Helsingin yliopisto
Tietojenkäsittelytieteen laitos
Ubicomp – tulevaisuudenkuvako? -seminaari
Ohjaaja FT Lea Kutvonen

ADAPTIIVISET SOVELLUKSET

Laatinut: Gitta Lehto

Tallennettu: 6.11.2000

TIIVISTELMÄ

Nykyajan tietojärjestelmät sisältävät kannettavia tietokoneita, jotka voivat olla joko täysin ilman verkkoyhteyttä tai vaihtoehtoisesti olla yhteydessä hitaaseen langattomaan (GSM) tai nopeaan kiinteään verkkoon (ethernet). Suuret suorituskyvyn vaihtelut verkkoyhteyden palvelun laadussa (*QoS, Quality of Service*) aiheuttavat erityisiä vaatimuksia yhteyttä käyttäville järjestelmille. Vaihto suorituskyvyltään erilaisten verkkojen välillä vaatii järjestelmiltä sopeutumista vaihtelevaan yhteyden laatuun. Toisaalta verkkoyhteyttä käyttävällä järjestelmällä voi olla omia vaatimuksia, esimerkiksi suorituskyky- tai kustannusvaatimuksia, verkkopalvelun tason suhteen, jolloin järjestelmällä pitäisi olla mahdollisuus valita palvelun taso mahdollisimman hyvin tarpeitansa vastaavaksi. Joustavuus verkkopalvelun laadussa yhdistettynä langattomien tietoverkkojen luontaiseen epäluotettavuuteen johtaa siihen, että saadakseen parhaan hyödyn irti tarjolla olevista verkkoyhteyksistä, tulee järjestelmien nyt ja tulevaisuudessa pystyä dynaamisesti sopeutumaan palvelutasoltaan vaihteleviin verkkoyhteyksiin.
Adaptoitumista, eli sopeutumista voi tapahtua järjestelmän eri tasoilla. Peruskerroksessa voidaan pyrkiä tasoittamaan langattoman ympäristön kenttävaihtelua sopeutuvalla virranhallinnalla. Siirtoyhteyskerroksessa suojaudutaan siirtovirheitä ja vuonvaihteluja vastaan sopeutuvilla vuonvaraustekniikoilla. Kuljetuskerroksessa voidaan uudelleenneuvotella dynaamisesti yhteyden parametreja. Sovelluskerroksessa voi sopeutumista tapahtua monilla eri tavoin, kuten hierarkisella uudelleenkoodauksella, tehokkaalla tiivistämisellä, kaistanleveyden tasoittamisella, nopeuden säätelyllä, virheiden hallinnalla ja adaptiivisella synkronoinnilla.
Tässä dokumentissa käsitellään adaptiivisuutta enimmäkseen systeemiarkkitehtuurin ylimmillä tasoilla, eli käyttäjän ja sovelluksen näkökannalta. Jonkin verran perehdytään myös käyttöjärjestelmän ja sovellusalustan adaptoitumismahdollisuuksiin, sillä vaikka nykyajan järjestelmät tukevat adaptiivisuutta yleensä vain joissakin järjestelmän osissa, tulevaisuuden tietojärjestelmissä adaptiivisuutta tulee esiintymään kaikilla tasoilla.

Sisällys

1. MIKSI ADAPTOITUA?

Teknologian kehittymisen myötä on tullut yleisesti saataville kohtuulliseen hintaan erilaisia verkkoyhteysvaihtoehtoja, kuten ethernet, lähiverkot (LAN), laaja-alueverkot (WAN) ja langattomat modeemit. Kannettavalla tietokoneella otetaan tavallisesti yhteyttä monenlaisiin tietoverkkoihin, jotka tarjoavat eritasoisia verkkopalveluja. Esimerkiksi kiinteät verkot, kuten ethernet, ovat halpoja ja laajakaistaisia yhteyksiä, mutta ne toimivat ainoastaan rajoitetulla alueella, kun taasen langattomat yhteydet ovat laaja-alaisia, mutta kalliimpia ja kapeakaistaisempia. Taulukossa 1 on esitetty langattomille palvelimille tavallisten verkkoyhteyksien QoS-parametrien arvoja (*Quality of Service*, palvelun laatu).

