

Itsestabiloituva johtajan valinta vakio-tilassa

Jouni Siren

Helsinki 29.10.2007

Seminaarikirjoitelma

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

1 Johdanto

Johtajan valinta (engl. *leader election*) [Lyn99, luku 3] on eräs hajautettujen järjestelmien perusongelmista. Keskenään samanlaisten toimijoiden yhteistoiminta on monissa tilanteissa vaikeaa tai jopa mahdotonta. Vastaavasti monen ongelman ratkaiseminen helpottuu, jos yksi toimijoista koordinoi järjestelmän toimintaa ongelman ratkaisemiseksi. Joissain tilanteissa tämä koordinoija voidaan määrittää etukäteen, kun taas toisinaan valinta joudutaan tekemään järjestelmän suorituksen aikana. Tätä koordinoijan valitsemista samankaltaisten toimijoiden keskuudesta kutsutaan johtajan valinnaksi.

Tarve koordinoijan valitsemiseen voi uusiutua, jos järjestelmä päätyy häiriöiden takia virheelliseen tilaan. Tällöin haasteena on paitsi virheestä toipuminen myös häiriöiden ilmaantumisen ja niiden päättymisen havaitseminen. Yleisluontoisin ratkaisu virheistä toipumiseen ovat itsestabiloituvat järjestelmät [Dij74, Sch93, Dol00], jotka pystyvät häiriöiden päätyttyä palautumaan mistä tahansa tilasta takaisin hyväksyttävään tilaan.

Eräs tavallisimmista häiriötyypeistä on järjestelmän rakenteen muuttuminen. Järjestelmään järjestelmään voi tulla lisää tai siitä voidaan poistaa koneita ja niiden välisiä yhteyksiä erilaisista syistä. Niinpä olisikin toivottavaa, että järjestelmä pystyisi sopeutumaan muutoksiin mahdollisimman läpinäkyvästi. Eräs ratkaisu tähän ovat vakio-tilassa toimivat protokollat, jotka eivät tarvitse minkäänlaista tietoa verkon rakenteesta kokonaisuudessaan vaan toimivat koneen itsensä ja sen välittömien naapureiden tilojen perusteella. Jos tällaisesta protokollasta voidaan tehdä itsestabiloituva, sopeutuu se suuriinkin muutoksiin verkon koossa ja rakenteessa ilman, että yksittäisten koneiden suorittamia protokollia tarvitsisi muuttaa.

Tämä kirjoitelma käsittelee itsestabiloituvaa johtajan valintaa vakio-tilassa. Luvussa 2 määritellään hajautetun laskennan mallit, joissa johtajan valintaa tarkastellaan. Luvussa 3 esitetään mahdottomuus- ja alarajatuloksia, jotka estävät ongelman ratkaisemisen joissain malleissa. Lopuksi luvussa 4 esitetään joitain protokollia, jotka ratkaisevat ongelman tietyin rajoituksin.

2 Hajautetun laskennan mallit

Hajautettuja järjestelmiä on mallinnettu monin erilaisin tavoin. Seuraavaksi esitellään lyhyesti malleja, joita käytetään tässä työssä.

2.1 Hajautettu järjestelmä

Tulkitaan hajautettu järjestelmä verkoksi G , jonka solmujen joukko on V ja kaarten joukko E . Verkko voi olla suunnattu tai suuntaamaton. Merkitään myös verkon solmujen määrää $|V| = n$. Koska tässä työssä tarkastellaan vakiotilassa toimivia protokollia, kunkin solmun voidaan olettaa olevan tilakone. Tyypillisesti oletetaan, että solmussa oleva tilakone riippuu vain solmuun saapuvien ja siitä lähtevien kaarten määrästä mutta ei verkon laajemmasta rakenteesta.

Verkkoa sanotaan **tasalaatuiseksi** (engl. *uniform*) [BP89, DIM93], jos kaikki sen solmut ovat keskenään samanlaisia. Tasalaatuksena voidaan pitää myös verkkoa, jossa tilakoneen tyyppi riippuu solmuun saapuvien ja siitä lähtevien kaarten määristä. Monien hajautetun laskennan ongelmien ratkaiseminen on tällaisissa verkoissa mahdotonta (esimerkiksi [Dij74, BP89]), sillä niiden ratkaiseminen edellyttää solmujen välisen symmetrian rikkomista. Eräs ratkaisu on antaa solmuille yksilölliset **tunnisteet** (engl. *identity*), joita hyödyntämällä symmetria voidaan rikkoa. Tavallisesti oletetaan, että mahdollisia tunnisteita on paljon enemmän kuin verkossa on solmuja, jolloin todennäköisyys kunkin tietyn tunnisteen esiintymiselle verkossa on hyvin pieni.

