



Formal Methods for Concurrent Systems

The design methods of sequential systems produce poor results for concurrent systems. Tests can only reveal errors, but they cannot prove that a system is correct. When a system exhibits concurrency, errors show up randomly in tests that cannot control all the inputs simultaneously. Formal methods offer some help. It is possible to check models of a system and locate concurrency related errors before any code exists.

Applications of concurrent systems include communication networks, distributed databases and control systems.

Concurrency is inherent in several levels, both in software and hardware. Even simple-looking systems can reach millions of states, as the order of events can vary.

Obviously, assuring the quality of such systems cannot rely solely on semi-formal reasoning and testing. Some errors can be reproduced and

fixed, but a fix may introduce other, more random errors.

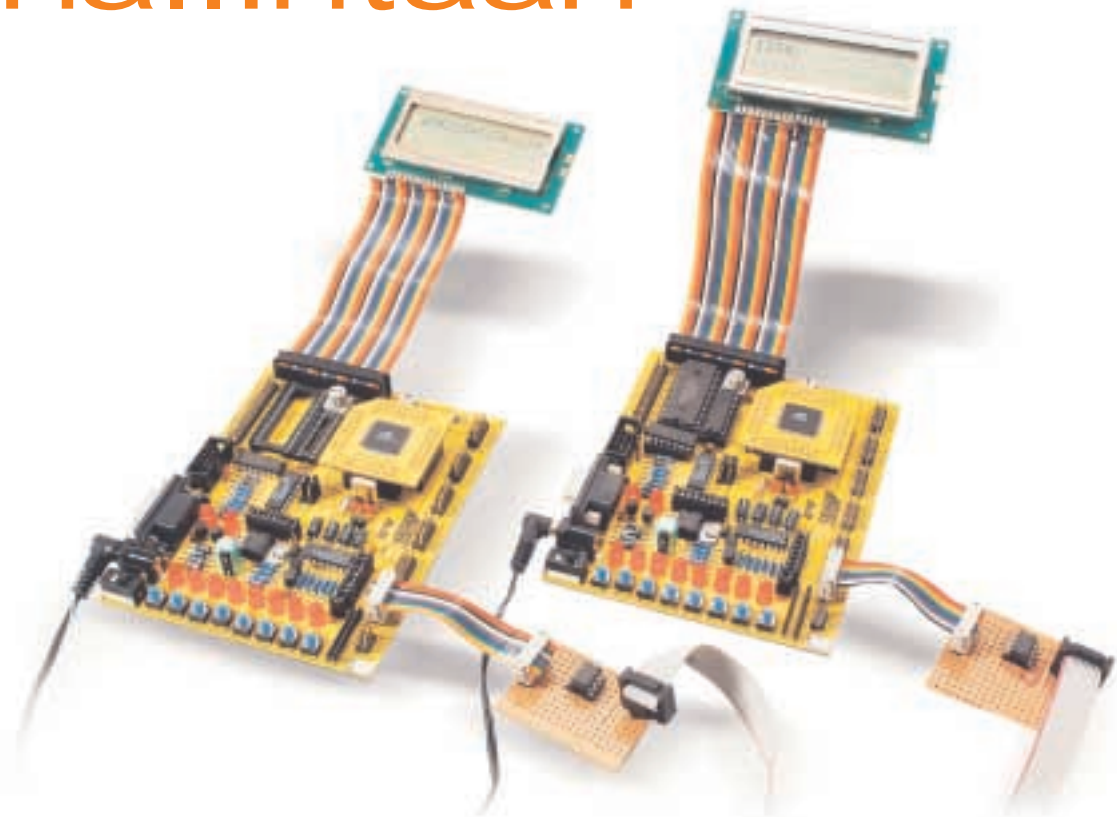
A suitably abstract system description helps to solve this kind of problems. When a model excludes details that do not affect the properties being observed, a computer can explore the whole model and either find an undesired execution or confirm that the properties hold.

Our research project belongs to the ETX program of Tekes, the National Technology Agency of Finland. The main outcome of the project is a freely distributable reachability analyser that supports high-level

programming languages. The research team is lead by Professor Nisse Husberg (Nisse.Husberg@hut.fi).

