä, myös käyttäytyä *mielivaltaisella* tavalla. Mielivaltaisuus tarkoittaa tässä, että prosessi voi käynnistyä mielivaltaisessa tilassa, joka ei välttämättä kuulu sen määritelyihin alkutiloihin; voi lähettää mielivaltaisia viestejä ja voi tehdä mielivaltaisia tilasiirtymiä. Ainoa rajoitus näiden virheellisten prosessien käyttäytymiselle on, että ne voivat vaikuttaa vain niihin prosesseihin viesteillään, mihin niillä on viestiyhteys, ja voivat vaikuttaa suoraan vain omaan tilaansa. Prosessien välinen viestinvälitys määritellään tarkemmin seuraavien oletuksien avulla:

#### Viestinvälitys

V1 Jokainen lähetetty viesti kuljetetaan perille oikein.

V2 Viestin vastaanottaja tietää mikä prosessi lähetti sen.

V3 Viestin tulematta jääminen havaitaan.

Oletukset V1 ja V2 estävät virheellistä prosessia häiritsemästä kahden prosessin välistä suoraa kommunikaatiota. Oletus V3 tekee tyhjäksi prosessin mahdollisuudet estää päätöksenteko viestin lähettämättä jättämisellä.

### Määritelmä 2.2 (Konsensusongelma Bysanttilaisessa virhemallissa)

**päätös** *Jokainen ei-virheellinen prosessi lopulta päättää arvon  $d_i \in V$ .*

**konsensus** *Jokainen ei-virheellinen prosessi päättää saman arvon  $d \in V$ .*

**validisuus** *Jos kaikkien ei-virheellisten prosessorien alkuarvot  $v_i$  ovat identtiset, niin  $d_i = v_i$  kaikilla ei-virheellisillä prosesseilla  $i$ .*

Toisessa tunnetussa variantissa bysanttilaisesta konsensusongelmasta on olemassa yksiselitteinen *johtaja*, jolla on yksi alkuarvo  $v$ . Päätös- ja konsensus ehdot pysyvät samoina, mutta validisuusehto on seuraava:

**validisuus** Jos johtaja on ei-virheellinen, kaikkien ei-virheellisten prosessien tulee päättää johtajan arvo  $v$

Menetelmät molemmille varianteille ovat käytännössä identtiset, ja kaikki mitä sanomme tässä pätee molemmille vain pienin modifikaatioin. Käytetään bysanttilaiselle konsensusongelmalle tästä lähtien lyhennettä *BK-ongelma* tai *BKO*.

### **Vaativuusmääreitä**

Fischer ja Lynch [FL82] osoittivat, että bysanttilaista konsensusongelmaa ei voida ratkaista alle  $f + 1$  kierroksessa pahimmassa tapauksessa. Aikavaativuuteen vaikuttaa kierrosten määrän ohella kierroksella tehtyjen laskenta-askeleiden määrä, joka taas riippuu prosessien välisen kommunikaation määrästä. Kommunikaatiovaativuudeksi määritetään lähetettyjen viestien ja niiden sisältämien bittien määrä. Pysähtymisvirhemallissa tähän luetaan mukaan kaikkien prosessien lähettämät viestit. Bysanttilaisessa virhemallissa luetaan mukaan vain ei-virheellisten prosessien lähettämät viestit. Tämä johtuu siitä, että virheellisten prosessien lähettämien viestien (ja bittien) lukumäärälle ei voida määrittää mitään ei-triviaalia ylärajaa bysanttilaisessa virhemallissa.

Yleinen ominaisuus kaikille bysanttilaisen vikasietoisuuden algoritmeille on, että niiden tarvitsemien prosessien lukumäärä on enemmän kuin kolme kertaa viallisten prosessien lukumäärä,  $n > 3f$ . Tämä saattaa tuntua yllättävältä, koska voisi luulla että  $2f + 1$  prosessia voisi kestää  $f$  bysanttilaista vikaa käyttämällä jonkinlaista enemmistöäänestystä (majority voting) perustuvaa algoritmia. Seuraavassa luvussa kuitenkin näytetään ettei tämä onnistu.