Network	Throughput	BER	Packet oriented	Multicast (group)	Dial-up
Ethernet	10 Mbps	10^{-9}	Yes	Yes	No
RF LANs (e.g. WaveLAN)	242 Kbps - 5.7 Mbps	10^{-8}	Yes	Yes	No
Infra-red	9.6 Kbps - 4 Mbps	$>10^{-6}$	Yes	Yes	No
GSM	9.6 Kbps (corrected)	10^{-8}	No	No	Yes
TETRA	7.2 Kbps-28.8 Kbps (raw, 3 protection levels)	$>10^{-3}$ (raw)	Depends on mode	Yes	Yes

Taulukko 1. QoS-parametrien arvoja yleisimmille verkoille [MOST99].

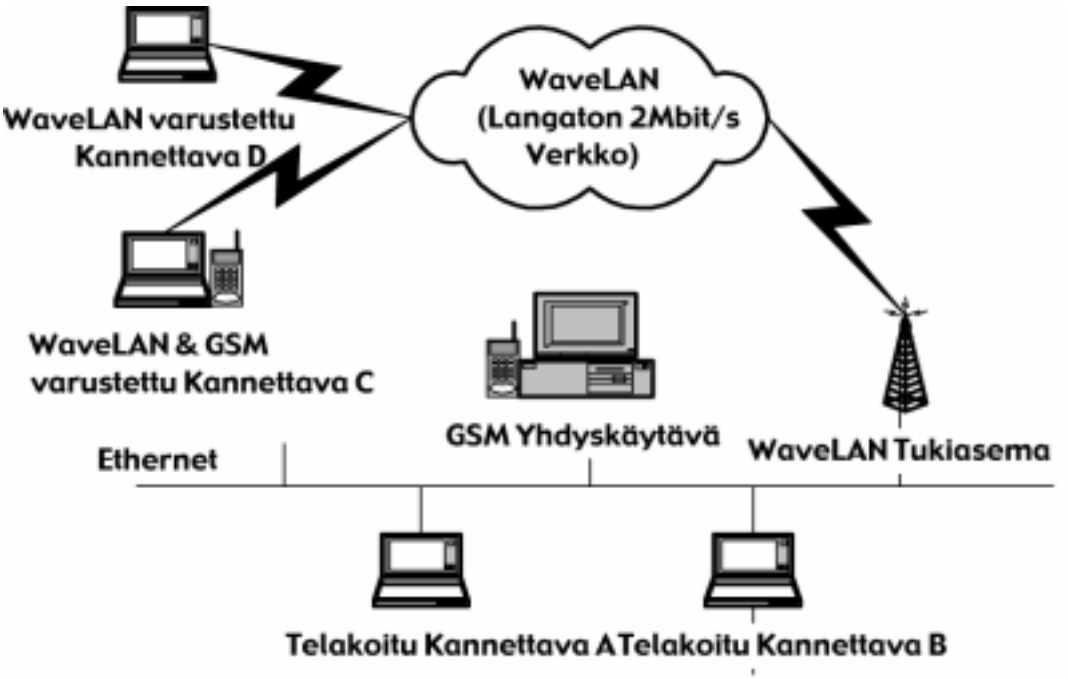
Kiinteissä verkoissa sovellukset ja protokollat on räätälöity vastaamaan verkon ominaisuuksia, mutta langattomissa ympäristöissä sovelluksilta vaaditaan dynaamista sopeutumista muutoksiin verkkoyhteydessä. Esimerkiksi modernissa toimisto- tai tutkimusympäristössä mobiili tietokone voi olla kiinteässä yhteydessä verkkoon käyttäjän pysytellessä omassa huoneessaan. Kun käyttäjä liikkuu toimistoympäristössä, voidaan yhteyttä ylläpitää infrapunan tai WaveLANin välityksellä. Jos käyttäjä liikkuu paikallisen toimistoympäristön ulkopuolelle, vaihtuu yhteyden status langattomaksi laaja-alueverkoksi. Tällaisessa tilanteessa verkkoyhteyden nopeus voi vaihdella GSM-verkkojen tarjoamasta 9,6 kbit/s siirtonopeudesta ATM-runkoverkolla saavutettavaan 2,4 Gbit/s nopeuteen.