Formaalit menetelmät mutkikkaiden järjestelmien suunnittelun apuna

Rinnakkaisuus hallintaan



Peräkkäisten, yhtä asiaa kerrallaan tekevien järjestelmien suunnittelu- ja toteutusperiaatteet soveltuvat huonosti rinnakkaisiin järjestelmiin. Virheet tuntuvat esiintyvän satunnaisesti, sillä testeissä on mahdotonta hallita kaikkia järjestelmään vaikuttavia asioita samanaikaisesti. Apua tulee formaaleista menetelmistä. Järjestelmää kuvaavista malleista voidaan löytää rinnakkaisuuteen liittyviä virheitä jo ennen kuin riviäkään koodia on kirjoitettu.

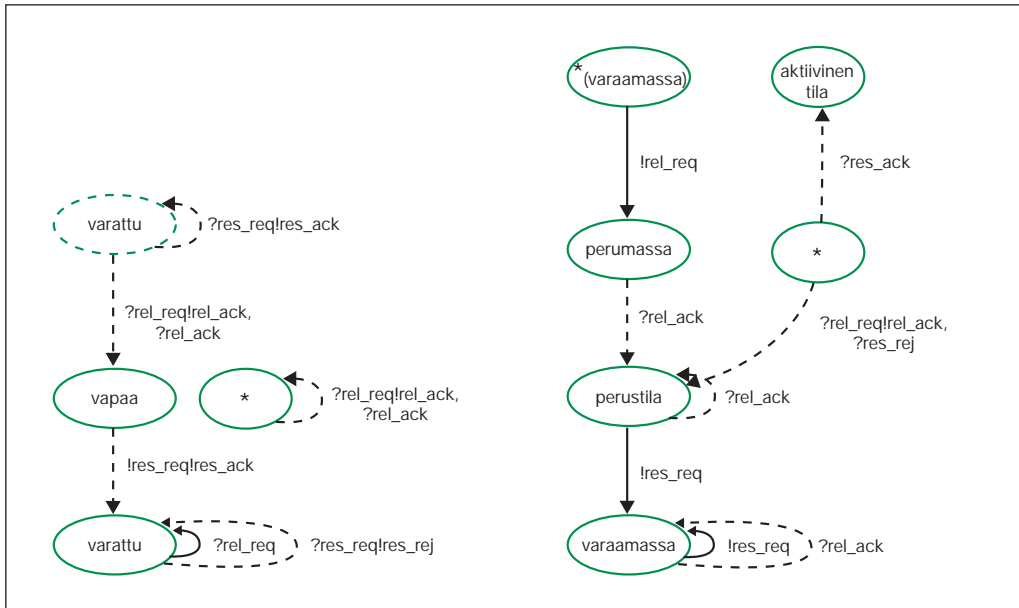
Rinnakkainen tietojenkäsittely hallitsee merkittäviä sovellusaloja: puhelin- ja tietoliikenneverkkoja, hajautettuja tiedonhallintajärjestelmiä ja erilaisia ohjausjärjestelmiä.

Rinnakkaisuutta esiintyy monella eri tasolla: laitteiden sisäisten lohkojen välillä, käyttöjärjestelmän ajamisen prosessin välillä, samaan tietoverkkoon kytkettyjen laitteiden välillä se-

kä laitteiden ja niiden ympäristön välillä. Yksinkertaisiltakin näyttävien järjestelmien on mahdollista saavuttaa miljoonia tai miljardeja erilaisia tiloja, koska tapahtumat voivat esiintyä hyvin monessa eri järjestyksessä.

On selvää, että näin monimutkaisten järjestelmien laatua ei voi jättää vain suunnittelijan huolellisuuden varaan. Kukaan ei pysty pitämään kaikkia mahdollisuuksia mielessään, ja usein jonkin virhetilanteen varalta tehty korjaus synnyttää uusia virhemahdollisuuksia.

Tietokoneet ovat vahvimmiltaan mekaanisessa sääntöjen soveltamisessa. Tietotekniikkaan



Kuva 1: Erään varausjärjestelmän keskusyksikköjen ja päätteiden toimintaa kuvaavat yksinkertaistetut tilakoneet. Keskusyksikkö voi pyytää resurssin varannutta päätettä vapauttamaan sen. Muuten keskusyksikkö reagoi ainoastaan päätteillä saamiinsa pyyntöihin.