## **2.2 Bysanttilaisten kenraalien ongelma**

Klassinen bysanttilaisten kenraalien ongelma on havainnollistus tietokonejärjestelmästä, jossa luotettavasti toimivien komponenttien on saatava aikaan järjestelmän toivottu toiminta, huolimatta viallisten komponenttien antamasta ristiriitaisesta informaatiosta. Tässä luvussa esitellyt tulokset ja algoritmit koskevat, oleellisesti korvaamalla “kenraali” prosessilla, mitä tahansa hajautettua järjestelmää bysanttilaisen häiriön piirissä. Kuvailen seuraavaksi tämän ongelman, kuten se on Lamportin [LSP82] artikkelissa esitelty. Ongelmassa joukko bysanttilaisen armeijan kenraaleja on ryhmittynyt divisioonineen viholliskaupungin ympäristöön. Kenraalien on lähetänsä avulla sovittava yhteisestä hyökkäyssuunnitelmasta, joka on *päätös* hyökätäkö vai vetäytyä. Jokainen kenraali tarkkailee vihollista ja keskusteleee havainnoistaan muiden kenraalien kanssa. Osa kenraaleista on kuitenkin pettureita, jotka yrittävät pilata yhteispäätöksen syntymisen kylvämällä ristiriitaista informaatiota kenraalien keskuuteen. Jos vain osa uskollisista kenraaleista hyökkää, hyökkäys on tuomittu epäonnistumaan. Tarkoituksena on löytää algoritmi, jolla uskolliset kenraalit pääsevät lopulta yksimielisyyteen hyökkäyspäätöksestä. Ongelman ehdot voidaan esittää seuraavalla tavalla:

1. Jokaisen uskollisen kenraalin täytyy saada sama informaatio  $v(1) \dots v(n)$ .

2. Jos kenraali  $i$  on uskollinen, kaikkien uskollisten kenraalien täytyy käyttää hänen lähettämäänsä arvoa arvona  $v(i)$ .

Ehto 1 voidaan uudelleenkirjoittaa seuraavassa muodossa, riippumatta siitä onko kenraali  $i$  uskollinen vai ei:

- 1' Mitkä tahansa kaksi uskollista kenraalia käyttävät samaa arvoa  $v(i)$ .

Ehdot 1' ja 2 koskevat molemmat yksittäistä arvoa jonka kenraali  $i$  lähettää. Tarastelu voidaan siten rajoittaa ongelmaan, kuinka yksittäinen kenraali voi lähettää arvonsa toisille kenraaleille. Tämä voidaan tapauksena, jossa komentava kenraali lähettää käskyjään muille kenraaleille – nimitettäköön niitä tässä yhteydessä luutnanteiksi. Nyt päästään käsiksi muotoiluun, joka on varsinainen bysanttilaisten kenraalien ongelma.

### Määritelmä 2.3 (Bysanttilaisten kenraalien ongelma)

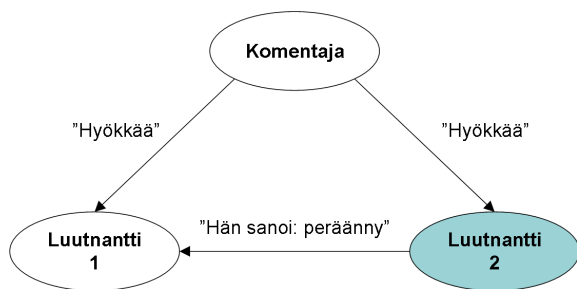
*IC1 Kaikki uskolliset kenraalit noudattavat samaa käskyä*

*IC2 Jos komentaja on uskollinen, kaikki uskolliset kenraalit noudattavat hänen käskyään.*

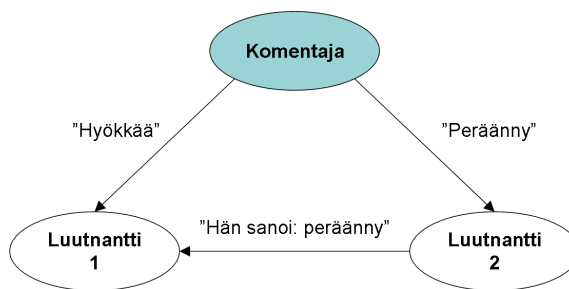
Ehtoja IC1 ja IC2 kutsuttiin *vuorovaikutteisen konsistenssin* (interactive consistency) ehdoiksi, ennen kuin termi “bysanttilainen” vakiintui alan termistöön. Huomattakoon, että jos komentaja on uskollinen, ehto IC1 seuraa ehdosta IC2. Tietenkään komentajan ei tarvitse olla uskollinen. Huomattakoon, että ehdot IC1 ja IC2 ovat yhtäpitävät Bysanttilaisen konsensusongelman määrittelyn (tapaus jossa johtaja on olemassa) kanssa.