Usein tarkastellaan erityisesti **rengasverkkoja** (engl. *ring*), joiden solmujoukko on $V = \{0, 1, \dots, n - 1\}$ ja kaarten joukko $E = \{(i, i + 1 \bmod n) \mid 0 \leq i < n\}$. Jos rengasverkko on suunnattu, määrittää kaarten suunta renkaan kiertosuunnan. Tällaisessa rajatussa verkossa toimivien protokollien käytännön merkitys vaikuttaa ensikatsomalta vähäiseltä. **Reilua koostamista** (engl. *fair protocol combination, fair composition*) [AV91, DIM93, Sch93, Dol00] käytämällä rengasverkon protokollat voidaan kuitenkin yhdistää itsestabiloituvan virittävän puun [AG94] virittämään sykliin, jolloin ne saadaan toimimaan myös yleisissä verkoissa.

Merkitään solmun $v \in V$ tilaa $s(v)$ ja tilojen joukkoa S_v . Solmun v edeltäjien joukko $p(v)$ koostuu niistä solmuista u , joista lähtee (suunnattu tai suuntaamaton) kaari $(u, v) \in E$ solmuun v . Kukin tilakone koostuu **siirtymäehdoista** (engl. *privilege, guard*) ja niitä vastaavista **tilasiirtymistä** (engl. *transition*). Siirtymäehto on totuusarvoinen funktio g solmun v ja sen edeltäjien $p(v)$ tiloilta. Jos siirtymäehto on tosi, jää tilakone odottamaan **suoritusvuoroa**, jonka saadessaan se muuttaa tilaansa siirtymäehtoa vastaavan tilasiirtymän mukaisesti. Jatkossa tilasiirtymiä merkitään

$$s(v)s(u_1)s(u_2) \dots s(u_k) \rightarrow s',$$

missä solmut u_1, u_2, \dots, u_k ovat solmun v edeltäjiä, siirtymäehto toteutuu solmun v ollessa tilassa $s(v)$ ja sen edeltäjien ollessa tiloissa $s(u_i)$ ja ehtoa vastaava siirtymä on tilaan s' .

Jos tilakoneen siirtymäehdoista voi toteutua samanaikaisesti korkeintaan yksi, sanotaan tilakonetta **deterministiseksi**. Muussa tapauksessa tilakone on **satunnainen** (engl. *randomized*), ja se suorittaa vuoron saadessaan toteutuneita siirtymäehtoja vastaavista tilasiirtymistä yhden satunnaisesti valitun. Satunnaisuutta voidaan usein käyttää symmetrian rikkomiseen tilanteissa, joissa ongelman ratkaiseminen deterministisiä tilakoneita käyttämällä ei olisi mahdollista.

2.2 Itsestabilointi

Merkitään hajautetun järjestelmän tilojen joukkoa $S = \prod_{v \in V} S_v$. Tavallista on, että oikein toimiessaan järjestelmä voi olla vain pienessä osassa tiloista $s \in S$. Kutsutaan näitä tiloja **turvallisiksi** (engl. *safe*). Turvallisesta tilasta alkava laskenta voi käydä vain turvallisissa tiloissa; kutsutaan tällaista laskentaa **kelvolliseksi**.

On kuitenkin tavallista, että hajautettu järjestelmä päätyy virheelliseen tilaan järjestelmässä tapahtuvien muutosten tai solmuissa tai tietoliikenneyhteyksissä tapahtuvien virheiden vuoksi. Siksi onkin tarpeen, että järjestelmä pystyy itse toipumaan tällaisista virheistä. Perinteinen tapa on ollut pyrkiä ennakoimaan erilaisia virhetyppejä ja kehittää erikseen menetelmiä niistä toipumiseen. Sen ongelmana on kuitenkin usein erilaisten virhetyyppien suuri määrä sekä mahdolliset ennakoimattomat virheet.