Client and server logic specified with abstract state machines. The states used in this model are: allocated (varaamassa), available (vapaa), reserving (varaamassa), canceling (perumassa), idle (perustila) and active (aktiivinen).

yhdistettynä formaalit menetelmät ovat hyödyllisimmillään. Suunnittelijan tarvitsee vain kuvata järjestelmä siitä tutkittavien ominaisuuksien edellyttämässä laajuudessa sekä muotoilla halutut ominaisuudet loogisilla kaavoilla, ja tietokone joko vahvistaa ominaisuuksien toteutuvan tai esittää niiden vastaisen tapahtumaketjun.

Esittelemme formaalien menetelmien soveltamista erään hajautetun sovelluksen suunnittelussa.

Eräs hajautettu järjestelmä

Esimerkijärjestelmämme on tarkoitus korvata analoginen järjestelmä, jossa on älykkäitä keskusyksikköjä, joihin on paksuilla johtokimpuilla liitetty tyhmiä päätteitä. Pääteen nappia painava käyttäjä saa käyttöönsä jonkin jaetun resurssin, esimerkiksi keskusyksikön hallitseman puhelinlinjan.

Kustannusten pienentämiseksi kaapelointia yksinkertaistetaan korvaamalla kultakin päätteeltä lähtevät omat johtokimpuat kaikille laitteille yhteisellä väylällä. Keskusyksiköt ja päätteet kytketään yhteiseen CAT5-tasoisilla RJ-45-kaapeleilla toteutettavaan verkkoon. Yksi kaapelin johtopari muodostaa RS-485-väylän, jota pitkin päätteiden ja keskusyksiköiden mikroohjaimet lähettävät toisilleen sanomia, joiden tarkoitus on huolehtia resurssin varaamisesta ja vapauttamisesta. Muu osa järjestelmästä voi yhä toimia perinteisellä analogisella tekniikalla.

Siirtokerros muistuttaa Ethernet-verkkoa, sillä päätteet jakavat saman RS-485-väylän keskusyksikköjen kanssa. Väylällä

ei ole keskitettyä "puheenjohtajaa", vaan jokainen laite voi milloin tahansa yrittää lähettää linjalle sanomia. Tästä aiheutuvat ongelmat ratkaistaan soveltamalla CSMA/CD-menettelyä ja sanomien kehystystä.

Sovelluskerros huolehtii resurssin jakamisesta. Yksinkertaisuuden vuoksi oletamme, että verkossa on päätteiden seurana vain yksi keskusyksikkö. Verkossa voi olla useita keskusyksikköjä, jos sanomiin lisätään keskusyksikön osoite.

Päätteiden on kyettävä varamaan keskusyksikön hallitsemaa resurssia. Enintään yksi päätte kerrallaan saa olla siinä käsityksessä, että sillä on valtuus käyttää resurssia. Järjestelmä ei saa seota päätteiden käynnistämistä ja sammuttamisista, ja päätteiden on voitava siirtyä virransäästötilaan.

On luontevaa tehdä keskusyksiköstä eräänlainen puheenjohtaja, jolta päätteet pyytävät resurssia käyttöönsä. Keskusyksikkö pitää kirjaa siitä, onko resurssi vapaana tai minkä päätteen hallussa se on.

Näyttää siltä, että päätteet tarvitsevat neljä toimintatilaa. Käyttäjän painaessa nappia ne siirtyvät perustilasta varaustilaan, josta ne voivat siirtyä keskusyksiköltä luvan saatuaan aktiiviseen tilaan. Resurssin palauttamista varten on perumistila, josta päätte siirtyy kuittauksen saatuaan perustilaan.

Tilakoneita ja sanomakaavioita

Kuvan 2 ensimmäinen kaavio esittää esimerkijärjestelmämme sovelluskerroksen sanomaliikennettä sellaisessa tapauksessa, että päätte pyytää aluksi vapaata resurssia käyttöönsä ja

kaikki sanomat menevät perille. Päätte saa varauspyyntönsä (res_req, reservation request) kuittauksen (res_ack, reservation acknowledgement). Jonkin ajan kuluttua päätte lähettää vapautuspyynnön (rel_req, release), jonka keskusyksikkö kuittaa.

Kuvassa 1 on päätteen ja keskusyksikön toimintaa kuvaavat tilakoneet. Kaavioissa on kahdenlaisia siirtymiä. Huutomerkillä alkavat yhtenäisellä viivalla merkityt siirtymät esittävät sanomien omatoimista lähetystä. Sanoman nimi seuraa huutomerkkin jälkeen. Katkoviivoilla merkityt siirtymät esittävät tulevia sanomia. Usein niihin vastataan lähettämällä kuittausanoma.