Hahmotellaan todistus sille, että bysanttilaisten kenraalien ongelmaa ei voida ratkaista kolmella kenraalilla, kun yksi on petturi. Ongelma voidaan jakaa kahteen tapaukseen, joista ensimmäisessä toinen luutnanteista on petturi, ja toisessa komentaja on petturi. Ensimmäisessä tapauksessa, kuva 2.1, komentaja on uskollinen ja lähettää hyökkäyskäskyn, mutta luutnantti 2 petturina raportoi luutnantille 1, että komentaja lähetti käskyn “peräänny”. Jotta ehto IC2 olisi nyt voimassa, luutnantin 1 täytyy noudattaa komentajan käskyä “hyökkää”. Toisessa tapauksessa, kuva 2.2, komentaja on petturi, ja lähettää luutnantille 1 hyökkäyskäskyn, ja luutnantille 2 perääntymiskäskyn. Luutnantti 1 ei tiedä kumpi kenraaleista on petturi, eikä sitä, minkä viestin komentaja todella lähetti luutnantille 2. Jos petturi valehtelee johdonmukaisesti koko ajan, luutnantilla 1 ei ole mitään keinoa erottaa äskeisiä tapauksia toisistaan, joten hänen täytyy noudattaa käskyä “hyökkää” molemmissa tapauksissa. Näin ollen luutnantin 1 täytyy noudattaa aina komentajan hyökkäyskäskyä.

Samanlaisella päättelyllä kuitenkin nähdään, että jos luutnantti 2 saa komentajalta perääntymiskäskyn, hänen täytyy noudattaa sitä riippumatta siitä, mitä luutnantti 1



Kuva 2.1: Luutnantti 2 on petturi



Kuva 2.2: Komentaja on petturi

kertoo komentajan käskeneen. Kuvan 2.2 tapauksessa luutnantin 2 täytyy noudattaa komentajan käskyä “perääny”, ja luutnantin 1 komentajan käskyä “hyökkää”, joten ehto IC1 ei pysy voimassa. Näin ollen *kolmen bysanttilaisen kenraalin ongelmaan ei ole olemassa ratkaisua kun yksi kenraaleista on petturi*. Yksityiskohtainen todistus löytyy esimerkiksi lähteestä [Lyn96].

Käyttämällä edellistä tulosta, voidaan näyttää, että bysanttilaisten kenraalien ongelmalle ei ole olemassa ratkaisua, jos kenraaleja on vähemmän kuin  $3f + 1$ , missä  $f$  on pettureiden lukumäärä.

Todistus, jonka tässä hahmottelemme, käyttää vastaoletusta:

*oletamme että bysanttilaisten kenraalien ongelmaan on olemassa ratkaisu, kun kenraaleita on  $3f$  tai vähemmän,*

ja käytämme sitä konstruoimaan ratkaisun ongelmaan kolmella kenraalilla ja yhdellä petturilla, jonka tiedämme olevan mahdoton. Välttääksemme sekaannusta, nimeämme vastaoletuksen ratkaisussa olevia kenraaleja *albanialaisiksi*, ja konstruoidun ratkaisun kenraaleja bysanttilaisiksi. Aloittamalla algoritmista joka ratkaisee vastaoletuksen, konstruoidumme siis ratkaisun bysanttilaisen kenraalin ongelmaan kolmella kenraalilla ja yhdellä petturilla. Oletetaan, että jokainen bysanttilainen kenraali simuloi yhtä kolmasosaa albanialaisista kenraaleista, joka tarkoittaa siis korkeintaan  $f$  kenraalia. Bysanttilainen komentaja simuloi albanialaista komentajaa ja korkeintaan  $f - 1$  albanialaista luutnanttia, ja kumpikin bysanttilainen luutnantti simuloi siis korkeintaan  $f$  albanialaista luutnanttia. Koska vain yksi bysanttilainen kenraali voi olla petturi, ja hän simuloi korkeintaan  $f$  albanialaista kenraalia, korkeintaan  $f$  albanialaisista kenraaleista on pettureita. Näin ollen vastaoletus takaa, että ehdot IC1 ja IC2 pätevät albanialaisille kenraaleille. Ehdon IC1 mukaan kaikki albanialaiset luutnantit joita uskollinen bysanttilainen luutnantti simuloi noudattavat samaa käskyä, joka on kyseisen bysanttilaisen luutnantin noudattama käsky. Ehdot IC1 ja IC2 albanialaisille kenraaleille implikoivat samat ehdot myös bysanttilaisille kenraaleille, joten vastaoletus on mahdoton. Täsmällinen todistus on esitetty Lynchin kirjassa [Lyn96].