Itsestabiloituvat (engl. *self-stabilizing*) järjestelmät [Dij74, Sch93, Dol00] tarjoavat yleisemmän lähestymistavan virheistä toipumiseen. Erikseen määritellyistä virheistä toipumisen sijaan itsestabiloituva järjestelmä pystyy palautumaan äärellisessä ajassa mistä tahansa tilasta $s \in S$ johonkin turvalliseen tilaan s' , jos uusia virheitä ei toipumisen aikana ilmene. Tällainen toipumismekanismi mahdollistaa myös ennalta tuntemattomista virheistä toipumisen, mutta varjopuolena saattaa olla hitaampi toipuminen tunnetuista virheistä.

2.3 Ajastimet

Hajautetussa järjestelmässä solmujen tilasiirtymät eivät tapahdu missään ennalta määrättyssä järjestyksessä. Tätä mallinnetaan tyypillisesti **ajastimella** (engl. *dae-*

mon), joka valitsee, mitkä solmut pääsevät suorittamaan tilasiirtymänsä seuraavaksi. Yleisin ajastin on **hajautettu ajastin** (engl. *distributed daemon, asynchronous daemon*) [Lyn99, luku 8], joka voi valita suoritusvuoroon minkä tahansa osajoukon sitä odottavista solmuista. Kaikki suoritusvuoron saaneet solmut suorittavat tällöin siirtymänsä samanaikaisesti.

Hajautetun ajastimen kaksi keskeistä erikoistapausta ovat **keskusajastin** (engl. *centralized daemon*) [Dij74, BP89, DIM93] ja **synkroninen ajastin** (engl. *synchronized daemon*) [Lyn99, luku 2]. Keskusajastinta käytettäessä suoritusvuoroon valitaan yksi solmu kerrallaan, joten järjestelmässä ei varsinaisesti esiinny rinnakkaisuutta. Synkroninen ajastin puolestaan valitsee kaikki suoritusvuoroa odottavat solmut. Jos jonkin ongelman ratkaiseminen ei ole mahdollista keskusajastinta (synkronista ajastinta) käytettäessä, ei se onnistu myöskään hajautetulla ajastimella. Jos nimittäin protokolla tuottaa väärän ratkaisun jollain keskusajastimen (synkronisen ajastimen) valitsemalla suoritusjärjestyksellä, saattaa myös hajautettu ajastin valita tämän järjestyksen.

Algoritmin pahimman tapauksen käyttäytymistä tarkasteltaessa ajastinta ajatellaan usein vastustajana, joka pyrkii valitsemaan tavoitteiden kannalta mahdollisimman huonon suoritusjärjestyksen. Usein ajastimen oletetaan kuitenkin olevan **reilu** (engl. *fair*) [Lyn99, luku 8]: äärettömässä suorituksessa reilu ajastin antaa jokaiselle suoritusvuoroa äärettömän usein odottavalle solmulle suoritusvuoron äärettömän monta kertaa. Jos ajastin ei ole reilu, sitä sanotaan **epäreiluksi** (engl. *unfair*). Voidaan ajatella, että tällainen ajastin pyrkii jättämään solmun, jonka suorituksesta protokollan eteneminen riippuu, vuorotta. Epäreilulla ajastimella toimivat protokollat suunnitellaankin usein niin, että vain yksi solmu kerrallaan odottaa suoritusvuoroa, jolloin ajastimen on pakko valita se.

3 Alarajoja ja mahdottomuustuloksia

Algoritmien suunnittelemisessa hajautettuihin järjestelmiin on monia haasteita, joita perinteisten algoritmien parissa ei ole. Eräs keskeisimmistä on tarve samassa tilassa olevien naapurustoltaan identtisten solmujen välisen symmetrian rikkomiseen. Usein voidaan osoittaa, ettei halutun ongelman ratkaisevaa protokollaa ole olemassa, jos symmetriaa ei voida rikkoa esimerkiksi satunnaislukuja tai solmujen yksilöllisiä tunnisteita hyödyntämällä. Myös samanaikaisuus luo haasteita: kahden eri solmun tilasiirtymät, joista kumpikin yksinään pitäisi järjestelmän turvallisessa ti-

lassa, voivat samanaikaisesti suoritettuina viedä järjestelmän turvattomaan tilaan. Tällaisten haasteiden voittaminen edellyttää sekin usein erilaisten oletusten tekemistä. Vaatimus vakio-tilassa toimimisesta asettaa sekin omat haasteensa: solmuihin ei esimerkiksi voida tallentaa toisten solmujen tunnisteita ja riittävän suuressa verkossa on aina useita solmuja, jotka ovat samassa tilassa. Tässä luvussa tarkastellaan sitä, millaisia oletuksia tekemällä vakio-tilassa toimivia protokollia itsestabiloituvaan johtajan valintaan on ylipäänsä olemassa.