Kaikilla verkkoyhteyden ominaisuuksilla on vaikutuksensa järjestelmän suorituskykyyn. Yhteyden laadun vaihtelu verkon eri osissa voi olla huomattavaa suuruusluokkaa. Esimerkiksi kaistanleveys voi vaihdella kilobiteistä megabitteihin sekunnissa ja latenssi mikrosekunneista sekunteihin. Osa verkosta voi olla lähes ruuhkaton ja häviötön ja toinen osa erittäin ruuhkainen ja lähes kaikki paketit hävittävä. Langattoman lähiverkon, kuten WaveLANin, ominaisuudet eroavat vastaavaan kiinteään verkkoon verrattuna suurempana bittivirheiden määränä, joka johtaa suurempaan kadonneiden pakettien määrään [MOST99].

Vaihto suorituskyvyltään erilaisten verkkojen välillä vaatii järjestelmiltä sopeutumista vaihtelevaan yhteyden laatuun. Joustavuus verkkopalvelun laadussa yhdistettynä langattomien tietoverkkojen luontaiseen epäluotettavuuteen johtaa siihen, että saadakseen parhaan hyödyn irti tarjolla olevista verkkoyhteyksistä, tulee järjestelmien pystyä dynaamisesti sopeutumaan palvelutasoltaan vaihteleviin verkkoyhteyksiin.

2. MIKSI ADAPTOITUVIA SOVELLUKSIA?

Hajautettujen järjestelmien sovellukset pyörivät ympäristöissä, jotka voivat sisältää erilaisia verkkoarkkitehtuureja sekä sovellusalustoja. Lisäksi hajautetuissa ympäristöissä sovellus harvemmin pääsee yksinään käyttämään resursseja, sillä yleensä ne jaetaan monen sovelluksen ja käyttäjän kesken. Tästä johtuen sovellukset saavat vaihtelevan tasoista palvelua ympäristöltään. Kuvassa 1 on esimerkki hajautetusta mobiilista järjestelmästä.



Kuva 1. Hajautettu mobiili järjestelmä

Monilla hajautettujen ympäristöjen sovelluksilla on kriittiset vasteaikavaatimukset. Suoritettavan tapahtuman vasteaika riippuu pitkälti resurssien saatavuudesta, eli saatavilla olevasta kaistanleveydestä ja prosessoriajasta. Esimerkiksi nopean prosessorin tai verkkoyhteyden suorituskyky voi heikentyä kuormituksen kasvaessa, mutta jos tällaisessa tilanteessa sovellus osaisi vaihtaa toiseen järjestelmän osaan lennossa, ei käyttäjä välttämättä huomaisi minkäänlaista hidastumista toiminnassa. Vastaavasti suoritettaessa pitkäkestoista hajautettua simulaatiota voidaan ruuhkaisten linkkien

vaikutusta välttää siirtämällä simulaatiota toiseen verkon osaan. Tietokanta, johon käyttäjät tekevät samanaikaisesti useita vaativia kyselyjä, voi käyttäjälle vaikuttaa täysin pysähtyneeltä, mutta jos data olisi replikoitu myös muille palvelimille, voisi sovellus vaihtaa palvelinta ja saada vastauksen sopivassa ajassa. Videokuvan siirtäminen ruuhkaisen verkon läpi voi johtaa moniin hylättyihin kehyksiin ja siten kuvan laadun heikkenemiseen, mutta älykäs filtteri voisi päätellä mitkä kehyksistä ovat vähemmän tärkeitä, ja hylkäämällä näitä kehyksiä ja täten vähentämällä tarvittavaa kaistanleveyttä, se pystyisi säilyttämään audio-video synkronisaation.