Kuvia on yksinkertaistettu käyttämällä erilaisia lyhennysmerkkintöjä. Resurssin varattuna olemista kuvaava keskusyksikön tila on jaettu kahtia. Pisteillä reunustettu tila kuvaa sitä tilannetta, että resurssin varanneelta päätteeltä tulee sanoma. Muilta päätteiltä tulevat sanomat eivät saa keskusyksikköä vaihtamaan tilaansa. Tähdellä merkitystä tilasta lähtevät nuolet kuvaavat kaikissa tiloissa mahdollisia toimintoja.

Päätteen kaaviossa oleva symboli, jossa on tähden jälkeen suluisia tilan nimi (varaamassa), kuvaa kaikissa muissa paitsi nimetyssä tilassa mahdollisia toimintoja. Varaamistilassa ollessaan päätte ei voi lähettää vapautuspyyntöä.

Joissakin tilanteissa keskusyksikkö voi vastata varauspyyntöön kieltävästi (res_rej) tai pyytää päätettä vapauttamaan resurssin (rel_req), mutta se voi toimia myös kuvan 2 kaavion mukaisesti. Järjestelmä tuntuu

siis toimivan, joten sen voi rauhallisin mielin ottaa käyttöön. Vai voiko?

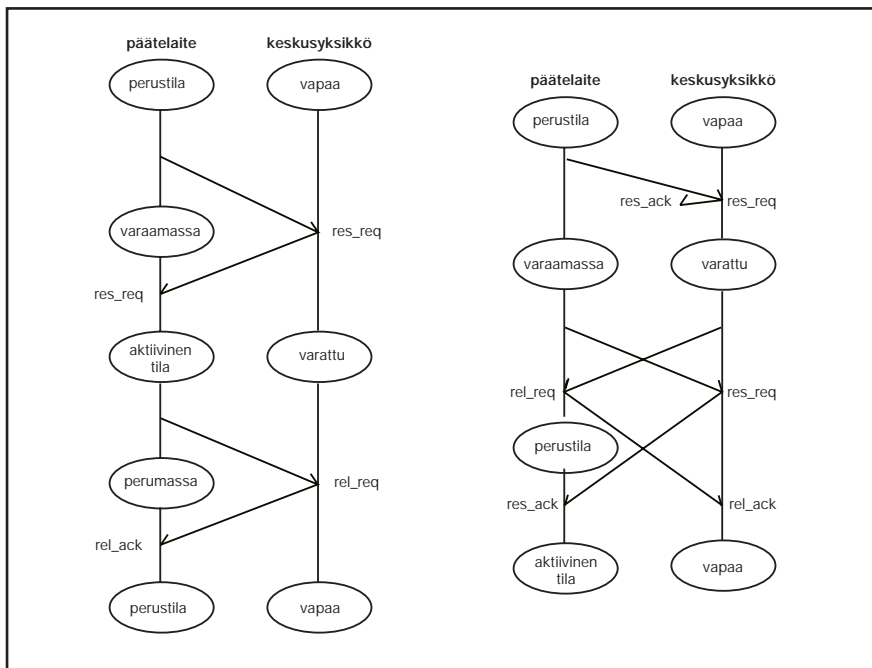
Valitettavasti myös toinen kuvan 2 esittämä tilanne on mahdollinen. Päätte varaa resurssin. Koska keskusyksikön kuittaus katoaa, päätte lähettää uuden varauspyynnön. Samoihin aikoihin keskusyksikkö pyytää resurssin vapauttamista. Päätte vastaa siirtymällä perustilaan ja lähettämällä kuittauksen, jonka saatuaan keskusyksikkö vapauttaa resurssin. Lopuksi päätte saa kuittauksen alussa toistamaansa varauspyyntöön ja siirtyy aktiiviseksi, vaikka resurssi on keskusyksikön käsityksen mukaan vapaa.

Kattavampaa kuin testaus

Kuvan 2 jälkimmäisen sanomakaavion virhetilannetta ei suinkaan löydetty kynän, paperin ja mielikuvituksen avulla vaan simulatioissa. Järjestelmän toiminta kuvattiin erityisellä mallinnuskielellä, joka perustuu erääseen tilakoneiden yleistykseseen, korkean tason Petri-verkkoihin. Tehdyssä mallissa on useita parametreja: päätteiden lukumäärä, sanomajonojen enimmäispituudet sekä erilaisia muunnelmia perusprotokollaan.