3.1 Rengasverkot

Ehkäpä kaikkein perustavinta laatua oleva rajoitus johtajan valinnalle on se, ettei deterministinen protokolla voi rikkoa symmetriaa synkronisesti ajastetuissa tasalaatuisissa rengasverkoissa [Lyn99, luku 3]. Jos verkon kaikki solmut ovat aluksi keskenään samassa tilassa, tulevat ne jatkossakin olemaan aina samassa tilassa. Sama tulos pätee luonnollisesti myös hajautetulla ajastimella. Näin ollen johtajan valinta rengasverkossa edellyttää aina satunnaisuutta, tunnisteita, toisenlaista ajastinta tai symmetrian rikkomista asettamalla solmut etukäteen eri tiloihin.

Edes keskusajastimen käyttäminen ei auta tasalaatuisissa rengasverkoissa, jos johtajan valinnan on tapahduttava itsestabiloituvasti [Dij74, BP89]¹. Jos nimittäin verkossa vallitsee k -kertainen symmetria millä tahansa solmujen määrän n aidolla tekijällä k , voi ajastin valita suoritusjärjestyksen niin, että symmetria säilyy. Tämän seurauksena verkossa voi olla k johtajaa, joista mikään deterministinen protokolla ei pääse eroon. Ongelmaa ei kuitenkaan esiinny silloin, kun solmujen määrä on alkuluku. Käytännössä tällaisen oletuksen tekeminen ei kuitenkaan ole kovin realistista.

Vaikka solmujen määrä olisi alkuluku, tarvitaan silti ainakin n tilaa, jos rengasverkko on suunnattu [BGJ99]. Todistetaan tämä vastaoletuksen kautta. Oletetaan, että on olemassa tasalaatuisissa suunnatuissa rengasverkoissa keskusajastimella toimiva deterministinen itsestabiloituva johtajanvalintaprotokolla, jossa solmuilla on $m < n$ tilaa. Tilakoneen tilat voidaan numeroida niin, että siinä on siirtymät

$$s_i s_i \rightarrow s_{(i+1) \bmod k} \quad \text{kaikilla } i < k$$

jollain $k \leq m$. Tätä k tilan sykliä hyödyntämällä voidaan muodostaa alkutilanne, josta lähdeittäessä keskusajastin voi jakaa suoritusvuoroja niin, ettei johtajaa saada koskaan valittua.

¹Tulos on esitetty yksityiskohtaisemmin Jukka Suomelan seminaarityössä.

Yksi tällainen alkutilanne on

$$C_0 = C_{0,0} = (s_0^{n-k+1}, s_1, s_2, \dots, s_{k-1}),$$

missä solmujen tilat luetellaan renkaan mukaisessa järjestyksessä. Aloittamalla oikeanpuoleisimmasta tilassa s_0 olevasta solmusta ja soveltamalla tilojen numerointiin käytettyjä siirtymiä siitä eteenpäin päästään $i < k$ askeleella tilanteeseen

$$C_{0,i} = (s_0^{n-k}, s_1, s_2, \dots, s_i, s_i, \dots, s_{k-1})$$

ja tilanteesta $C_{0,k-1}$ edelleen yhdellä askeleella tilanteeseen

$$C_1 = (s_0^{n-k}, s_1, s_2, \dots, s_{k-1}, s_0).$$

Vastaavaa toistamalla voidaan aina k askeleella saavuttaa seuraava tilanteista

$$C_i = (s_0^{n-k+1-i}, s_1, s_2, \dots, s_{k-1}, s_0^i),$$

missä $i \leq n - k + 1$, kunnes aikanaan päädytään tilanteeseen

$$C_{n-k+1} = (s_1, s_2, \dots, s_{k-1}, s_0^{n-k+1}).$$

Samaa edelleen jatkamalla edelleen päästään k askeleen välein tilanteisiin

$$C_{n-k+i} = (s_i, s_{i+1}, \dots, s_{k-1}, s_0^{n-k+1}, s_1, \dots, s_{i-1}),$$

missä $i \leq k$, kunnes lopulta saavutetaan tilanne $C_n = C_0$.

