

Itsestabilointi: perusmääritelmiä ja klassisia tuloksia

Jukka Suomela

Helsinki 8.10.2007

seminaarikirjoitelma

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

1 Johdanto

Yhdessä tietokoneessa toimivaa ohjelmaa suunniteltaessa lähdetään yleensä oletuksesta, että laitteisto toimii oikein eikä sen kokoonpano muutu kesken ohjelman suorituksen. Kun siirrytään kohti laajoja hajautettuja järjestelmiä, tilanne muuttuu oleellisesti.

Jo yksittäisten laitteiden *suuri määrä* merkitsee suurta todennäköisyyttä sille, että jokin laitteista vikaantuu. Hajautettuun järjestelmään liittyy uusia vikamahdollisuuksia, joita yksittäisessä tietokoneessa ei ole: keskeisimpiä näistä ovat *tietoliikenneyhteydet*.

Erityisesti kannettavista laitteista koostuvissa hajautetuissa järjestelmissä kokoonpanon muutokset ovat tavallisia. Kannettavat laitteet ovat tyypillisesti *akku- tai paristokäyttöisiä*, jolloin on odotettavissakin, että joistain laitteista voi loppua virta kesken järjestelmän käytön. Tietoliikenneyhteydet hoidetaan yleensä *radiolla*, ja tällöin jo ympäristön muutokset — esimerkiksi kulkuneuvojen liikkeitä tai sään vaihtelu — voivat kesken järjestelmän käytön katkaista aiemmin toimineen tietoliikenneyhteyden kahden laitteen välillä. *Laitteiden liike* voi muuttaa järjestelmän kokoonpanoa: uusia laitteita voi tulla radion kantaman päähän, ja nämä halutaan liittää osaksi järjestelmään, ja vastaavasti laitteita voi poistua radion kantaman ulottumattomiin. Järjestelmä voi jakautua erillisiin osiin, joiden välillä ei ole tietoliikenneyhteyttä, ja erilliset osat voivat vastaavasti liittyä yhdeksi järjestelmäksi.

Kun vikatilanteita ja järjestelmän kokoonpanon muutoksia on odotettavissa, tietokoneohjelman suunnittelu muuttuu oleellisesti. Yksi lähestymistapa olisi kartoittaa erilaiset mahdolliset vikatilanteet järjestelmän suorituksen eri vaiheissa, ja huolellisesti suunnitella, miten näistä kustakin toivutaan. Karkea mutta elegantti vaihtoehto tälle on järjestelmän suunnitteleminen *itsestabiloivaksi* (self-stabilising) [Dij74, Sch93, Dol00]

Itsestabiloivuus. Hajautettu järjestelmä on itsestabiloiva, jos se palautuu äärellisessä ajassa kelvolliseen suoritukseen riippumatta siitä, mikä on järjestelmän alkutila. Kelvollinen suoritus tarkoittaa järjestelmän haluttua toimintaa — mikä ikinä se kussakin sovelluksessa sitten onkaan.

Voidaan esimerkiksi ajatella, että hajautettu järjestelmä koostuu tilakoneista. Järjestelmä on itsestabiloiva, jos tilakoneiden alkutila voidaan valita mielivaltaisesti, ja järjestelmä silti päätyy kelvolliseen suoritukseen. Vaihtoehtoisesti voidaan ajatella,

että kukin laite on tietokone, jonka ohjelma on pysyväismuistissa; kunkin tietokoneen työmuistin sisältö, rekisterien sisältö ja ohjelmalaskurin arvo voidaan asettaa mielivaltaiseen tilaan ja järjestelmä silti päätyy kelvolliseen suoritukseen.

Kun järjestelmässä on tapahtunut vika tai sen rakenne on muuttunut, voimme aina tulkita tämän muutoksen jälkeisen ajanhetken alkutilaksi. Jos järjestelmä on itsestabiloiva, se päätyy tästäkin nimenomaisesta alkutilasta kelvolliseen suoritukseen. Toki kelvolliseen suoritukseen päätymisessä voi kestää jonkin aikaa, ja jos muutoksia tulee jatkuvasti riittävän tiheään, siihen ei välttämättä päästä koskaan.

Itsestabiloivuus on sikäli karkea ratkaisu, että siinä varaudutaan *kaikkiin* alkutiloihin eikä pelkästään niihin, joihin järjestelmä voi päätyä joidenkin mahdollisten vikatilanteiden ja muutosten seurauksena; tämä voi jopa tarkoittaa, että itsestabiloivuus on liian vahva vaatimus. Itsestabiloivan järjestelmän toteuttaminen tiettyyn sovellukseen voikin olla vaikeaa tai mahdotonta, vaikka tavallisen vikasietoisen järjestelmän voisi toteuttaa.

Toisaalta itsestabiloivuus on sikäli elegantti ratkaisu, että siinä varaudutaan yhdellä kertaa kaikkiin vikatilanteisiin ja muutoksiin ilman, että niitä tarvitsee erikseen ennakoida. Kuten tässä työssä tulemme näkemään, itsestabiloivia järjestelmiä on mahdollista toteuttaa käytännössä kiinnostaviin hajautetun laskennan ongelmiin.

Työn sisältö. Tässä tutkielmassa tarkastellaan sovellusesimerkkinä *keskinäistä poissulkemista* hajautetussa järjestelmässä. Tavoitteena on kierrättää järjestelmässä *vuoromerkkiä* laitteelta toiselle. Kelvollisessa suorituksessa vuoromerkki on täsmälleen yhdellä laitteella kerrallaan. Vuoromerkin saanut laite pitää vuoromerkkiä hallussaan haluamansa ajan ja lopulta vapauttaa vuoromerkin. Luonnollisesti myös vaaditaan, että jokainen laite saa aikanaan vuoromerkin. Ongelma muotoillaan täsmällisemmin luvussa 3.

Aloitamme keskinäisen poissulkemisen tarkastelun luvussa 4 yksinkertaisesta erikoistapauksesta ja etenemme kohti yleisempää tapausta. Matkalla joudumme ratkaistaan muitakin tavallisia hajautetun laskennan ongelmia: *johtajan valinta* ja *virittävän puun muodostaminen*. Tutustumme yhteen itsestabiloivien algoritmien suunnittelun perustekniikkaan: *reiluun koostamiseen*.

Tämän työn sisältö seuraa pääpiirteissään Dolevin kirjan [Dol00] lukua 2; mukana on joitain kirjan ulkopuolisia asioita, mm. luvussa 5 mainittu tekniikka paikallisten algoritmien muuttamiseksi itsestabiloiviksi.

2 Hajautetun laskennan malli

Käymme aluksi läpi hajautetun laskennan mallin, jota tämän tutkielman luvuissa 3–4 sovelletaan. Tässä esitettävää keskusajastimeen ja tilakoneisiin perustuvaa mallia on käytetty mm. Dijkstran [Dij74] klassisessa työssä. Luvussa 5 tarkastellaan hiukan toisentyyppisenä esimerkkinä viestinvälitykseen perustuvaa mallia.

Verkko. Hajautettu järjestelmä tulkitaan tässä työssä *verkoksi* \mathcal{G} . Verkon kukin solmu vastaa laitetta; solmujen välillä on kaari, jos kyseisten solmujen välillä on tiedonsiirtoyhteys. Tässä keskitytään yksinkertaisuuden vuoksi tapaukseen, jossa tiedonsiirtoyhteydet ovat kaksisuuntaisia; verkko on siis *suuntaamaton*. Tarkastelemme *yhtenäistä* verkkoa tai verkon yhtä yhtenäistä komponenttia.

Verkon \mathcal{G} solmujen joukkoa merkitään kirjaimella V . Yksittäistä solmua merkitään kirjaimella v , ja solmujen lukumäärää merkitään kirjaimella n ; siis $v \in V$ ja $n = |V|$.

Usein tarkastellaan rengasta. Tällöin nimeämme solmut järjestyksessä $0, 1, \dots, n-1$. Lisäksi sovimme, että $v+1$ tarkoittaa renkaassa aina solmua v seuraavaa solmua ja $v-1$ edellistä solmua; laskemme siis solmujen numeroilla modulo- n -aritmetiikalla.

Tilakoneet. Jokainen laite tulkitaan *tilakoneeksi*. Solmun v tilasta käytetään tässä työssä merkintää $s(v)$. Jokaiselle tilakoneelle on määritelty yksi tai useampia *siirtymäehtoja* (privilege); nämä ovat totuusarvoisia funktioita, joiden syötteenä on solmun ja sen naapurisolmujen tila. Jokaiseen siirtymäehtoon liittyy *tilasiirtymä* (move, transition), jossa valitaan solmun uusi tila funktiolla, jossa on jälleen syötteenä solmun ja sen naapurisolmujen tila.

Esimerkiksi renkaassa solmun v siirtymäehdot ja niihin liittyvät tilasiirtymät voisivat näyttää tältä:

$$(s(v) + 1) \bmod 3 = s(v-1) : \quad s(v) \leftarrow (s(v) - 1) \bmod 3, \quad (1)$$

$$s(v) = s(v-1) : \quad s(v) \leftarrow (s(v) + 1) \bmod 3. \quad (2)$$

Ehto (2) siis toteutuu, jos solmu on samassa tilassa kuin edellinen; vastaava tilasiirtymä kasvattaa solmun tilaa yhdellä modulo 3.

Keskusajastin. Seuraavaksi tehdään yksinkertaistava oletus, joka on todellisten järjestelmien kannalta ehkä kaikkein epärealistisin. Oletamme, että järjestelmässä

on olemassa *keskusajastin* (central daemon) [Dij74, BP89, DIM93]. Järjestelmän tila päivittyy tämän ajastimen tahdittamana. Joka askeleessa ajastin valitsee yhden solmun, jonka jokin siirtymäehto on voimassa, ja edelleen jonkin näistä siirtymäehdoista. Solmu tekee tilasiirtymän ja keskusajastin valitsee tämän jälkeen seuraavan solmun. Erityisesti keskusajastimen olemassaolo tarkoittaa, ettei verkossa useampi solmu voi tehdä tilasiirtymiään täsmälleen samanaikaisesti rinnakkain.

Koko järjestelmän tila (state, configuration) määräytyy, kun kiinnitetään kunkin tilakoneen tila. Muodostetaan jono koko järjestelmän tiloja lähtemällä liikkeelle mielivaltaisesta alkutilasta ja tekemällä tilasiirtymiä, jotka jokin keskusajastin voisi tehdä. Tällaista jonoa sanotaan *suorituksiksi* (computation, execution) ja kukin siirtymä on laskennan *askel* (step).