Tietokone tutkii mallin kaikki mahdolliset tilat ja käyttäytymiset ja hälyttää, jos järjestelmän on mahdollista lukkiutua tai toimia muuten virheellisesti. Esimerkkitapauksessamme virheeksi määriteltiin muun muassa se, että päätte on aktiivinen vaikka keskusyksikkö on vapauttanut resurssin.

Virheen löytäessään tietokone esittää järjestelmän alkutilasta virheelliseen käyttäytymiseen johtavan suorituspolun, joka



Kuva2: Järjestelmän toimintaa kahdessa eri tilanteessa kuvaavat sanomakaaviot. Perustapauksessa pääte varaa ja vapauttaa resurssin. Järjestelmä voi toimia virheellisesti, jos sanomia katoaa.

Message charts that demonstrate the system. The states used in this model are: idle (perustila), reserving (varaamassa), active (aktiivinen), canceling (perumassa), available (vapaa) and allocated (varattu).

auttaa suunnittelijaa korjaamaan järjestelmää. Yhteyskäytäntöjen eli protokollien toimintaa on havainnollista esittää kuvan 2 kaltaisina sanomakaavioina, joissa eri vastinolioiden tilasiirtymät yhdistyvät sanomaliikenteeseen.

Tutkiessaan järjestelmää kuvaavan mallin tila-avaruuden eli kaikki sen saavutettavissa olevat tilat ja siirtymät verifiointityökalu luo mallin käyttäytymistä kuvaavan saavutettavuusgraafin. Suunnatun graafin solmut ovat järjestelmän saavutettavissa olevia tiloja ja kaaret tilojen välisiä siirtymiä. Esimerkkijärjestelmämme tila koostuu paitsi päätteiden ja keskusyksiköiden paikallisista tiloista myös siirtoyhteyksien ja sanomajonojen tiloista. Verifiointia varten on tehtävä yleistyksiä ja yksinkertaistavia ole-

tuksia, jotta tila-avaruus rajoituisi hallittavissa olevan kokoiseksi.

Yksinkertaistuksia on tehtävä tarkkaan harkiten, jotta yksinkertaistettu malli kuvaaisi järjestelmän toimintaa tutkittavien ominaisuuksien kannalta. Teoreettisen, käytännöstä vieraantuneen mallin tutkimisesta voi olla vain teoreettista hyötyä.

Protokollia tutkittaessa on hyödyllistä olettaa, että alemmat kerrokset toimivat vaatimusmäärittelyjensä mukaisesti. Esimerkkimme sovelluskerroksen mallista jätettiin kaikki siirto- ja laitteistokerroksen yksityiskohdat pois. Malliin jäi ainoastaan mahdollisuus sanomien katoamiseen ja monistumiseen.

Järjestelmältä edellytettävät ominaisuudet voidaan jakaa kahteen osa-alueeseen: turval-

lisuusominaisuuksiin ("mitään pahaa ei tapahdu") ja elävyysominaisuuksiin ("järjestelmässä tapahtuu edistystä"). Elävyysominaisuudet kulkevat käsi kädessä reiluusoletusten kanssa. Tutkittaessa sitä, saavuttavatko lähetetyt sanomat vastaanottajan, on mielekäästä olettaa, ettei yhteys ole jatkuvasti poikki.

Jos järjestelmää mallinnettaessa on tehty järjeviä yksinkertaistuksia, koko järjestelmä voidaan analysoida nopeimmillaan minuuteissa, yleensä tunneissa. Formaali menetelmät eivät korvaa perinteistä testaamista, sillä järjestelmissä on yleensä myös peräkkäistä laskentaa, joka on jätettävä rinnakkaisuutta kuvaavista malleista pois. Liian yksityiskohtaisella mallilla on helposti huomattavasti enemmän tiloja kuin nykyiset työkalut

pystyvät käsittelemään.