Verkon solmujen tiloissa toistuu siis kn askeleen sykli, jonka alussa ja lopussa verkko on samassa tilanteessa ja jonka aikana kukin solmu käy läpi k eri tilaa. Koska ajastin voi pakottaa saman toistumaan loputtomiin, ei johtajaa saada koskaan valittua. Oletus halutunlaisen protokollan olemassaolosta oli siis väärä, joten johtajan valinta näillä edellytyksillä vaatii solmuilta ainakin n tilaa.

3.2 Tunnisteelliset verkot

Jos itsestabiloituvan deterministisen protokollan on toimittava vakio-tilassa, voidaan solmujen yksilöllisiä tunnisteita hyödyntävä protokolla toteuttaa myös tasalaatuisessa verkossa [BGJ99]. Tämä seuraa siitä, että jos solmulla on k edeltäjää ja S mahdollista tilaa, vastaa sen toiminta jotain $(S+1)^{S^{k+1}}$ mahdollisesta tilakoneesta. Koska käytettävä tilakone perustuu vain edeltäjien määrään ja solmun tunnisteeseen, on olemassa ainakin yksi tilakone M_k , jota mielivaltaisen moni k edeltäjän solmu käyttää tällaisten solmujen määrän kasvaessa rajatta.

Kaikilla k on siis olemassa äärettömän monta tunnistetta, joilla k edeltäjän solmun toiminta vastaa tilakonetta M_k . Verkon solmujen tunnisteet voidaan siis jakaa uudelleen niin, että kukin k edeltäjän solmu saa tilakoneeseen koneen M_k . Protokollan toiminta ei siis riipukaan solmujen tunnisteista vaan ainoastaan niiden naapureiden määrästä, joten se voitaisiin toteuttaa myös tasalaatuisessa verkossa.

Kaikki tasalaatuisessa verkossa toimivia deterministisiä itsestabiloituvia vakiotilan protokollia koskevat mahdottomuustulokset ovat siis voimassa myös tunnisteita käyttävissä verkoissa. Erityisesti siis johtajan valinta onnistuu tällä tavalla rengasverkossa vain, jos solmujen määrä on alkuluku, vastustajana on keskusajastin ja verkko on suuntaamaton. Tällaisen protokollan ovat esittäneet Itkis, Lin ja Simon [ILS95].

Esitetty mahdottomuustulos perustuu kuitenkin hyvin voimakkaasti siihen, että mahdollisia tunnisteita on rajattomasti, eikä vakiotilassa toimiva solmu siten pysty hyödyntämään tunnistetta millään merkityksellisellä tavalla. Käytännössä tunnisteet ovat kuitenkin lähes aina rajatun mittaisia, kuten esimerkiksi verkkosovittimen 48-bittinen MAC-osoite. Seuraavassa luvussa esitetäänkin vakiotilassa toimiva protokolla, joka hyödyntää rajatun mittaisia tunnisteita ja ratkaisee johtajan valinta-ongelman tilanteessa, jossa se ei rajoittamattomilla tunnisteilla ole mahdollista.

Hiljaisen itsestabiloituvan johtajan valinnan on osoitettu vaativan $\Omega(\log n)$ bittia linkkiä kohti riippumatta siitä, onko verkko tasalaatuinen vai tunnisteita käyttävä, onko protokolla deterministinen vai satunnainen, millainen on verkon rakenne ja millaista ajastinta käytetään vastustajana [DGS99]. Rajatun mittaisten tunnisteiden käyttäminen mahdollistaa kuitenkin tämänkin rajan kiertämisen ainakin joissain tapauksissa — asettamalla raja tunnisteiden pituudelle asetetaan samalla (hyvin suuri) raja verkon koolle, jolloin $\Omega(\log n)$ onkin sama kuin $\Omega(1)$.

4 Protokollia

Tässä luvussa esitellään joitain protokollia itsestabiloituvaan johtajan valintaan vakiotilassa.