Tarkastellaan seuraavaksi erästä ratkaisualgoritmia bysanttilaisten kenraalien ongelmaan [LSP82]. Määritellään *rekursiivisesti* algoritmi  $OM(f)$  (Oral Message) kaikille ei-negatiivisille kokonaisluvuille  $f$ , missä komentaja lähettää käskynsä  $n - 1$  luut-

nantille. Algoritmi  $OM(f)$  ratkaisee bysanttilaisten kenraalien ongelman kun kenraaleita on vähintään  $3f + 1$ , missä  $f$  on pettureiden lukumäärä. Käytetään algoritmin kuvauksessa ilmauksen “luutnantti noudattaa käskyä” tilalla ilmausta “luutnantti käyttää arvoa”. Algoritmi käyttää funktiota *majority*, jolla on seuraava ominaisuus: jos enemmistölle arvoista  $v_i$  pätee,  $v_i = v$ , niin  $majority(v_1, \dots, v_{n-1}) = v$ .

**Algoritmi  $OM(0)$**

1. Komentaja lähettää arvonsa kaikille luutnanteille.
2. Jokainen luutnantti käyttää komentajalta saamaansa arvoa, jos sellainen sai, muuten arvoa *PERÄÄNNY*.

**Algoritmi  $OM(f)$ ,  $f > 0$**

1. Komentaja lähettää arvonsa kaikille luutnanteille.
2. Kaikille luutnanteille  $i$ , olkoon  $v_i$  komentajalta saatu arvo, tai *PERÄÄNNY* jos komentajalta ei saatu arvoa. Luutnantti  $i$  toimii komentajana algoritmissa  $OM(f - 1)$  lähettäen arvonsa  $v_i$  muille  $n - 2$  luutnantille.
3. Kaikille  $i, j$ ,  $i \neq j$ , olkoon  $v_j$  se arvo jonka luutnantti  $i$  saa luutnantilta  $j$  askeleessa (2) (algoritmillä  $OM(f-1)$ ), tai muuten *PERÄÄNNY* jos hän ei saa arvoa. Luutnantti  $i$  käyttää sitten arvoa  $majority(v_1, \dots, v_{n-1})$ .

Algoritmin oikeellisuus on osoitettu sen esitelleessä artikkelissa [LSP82]. Algoritmin hyvänä puolena on sen yksinkertaisuus, huonona puolena laskennallinen vaativuus. Sen suorituksen aikana lähetetään jopa  $(n - 1)(n - 2) \dots (n - f - 1)$  viestiä kenraalien välillä, joka tarkoittaa *eksponentiaalista* aikavaativuutta. Jotta bysanttilaiseen vikasietoisuuteen saataisiin kelvollisia ratkaisuja, algoritmin pitäisi toimia *polynomisessa* ajassa; seuraavassa luvussa näytän esimerkin tällaisesta algoritmista.

## 2.3 Kommunikaatiotarpeen rajoittaminen

Bysanttilaiseen konsensusongelmaan kehitettyjen algoritmien kompleksisuutta hallitsee useissa tapauksissa prosessien välisen kommunikaation määrä, joka oli edellisen luvun algoritmissa eksponentiaalinen, tehden siitä käytännön sovelluksiin sopimattoman. Kommunikaatiotarpeeltaan polynomisia BKO-algoritmeja on kuitenkin olemassa [DS82] [GM98]. Käyn läpi Lynchin kirjassa [Lyn96] esiteltyä *Poly-Byz*-algoritmia, joka vaatii  $2f + 1$  kierrosta ollen samalla kommunikaatiotarpeeltaan polynomisen.