Suuret suorituskyvyn vaihtelut verkkoyhteyden palvelun laadussa aiheuttavat erityisiä vaatimuksia yhteyttä käyttäville sovelluksille. Toisaalta verkkoyhteyttä käyttävällä sovelluksella voi olla omia vaatimuksia, esimerkiksi suorituskyky- tai kustannusvaatimuksia, verkkopalvelun tason suhteen, jolloin sovelluksella pitäisi olla mahdollisuus valita palvelun taso mahdollisimman hyvin tarpeitansa vastaavaksi. Adaptiivisuuden tukemisella voidaan myös säästää, sillä sovellus voi valita tarpeitaan vastaavan alemman palvelutason, esimerkiksi taatun palvelutason sijasta parhaan saatavilla olevan palvelutason.

3. ADAPTIIVISUUS JÄRJESTELMÄN ERI TASOILLA

Adaptoitumista, eli sopeutumista, voi tapahtua järjestelmän eri tasoilla. OSI-viitemallin (*Open Systems Interconnection*) kerrosrakenteen mukaisesti jaoteltuna voidaan peruskerroksessa pyrkiä tasoittamaan langattoman ympäristön kenttävaihtelua sopeutuvalla virran hallinnalla. Siirtoyhteyskerroksessa suojaudutaan siirtovirheitä ja vuonvaihteluja vastaan sopeutuvilla vuonvaraustekniikoilla. Kuljetuskerroksessa voidaan uudelleenneuvotella dynaamisesti yhteyden parametreja. Sovelluskerroksessa voi sopeutumista tapahtua monilla eri tavoin, kuten datan hierarkisella uudelleenkoodauksella, tehokkaalla tiivistämisellä, kaistanleveyden tasoittamisella, nopeuden säätelyllä, virheiden hallinnalla ja adaptiivisella synkronoinnilla [STREAM99].

Jotta suuria muutoksia verkkoyhteydessä pystyttäisiin käsittelemään, täytyy protokollapinon kerroksien muuttaa toimintamalliaan. Yleensä sovellukset ja sovellustason protokollat ovat parhaita sopeutujia. Esimerkiksi www-selain voi päättää vaihtaa värikuvat musta-valkokuviksi jos verkkoyhteys on huono, tai jättää kuvat kokonaan pois. Samoin, sopeutuva FTP voi lykätä tiedostojen latausta, jos verkkoyhteys on kallista. Sovellustason adaptoituminen ei kuitenkaan ole aina tarkoituksenmukaista tai mahdollista. Datan semanttisuudesta riippumaton sopeutuminen voidaan joskus suorittaa tehokkaammin alemmilla tasoilla. Esimerkiksi pakettien uudelleenlähetys langattomassa ympäristössä voidaan toteuttaa sovellukselle näkymättömällä tavalla ja näin vähentää häviämisiä tukiaseman vaihtumisen aikana [PROTO99].

3.1. Adaptiivisuus systeemitasolla

Käyttöjärjestelmä ja erityisesti protokollapino ovat luonnollinen adaptointitekniikoiden toteutusympäristö. Yksinkertaisimmillaan käyttöjärjestelmä voi tukea valtavaa määrää protokollia, joista jokainen on erityisesti räätälöity vastaamaan tietynlaista kommunikaatioteknologiaa. Kun verkkoympäristö vaihtuu, voi järjestelmä dynaamisesti vaihtaa uutta ympäristöä parhaiten tukevaan protokollaan.