Käytännön toteutus. Todellisessa järjestelmässä on toki ajateltava, että tämä edellä kuvattu tilakone on vain pieni osa solmun toiminnallisuutta. Jotta määrittelmistä saadaan käytännössä mielekkäitä, voidaan ajatella, että solmulla on esimerkiksi jotain valtaa sen suhteen, suoritetaanko tilasiirtymä vai ei. Keskusajastin kertoo, milloin solmu voisi suorittaa tilasiirtymän; solmu itse päättää, suoritetaanko sitä.

Esimerkiksi keskinäisessä poissulkemisessa tietyn siirtymäehdon toteutuminen voidaan tulkita vuoromerkin hallussapitämiseksi; vastaavan tilasiirtymän suorittaminen voidaan tulkita vuoromerkin vapauttamiseksi. Jotta jokainen solmu saa aikanaan ajovuoron, solmujen on toki toimittava sikäli reilusti, että vuoromerkki vapautetaan äärellisessä ajassa.

Jatkossa nämä yksityiskohdat sivuutetaan. On helpompi kuvata täsmällisesti esimerkiksi keskinäiseen poissulkemiseen liittyvät vaatimukset, jos ajatellaan, että siirtymä suoritetaan välittömästi, kun siirtymäehto sen mahdollistaa ja keskusajastin antaa ajovuoron.

Vaihtoehtoinen tulkinta. Tilakoneiden, siirtymäehtojen ja tilasiirtymien sijaan voidaan myös ajatella, että kukin solmu suorittaa ohjelmaa päättymättömässä silmukassa [DIM93, Dol00]. Kussakin solmussa on tilarekistereitä, ja solmu voi käsitellä niitä tilarekistereitä, jotka sijaitsevat solmussa itsessään tai sen naapurisolmuissa. Tilarekisteri voidaan siis tulkita jaetuksi muistiksi.

Palataan edellä olevaan tilasiirtymäesimerkkiin kaavoissa (1)–(2). Tämä voidaan suoraan kääntää ohjelmaksi $SOLMU[v]$, jota solmu v suorittaa:

```

SOLMU[v]
1  while TRUE
2      do if  $(s + 1) \bmod 3 = \text{HAE-TILA}[v-1]$ 
3          then  $s \leftarrow (s - 1) \bmod 3$ 
4          if  $s = \text{HAE-TILA}[v-1]$ 
5              then  $s \leftarrow (s + 1) \bmod 3$ 

```

Tässä s on solmussa itsessään sijaitseva tilarekisteri, jota käsitellään aivan kuten paikallista muuttujaa. Funktiolla $\text{HAE-TILA}[u]$ voidaan sitten hakea naapurisolmun u tilarekisterin arvo hyödyntämällä solmujen v ja u välistä tiedonsiirtoyhteyttä.

Kukin **if-then**-lause ajatellaan atomiseksi; kyseessä on ns. *koosteaskel* (aggregate step) [Dol00, §2.6]. Jälleen tehdään keskusajastinta vastaava epärealistinen oletus: koosteaskelten suoritus lomittuu, eli kaksi eri solmua ei suorita koosteaskeleitaan samanaikaisesti.

Kun jokin solmu törmää päättymätöntä silmukkaa suorittaessaan omalla ajovuorollaan **if**-lauseeseen, joka on tosi, solmu suorittaa vastaavassa **then**-lauseessa olevan tilasiirtymän. Tämä vastaa suoraan sitä, että keskusajastin valitsee jonkin voimassa olevan siirtymäehdon ja solmu suorittaa vastaavan tilasiirtymän.

Tässä työssä käytetään lähinnä tilakonemallia, sillä malli on yksinkertaisempi ja siihen liittyvät oletukset on helpompi muotoilla täsmällisesti. On kuitenkin hyvä muistaa, että tilakone on suoraan käännettävissä ylläolevan esimerkin tapaan päättymätömässä silmukassa suoritettavaksi ohjelmaksi. Muunnos päinvastaiseen suuntaan ei välttämättä ole aivan suoraviivainen.

Tilakonemallissa ei yleensä tarvitse murehtia keskusajastimen reiluudesta. Määrittelistä seuraa suoraan, että jos esimerkiksi vain yhdellä solmulla on jokin siirtymäehto voimassa, tämä siirtymäehto myös suoritetaan. Kun ajatellaan ohjelmia ja niiden koosteaskeleiden lomittumista, joudutaan erikseen vaatimaan, että jokainen solmu saa äärettömässä suorituksessa ajovuoron äärettömän monesti [DIM93, Dol00]. Muutoinhan järjestelmä voisi jäädä suorittamaan pelkästään epätosia **if**-lauseita.

3 Itsestabiloivuus

Määrittelemme nyt, mitä tarkalleen ottaen tarkoitamme, jos esimerkiksi sanomme, että ”tämä järjestelmä on itsestabiloiva toteutus keskinäisestä poissulkemisesta renkaissa”. Nojaudumme luvussa 2 kuvattuun hajautetun laskennan malliin.

Verkkojen perhe ja lisätiedot. Kiinnitämme ensin sen verkkojen perheen, jota tarkastelemme. Jatkossa saatamme esimerkiksi olettaa, että verkko \mathcal{G} on rengas.

Joissain tapauksissa oletamme, että solmuilla on käytettävissä lisätietoja verkon \mathcal{G} rakenteesta. Voimme esimerkiksi tarkastella tapausta, jossa solmuilla on tiedossa jokin yläraja verkon halkaisijalle tai solmujen lukumäärälle.

Kuhunkin verkon solmuun voi myös liittyä jotain solmukohtaista lisätietoa. Kullakin solmulla voi olla *yksilöllinen tunniste* — esimerkiksi renkaan tapauksessa emme tällöin tiedä, mitkä tunnistheet solmuille on valittu ja missä järjestyksessä ne renkaassa esiintyvät, mutta kukin solmu tietää oman tunnisteesa ja voi luottaa siihen, ettei samaa tunnistetta esiinny toista kertaa samassa verkossa.

Toinen esimerkki solmukohtaisesta lisätiedosta on se, että verkossa on *yksi erityinen solmu*. Siis jokaiseen solmuun liittyy yksi bitti lisätietoa, ja tiedämme, että tämä bitti on päällä täsmälleen yhdessä solmussa. Toisin sanoen olemme valinneet verkossa etukäteen yhden solmun *johtajaksi*.

Jos verkon solmuissa ei ole käytettävissä mitään solmukohtaista lisätietoa, sanomme, että verkko on *tasalaatuinen* (uniform) [BP89, DIM93].

Toteutus. Kiinnitetään seuraavaksi jokin hajautetun järjestelmän toteutus. Käytännössä siis määrittelemme verkon kullekin solmulle siirtymäehdot ja niitä vastaavat tilasiirtymät.

Tehtävä. Hajautetulla järjestelmällä on jokin laskennallinen tehtävä, jota se suorittaa. Kiinnitetään seuraavaksi tämä tehtävä. Koska käyttämässämme mallissa solmuilla ei varsinaisesti ole mitään ulostulosignaalia, laskennallisen tehtävän tavoite määritellään täsmällisesti tilakoneiden siirtymäehtojen ja tilojen avulla.

Kun tavoite on kiinnitetty, voidaan tarkastella suorituksia. Sanomme, että suoritus on *kelvollinen* (legal execution, legitimate behaviour), jos järjestelmä toteuttaa tehtävän vaatimukset koko suorituksen ajan [DIM93, Dol00].

Esimerkki: keskinäinen poissulkeminen. Keskinäisessä poissulkemisessa voimme tulkita jonkin tietyn siirtymäehdon vuoromeriksi. Sanomme suoritusta kelvolliseksi, jos se toteuttaa seuraavat vaatimukset [Dol00, §2.6]:

1. Kullakin ajanhetkellä täsmälleen yhdellä suorittimella on vuoromerkki.

2. Äärettömässä suorituksessa kukin solmu saa vuoromerkin äärettömän monesti.

Jälkimmäinen ehto on reiluusvaatimus. Ilman sitä voisimme muodostaa hyvin triviaalin algoritmin, jossa esimerkiksi johtajasolmu pitää vuoromerkkiä hallussaan jatkuvasti ja kukaan muu ei saa koskaan vuoromerkkiä.

Useat lähteet [Dij86, BP89, DIM93] käyttävät oleellisilta osiltaan samoja määritelmiä. Dijkstra [Dij74] ei mainitse reiluusehtoa erikseen, mutta esitetyt algoritmit toteuttavat silti sen.

Muunkinlaiset määritelmät olisivat mahdollisia. Voisimme esimerkiksi vaatia, että jokainen solmu saisi vuoromerkin pitkässä suorituksessa likimain yhtä monta kertaa; näin tiukkaa vaatimusta emme kuitenkaan tässä aseta.

Turvallinen tila. Sanomme, että järjestelmän tila on *turvallinen* (safe), jos jokainen kyseisestä tilasta alkava suoritus on kelvallinen [DIM93, Dol00].

Vielä emme tarkastele varsinaisesti itsestabiloivia järjestelmiä. Yksinkertaisesti tämä määritelmä tarkoittaa, että turvallinen tila olisi hyvä alkutila, jos voisimme valita, missä alkutilassa järjestelmä laitetaan käyntiin. On syytä huomata, että jo turvallisen tilan määritelmään sisältyy pahimman tapauksen analyysia: turvallisesta tilasta alkavan suorituksen tulee olla kelvallinen riippumatta siitä, mitä valintoja keskusajastin tekee silloin, kun useampi eri siirtymäehto on tosi. Voidaan ajatella, että keskusajastin on vastustaja pelissä, jota solmuissa olevat tilakoneet pelaavat.

Itsestabiloivuus. Määrittelemme lopulta, mitä tarkoitamme itsestabiloivalla järjestelmällä. Sanomme, että järjestelmä on itsestabiloiva, jos mikä tahansa äärettömän pituinen suoritus saavuttaa jossain vaiheessa turvallisen tilan [DIM93, Sch93, Dol00]. Tästä eteenpäin suoritus onkin määritelmän mukaan kelvallinen.