Algoritmi käyttää mekanismia nimeltä *konsistentti lähetys* (consistent broadcast) kaikessa prosessienvälisessä kommunikaatiossa. Käyttämällä konsistenttia lähetystä prosessi  $i$  voi lähettää viestin muodossa  $(m, i, r)$  kierroksella  $r$ , ja mikä tahansa

prosessi voi *hyväksyä* viestin millä tahansa myöhemmällä kierroksella. Konsistentin lähetyksen mekanismi täyttää seuraavat kolme ehtoa:

1. Jos luotettava prosessi  $i$  lähettää viestin  $(m, i, r)$  kierroksella  $r$ , niin kaikki luotettavat prosessit hyväksyvät viestin viimeistään kierroksella  $r + 1$  (viesti siis hyväksytään joko kierroksella  $r$  tai  $r + 1$ ).
2. Jos luotettava prosessi  $i$  ei lähetä viestiä  $m, i, r$  kierroksella  $r$ , niin mikään luotettava prosessi ei ikinä hyväksy viestiä  $(m, i, r)$ .
3. Jos jokin luotettava prosessi  $j$  hyväksyy viestin  $(m, i, r)$  kierroksella  $r'$ , silloin jokainen luotettava prosessi hyväksyy tämän viestin kierrokseen  $r' + 1$  mennessä

Ensimmäisen ehdon mukaan luotettavien prosessien lähetykset hyväksytään nopeasti. Toisen ehdon mukaan mitään viestiä ei ikinä virheellisesti uskota luotettavan prosessin lähettämäksi. Kolmannen ehdon mukaan mikä tahansa luotettavan prosessin hyväksymä viesti, riippumatta siitä onko sen lähettäjä luotettava vai virheellinen prosessi, hyväksytään kaikissa luotettavissa prosesseissa nopeasti tämän jälkeen.

### ***PolyByz-algoritmi***

PolyByz-algoritmi käyttää apunaan konsistentin lähetyksen toteuttavaa algoritmia, joka käyttää yhden viestin  $(m, i, r)$  konsistenttiin lähetykseen  $O(n^2)$  apuviestiä. PolyByz-algoritmin eteneminen muodostuu  $f + 1$  vaiheesta, missä jokainen vaihe koostuu kahdesta kierroksesta. Kaikki lähetetyt viestit (käyttämällä konsistenttia lähetystä) ovat muotoa  $(1, i, r)$ , missä  $i$  on prosessin indeksi, ja  $r$  on pariton kierrosnumero. Näin ollen viestejä lähetetään vain vaiheiden ensimmäisillä kierroksilla, ja ainoa informaatio jota lähetetään on arvo 1.

Prosessi  $i$  lähettää viestin seuraavilla ehdoilla. Kierroksella 1,  $i$  lähettää viestin  $(1, i, 1)$  jos ja vain jos sen alkuarvo on 1. Kierroksella  $2s - 1$ , vaiheen  $s$  ensimmäisellä kierroksella, missä  $2 \leq s \leq f + 1$ ,  $i$  lähettää viestin  $(1, i, 2s - 1)$  jos ja vain jos  $i$  on hyväksynyt viestejä vähintään  $f + s - 1$  eri prosessilta ennen kierrosta  $2s - 1$  ja  $i$  ei ole vielä lähettänyt viestiä.

$2(f + 1)$  kierroksen päätyttyä, prosessi  $i$  päättää arvon 1 jos ja vain jos  $i$  on hyväksynyt viestejä vähintään  $2f + 1$  eri prosessilta kierroksen  $2(f + 1)$  loppuun mennessä. Muuten  $i$  päättää arvon 0.

PolyByz-algoritmi ratkaisee binäärisen bysanttilaisen konsensusongelman, jos  $n > 3f$ . Todistusta ei sen pituuden vuoksi tässä esitetä, se löytyy lähteestä [Lyn96]. PolyByz ei ole kompleksisuudeltaan optimaalinen, esittelin sen tässä sen suhteellisen yksinkertaisuuden vuoksi. Garay ja Moses esittivät 1998  $O(f + 1)$  kierroksessa toimivan kommunikaatiovaativuudeltaan polynomisen algoritmin [GM98], mutta se on jonkin verran monimutkaisempi kuin äsken esitetty.