4.1 Hiljainen protokolla rajoitetuilla tunnisteilla

Vakiotilan itsestabiloituva johtajan valinta deterministisellä protokollalla ei normaalisti onnistu rengasverkossa eikä hiljaisena missään verkossa. Beauquier, Gradinariu

ja Johnen ovat kuitenkin esittäneet protokollan, joka täyttää kaikki nämä vaatimukset keskusajastinta vastaan [BGJ99]. Protokolla perustuu rajatun mittaisten tunnisteiden käyttämiseen.

Olkoot solmujen tunnisteet kokonaislukuja väliltä $[0, n + k]$ jollain kokonaisluvulla k . Tällöin pienin verkossa oleva tunniste on korkeintaan $k + 1$. Kaikki solmut, joiden tunniste on välillä $[0, k + 1]$, ovat potentiaalisia johtajia. Tällaisia solmuja on verkossa korkeintaan $k + 2$. Protokolla perustuu siihen, että kukin solmu ylläpitää listaa käsityksestään siitä, mitkä potentiaaliset johtajat ovat verkossa, sekä siitä, onko solmu itse mielestään johtaja.

Solmu, joka ei ole potentiaalinen johtaja, kopioi edeltäjänsä listan itselleen, jos se on muuttunut. Potentiaalinen johtaja puolestaan laittaa itsensä listan alkuun ja kopioi perään $k + 1$ ensimmäistä tunnistetta edeltäjänsä listalta, jos ne ovat muuttuneet. Käytännössä listalle tulee siis $k + 2$ viimeisimmän potentiaalisen johtajan tunnisteet renkaan suunnan vastaisessa järjestyksessä. Johtajaksi valitaan se potentiaalinen johtaja, jonka tunniste on listalla olevista pienin.

Protokolla on hiljainen, sillä kaikilla solmuilla on aikanaan totuudenmukaiset listat niitä edeltävistä potentiaalisista johtajista, eikä mikään solmu tämän jälkeen enää päivitä listaansa. Kukin solmu tarvitsee $2 \cdot (k + 2)^{k+2}$ tilaa, mikä on vakio suhteessa solmujen määrään n , jos luku k on. Tästä syystä protokolla ei ole kuitenkaan kovin käytännöllinen: mahdollisten tunnisteiden määrän tulee olla lähellä solmujen määrää. Protokolla ei siis sopeudu kovin hyvin muutoksiin verkon koossa, minkä lisäksi tunnisteiden jakaminen itsessään saattaa olla vaikea tehtävä.

Käytännössä toimivampaa onkin luultavasti käyttää protokollia, jotka perustuvat siihen, että kukin solmu pitää yllä käsitystään siitä, mikä on senhetkisen johtajan tunniste (esimerkiksi [AG94]). MAC-osoitteen kaltaisilla todellisilla rajoitetun mittaisten tunnisteilla tällaiset protokollat toimivat vakiotilassa.

4.2 Satunnaisprotokolla hajautetulla ajastimella

Beauquier, Gradinariu ja Johnen esittävät myös satunnaisprotokollan, joka toimii tasalaatuisissa suunnatuissa rengasverkoissa jopa hajautetulla epäreilulla ajastimella [BGJ99]. Protokolla edellyttää tietoa verkon koosta: jokaisella solmulla on $2 \cdot m^3$ tilaa jollain kokonaisluvulla m , joka ei ole solmujen määrän n tekijä. Tästä syystä sen tilavaativuus ei ole vakio, vaikka onkin käytännössä hyvin pieni, jos m on pienin tällainen luku.

Jokaisella solmulla on kolme muuttujaa, jotka saavat arvoja väliltä $[0, m)$: deterministinen vuoromerkki, johtajan merkki ja värimerkki. Näiden lisäksi solmulla on värinään joko sininen tai vihreä. Solmulla v on muuttujaa $x(v)$ vastaava vuoromerkki, jos $(x(v) - x(p(v))) \not\equiv 1 \pmod{m}$, missä $p(v)$ on solmun v edeltäjä. Kun solmu antaa vuoromerkin eteenpäin, se kasvattaa sitä vastaavaa muuttujaa yhdellä \pmod{m} . Jokaisista vuoromerkkityypistä on verkossa aina vähintään yksi, koska m ei ole solmujen määrän tekijä.