Esimerkki protokollatason sopeutumisesta on vaihtoehtoisten protokollien käyttö sovitettaessa pakettien ajoitusta kuljetusteknologiaan. Suurinopeuksisissa verkoissa pakettien lähettämisen hallinnalla pyritään välttämään ruuhkaa ja ylläpitämään jatkuvan datan lähettämiseen tarvittavia QoS-parametreja. Esimerkiksi modeemilinjaa käytettäessä latenssia ja yhteyden kustannuksia voidaan vähentää ryhmittelemällä dataa. Järjestelmissä, joissa kaistanleveys on erittäin rajoitettu, esimerkiksi langatonta WANia käyttävissä järjestelmissä, on syytä priorisoida lähetettävä data jonkin olennaisen kriteerin perusteella.

Protokollatason sopeutuminen on monimutkaista tapauksissa, joissa on samaan aikaan käytettävissä useita ”päällekkäisiä” verkkoja, jolloin protokollatason täytyy päätellä mikä tarjolla olevista kujetustavoista sopii parhaiten kyseessä olevalle datalle [MOST99].

Adaptoituminen muutoksiin verkkoyhteydessä ei ole ideana kovinkaan uusi, sillä verkkotason sopeutuminen on ollut osa Internet-protokollaa alusta alkaen. TCP (*Transmission Control Protocol*), internetin siirtoprotokolla, sopeuttaa lähettämänsä datan määrän verkon kapasiteettia vastaavaksi mittaamalla pakettien häviötä ja siirtoviivettä.

3.2. Adaptiivinen sovellusalusta (*Middleware*)

Monet projektit ovat tutkineet adaptiivisuutta tukevien palvelujen toteuttamista sovellusalustatasolla ja kolme perusratkaisua on noussut esille. Sovellusalustan palvelut voivat pyrkiä vähentämään sovelluksen tarvitsemaa kaistanleveyttä esimerkiksi lisäämällä datan tiivistystä ennen lähettämistä. Tämä tekniikka on erityisen tehokas silloin kun lähetetään jatkuvaa dataa, johon voidaan soveltaa tietoa hukkaavaa tiivistämistä tai epäolennaisen tiedon pois suodattamista, eli filtteröintiä.

Toinen vaihtoehto on tuottaa palveluja, jotka voivat noutaa korkealaatuisen yhteyden aikana dataa välimuistiin ja pystyvät näin etukäteen varautumaan heikon yhteyden varalle. Jalostettu versio edellä mainitusta on priorisoida data etukäteen ja noutaa tärkeät tiedot valmiiksi hyvän yhteyden aikana. Esimerkiksi yhteydettömät postin- ja uutistenlukuohjelmat noutavat etukäteen yhteenvetoinformaation (otsikot ja aiheet) ja loput sanomasta tai artikkelista jos ja kun sitä tarvitaan. Tukipalvelut voivat soveltaa tätä tekniikkaa käytännössä tuomalla dokumentin ensimmäisen sivun näytölle ja noutamalla muita sivuja sillä aikaa kuin käyttäjä lukee ensimmäistä.

Kolmas lähestymistapa on dynaamisesti luoda uusi yhteys clientin ja palvelimen välille aina kun verkon tila muuttuu. Yhteydettömänä aikana clientti voi kytkeytyä

paikalliseen proxy-palvelimeen, kunnes verkkoyhteys voidaan jälleen tarjota. Tällä hetkellä suurin osa sovellusalustoista perustuu RPC-protokollan (*Remote Procedure Call*) mukaiseen prosessien väliseen kommunikointimuotoon, joka ei suvaitse keskeytyksiä yhteydessä. Perinteisen RPC:n semantiikan muokkaamiseksi vähemmän synkroniseksi on tehty paljon työtä, sillä erityisesti langattomassa ympäristössä on suotavaa pystyä sopeutumaan katkoksiin yhteydessä [MOST99].

3.3. Adaptiivisuus sovellustasolla

Sovellusalustan tukipalvelut pyrkivät lieventämään yhteyskerroksen muutosten vaikutusta sovelluksen toimintaan, mutta monissa tapauksissa sovellus itse pystyy parhaiten sopeutumaan muutoksiin verkon tilassa. Jotta adaptiivisuutta voidaan tukea sovellustasolla, vaatii se erityisesti dynaamisessa ympäristössä toimimaan suunniteltujen sovellusten toteuttamista.