Itsestabiloivuuden määritelmän yksityiskohdat eroavat hiukan eri lähteissä, ja usein mukaan sekoitetaan myös tehtäväkohtaisia vaatimuksia [Dij74, BP89]. Esimerkiksi Dijkstran [Dij74] työ keskittyy keskinäiseen poissulkemiseen, jolloin on luonnollista vaatia, ettei järjestelmä koskaan lukkiudu, ts. päädy tilaan, jossa mikään siirtymäehto ei ole voimassa. Averbuch ja Varghese [AV91] taas keskittyvät laskentatehtäviin, joissa tavoitteena on nimenomaan lukkiutuminen johonkin turvalliseen tilaan; tällaisia tehtäviä on esimerkiksi johtajan valinta. Perusidea on kuitenkin kaikissa näissä määritelmissä sama: jossain äärellisessä määrässä askelia päädytään aina tilaan, josta alkaen suoritus on tehtävän kannalta kelvallinen.

4 Keskinäinen poissulkeminen

Sovellusesimerkkinä tarkastelemme itsestabiloivaa toteutusta keskinäiseen poissulkemiseen. Aloitamme ensin yksinkertaisesta erikoistapauksesta: verkko \mathcal{G} on rengas, ja yksi renkaan solmuista on erityisasemassa. Siirrymme tästä asteittain kohti yleisempää tapausta.

Rengas, jossa yksi erityinen solmu. Dijkstran klassisessa työssä [Dij74] esitetään kolme eri algoritmia keskinäiseen poissulkemiseen. Yksinkertaisimmassa algoritmissa kullakin solmulla on n mahdollista tilaa. Siirtymäehdot ja tilasiirtymät ovat seuraavat.

$$\text{Johtajasolmu:} \quad s(v) = s(v-1) : \quad s(v) \leftarrow s(v) + 1 \pmod n \quad (3)$$

$$\text{Muut solmut:} \quad s(v) \neq s(v-1) : \quad s(v) \leftarrow s(v-1). \quad (4)$$

Tässä toteutuksessa solmulla on vuoromerkki hallussa, kun sen ainoa siirtymäehto on tosi. Esimerkkilaskenta on esitetty liitteessä 1 kuvassa 2; todistus algoritmin oikeellisuudesta on esitetty esimerkiksi Dolevin kirjassa [Dol00, §2.6].

Edellä kuvatun algoritmin tila-avaruuden valinnassa tarvitaan etukäteistietoa verkon solmujen määrästä. On kuitenkin olemassa algoritmi, jossa tarvitaan vain neljä tilaa kussakin solmussa [Dij74]; itseasiassa tilojen määrä on mahdollista puristaa kolmeen tilaan solmua kohti [Dij74, Dij86]. Näissä algoritmeissa ei enää vaadita etukäteistietoa verkon koosta. Algoritmeissa hyödynnetään kahta erityistä solmu, ”top” ja ”bottom”, mutta jos käytettävissä on johtajasolmu v , solmuksi ”top” voidaan valita v ja solmuksi ”bottom” voidaan valita $v-1$. Algoritmien esimerkkiajot on esitetty liitteen 1 kuvissa 3 ja 4.

Rengas ilman erityistä solmua. Seuraava luonnollinen askel olisi hankkiutua eroon erityisen solmun tarpeesta. Joissain tapauksissa tämä onnistuukin. Esimerkiksi kahden solmun renkaassa voidaan tilasiirtymät järjestellä siten, että jos molempien solmujen tila on sama, symmetria rikotaan [BP89]; tämän yksinkertaisen algoritmin esimerkkilaskenta on esitetty kuvassa 5. Yleisemminkin, jos renkaan solmujen lukumäärä on alkuluku, itsestabiloiva algoritmi on olemassa [BP89]; ks. esimerkkilaskenta kuvassa 6.

Tämän pitemmälle ei kuitenkaan päästä tasalaatuisessa verkossa. Voidaan nimittäin todistaa, että keskinäinen poissulkeminen ei onnistu samanlaisista solmuista koostuvassa renkaassa, jos n ei ole alkuluku (ks. esim. Burns ja Pahl [BP89]). Tällöin

siis $n = ab$ jollain kokonaisluvulla $a \geq 2$, $b \geq 2$. Tehdään vasta oletus: on olemassa itsestabiloiva algoritmi keskinäiseen poissulkemiseen. Merkitään

$$V_k = \{k, k + b, k + 2b, \dots, k + (a-1)b\} \subseteq V.$$

Lähdetään alkutilanteesta, jossa kaikille $k \in \{0, 1, \dots, b-1\}$ pätee, että solmut V_k ovat keskenään samassa tilassa. Itsestabiloivan algoritmin tulee toki päästä tästäkin tilanteesta kelvolliseen tilaan.

Esimerkki. Jos $n = 12$, voidaan valita $a = 3$ ja $b = 4$; tällöin $V_0 = \{0, 4, 8\}$, $V_1 = \{1, 5, 9\}$, $V_2 = \{2, 6, 10\}$ ja $V_3 = \{3, 7, 11\}$. Alussa solmujen tilat voisivat näyttää vaikkapa tältä: 1, 3, 5, 6, 1, 3, 5, 6, 1, 3, 5, 6.

Nyt kaikille k pätee, että jos $u \in V_k$ ja $v \in V_k$, niin solmujen u ja v lähiympäristöjen tilat ovat samoja. Täsmällisemmin $s(u-1) = s(v-1)$, $s(u) = s(v)$ ja $s(u+1) = s(v+1)$. Jos siis solmun u jokin siirtymäehto on voimassa, myös solmun v vastaava siirtymäehto on voimassa.

Tämä ei selvästikään ole kelvollinen tila. Riippumatta siitä, määritelläänkö vuoromerkki siirtymäehtojen vai tilojen kautta, joko millään solmulla ei ole vuoromerkkiä tai vaihtoehtoisesti on olemassa ainakin yksi k siten, että vuoromerkki on kaikilla solmuilla $v \in V_k$.

Itsestabiloivuusoletuksen vuoksi järjestelmä ei voi lukkiutua tähän kelvottomaan tilaan. On siis olemassa jokin solmu u , jonka jokin siirtymäehto on tosi. Siis on jokin joukko solmuja V_k , joissa kaikissa on jokin siirtymäehto tosi.

Nyt on mahdollista, että keskusajastin antaa kaikkien näiden solmujen $v \in V_k$ suorittaa vuorollaan saman siirtymän. Kaikki nämä solmut myös päätyvät samaan tilaan, sillä solmut ovat identtisiä; solmun $v_1 \in V_k$ tekemä siirtymä ei vaikuta solmun $v_2 \in V_k$ siirtymäehtoihin, sillä nämä eivät ole naapurisolmuja, vaan välissä on johonkin joukkoon V_ℓ , $\ell \neq k$, kuuluvia solmuja.

Esimerkki jatkuu. Olkoon vaikkapa solmu 3 sellainen, jonka eräs siirtymäehto on tosi; tämä siirtymäehto vaihtaisi solmun tilaksi 0. Siis tässä solmussa on esimerkiksi siirtymäehto

$$s(v-1) = 5 \quad \text{ja} \quad s(v) = 6 \quad \text{ja} \quad s(v+1) = 1 : \quad s(v) \leftarrow 0.$$

Mutta sama siirtymäehto on siis kaikissa muissakin solmuissa, ja kaikissa solmuissa $v \in V_3 = \{3, 7, 11\}$ tämä ehto on myös tosi. Keskusajastin voi siis antaa esimerkiksi ensin solmun 3 suorittaa tämän siirtymän, sen jälkeen solmun 7 ja lopuksi solmun 11. Järjestelmä päätyy tilaan 1, 3, 5, 0, 1, 3, 5, 0, 1, 3, 5, 0.

Huomataan, että olemme alkupisteessä. Jokaiselle k pätee, että solmut V_k ovat keskenään samassa tilassa. Siis tämäkään ei voi olla kelvollinen tila. Voimme toistaa samanlaisia askeleita loputtomiin; oletus itsestabiloivan algoritmin olemassaolosta on siis väärä. Edes vahvempi oletus keskusajastimen reiluudesta ei auta.

Johtajan valinta. Oletus siitä, että verkossa olisi valmiiksi valittuna yksi erityinen solmu, on varsin epärealistinen, mutta toisaalta edellä näimme, että edes renkaassa emme voi ratkaista keskinäistä poissulkemista millään itsestabiloivalla algoritmilla, jos solmut ovat samanlaisia. Olemmeko siis osoittaneet, että käytännössä keskinäistä poissulkemista ei voida toteuttaa itsestabiloivasti?

Onneksi tilanne ei ole näin murheellinen. Hajautetuissa järjestelmissä on varsin tyyppillistä, että solmut eivät ole keskenään identtisiä, vaan kullakin on käytettävissään jokin yksilöllinen tunniste. Tällainen tunniste voi olla esimerkiksi laitteen verkkosovittimen laitteisto-osoite (ns. MAC-osoite), joksi on jo tehtaalla valittu yksilöllinen 48-bittinen tunniste.

Hyödyntämällä yksilöllistä tunnistetta voimme puolestaan valita verkosta johtajasolmun itsestabiloivalla algoritmilla. Erään yksinkertainen algoritmi tähän ovat esittäneet Arora ja Gouda [AG94]; ks. myös Dolev [Dol00, kuva 2.9]. Perusideana on valita johtajaksi solmu, jonka yksilöllinen tunniste on arvoltaan suurin. Tässä ratkaisussa solmut tarvitsevat lisätietoa verkon enimmäiskoosta.

Kuvassa 7 on esitetty tämän algoritmin kulku yksinkertaisessa viiden solmun renkaassa. Alussa verkossa vaeltelee pitkään virheellistä tietoa siitä, että olisi olemassa solmu, jonka tunniste on 42. Tämä olisi kaikkien mielestä kelvollinen ehdokas johtajaksi, kunnes lopulta paljastuu, että etäisyys tähän juurisolmuun olisi välttämättä suurempi kuin verkon solmujen enimmäismäärä (tässä esimerkissä 50 solmua).

Kun virheellinen tieto on saatu siivottua pois verkosta, järjestelmä alkaa stabiloitua nopeasti. Oikeanpuoleisin solmu (tunniste 40) julistautuu johtajaksi, ja tieto tästä alkaa levitä puumaisesti koko verkkoon. Tässä esimerkissä toki puu on varsin surkastunut, kun verkko on pelkkä rengas.

Koostaminen. Edellä olemme nähneet itsestabiloivan algoritmin johtajan valintaan; toisaalta olemme nähneet keskinäiseen poissulkemiseen itsestabiloivan algoritmin, joka tarvitsee tietoa johtajasta. Voimme nyt muodostaa *reilulla koostamisella* (fair protocol combination, fair composition, transitivity, layering) näistä uuden itsestabiloivan algoritmin, joka toteuttaa keskinäisen poissulkemisen ilman etukäteistietoa johtajasta [AV91, DIM93, Sch93, Dol00].