Solmu v voi toimia vain, jos sillä on hallussaan deterministinen vuoromerkki. Toimituaan se antaa tämän vuoromerkin eteenpäin. Tehdäkseen mitään muuta solmulla täytyy olla hallussaan vuoromerkin lisäksi myös värimerkki. Tällöin protokolla valitsee satunnaisesti, tekeekö v jotain ennen kuin se antaa vuoromerkin seuraavalle solmulle. Solmun v tilasta riippuen sillä on kolme toimintavaihtoehtoa:

1. Solmulla v ei ole johtajamerkkiä. Tällöin se omaksuu edeltäjänsä värin ja antaa värimerkin eteenpäin.
2. Solmulla v on johtajamerkki ja sen väri poikkeaa edeltäjän väristä. Tällöin v omaksuu edeltäjänsä värin ja antaa väri- ja johtajamerkit eteenpäin.
3. Solmulla v on johtajamerkki ja sen väri on sama kuin edeltäjällä. Tällöin se valitsee satunnaisesti uuden värin ja antaa värimerkin eteenpäin.

Väri- ja johtajamerkkien etenemisnopeudet ovat satunnaisia. Näin ollen jos verkossa on useampi samanlainen merkki, kohtaavat ne ennemmin tai myöhemmin toisensa ja sulautuvat yhteen, kunnes jäljelle jää enää yksi merkki kumpaakin tyyppiä.

Potentiaallinen johtaja käyttää värimerkkiä sen selvittämiseen, onko verkossa muita johtajia. Värimerkki värjää solmut samalla värillä kuin sen lähettänyt potentiaallinen johtaja (tapaus 1). Jos potentiaallinen johtaja saa haltuunsa eri värisen värimerkin kuin se itse lähetti, se tietää, että verkossa on muitakin potentiaalisia johtajia. Tällöin se valitsee satunnaisesti, antaako se johtajamerkin eteenpäin ja omaksuu toisen potentiaalisen johtajan värin (tapaus 2). Jos taas potentiaallinen johtaja saa haltuunsa omaa väriään vastaavan värimerkin, se pitää johtajamerkin hallussaan ja kokeilee uudestaan satunnaisella värillä, onko verkossa muita potentiaalisia johtajia (tapaus 3). Jos siis verkossa on vain yksi potentiaallinen johtaja, pysyy johtajamerkki koko ajan tämän hallussa.

4.3 Satunnaisprotokolla yleisissä verkoissa

Itkis ja Levin ovat esittäneet itsestabiloituvan satunnaisprotokollan johtajan valitsemiseksi missä tahansa tasalaatuisessa verkossa [IL94]. Protokolla perustuu virittävän puun muodostamiseen. Verkon on oltava suuntaamaton, sillä solmujen syvyydet on tallennettu vakiotilan saavuttamiseksi hajautetusti ja puun eheyden tarkistaminen edellyttää siksi kaksisuuntaista liikennettä solmun ja sen naapureiden välillä. Vastustajana Itkis ja Levin käyttävät reilua keskusajastinta, joka tuntee ennalta kunkin solmun satunnaisbitit ja voi myös rajoitetusti vaikuttaa niiden jakaumaan.

Satunnaisuuden käytön osalta Itkisin ja Levinin protokolla on poikkeuksellinen. Tavallisesti oletetaan, ettei vastustaja tunne ennalta käytettyjä satunnaisbittejä. Tämän seurauksena vastustaja ei voi mukauttaa käyttämäänsä strategiaa protokollan tulevan toiminnan mukaiseksi, vaan joutuu valitsemaan yleisluontoisemman strategian, joka on riittävän suurella todennäköisyydellä huono protokollan tavoitteiden kannalta. Itkis ja Levin kuitenkin olettavat vain, ettei mikään solmu saa liian monta samaa satunnaisbittiä peräkkäin, vaan kukin solmu vaihtaa toimintatapaansa riittävän usein. Tavallaan he siis käyttävät satunnaisuutta symmetrian rikkomiseen antamalla jokaiselle solmulle satunnaisen tunnisteiden, joka vaikuttaa solmun toimintaan.