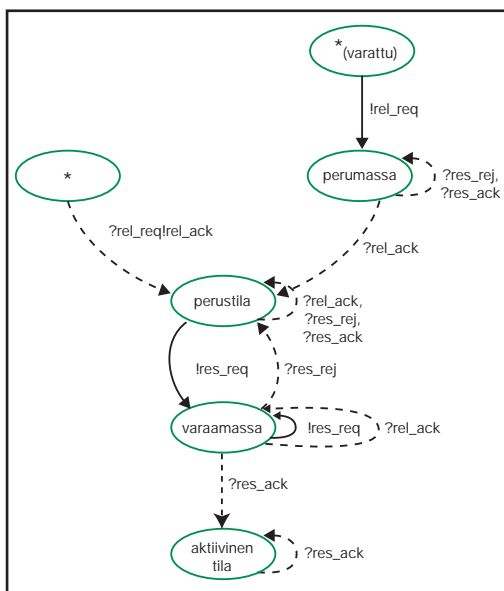
Testaamalla voidaan varmentaa rinnakkaisuutta ja satunnaisuutta sisältämättömän järjestelmän virheetön toiminta testien kattamissa tapauksissa. Oletuksiin ja yksinkertaistuksiin perustuvaa verifiointia voidaan pitää eräänlaisena laajennettuna testausmenetelmänä rinnakkaisille tai satunnaisesti käyttäytyville järjestelmille.

Formaalin analyysin tulokset pätevät järjestelmästä tehdyille mallille, joka käyttäytyy yleensä hieman eri tavalla kuin kaikki yksityiskohdat huomioon ottaessa järjestelmän toteutus. Jos verifiointimallia pidetään toteutuksen vaatimusmäärittelyinä, mallista saadut tulokset ovat melko varmasti yleistettävissä toteutukseen.

Verifiointituloksia

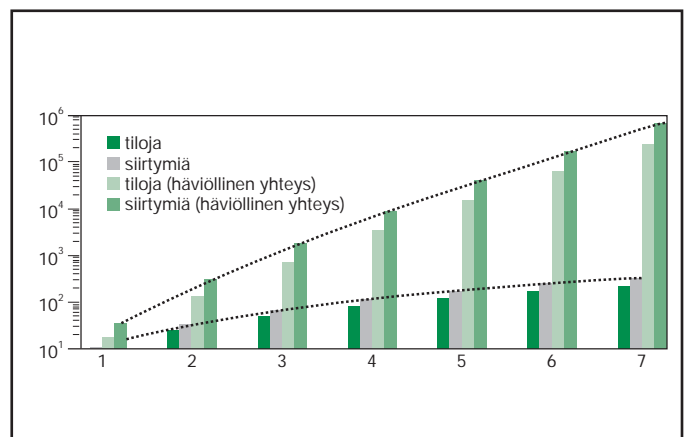
Kun kuvan 3 mukaisesti korjatun järjestelmän eri muunnelmia tutkittiin Teknillisen korkeakoulun tietojenkäsittelyteorian laboratoriossa kehitetyllä työkalulla, havaittiin, että jos sanomat eivät katoa eivätkä toistu, sanomajonojen enimmäispituus on äärellinen: päätteillä 3 sanomaa ja keskusyksiköllä 2n-1, missä n on päätteiden määrä.

Protokolla tuntuu toimivan erilaisissa verkkoympäristöissä, jopa laitteiden ollessa kytkettyinä toisiinsa kahdenvälisillä mielivaltaisesti nopeuttamalla muuttavilla yhteyksillä. Siirtokerros saa hävittää ja toistaa viestejä mutta ei muuttaa niiden järjestystä. Päätteitä voidaan sammuttaa ja kytkeä uudelleen toimintaan milloin tahansa, kunhan ne eivät ole juuri varaamassa resurssia. Sekin rajoitus poistuu lisäämällä päätteiden tilakoneeseen erillinen käynnistystila perustilan rinnalle.



Kuva3: Pääteiden tilakone korjattuna sellaiseksi, että virheitä ei esiinny, vaikka sanomia katoaisi tai toistuisi mielivaltaisesti.

A corrected state machine of the client logic. The states used in this model are: reserving (varaamassa), canceling (perumassa), idle (perustila) and active (aktiivinen).



Kuva4: Korjatun järjestelmän mallin kahden muunnelman tila-avaruuden koko päätteiden määrän funktiona. Mallissa on Ethernetin kaltainen väylä ilman sanomajonoja. Nopeammin kasvavassa mallissa sanomat voivat kadota. Huomaa logaritminen asteikko.

The state space grows rapidly when clients are added to the model.