Koostaminen on pohjimmiltaan äärimmäisen yksinkertaista: jokainen solmu ajaa rinnan molempia algoritmeja. Kun keskinäisen poissulkemisen toteutus tarvitsee tietoa siitä, onko solmu johtaja vai ei, se kysyy johtajanvalinta-algoritmilta, onko solmu sen mielestä johtaja. Oleellista on, että johtajanvalinta-algoritmin toiminta ei missään vaiheessa riipu keskinäisen poissulkemisen algoritmista, vaan riippuvuus on ainoastaan yksisuuntaista.

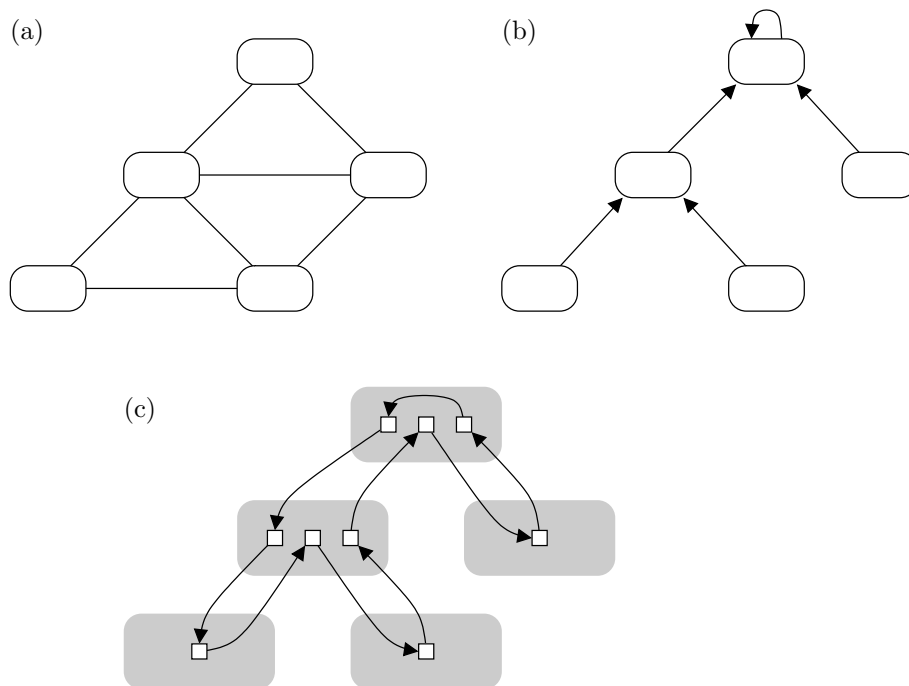
Esimerkkiajo on kuvassa 8. Alussa johtajanvalinta-algoritmi ei ole vielä stabiloitunut, joten keskinäisen poissulkemisen toteutus voi tehdä mielivaltaisia virheitä. Esimerkiksi askeleissa 18–37 mikään solmu ei ole mielestään johtaja, ja keskinäisen poissulkemisen algoritmi on lukkiutuneena. Lopulta kuitenkin askeleessa 50 johtajanvalinta-algoritmi valmistuu ja oikeanpuoleisin solmu valitaan johtajaksi. Tästä eteenpäin mikään ei myöskään voi häiritä johtajanvalintaa.

Hetkeä, jolloin johtajanvalinta-algoritmi valmistuu, voidaan nyt pitää keskinäisen poissulkemisen algoritmin kannalta alkutilana. Algoritmi voi olla tähän mennessä päätyneet mielivaltaiseen virhetilaan, mutta koska kyseessä on itsestabiloiva algoritmi, se toipuu tästäkin tilasta. Esimerkkiajossa itseasiassa toipuminen sattuu olemaan hyvin nopeaa, ja jo tilasta 52 alkaen suoritus on kelvollinen.

Koostamisen reiluus. Edellä mainittiin koostamisen yhteydessä termi reiluus. Tässä on kyse yksinkertaisesti siitä, että jokaisen solmun osalta annetaan ajovuoro molemmille algoritmeille. Muutoinhan edellä kuvatussa tapauksessa voisi käydä niin, että jokaisen siirron tekee keskinäisen poissulkemisen algoritmi eikä johtajanvalinta koskaan valmistu.

Yleisessä tapauksessa puhtaassa tilakone–keskusajastin-mallissa koostamisen reiluus on toteutettavissa esimerkiksi lisäämällä solmuihin ylimääräinen tila, joka kertoo, kumpaa algoritmia solmu kullakin siirrolla suorittaa. Viimeistään tällöin vaaditaan oletus keskusajastimen reiluudesta: jokaiselle solmulle on annettava äärettömässä suorituksessa äärettömän monesti ajovuoro.

Jos taas ajatellaan päättymätöntä silmukkaa suorittavia tietokoneohjelmia, koosta-



Kuva 1: (a) Verkko. (b) Eräs virittävä puu; nuolet osoittavat kohti vanhempaa eli kohti juurisolmua. (c) Puuhun muodostettu kulku; pienet neliöt ovat virtuaalisia solmuja.

misen reiluus voidaan hoitaa lomittamalla päättymättömien silmukoiden sisältämä laskenta. Tästä mallista todettiin jo aikaisemmin, että jokaiselle solmulle on taattava reilusti ajovuoroja, joten reilu koostaminen ei sikäli aiheuta lisärasitteita.

Yleiset verkot. Nyt osaamme siis tehdä keskinäisen poissulkemisen itsestabiloivasti renkaassa, ja riittää, että kullakin solmulla on yksikäsitteinen tunnus. Tästä on vielä matkaa siihen, että saamme toteutettua keskinäisen poissulkemisen mielivaltaisessa verkossa.

Tarvittavat rakennuspalikat ovat kuitenkin jo koossa. Edellä näimme, että Aroran ja Goudan [AG94] johtajanvalinta-algoritmissa tieto johtajasta etenee puumaisesti verkossa. Verkkoon muodostuu itsestabiloivasti virittävä, juurellinen puu: jokainen solmu tietää, onko se puun juuri, ja jos ei, minkä naapurisolmun suunnasta löytyy puun juuri. Vaikka edellä esitimme algoritmin ajon renkaassa, se yleistyy myös mielivaltaisiin verkkoihin. Kuvassa 1a on esimerkki verkosta ja kuvassa 1b on esimerkki syntyvästä virittävästä puusta.

Puu ei ehkä tunnu erityisen hyödylliseltä lähtökohdalta. Osaamme ratkaista kes-

kinäisen poissulkemisen verkossa, joka koostuu pelkästä syklistä, mutta virittävä puu on maksimaalinen verkko, jossa nimenomaan ei ole ainoatakaan sykliä. Pienellä tempulla voimme kuitenkin virittävän puun avulla muodostaa rengasmaisen kulun verkossa. Kuvassa 1c on esimerkki tästä [Dol00, §2.7].

Jaamme ensin kunkin solmun useampaan *virtuaaliseen solmuun*: yksi virtuaalinen solmu kutakin oikean solmun naapuria kohti. Tämän jälkeen ketjutamme virtuaaliset solmut esimerkkikuvan osoittamaan tapaan. Kunkin solmun virtuaalisten solmujen numero 1 ja 2 välissä on kulku, joka käy solmun ensimmäisen lapsen alipuun läpi; virtuaalisten solmujen numero 2 ja 3 välissä on kulku, joka käy solmun toisen lapsen alipuun läpi; näin jatketaan kaikille lapsille kunnes lopulta virtuaalisten solmujen numero N ja 1 välissä on kulku, joka käy puun muut osat läpi. Kun tarkastelemme näin luotua kulkua, havaitsemme, että se käy kunkin virtuaalisen solmun läpi täsmälleen yhden kerran. Olemme muodostaneet virtuaalisista solmuista renkaan.

Huomattavaa on, että tämä rengas voidaan muodostaa solmujen *sisäisellä* laskennalla: kun jokainen solmu tietää, mikä naapureista on vanhempi ja mitkä lapsia, solmu voi sisäisesti muodostaa virtuaaliset solmut ja ketjuttaa naapureista tulevat linkit kulkemaan virtuaalisten solmujen kautta. Tämä algoritmi on toki myös itsestabiloiva. Jos esimerkiksi ajatellaan tilakoneiden sijaan tietokoneohjelmia, riittää, että solmu toistaa sisäistä laskentaansa päättymättömässä silmukassa: vikatilanteen jälkeinen ensimmäinen kokonainen laskentakierros palauttaa tilan halutuksi. Sisäinen laskenta on äärimmäinen esimerkki paikallisista algoritmeista, joihin palataan lyhyesti tutkielman lopussa luvussa 5.

Virtuaalisista solmuista muodostuvassa renkaassa voimmekin nyt ajaa keskinäisen poissulkemisen algoritmia. Johtajasolmuna voimme käyttää esimerkiksi puun juuri-solmun virtuaalista solmua numero 1. Sovelluksessa voimme esimerkiksi tulkita, että solmulla on vuoromerkki, jos sen virtuaalinen solmu 1 pitää vuoromerkkiä.

Olemme tässä soveltaneet jälleen koostamista. Tällä kertaa muodostimme koosteen kolmesta itsestabiloivasta algoritmista. Pohjimmaisena on itsestabiloiva algoritmi virittävän puun muodostaminen; tämän ajo ei riipu muista algoritmeista. Virittävää puuta hyödyntää itsestabiloiva algoritmi virtuaalisista solmuista koostuvan renkaan muodostamiseen. Rengasta puolestaan hyödyntää itsestabiloiva algoritmi keskinäiseen poissulkemiseen. Jälleen joudumme huolehtimaan reiluudesta: kullekin algoritmille on taattava ajoaika.

Laajennuksia. Osoitimme edellä, että jo renkaassa joudumme välttämättä oletta-
maan, että solmut eivät ole tasalaatuisia. Kaikki tekemämme oletukset järjestelmäs-
tä eivät kuitenkaan ole välttämättömiä, vaan niitä voi hiukan lieventää. Ei esimer-
kiksi ole välttämätöntä olettaa, että koko koosteaskel (naapurien tilojen luku ja tilan
päivitys) on atominen [DIM93]. Johtajan valinta ja virittävän puun muodostaminen
eivät tarvitse välttämättä tietoa solmujen määrän ylärajasta [AB97].