Monet sovellukset pystyvät yhtenään tarkentamaan vaatimuksiaan tarvitsemiensa resurssien suhteen. Adaptiivinen sovellus voi lisätä tai vähentää käyttämiensä prosessorien tai solmujen lukumäärää tai siirtyä suorittumaan kokonaan toiseen verkon osaan reagoidessaan muutoksiin verkon tilassa. Myös verkon resurssit antavat sovellukselle mahdollisuuden toteuttaa adaptiivisuutta. Kaistanleveydellä voidaan tehdä lehmänkauppoja muiden parametrien, kuten tiedon tarkkuuden tai siirrettävien objektien laadun kanssa. Sovellus voi esimerkiksi muuttamalla siirrettävän videokuvan kehyksen kokoa tai niiden lukumäärää suurentaa tai pienentää siirtoon vaadittavaa kaistanleveyttä. Vaihtoehtoisesti sovellus voi tiivistämisellä pienentää kaistanleveysvaatimuksiaan, mutta tällöin se tarvitsee vastakaupaksi prosessoriaikaa tiedon tiivistämiseen ja purkamiseen.

3.4. Adaptiivinen käyttäjä

Järjestelmän käyttäjän voisi ajatella olevan sopeutumisen ensimmäisellä tasolla, sillä käyttäjän teot vaikuttavat epäsuorasti suurimpaan osaan verkon resurssivaatimuksista. Jos käyttäjä olisi tietoinen valintojensa aiheuttamista toimenpiteistä verkkotasolla, voitaisiin monet verkon rajoituksista aiheutuvat ongelmat välttää. Jos esimerkiksi internet-selain tai FTP-palvelu antaa käyttäjälle palautetta siirron edistymisestä tai jäljellä olevasta ajasta, voi käyttäjä helposti päättää haluaako jatkaa operaatiota.

Helsingin Yliopistolla kehitetty Mowgli WWW-ohjelmisto mobiilille Internetin käyttäjälle tarjoaa hyvän esimerkin adaptiivisuutta tukevasta käyttöliittymästä. Käyttöliittymän dokumenttiasetusten avulla käyttäjä voi esimerkiksi valita haetaanko upotetut kuvat tai taustakuvat dokumentin yhteydessä ja asettaa kokorajoituksia dokumentin objekteille. Lisäksi käyttäjä pääsee vaikuttamaan mm. tiedon tiivistykseen, esitystapaan, verkkoyhteyteen, välimuistin kokoon ja erätiedonsiirron ajastamiseen vaikuttaviin parametreihin. Mowglin WWW agentti muokkaa jokaisen käyttäjälle näytettävän dokumentin, lisäämällä sen ylälaitaan rivin painikkeita, joilla käyttäjä voi hallita dokumenttia esimerkiksi lukitsemalla sen välimuistiin tai tuhoamalla sen välimuistista. Lisäksi agentti lisää dokumenttiin jokaisen hypertekstilinkin perään pienen ikonin, jonka ilmentymä paljastaa onko

dokumentti noutamatta, parhaillaan noudossa vai noudettu jo. Käyttäjä voi lisätä noutamattoman dokumentin eräsiirtojonoon klikkaamalla ikonia [MOWGLI96].

Tavallisen käyttäjän kannalta ajatellen siirtyy adaptiivisuuden tukeminen takaisin sovellustasolle, sillä huolellisesti suunnitellut informatiiviset käyttöliitymät ovat helpoin tapa saada käyttäjä sopeutumaan muutoksiin järjestelmäympäristössä.