Kaikissa mallin kaikki käytetyt kättymiset kattavissa analyysissä ilmeni, että järjestelmä pääsee aina globaaliin alkutilaansa eli mitkään sen osat eivät voi jumittua. Vaatimus siitä, että pääte voi olla aktiivisena ainoastaan resurssin ollessa varattuna sille, toteutui niin ikään kaikissa mallin muunnelmissa.

Kuva 4 havainnollistaa yksinkertaistusten vaikutusta mallin hallittavuuteen. Erittäin rajoitettu kommunikaation malli, jossa ei ole minkäänlaisia sanomajonoja eikä sanomien häviöitä, saa graafin kasvamaan kutaquinkin lineaarisesti pääteiden määrään nähden. Kaikkien muiden tutkittujen muunnelmien tila-avaruus kasvaa eksponentiaalisesti kasvatettaessa sanomajonon enimmäispituutta tai pääteiden määrää.

Mitä enemmän järjestelmän osat voivat suorittaa toimintoja toisistaan riippumatta, sen hankalammaksi analyysi muuttuu. Eräs esimerkimmillämme muunnelma koettelee työkalumme suorituskyvyn rajoja jo kolmen päätteen tapauksessa. ●

Aiheesta enemmän

Maria-työkälun kotisivu:

www.tcs.hut.fi/maria/

World of Petri Nets:

www.daimi.au.dk/PetriNets/
Gerard J. Holzmann. Design and Validation of Computer Protocols. Prentice Hall, Englewood Cliffs, NJ, Yhdysvallat, 1991.

Wolfgang Reisig. Elements of Distributed Algorithms. Modeling and Analysis with Petri Nets. Springer-Verlag, Berliini, Saksa, 1998.

Taustat

Kirjoittaja: TkL Marko Mäkelä toimii tutkijana TKK:n Tietojenkäsittelyteorian laboratoriossa.

Yhteystieto:

marko.makela@hut.fi

Tutkimus: Maria – Modular Reachability Analyser

Yhteistyössä: EKE Electronics, Genera, Nokia Research Center ja Nokia Networks.

Teknologiaohjelma: ETX

Pakettikytkentäisten järjestelmien kerrosmalli

Sanomien välitykseen perustuvien eli pakettikytkentäisten järjestelmien ymmärtämistä helpottaa niiden jakaminen kerroksiin. Standardointijärjestö ISO on kuvannut yhteyskäytäntöjä eli protokollia seitsemänkerroksisella OSI-mallilla. Protokollapino koostuu kerroksista, jotka toteuttavat ylöspäin tarjoamansa palvelut alempien kerrosten palveluilla. Viisi kerrosta riittää useimpien yhteyskäytäntöjen kuvailemiseen.

Fyysinen kerros välittää sanomia ympäristön ja ylempien kerrosten välillä yleensä sähkömagneettisesti. Sanomat voivat vääristyä.

Siirtokerros mahdollistaa laitteiden välisen sanomaliikenteen lisäämällä välitettävien sanomien osoitteet ja tarkistussumman. Jos tarkistussumma on luotettava, sanomat eivät voi käytännössä koskaan vääristyä. Sen sijaan ne voivat kadota, sillä siirtokerros hävittää vastaanottamansa tunnistamattomat tai tarkistussummaltaan virheelliset viestit.

Verkkokerros huolehtii useita verkkoja käsittävissä järjestelmissä sanomien reitittämisestä verkkojen välillä. Yksinkertaisissa sovelluksissa tällaista tarvetta ei ole.

Kuljetuskerros tarjoaa katkeamattoman tietovuon kahden laitteen välille sanomajonon ja kuittausmekanismien avulla. Tunnetuimpia kuljetuskerroksen toteutuksia on Internetin useimpien sovellusten käyttämä TCP (Transmission Control Protocol).

Sovelluskerros on protokollapinin varsinainen käyttäjä. Jos sovellus huolehtii kadonneiden viestien lähettämisestä uudelleen eikä sekoja, vaikka viestit kahdentuisivat, monimutkainen kuljetus voidaan jättää toteuttamatta. Esimerkiksi Internetin nimipalvelu (DNS) käyttää tehokkuuden vuoksi epäluotettavaa sanomavälitystä UDP (User Datagram Protocol).