5 Paikallisuus

Edellä luvussa 4 teimme varsin triviaalin havainnon: hajautettu algoritmi, jossa teh-
dään vain solmujen sisäistä laskentaa solmusta itsestään löytyvän syötteen perusteel-
la, on suoraan muunnettavissa itsestabiloivaksi algoritmiksi. Tarkastelemme lopuksi
huomattavasti laajempaa hajautettujen algoritmien perhettä ja esittelemme yleisen
tekniikan, jolla näistä kaikista saadaan itsestabiloivia; apuna käytämme jälleen rei-
lua koostamista.

Paikalliset algoritmit. *Paikallinen algoritmi* (local algorithm) [Lin92, NS95] on
hajautettu algoritmi, jossa kunkin solmun tekemä päätös on funktio syöttestä, joka
oli alkutilanteessa saatavilla solmun lähiympäristössä enintään tietyllä vakioetäisyy-
dellä R solmusta.

Tarkastelemme jatkossa vain astelukurajoitettuja verkkoja. Oletamme, että solmuil-
la on tunnisteet, jotka ovat yksilöllisiä kussakin R -säteisessä ympäristössä. Syötteen
koko solmua kohti on rajoitettu. Nyt paikallisen algoritmin olemassaolo tarkoittaa,
että on myös olemassa yksinkertainen vakioajassa toimiva hajautettu algoritmi: jo-
kainen solmu tulvittaa tilatietonsa lähiympäristöön; viesteissä pidetään mukana etäi-
syyslaskuria ja solmut eivät välitä pakettia enää eteenpäin, kun laskuri saavuttaa
arvon R . Jokainen solmu käsittelee ja välittää vain vakiomäärän dataa.

Tämän hajautetun, vakioaikaisen algoritmin voisi nyt muuntaa itsestabiloivaksi al-
goritmiksi, joka stabiloituu vakioajassa [AS88, AV91]. Muunnos on kuitenkin hiu-
kan raskas; paikallisen algoritmin pohjalta voi muodostaa suoraankin itsestabiloivan
algoritmin.

Viestinvälitysmalli. Tähän asti olemme tarkastelleet mallia, jossa tilakoneet nä-
kevät suoraan naapurisolmujen tilan; solmuilla on ollut jaettava muistia, jota naa-
purisolmut ovat suoraan päässeet lukemaan. Esimerkin vuoksi tarkastelemme tässä

luvussa *viestinvälitysmallia*. Tiedonsiirtoyhteys ajatellaan rajallisen kokoiseksi viestijonoksi. Kukin solmu suorittaa ohjelmaa, jossa viestijonoon kirjoittaminen ja viestijonosta lukeminen ovat eksplisiittisiä toimenpiteitä; naapurin tilaa ei enää näe suoraan. Jokaiseen verkon \mathcal{G} kaareen $\{u, v\}$ liittyy siis kaksi viestijonoa: yksi jono, johon u kirjoittaa ja jota v lukee, ja yksi jono, johon v kirjoittaa ja jota u lukee.

Lisäksi viestejä voi hukkuu viestijonoista. Oletamme kuitenkin, että kommunikointi onnistuu edes joskus: jos samaa viestiä lähetetään äärettömän monesti, se myös vastaanotetaan äärettömän monesti [Dol00, §2.1].

Paikallisesta itsestabiloivaksi. Tavoitteenamme on siis muuntaa paikallinen algoritmi itsestabiloivaksi ja vieläpä tässä varsin heikossa viestinvälitysmallissa. Kukin solmu suorittaa rinnan seuraavia toimenpiteitä; näitä toistetaan päättymättömässä silmukassa. Tämänkaltaisen algoritmin olemassaoloa voinee pitää alalla perimätietona; kuulen mielelläni, jos joku sattuu löytämään alkuperäislähteen.

1. Solmu lähettää kaikille naapureilleen viestin, joka koostuu seuraavista tiedoista: solmun paikallisesti yksilöllinen tunniste; solmussa oleva syöte; solmun naapurit; etäisyys viestin lähettäjään, aluksi 0.
2. Solmu lukee naapureiltaan tulevia viestejä. Kunkin viestin kohdalla solmu päivittää omaan kirjanpitoonsa, mitkä ovat viestin lähettäneen solmun naapurit ja syöte. Jos etäisyystieto oli alle R , solmu kasvattaa viestissä olevaa etäisyyttä yhdellä ja välittää viestin eteenpäin muille naapureille.
3. Solmu luo oman kuvansa R -säteisestä ympäristöstä. Aloitetaan solmusta itsestään ja sen naapureista; tämän jälkeen katsotaan kirjanpidosta näiden naapurit jne. kunnes koko verkkotopologia säteeseen R asti on koossa tai kirjanpito havaitaan vaillinaiseksi.
4. Jos on saavutettu sisäisesti konsistentti kuva R -säteisestä ympäristöstä, suoritetaan alkuperäinen paikallinen algoritmi ja päivitetään tulosrekisterin arvo.

Tilätiedon tulvitus (askeleet 1–2) ei riipu sisäisestä laskennasta (askeleet 3–4). Jos viestejä ei huku, algoritmi stabiloituu vakioajassa. Solmun paikalliseen kirjanpitoon voi jäädä virheellisiä tietoja solmuista, joita ei verkossa (ainakaan säteellä R) ole olemassakaan, mutta nämä jäävät yksinkertaisesti huomioimatta, kun luodaan kuvaa R -säteisestä ympäristöstä lähtemällä solmusta itsestään liikkeelle.

Lähteet

- AB97 Afek, Y. ja Bremler, A., Self-stabilizing unidirectional network algorithms by power-supply. *Proc. 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA, New Orleans, Louisiana, LA, January 1997)*, Philadelphia, PA, USA, 1997, Society for Industrial and Applied Mathematics, sivut 111–120.
- AG94 Arora, A. ja Gouda, M., Distributed reset. *IEEE Transactions on Computers*, 43,9(1994), sivut 1026–1038.
- AS88 Awerbuch, B. ja Sipser, M., Dynamic networks are as fast as static networks. *Proc. 29th Annual Symposium on Foundations of Computer Science (FOCS, White Plains, NY, USA, October 1988)*, Piscataway, NJ, USA, 1988, IEEE, sivut 206–219.
- AV91 Awerbuch, B. ja Varghese, G., Distributed program checking: a paradigm for building self-stabilizing distributed protocols. *Proc. 32nd Annual Symposium on Foundations of Computer Science (FOCS, San Juan, Puerto Rico, October 1991)*, Piscataway, NJ, USA, 1991, IEEE, sivut 258–267.
- BP89 Burns, J. E. ja Pachl, J. K., Uniform self-stabilizing rings. *ACM Transactions on Programming Languages and Systems*, 11,2(1989), sivut 330–344.
- Dij74 Dijkstra, E. W., Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17,11(1974), sivut 643–644.
- Dij86 Dijkstra, E. W., A belated proof of self-stabilization. *Distributed Computing*, 1,1(1986), sivut 5–6.
- 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, Cambridge, MA, USA, 2000.
- Lin92 Linial, N., Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21,1(1992), sivut 193–201.

- NS95 Naor, M. ja Stockmeyer, L., What can be computed locally? *SIAM Journal on Computing*, 24,6(1995), sivut 1259–1277.
- Sch93 Schneider, M., Self-stabilization. *ACM Computing Surveys*, 25,1(1993), sivut 45–67.

Liite 1. Algoritmien ajoesimerkkejä

Ajoesimerkeissä on listattu kukin askel omalla rivillään. Rivin alussa on askeleen numero. Tämän jälkeen tulee merkki "★", jos on saavutettu tila, jossa vain yksi tilasiirtymä on mahdollinen. Jos taas on saavutettu tila, jossa mikään solmu ei voi enää tehdä mitään, rivillä on merkki "S", ja ajoesimerkki päättyy.

Tämän jälkeen on lueteltu järjestyksessä kunkin solmun tila sekä solmuun liittyvät siirtymäehdot. Siirtymäehdoissa käytetään seuraavia merkintöjä: "." on siirtymäehto, joka on epätosi; "o" on siirtymäehto, joka on tosi; ja "●" on siirtymäehto, joka on tosi ja jonka keskusajastin sattui valitsemaan tällä kierroksella.

Kaikissa esimerkeissä verkko on rengas, jossa on n solmua. Johtajan valinnan yhteydessä on kiintoisaa tarkastella tilannetta, jossa solmuille on annettu jokin yksilöllinen tunniste. Näissä esimerkeissä solmujen yksilöllisiksi tunnisteiksi on valittu 00, 10, 20, ... tässä järjestyksessä. Solmujen suorittama algoritmi ei tätä valintaa tietenkään tiedä.

0		2·	1○	4○	0○	3●	2○	0○
1		2·	1○	4○	0●	0·	2○	0○
2		2·	1●	4○	4·	0○	2○	0○
3		2·	2·	4○	4·	0○	2○	0●
4		2●	2·	4○	4·	0○	2○	2·
5		3·	2○	4○	4·	0○	2●	2·
6		3·	2○	4●	4·	0○	0·	2○
7		3·	2●	2·	4○	0○	0·	2○
8		3·	3·	2○	4●	0○	0·	2○
9		3·	3·	2●	2·	0○	0·	2○
10		3·	3·	3·	2○	0○	0·	2●
11		3·	3·	3·	2○	0●	0·	0·
12		3·	3·	3·	2●	2·	0○	0·
13		3·	3·	3·	3·	2●	0○	0·
14	*	3·	3·	3·	3·	3·	0●	0·
15	*	3·	3·	3·	3·	3·	3·	0●
16	*	3●	3·	3·	3·	3·	3·	3·
17	*	4·	3●	3·	3·	3·	3·	3·
18	*	4·	4·	3●	3·	3·	3·	3·
19	*	4·	4·	4·	3●	3·	3·	3·
20	*	4·	4·	4·	4·	3●	3·	3·
21	*	4·	4·	4·	4·	4·	3●	3·
22	*	4·	4·	4·	4·	4·	4·	3●
23	*	4●	4·	4·	4·	4·	4·	4·
24	*	5·	4●	4·	4·	4·	4·	4·
25	*	5·	5·	4●	4·	4·	4·	4·
26	*	5·	5·	5·	4●	4·	4·	4·
27	*	5·	5·	5·	5·	4●	4·	4·
28	*	5·	5·	5·	5·	5·	4●	4·
29	*	5·	5·	5·	5·	5·	5·	4●
30	*	5○	5·	5·	5·	5·	5·	5·

Kuva 2: Keskinäinen poissulkeminen renkaassa, yksi solmu erityisasemassa, kussakin solmussa n tilaa [Dij74]. Esimerkkilaskenta, $n = 7$. Johtajan ainoa tilasiirtymä on esitetty kaavassa (3) ja muiden solmujen ainoa tilasiirtymä kaavassa (4).