## **2.4 Muunnoksia ja laajennoksia**

### **Moniarvoinen päätöksenteko**

Edellä esitetyt algoritmit soveltuvat binääriseen päätöksentekoon; on valittava arvon 0 ja 1 välillä. Esittelen moniarvoiseen bysanttilaiseen konsensusongelmaan soveltuvan algoritmin, joka käyttää alirutiininaan binääristä BKO-algoritmia.

### ***TurpinCoan algoritmi***

Jokaisella prosessilla on paikalliset muuttujat  $x$ ,  $y$ ,  $z$  ja  $vote$ , missä  $x$ :lle alustetaan prosessin syötteenä saama arvo, ja  $y$ ,  $z$  ja  $vote$  alustetaan mielivaltaisesti.

1. *Kierros 1* Prosessi  $i$  lähettää arvonsa  $x$  kaikille prosesseille, myös itselleen. Jos tällä kierroksella vastaanotettujen viestin joukossa on  $\geq n - f$  kopiota jostain arvosta  $v \in V$ , prosessi  $i$  asettaa  $y = v$ , muuten  $y = null$ .
2. *Kierros 2* Prosessi  $i$  lähettää arvonsa  $y$  kaikille prosesseille, myös itselleen. Jos tällä kierroksella vastaanotettujen viestin joukossa on  $\geq n - f$  kopiota jostain  $V$ :n arvosta, prosessi  $i$  asettaa  $vote = 1$ , muuten  $vote = 0$ . Prosessi  $i$  asettaa arvoksi  $z$  sen  $ei - null$ -arvon joka esiintyy useimmin  $i$ :n tällä kierroksella vastaanottamien viestien joukossa, tasatilanteet ratkotaan arvalla. Jos kaikki vastaanotetut viestit ovat  $null$ -arvoisia, pysyy  $z$  määrittelemättömänä.
3. *Kierros  $r$* ,  $r \geq 3$  Prosessit ajavat binäärisen BKO-alirutiinin käyttämällä arvoja  $vote$  syötearvoina. Jos prosessin  $i$  päätös alirutiinissa on 1, ja  $z$  on määritelty, niin prosessin  $i$  lopullinen päätös on  $z$ , muuten se on oletusarvo  $v_0$ .

TurpinCoan-algoritmin tarvitsemien kierroksien lukumäärä on  $r + 2$ , missä  $r$  on binäärisen BKO-alirutiinin tarvitsemien kierroksien määrä. Kommunikaation tarve BKO-alirutiinin lisäksi on  $2n^2$  viestiä, viestiä kohden korkeintaan  $b$  bittiä, ollen siis yhteenlaskettuna  $O(n^2b)$  bittiä.

### **Autentikoitu viestinvälitys**

Viestien *autentikointi* on mielenkiintoinen rajoite bysanttilaiseen konsensusongelmaan, joka ansaitsee oman tarkastelunsa. Tässä mallissa jokainen prosessi  $i$  voi lisätä jokaiseen lähettämäänsä viestiin eräänlaisen digitaalisen allekirjoituksen, jolla viestin voi todentaa olevan todella peräisin prosessilta  $i$ . Oletuksena on, että vialliset prosessit eivät voi väärentää toisten prosessien allekirjoituksia, eivätkä näin voi väittää jotain viestiä tai informaatiota toisen prosessin lähettämäksi. Oletamme myös, että prosessien alkuarvot tulevat jostakin yhteisestä *lähteestä* joka myös allekirjoittaa nämä arvot. Luotettavat prosessit aloittavat alkutilassa yhden alkuarvon kanssa. Vialliset prosessit aloittavat mielivaltaisessa tilassa sisältäen jonkin joukon lähteen allekirjoittamia alkuarvoja. Tämän mallin [Lyn96] oikeellisuusehdoista päätös ja konsensus ovat samat kuin bysanttilaiselle konsensusongelmalle (määritelmä 2.2), mutta validiusehto on seuraavanlainen:

**validisuus** Jos kaikki prosessit aloittavat täsmälleen yhdellä alkuarvolla  $v \in V$ , joka on lähteen allekirjoittama, niin  $v$  on ainoa mahdollinen päätösarvo luotettaville prosesseille.