Eräs siirtokerroksen toteutustapa

Ethernet-verkon vuoronjakamiskäytännössä CSMA/CD (Carrier Sense Multiple Access with Collision Detection) lähettämiseen kiinnostunut laite kuulostee linjaa. Mikäli linja vaikuttaa olevan vapaana, se lähettää sanomansa mitaten samalla lähettämäänsä signaalia. Jos signaali vääristyy, laite toteaa törmäyksen ja jää odottamaan linjan vapautumista. Linjan vapauduttua se odottaa satunnaisen ajan ennen uuteen lähetyksyritykseen ryhtymistä. Odotusaika on satunnainen, jotta törmäyksen osapuolet eivät aloittaisi uutta lähetystä samanaikaisesti.

Ethernet-tyylinen vuoronjakamiskäytäntö soveltuu kaksijohtimiseen RS-485-väylään erittäin hyvin. Siirtokerros ottaa lähetysten aikana RS-485-puskurin vastaanottimelta tulevaa signaalia vastaan. Jos vastaanotettu merkki eroaa juuri lähetetystä, siirtokerros toteaa törmäyksen tapahtuneeksi ja jää odottamaan linjan vapautumista.

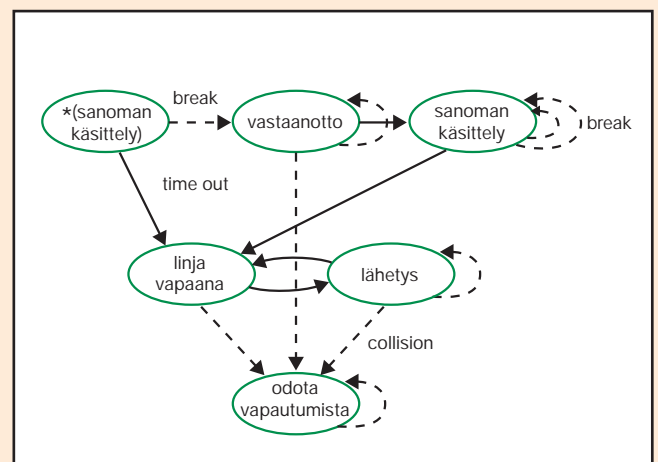
Kuvassa 5 on yksinkertaisesti kaavio siirtokerroksen toiminnasta. Tarkistussummaan ja osoitteisiin liittyvät yksityiskohdat on jätetty pois. Siirtokerroksessa on viisi tilaa: linja vapaana, lähetys, vastaanotto ja vastaanotetun sanoman käsittelyminen.

Tähdellä merkitty tila on lyhennysmerkintä: kaikki siinä kuvatut siirtymät ovat mahdollisia muissa paitsi käsittelytilassa. Siirtymä break liittyy sanomien kehystykseen. Jotta laitteet tunnistaisivat sanomien rajat kaikissa tilanteissa, sanomat alkavat RS-232:n break-merkillä eli pelkkiä nollabittejä sisältävällä sanalla, jonka päätävää ykkösbittii on viivästetty.

Lisäksi laitteet valvovat, että sanomat ovat yhtenäisiä lähetysryöpyjä. Jos sanomalle ei tule jatkoa kahden sanan lähettämiseen kuluvana aikana, lähetysten oletetaan keskeytyneen ja linjan vapautuneen. Tähän liittyy aikavalvontaa kuvaava siirtymä timeout.

Koska järjestelmä ei ota käsittelytilassa mitään vastaan, käsittely on syytä olla nopeaa. Jos sovellus on hidas, se voidaan jakaa prosesseihin, jolloin käsittelytila ainoastaan siirtää sanomat erillisen käsittelyprosessin syötejonoon.

Siirtokerroksen toteutus Atmelin AVR-sarjan mikroohjaimille koostuu viidestä keskeytyskäsittelijästä ja noin 230 assembler-koodirivistä. Tilakonemuotoinen esitys auttaa ymmärtämään järjestelmän toimintaa huomattavasti paremmin kuin selkeäkin ohjelmistauksen yhteyteen kirjoitetut kommentit.



Kuva5: Siirtokerroksen vastaanottopuolen toiminta. Yhtenäiset nuolet kuvaavat sisäisiä siirtymiä. Nuoli timeout havaitsee sanoman enneaikaisen päättymisen. Kaikki sanomat alkavat erityisellä break-merkillä.

The receiver part of a data link layer. The states used in this model are; process (sanoman käsittely), receive (vastaanotto), idle (perustila), send (lähetyks) and wait (odota vapautumista).