0		01 ·	10 ○ ·	11 · ·	00 ○ ·	10 ○ ·	01 ● ○	00 ·
1		01 ·	10 ○ ·	11 · ·	00 ○ ·	10 ● ·	11 · ·	00 ○
2		01 ·	10 ○ ·	11 · ·	00 ○ ·	01 · ·	11 ● ·	00 ○
3		01 ·	10 ○ ·	11 · ·	00 ● ·	01 · ·	01 · ○	00 ·
4		01 ·	10 ○ ·	11 · ·	11 · ·	01 ○ ·	01 · ●	00 ·
5		01 ·	10 ○ ·	11 · ·	11 · ·	01 ○ ●	00 · ·	00 ·
6		01 ·	10 ● ·	11 · ·	11 · ·	00 ○ ·	00 · ·	00 ·
7		01 ·	01 · ·	11 ○ ·	11 · ·	00 ● ·	00 · ·	00 ·
8		01 ·	01 · ·	11 ○ ·	11 · ·	11 · ·	00 ● ·	00 ·
9		01 ·	01 · ·	11 ○ ·	11 · ·	11 · ·	11 · ·	00 ●
10		01 ·	01 · ·	11 ● ·	11 · ·	11 · ·	11 · ○	10 ·
11		01 ·	01 · ·	01 · ·	11 ● ·	11 · ·	11 · ○	10 ·
12		01 ·	01 · ·	01 · ·	01 · ·	11 ○ ·	11 · ●	10 ·
13		01 ·	01 · ·	01 · ·	01 · ·	11 ● ○	10 · ·	10 ·
14	★	01 ·	01 · ·	01 · ·	01 · ·	01 · ·	10 ● ·	10 ·
15	★	01 ·	01 · ·	01 · ·	01 · ·	01 · ·	01 · ·	10 ●
16	★	01 ·	01 · ·	01 · ·	01 · ·	01 · ·	01 · ●	00 ·
17	★	01 ·	01 · ·	01 · ·	01 · ·	01 · ●	00 · ·	00 ·
18	★	01 ·	01 · ·	01 · ·	01 · ●	00 · ·	00 · ·	00 ·
19	★	01 ·	01 · ·	01 · ●	00 · ·	00 · ·	00 · ·	00 ·
20	★	01 ·	01 · ●	00 · ·	00 · ·	00 · ·	00 · ·	00 ·
21	★	01 ●	00 · ·	00 · ·	00 · ·	00 · ·	00 · ·	00 ·
22	★	11 ·	00 ● ·	00 · ·	00 · ·	00 · ·	00 · ·	00 ·
23	★	11 ·	11 · ·	00 ● ·	00 · ·	00 · ·	00 · ·	00 ·
24	★	11 ·	11 · ·	11 · ·	00 ● ·	00 · ·	00 · ·	00 ·
25	★	11 ·	11 · ·	11 · ·	11 · ·	00 ● ·	00 · ·	00 ·
26	★	11 ·	11 · ·	11 · ·	11 · ·	11 · ·	00 ● ·	00 ·
27	★	11 ·	11 · ·	11 · ·	11 · ·	11 · ·	11 · ·	00 ●
28	★	11 ·	11 · ·	11 · ·	11 · ·	11 · ·	11 · ●	10 ·
29	★	11 ·	11 · ·	11 · ·	11 · ·	11 · ●	10 · ·	10 ·
30	★	11 ·	11 · ·	11 · ·	11 · ○	10 · ·	10 · ·	10 ·

Kuva 3: Keskinäinen poissulkeminen renkaassa, yksi solmu erityisasemassa, kussakin solmussa 4 tilaa [Dij74]. Esimerkkilaskenta, $n = 7$. Vasemmanpuoleisimman suorittimen jälkimmäinen tilabitti on pakotettu arvoon 1 ja oikeanpuoleisimmassa taas arvoon 0; näin taataan, että tilojen jonossa on ainakin yksi rajakohta, jossa jälkimmäinen tilabitti vaihtuu ykkösestä nollaksi. Kohdat, joissa ensimmäinen tilabitti eroaa, liikkuvat oikealle; samalla korjataan jälkimmäistä tilabittia ykköseksi (ensimmäinen siirtymäehto). Kohdat, joissa pelkästään jälkimmäinen tilabitti vaihtuu ykkösestä nollaksi, liikkuvat vasemmalle (toinen siirtymäehto). Reunat heijastavat.

0		2·	2··	1○·	0●·	0·○	1··	1·○	2··	1○○	2·
1		2·	2··	1○·	1··	0○○	1··	1·●	2··	1○○	2·
2		2·	2··	1○·	1··	0○○	1·●	2··	2··	1○○	2·
3		2·	2··	1●·	1··	0○·	2○·	2··	2··	1○○	2·
4		2·	2··	2··	1○·	0○·	2○·	2··	2··	1●○	2·
5		2·	2··	2··	1○·	0○·	2○·	2··	2··	2··	2●
6		2·	2··	2··	1○·	0○·	2○·	2··	2··	2·●	0·
7		2·	2··	2··	1●·	0○·	2○·	2··	2·○	0··	0·
8		2·	2··	2··	2·○	0··	2○·	2··	2·●	0··	0·
9		2·	2··	2··	2·●	0··	2○·	2·○	0··	0··	0·
10		2·	2··	2·○	0··	0··	2●·	2·○	0··	0··	0·
11		2·	2··	2·○	0··	0··	0··	2○●	0··	0··	0·
12	★	2·	2··	2·●	0··	0··	0··	0··	0··	0··	0·
13	★	2·	2·●	0··	0··	0··	0··	0··	0··	0··	0·
14	★	2●	0··	0··	0··	0··	0··	0··	0··	0··	0·
15	★	1·	0●·	0··	0··	0··	0··	0··	0··	0··	0·
16	★	1·	1··	0●·	0··	0··	0··	0··	0··	0··	0·
17	★	1·	1··	1··	0●·	0··	0··	0··	0··	0··	0·
18	★	1·	1··	1··	1··	0●·	0··	0··	0··	0··	0·
19	★	1·	1··	1··	1··	1··	0●·	0··	0··	0··	0·
20	★	1·	1··	1··	1··	1··	1··	0●·	0··	0··	0·
21	★	1·	1··	1··	1··	1··	1··	1··	0●·	0··	0·
22	★	1·	1··	1··	1··	1··	1··	1··	1··	0●·	0·
23	★	1·	1··	1··	1··	1··	1··	1··	1··	1··	0●
24	★	1·	1··	1··	1··	1··	1··	1··	1··	1·●	2·
25	★	1·	1··	1··	1··	1··	1··	1··	1·●	2··	2·
26	★	1·	1··	1··	1··	1··	1··	1·●	2··	2··	2·
27	★	1·	1··	1··	1··	1··	1·●	2··	2··	2··	2·
28	★	1·	1··	1··	1··	1·●	2··	2··	2··	2··	2·
29	★	1·	1··	1··	1·●	2··	2··	2··	2··	2··	2·
30	★	1·	1··	1·○	2··	2··	2··	2··	2··	2··	2·

Kuva 4: Keskinäinen poissulkeminen renkaassa, yksi solmu erityisasemassa, kussakin solmussa 3 tilaa [Dij74]. Esimerkkilaskenta, $n = 10$. Tarkastellaan nuolia, joka piirretään solmusta u solmuun v , jos $s(u)$ on yhtä suurempi kuin $s(v)$ modulo 3 [Dij86]. Turvallisia tiloja ovat ne, joissa esiintyy täsmälleen yksi nuoli; tämä nuoli kulkee edestakaisin verkossa ja heijastuu laidoista. Ylimääräiset nuolet poistuvat äärellisessä ajassa: esimerkiksi rivien 7 ja 8 välillä kaksi oikealle osoittavaa nuolta korvattiin yhdellä vasemmalle osoittavalla; rivien 11 ja 12 välillä kaksi nuolta kohtasi ja molemmat poistuivat. Toisaalta oikeanpuolimmainen erityinen solmu pitää huolen siitä, että symmetria rikotaan aina: esimerkiksi rivien 5 ja 6 välillä oikeanpuolimmaisen solmun lähiympäristö näytti liian symmetriseltä, joten solmu tuotti verkkoon uuden vasemmalle osoittavan nuolen.

0		0 · ●	0 · ○
1	★	1 · ·	0 ● ·
2	★	1 ● ·	2 · ·
3	★	0 · ·	2 ● ·
4	★	0 ● ·	1 · ·
5	★	2 · ·	1 ● ·
6	★	2 ● ·	0 · ·
7	★	1 · ·	0 ● ·
8	★	1 ● ·	2 · ·
9	★	0 · ·	2 ● ·
10	★	0 ○ ·	1 · ·

Kuva 5: Keskinäinen poissulkeminen kahden solmun renkaassa, samanlaisia solmuja, kussakin solmussa 3 tilaa [BP89]. Esimerkkilaskenta. Solmujen tilasiirtymät on esitetty kaavoissa (1)–(2). Tilasiirtymää (2) tarvitaan vain symmetrian rikkomiseen, jos molempien solmujen tilat sattuvat olemaan alussa samoja. Keskusajastimen olemassaolo on tässä kohtaa oleellista: symmetriaa ei saataisi näillä tilasiirtymillä rikottua, jos molemmat solmut suorittaisivat tämän siirtymän täsmälleen samanaikaisesti. Kun symmetria on saatu rikottua, tilanne helpottuu huomattavasti; sovellamme vain toistuvasti tilasiirtymää (1), eikä keskusajastinkaan enää olisi välttämätön, sillä vain yksi siirtymäehto on kullakin ajanhetkellä tosi.