5 Lopuksi

Vakiotilassa toimivat itsestabiloituvat protokollat edustavat eräänlaista läpinäkyvyyden huipentumaa. Ne pystyvät toipumaan automaattisesti mielivaltaisista virheistä ja mukautuvat samalla mielivaltaisiin muutoksiin verkossa, koska eivät tarvitse verkon koosta tai rakenteesta naapurustoaan laajempaa tietoa. Käytännössä ainakaan johtajan valinnan kohdalla näin pitkälle ei kuitenkaan päästä, sillä halutunlaisen protokollan muodostaminen on osoittautunut mahdottomaksi monissa malleissa.

Determinististen protokollien heikkoutena on tarve tunnisteiden käyttöön. Vaikka monessa tilanteissa verkon solmuilla on luonnolliset tunnisteet, ei sellaisten olemassaolo ole taattua kaikissa sovelluksissa (esimerkiksi [FJ06]). Käytännöllinen ja usein myös yksinkertaisempi vaihtoehto symmetrian rikkomiseen onkin satunnaisuuden hyödyntäminen.

Niin determinististen kuin satunnaisprotokollienkin haasteena on hajautetuille järjestelmille ominainen samanaikaisuus. Monissa protokollissa oletetaan, ettei tämä

samanaikaisuus ole aitoa, vaan samojen solmujen tilatietoja käyttävistä solmuista korkeintaan yksi on kerrallaan suoritusvuorossa. Käytännössä tämän oletuksen paikkansapitävyys voi olla kyseenalaista etenkin löyhästi kytketyissä järjestelmissä. Rengasverkoille on pystytty esittämään protokolla, jonka oikeellisuus ei edellytä oletuksia rajoitetusta samanaikaisuudesta, mutta ainakin toistaiseksi on epäselvää, säilyykö protokollan kyky sietää samanaikaisuutta myös yritettäessä muuttaa se toimimaan yleisissä verkoissa reilua koostamista käyttämällä.

Lähteet

- AG94 Arora, A. ja Gouda, M. G., Distributed reset. *IEEE Transactions on Computers*, 43,9(1994), sivut 1026–1038.
- AV91 Awerbuch, B. ja Varghese, G., Distributed program checking: a paradigm for building self-stabilizing distributed protocols. *Proceedings of the 32nd annual symposium on Foundations of computer science (FOCS 1991)*. IEEE, 1991, sivut 258–267.
- BGJ99 Beauquier, J., Gradinariu, M. ja Johnen, C., Memory space requirements for self-stabilizing leader election protocols. *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing (PODC 1999)*. ACM Press, 1999, sivut 199–207.
- BP89 Burns, J. E. ja Pachl, J., Uniform self-stabilizing rings. *ACM Transactions on Programming Languages and Systems*, 11,2(1989), sivut 330–344.
- DGS99 Dolev, S., Gouda, M. G. ja Schneider, M., Memory requirements for silent stabilization. *Acta Informatica*, 36,6(1999), sivut 447–462.
- Dij74 Dijkstra, E. W., Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17,11(1974), sivut 643–644.
- DIM93 Dolev, S., Israeli, A. ja Moran, S., Self-stabilization of dynamic systems assuming only read/write atomicity. *Distributed Computing*, 7,1(1993), sivut 3–16.
- Dol00 Dolev, S., *Self-stabilization*. The MIT Press, ensimmäinen painos, 2000.

- FJ06 Fischer, M. ja Jiang, H., Self-stabilizing leader election in networks of finite-state anonymous agents. *Principles of Distributed Computing (OPODIS 2006)*, osa 4305 sarjasta *Lecture Notes in Computer Science*. Springer-Verlag, 2006, sivut 395–409.
- IL94 Itkis, G. ja Levin, L., Fast and lean self-stabilizing asynchronous protocols. *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1994)*. IEEE, 1994, sivut 226–239.
- ILS95 Itkis, G., Lin, C. ja Simon, J., Deterministic, constant space, self-stabilizing leader election on uniform rings. *Proceedings of the 9th International Workshop on Distributed Algorithms*, osa 972 sarjasta *Lecture Notes in Computer Science*. Springer-Verlag, 1995, sivut 288–302.
- Lyn99 Lynch, N. A., *Distributed Algorithms*. Morgan Kaufmann Publishers, ensimmäinen painos, 1999.
- Sch93 Schneider, M., Self-stabilization. *ACM Computing Surveys*, 25,1(1993), sivut 45–67.