Autentikoidulla protokollalla viallisten prosessien mielivaltainen käyttäytyminen voidaan tulkita virheeksi toimittaa kaikkia viestejä tarkoitetuille kohteilleen, mikä voidaan rinnastaa pysähtymisvirhemalliin. BK-ongelma autentikoidulla viestinvälityksellä,  $f$  virheellisellä prosessilla, ratkeaa jo  $f + 2$  prosessilla. Vaadittavien kierrosten lukumäärä pysyy kuitenkin samana kuin yleisessä tapauksessa, ollen  $f + 1$  [Lyn96].

### Ei-täydelliset verkot

Tähän mennessä olemme tarkastelleet bysanttilaista konsensusongelmaa täydellisillä verkoilla. Ongelman voi eräin rajoituksin yleistää yleisille verkoille. Mikäli verkko on puu, bysanttilaista konsensusongelmaa ei voida ratkaista edes yhdellä vialliselä prosessilla, koska jos tämä prosessi ei ole puun lehtisolmu, se voi korruptoida täysin puun eri puolilla olevien prosessien välisen kommunikaation. Vastaavasti jos  $f$  solmun poistaminen tekee verkon epäyhtenäiseksi, BKO:ta ei voida ratkaista  $f$  viallisella prosessilla kyseisessä verkossa.

Verkon  $G$  yhteydellisyys (connectivity),  $conn(G)$  on minimimäärä solmuja, joiden poistaminen tekee verkon epäyhtenäiseksi. Verkko on *c-yhdistetty* (*c-connected*), jos  $conn(G) \geq c$ . Mengerin teoreeman mukaan verkko  $G$  on *c-yhdistetty* jos ja vain jos mitkä tahansa kaksi solmua verkossa voidaan yhdistää toisiinsa vähintään  $c$  solmultaan erillisellä polulla. Bysanttilainen konsensusongelma voidaan ratkaista  $n$  solmun verkossa  $G$  jossa on korkeintaan  $f$  virheellistä solmua, jos ja vain jos  $n > 3f$  ja  $conn(G) > 2f$ . Todistus löytyy esimerkiksi Lynchin kirjasta [Lyn96]. Jotain kuvaa todistuksesta saa ajattelemalla, että ehdon  $conn(G) > 2f$  täytyessä jokaisen kahden solmun  $i, j$  välillä on vähintään  $f + 1$  kokonaan luotettavista solmuista koostuvaa reittiä, joten enemmistö  $i$ :n vastaanottamista  $j$ :n lähettämistä viesteistä on luotettavia.

## Lähteet

- DS82 Dolev, D. ja Strong, H. R., Polynomial algorithms for multiple processor agreement. *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, New York, NY, USA, 1982, ACM Press, sivut 401–407.
- FL82 Fischer, M. J. ja Lynch, N. A., A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14,4(1982), sivut 183–186. URL [citeseer.ist.psu.edu/fischer81lower.html](http://citeseer.ist.psu.edu/fischer81lower.html).
- GM98 Garay ja Moses, Fully polynomial byzantine agreement for  $n > 3t$  processors in  $t+1$  rounds. *SICOMP: SIAM Journal on Computing*, 27(1998). URL [citeseer.ist.psu.edu/garay98fully.html](http://citeseer.ist.psu.edu/garay98fully.html).
- LHA91 Lala, J. H., Harper, R. E. ja Alger, L. S., A design approach for ultra-reliable real-time systems. *Computer*, 24,5(1991), sivut 12–22.
- LSP82 Lamport, L., Shostak, R. ja Pease, M., The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4,3(1982), sivut 382–401.

- Lyn96 Lynch, N. A., *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.
- WLG<sup>+</sup>78 Wensley, J., Lamport, L., Goldberg, J., Green, M., Levitt, K., Melliar-Smith, P., Shostak, R. ja Weinstock, C., Sift: Design and analysis of a fault-tolerant computer for aircraft control. *Proceedings of the IEEE*, 66(1978), sivut 1240–1255.