0	0.0 · ·	1.0 · ·	0.2 ○ ·	3.3 ● ·	3.0 ○ ·	
1	0.0 · ·	1.0 · ·	0.2 ● ·	0.3 ○ ·	3.0 ○ ·	
2	0.0 · ·	1.0 · ·	1.3 ● ·	0.3 · ·	3.0 ○ ·	
3	0.0 · ·	1.0 · ·	2.0 · ·	0.3 ○ ·	3.0 ● ·	
4	0.0 ○ ·	1.0 · ·	2.0 · ·	0.3 ○ ·	0.3 ● ·	
5	0.0 ● ·	1.0 · ·	2.0 · ·	0.3 ○ ·	1.0 ○ ·	
6	1.3 ○ ·	1.0 ○ ·	2.0 · ·	0.3 ● ·	1.0 ○ ·	
7	1.3 ○ ·	1.0 ○ ·	2.0 · ·	1.2 ○ ·	1.0 ● ·	
8	1.3 ○ ·	1.0 ○ ·	2.0 · ·	1.2 ● ·	2.0 ○ ·	
9	1.3 ○ ·	1.0 ○ ·	2.0 · ·	2.3 ● ·	2.0 ○ ·	
10	1.3 ○ ·	1.0 ● ·	2.0 · ·	3.0 · ·	2.0 ○ ·	
11	1.3 ○ ·	2.0 ○ ·	2.0 ○ ·	3.0 · ·	2.0 ● ·	
12	1.3 ○ ·	2.0 ○ ·	2.0 ● ·	3.0 · ·	3.3 ○ ·	
13	1.3 ○ ·	2.0 ○ ·	3.0 · ·	3.0 ○ ·	3.3 ● ·	
14	1.3 · ○	2.0 ○ ·	3.0 · ·	3.0 ● ·	0.0 · ·	
15	1.3 · ○	2.0 · ●	3.0 · ·	0.0 · ·	0.0 ○ ·	
16	1.3 · ○	2.3 · ·	3.0 · ●	0.0 · ·	0.0 ○ ·	
17	1.3 · ●	2.3 · ·	3.3 · ·	0.0 · ·	0.0 ○ ·	
18	1.0 · ·	2.3 · ○	3.3 · ·	0.0 · ·	0.0 ● ·	
19	1.0 ● ·	2.3 · ○	3.3 · ·	0.0 · ·	1.0 · ·	
20	★	2.0 · ·	2.3 ● ·	3.3 · ·	0.0 · ·	1.0 · ·
21	★	2.0 · ·	3.0 · ·	3.3 ● ·	0.0 · ·	1.0 · ·
22	★	2.0 · ·	3.0 · ·	0.0 · ·	0.0 ● ·	1.0 · ·
23	★	2.0 · ·	3.0 · ·	0.0 · ·	1.0 · ·	1.0 ● ·
24	★	2.0 ● ·	3.0 · ·	0.0 · ·	1.0 · ·	2.0 · ·
25	★	3.0 · ·	3.0 ● ·	0.0 · ·	1.0 · ·	2.0 · ·
26	★	3.0 · ·	0.0 · ·	0.0 ● ·	1.0 · ·	2.0 · ·
27	★	3.0 · ·	0.0 · ·	1.0 · ·	1.0 ● ·	2.0 · ·
28	★	3.0 · ·	0.0 · ·	1.0 · ·	2.0 · ·	2.0 ● ·
29	★	3.0 ● ·	0.0 · ·	1.0 · ·	2.0 · ·	3.0 · ·
30	★	0.0 · ·	0.0 ○ ·	1.0 · ·	2.0 · ·	3.0 · ·

Kuva 6: Keskinäinen poissulkeminen renkaassa, samanlaisia solmuja. Solmujen lukumäärä n on alkuluku. Kussakin solmussa on $\Theta(n^2)$ tilaa [BP89]. Esimerkkilas-kenta, $n = 5$. Turvallisessa tilassa solmujen tilojen ensimmäiset osat muodostavat jonon $0, 1, \dots, a - 1, a, a, a + 1, \dots, n - 3, n - 2$, jossa täsmälleen yksi luku toistuu; vuoromerkki on tämän solmuparin jälkimmäisellä solmulla. Tilan jälkimmäistä osaa hyödynnetään vain itsestabilointiin.

0	?	38.42	○	...	?	22.24	○	...	?	4.39	○	...	?	21.41	●	...	?	22.00	○	...	○
1	?	38.42	○	...	?	22.24	○	...	?	4.39	○	...	○	0.30	...	○	?	22.00	○	...	●
2	?	38.42	●	...	?	22.24	○	...	?	4.39	○	...	○	0.30	...	○	→	39.42
3	○	0.00	...	○	?	22.24	○	...	?	4.39	●	...	○	0.30	...	○	→	39.42
4	○	0.00	...	○	?	22.24	●	...	○	0.20	...	○	○	0.30	...	○	→	39.42
5	○	0.00	...	○	○	0.10	...	○	○	0.20	...	●	○	0.30	...	○	→	39.42
6	○	0.00	...	●	○	0.10	...	○	→	1.30	○	0.30	...	○	→	39.42
7	←	40.42	○	0.10	...	○	→	1.30	○	0.30	...	●	→	39.42
8	←	40.42	○	0.10	...	●	○	→	1.30	...	○	→	40.42	→	39.42
9	←	40.42	←	41.42	→	1.30	...	●	○	→	40.42	→	39.42
10	★	←	40.42	...	←	41.42	→	41.42	→	40.42	→	39.42
11	←	40.42	...	○	←	41.42	→	41.42	→	40.42	...	●	→	41.42
12	←	40.42	...	●	←	41.42	→	41.42	→	42.42	→	41.42
13	←	42.42	←	41.42	...	○	→	41.42	→	42.42	→	41.42	●
14	←	42.42	...	○	←	41.42	...	●	→	41.42	→	42.42	→	43.42
15	←	42.42	...	○	←	43.42	○	→	41.42	...	●	→	42.42	→	43.42
...																					
30	←	48.42	...	○	←	49.42	○	→	47.42	←	46.42	...	●	→	49.42
31	←	48.42	...	●	←	49.42	○	→	47.42	←	48.42	→	49.42
32	←	50.42	...	○	←	49.42	...	○	→	47.42	←	48.42	→	49.42	●
33	←	50.42	...	○	←	49.42	...	○	→	47.42	...	●	←	48.42	→	51.42	○
34	←	50.42	...	●	←	49.42	○	→	49.42	←	48.42	...	○	→	51.42
35	○	0.00	←	49.42	○	→	49.42	←	48.42	→	51.42
36	○	0.00	←	49.42	...	●	→	49.42	←	48.42	←	49.42
37	○	0.00	←	1.00	○	→	49.42	←	48.42	←	49.42
38	○	0.00	→	50.42	○	→	49.42	←	48.42	←	49.42
39	←	50.42	→	50.42	○	→	49.42	←	48.42	←	49.42
40	←	50.42	→	50.42	○	→	49.42	←	50.42	←	49.42
41	←	50.42	→	50.42	○	→	49.42	○	0.30	←	49.42
42	←	50.42	→	50.42	○	→	1.30	○	0.30	←	49.42
43	←	50.42	○	0.10	○	→	1.30	○	0.30	←	49.42
44	←	50.42	○	0.10	○	→	1.30	○	0.30	←	1.30
45	○	0.00	○	0.10	○	→	1.30	○	0.30	←	1.30
46	○	0.00	→	2.30	○	→	1.30	○	0.30	←	1.30
47	→	3.30	→	2.30	○	→	1.30	○	0.30	←	1.30
48	→	3.30	→	2.30	○	→	1.30	○	0.30	○	0.40
49	←	1.40	→	2.30	○	→	1.30	○	0.30	○	0.40
50	←	1.40	←	2.40	○	→	1.30	○	0.30	○	0.40
51	←	1.40	←	2.40	○	→	1.30	○	0.30	○	0.40
52	S	←	1.40	...	←	2.40	○	→	2.40	○	0.40	○	0.40

Kuva 7: Johtajan valinta ja virittävän puun muodostaminen [AG94]. Esimerkkilas-kenta renkaassa, $n = 5$. Solmun tilassa on ensimmäisenä tieto vanhemmasta: \circ , jos solmu itse on mielestään juurisolmu, \leftarrow , jos vasen naapuri on solmun vanhempi, ja \rightarrow , jos oikea naapuri on solmun vanhempi. Seuraavana tulee luku, joka on etäisyys juurisolmuun ja tämän jälkeen tulee juurisolmun yksilöllinen tunniste. Tilasiirtymät: (i) paikallisesti epäkonsistentti tila; (ii)–(iii) vasen/oikea naapuri on solmun vanhempi, mutta tämän naapurin tilatieto ei ole yhteensopiva oman tilatiedon kanssa; (iv)–(v) vasemman/oikean naapurin kautta löytyi parempi polku johtajaan.

0	?	29.33 : 1 ○ ○ ○ ○ ○	?	15.17 : 3 ○ ○ ○ ○ ○	←	1.01 : 1 ○ ○ ○ ○ ●	?	0.09 : 0 ○ ○ ○ ○ ○	
1	?	29.33 : 1 ○ ○ ○ ○ ○	?	15.17 : 3 ○ ○ ○ ○ ○	←	1.01 : 3 ● ○ ○ ○ ○	?	0.09 : 0 ○ ○ ○ ○ ○	
2	?	29.33 : 1 ○ ○ ○ ○ ○	?	15.17 : 3 ○ ○ ○ ○ ○	○	0.20 : 3 ○ ○ ○ ○ ○	?	0.09 : 0 ○ ○ ○ ○ ○ ●	
3	?	29.33 : 1 ○ ○ ○ ○ ○	?	15.17 : 3 ○ ○ ○ ○ ○	○	0.20 : 3 ○ ○ ○ ○ ●	?	0.09 : 3 ○ ○ ○ ○ ○	
4	?	29.33 : 1 ○ ○ ○ ○ ○	?	15.17 : 3 ○ ○ ○ ○ ○	○	0.20 : 0 ○ ○ ○ ○ ○	?	0.09 : 3 ○ ○ ○ ○ ○ ●	
5	?	29.33 : 1 ○ ○ ○ ○ ○	?	15.17 : 3 ○ ○ ○ ○ ○ ●	○	0.20 : 0 ○ ○ ○ ○ ○	?	0.09 : 0 ○ ○ ○ ○ ○	
6	?	29.33 : 1 ○ ○ ○ ○ ○	?	15.17 : 1 ○ ○ ○ ○ ○	○	0.20 : 0 ○ ○ ○ ○ ○	?	0.09 : 0 ● ○ ○ ○ ○ ○	
7	?	29.33 : 1 ○ ○ ○ ○ ○	?	15.17 : 1 ○ ○ ○ ○ ○	○	0.20 : 0 ○ ○ ○ ○ ○	○	0.30 : 0 ○ ○ ○ ○ ○ ●	
8	?	29.33 : 1 ○ ○ ○ ○ ○	?	15.17 : 1 ○ ○ ● ○ ○	○	0.20 : 0 ○ ○ ○ ○ ○	○	0.30 : 1 ○ ○ ○ ○ ○	
9	?	29.33 : 1 ○ ○ ○ ○ ○	←	30.33 : 1 ○ ○ ○ ○ ○	○	0.20 : 0 ○ ○ ○ ○ ●	○	0.30 : 1 ○ ○ ○ ○ ○	
10	?	29.33 : 1 ○ ○ ○ ○ ○	←	30.33 : 1 ○ ○ ○ ○ ○	→	1.30 : 0 ○ ○ ○ ○ ●	○	0.30 : 1 ○ ○ ○ ○ ○	
11	?	29.33 : 1 ○ ○ ○ ○ ○	←	30.33 : 1 ○ ○ ○ ○ ○	→	1.30 : 1 ○ ○ ○ ○ ●	○	0.30 : 1 ○ ○ ○ ○ ○	
12	?	29.33 : 1 ○ ○ ○ ○ ○	←	30.33 : 1 ○ ○ ○ ○ ○	←	31.33 : 1 ○ ○ ○ ○ ○	○	0.30 : 1 ○ ○ ○ ○ ○ ●	
13	★	?	29.33 : 1 ● ○ ○ ○ ○	←	30.33 : 1 ○ ○ ○ ○ ○	←	31.33 : 1 ○ ○ ○ ○ ○	→	30.33 : 1 ○ ○ ○ ○ ○
14	○	0.00 : 1 ○ ○ ○ ○ ○	←	30.33 : 1 ○ ○ ○ ○ ○	←	31.33 : 1 ○ ○ ○ ○ ○	→	30.33 : 1 ○ ○ ● ○ ○	
15	○	0.00 : 1 ○ ○ ○ ○ ●	←	30.33 : 1 ○ ○ ○ ○ ○	←	31.33 : 1 ○ ○ ○ ○ ○	→	1.00 : 1 ○ ○ ○ ○ ○	
16	→	31.33 : 1 ○ ○ ○ ○ ○	←	30.33 : 1 ○ ○ ○ ○ ○	←	31.33 : 1 ○ ○ ○ ○ ○	→	1.00 : 1 ● ○ ○ ○ ○	
17	→	31.33 : 1 ○ ○ ○ ○ ○	←	30.33 : 1 ○ ○ ○ ○ ○	←	31.33 : 1 ○ ○ ○ ○ ○	○	0.30 : 1 ○ ○ ○ ○ ●	
18	★	→	31.33 : 1 ○ ○ ○ ○ ○	←	30.33 : 1 ○ ● ○ ○ ○	←	31.33 : 1 ○ ○ ○ ○ ○	←	32.33 : 1 ○ ○ ○ ○ ○
19	→	31.33 : 1 ○ ○ ○ ○ ○	←	32.33 : 1 ○ ○ ○ ○ ○	←	31.33 : 1 ○ ● ○ ○ ○	←	32.33 : 1 ○ ○ ○ ○ ○	
20	→	31.33 : 1 ○ ○ ○ ○ ○	←	32.33 : 1 ○ ○ ○ ○ ○	←	33.33 : 1 ○ ○ ○ ○ ○	←	32.33 : 1 ○ ● ○ ○ ○	
...									
35	→	39.33 : 1 ○ ○ ● ○ ○	←	40.33 : 1 ○ ○ ○ ○ ○	←	39.33 : 1 ○ ○ ○ ○ ○	→	38.33 : 1 ○ ○ ○ ○ ○	
36	→	41.33 : 1 ○ ○ ○ ● ○	←	40.33 : 1 ○ ○ ○ ○ ○	←	39.33 : 1 ○ ○ ○ ○ ○	→	38.33 : 1 ○ ○ ○ ○ ○	
37	←	39.33 : 1 ○ ○ ○ ○ ○	←	40.33 : 1 ○ ● ○ ○ ○	←	39.33 : 1 ○ ○ ○ ○ ○	→	38.33 : 1 ○ ○ ○ ○ ○	
38	←	39.33 : 1 ○ ○ ○ ○ ○	○	0.10 : 1 ○ ○ ○ ● ○	←	39.33 : 1 ○ ○ ○ ○ ○	→	38.33 : 1 ○ ○ ○ ○ ○	
39	←	39.33 : 1 ○ ○ ○ ○ ○	←	40.33 : 1 ○ ○ ○ ○ ○	←	39.33 : 1 ○ ○ ○ ○ ○	→	38.33 : 1 ○ ○ ● ○ ○	
40	←	39.33 : 1 ○ ● ○ ○ ○	←	40.33 : 1 ○ ○ ○ ○ ○	←	39.33 : 1 ○ ○ ○ ○ ○	→	40.33 : 1 ○ ○ ○ ○ ○	
41	←	41.33 : 1 ○ ○ ○ ○ ○	←	40.33 : 1 ○ ● ○ ○ ○	←	39.33 : 1 ○ ○ ○ ○ ○	→	40.33 : 1 ○ ○ ○ ○ ○	
42	←	41.33 : 1 ○ ○ ○ ○ ○	○	0.10 : 1 ○ ○ ○ ○ ○	←	39.33 : 1 ○ ○ ○ ○ ○	→	40.33 : 1 ○ ● ○ ○ ○	
43	←	41.33 : 1 ○ ● ○ ○ ○	○	0.10 : 1 ○ ○ ○ ○ ○	←	39.33 : 1 ○ ○ ○ ○ ○	○	0.30 : 1 ○ ○ ○ ○ ○	
44	○	0.00 : 1 ○ ○ ○ ○ ○	○	0.10 : 1 ○ ○ ○ ○ ○	←	39.33 : 1 ○ ● ○ ○ ○	○	0.30 : 1 ○ ○ ○ ○ ○	
45	○	0.00 : 1 ○ ○ ○ ○ ●	○	0.10 : 1 ○ ○ ○ ○ ○	←	1.10 : 1 ○ ○ ○ ○ ○	○	0.30 : 1 ○ ○ ○ ○ ○	
46	→	1.10 : 1 ○ ○ ○ ● ○	○	0.10 : 1 ○ ○ ○ ○ ○	←	1.10 : 1 ○ ○ ○ ○ ○	○	0.30 : 1 ○ ○ ○ ○ ○	
47	←	1.30 : 1 ○ ○ ○ ○ ○	○	0.10 : 1 ○ ○ ○ ○ ○	←	1.10 : 1 ○ ○ ○ ○ ○	○	0.30 : 1 ○ ○ ○ ○ ●	
48	←	1.30 : 1 ○ ○ ○ ○ ○	○	0.10 : 1 ○ ○ ○ ○ ○	←	1.10 : 1 ○ ○ ○ ● ○	○	0.30 : 2 ○ ○ ○ ○ ○	
49	←	1.30 : 1 ○ ○ ○ ○ ○	○	0.10 : 1 ○ ○ ○ ○ ●	→	1.30 : 1 ○ ○ ○ ○ ○	○	0.30 : 2 ○ ○ ○ ○ ○	
50	←	1.30 : 1 ○ ○ ○ ○ ○	○	0.10 : 2 ○ ○ ○ ● ○	→	1.30 : 1 ○ ○ ○ ○ ○	○	0.30 : 2 ○ ○ ○ ○ ○	
51	←	1.30 : 1 ○ ○ ○ ○ ○	←	2.30 : 2 ○ ○ ○ ○ ●	→	1.30 : 1 ○ ○ ○ ○ ○	○	0.30 : 2 ○ ○ ○ ○ ○	
52	★	←	1.30 : 1 ○ ○ ○ ○ ●	←	2.30 : 1 ○ ○ ○ ○ ○	→	1.30 : 1 ○ ○ ○ ○ ○	○	0.30 : 2 ○ ○ ○ ○ ○
53	★	←	1.30 : 2 ○ ○ ○ ○ ○	←	2.30 : 1 ○ ○ ○ ○ ●	→	1.30 : 1 ○ ○ ○ ○ ○	○	0.30 : 2 ○ ○ ○ ○ ○
54	★	←	1.30 : 2 ○ ○ ○ ○ ○	←	2.30 : 2 ○ ○ ○ ○ ○	→	1.30 : 1 ○ ○ ○ ○ ●	○	0.30 : 2 ○ ○ ○ ○ ○
55	★	←	1.30 : 2 ○ ○ ○ ○ ○	←	2.30 : 2 ○ ○ ○ ○ ○	→	1.30 : 2 ○ ○ ○ ○ ○	○	0.30 : 2 ○ ○ ○ ○ ●
56	★	←	1.30 : 2 ○ ○ ○ ○ ●	←	2.30 : 2 ○ ○ ○ ○ ○	→	1.30 : 2 ○ ○ ○ ○ ○	○	0.30 : 3 ○ ○ ○ ○ ○
57	★	←	1.30 : 3 ○ ○ ○ ○ ○	←	2.30 : 2 ○ ○ ○ ○ ●	→	1.30 : 2 ○ ○ ○ ○ ○	○	0.30 : 3 ○ ○ ○ ○ ○
58	★	←	1.30 : 3 ○ ○ ○ ○ ○	←	2.30 : 3 ○ ○ ○ ○ ○	→	1.30 : 2 ○ ○ ○ ○ ●	○	0.30 : 3 ○ ○ ○ ○ ○
59	★	←	1.30 : 3 ○ ○ ○ ○ ○	←	2.30 : 3 ○ ○ ○ ○ ○	→	1.30 : 3 ○ ○ ○ ○ ○	○	0.30 : 3 ○ ○ ○ ○ ●
60	★	←	1.30 : 3 ○ ○ ○ ○ ○	←	2.30 : 3 ○ ○ ○ ○ ○	→	1.30 : 3 ○ ○ ○ ○ ○	○	0.30 : 0 ○ ○ ○ ○ ○

Kuva 8: Koostaminen: johtajan valinta kuten kuvassa 7 ja tämän päälle keskinäinen poissulkeminen kuten kuvassa 2. Kunkin solmun kohdalla on johtajan valintaan liittyvät tilatiedot erotettu kaksoispisteellä keskinäiseen poissulkemiseen liittyvästä tilatiedosta; siirtymäehdot (5 + 1 kappaletta) on merkitty peräjälkeen.