4. ADAPTIIVINEN SOVELLUS

Nykyään monet verkkosovellukset ovat adaptiivisia ja pystyvät tekemään sovellustason päätöksiä, jotka perustuvat verkon ja yhteyden päätepisteen tilaan. Tällaiset sovellukset jaetaan yleensä kahteen kategoriaan. Ensimmäisessä kategoriassa ovat sovellukset, jotka valitsevat palvelimen, johon olla yhteydessä palvelimen tämän hetkisen tilan perusteella. Palvelimen valitsevan sovelluksen täytyy olla tietoinen verkon samansisältöisistä palvelimista, joista se valitsee saamansa verkko- ja palvelininformaation perusteella sopivan palvelimen. Mekanismi, jolla sovellukselle tuotetaan lista palvelimista tai jolla sovellus valitsee niistä sopivan, vaihtelee sovelluksesta toiseen. Toiseen kategoriaan kuuluvat sovellukset, jotka kommunikoidessaan kaukaisten palvelimien kanssa, valitsevat tiedon esitystavan sen hetkisiin olosuhteisiin perustuen. Näiden sovellusten päämääränä on yleensä pitää vasteaika tai tehokkuus sopivana sisällön tarkkuuden kustannuksella. Myös tässä kategoriassa sovellukset käyttävät erilaisia mekanismeja päättäessään tiedon esitystavasta [STEMM99].

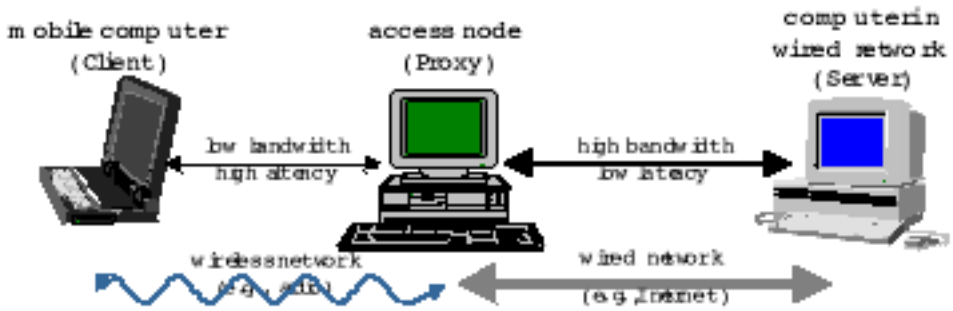
4.1. Adaptiivisuuden tukeminen

Sovellusten adaptiivisuuden tukemiseen on pyritty kahdella eri tekniikalla: joko laajentamalla systeemialustaa siten, että sovellukset voivat saada palautetta verkon tilasta, tai käyttämällä proxyjä suorittamaan adaptoituminen sovelluksen puolesta. Ensimmäinen vaihtoehto yleensä edellyttää uusien sovellusalustojen kehittämistä, jotta verkon tilan näkymättömyys sovellukselle saadaan häivytettyä. Näihin laajennoksiin yhdistetään usein muitakin hienouksia, kuten datan puskurointia, jotta sovellus voisi jatkaa operaatioitaan myös verkkoyhteyden katkosten aikana. Toinen vaihtoehto perustuu seuraavalla sivulla olevan kuvan 2 kaltaisiin proxy-arkkitehtuureihin, joissa taas mahdollistetaan suodatus-, välimuisti- ja käännöskomponenttien sijoittaminen palvelimen ja clientin väliselle kommunikaatioväylälle. Proxy-palvelin pystyy muokkaamaan palvelimen ja clientin välistä kommunikaatiovirtaa, sekä palvelimelta clientille että toisin päin, vallitseviin QoS-ominaisuuksiin sopeutuvaksi. Tämä sopeuttaminen tapahtuu joillakin seuraavista proxy-arkkitehtuurin perustoimenpiteistä [GENERA99].

- Suodattaminen (*filtering*): Suodattava proxy vähentää kuljetettavan datan määrää poistamalla siitä epäolennaisempia osia.
- Muuntaminen (*transforming*): Muuntava proxy muuttaa prosessoimansa datavirran tyyppiä vähentääkseen sen voimakkuutta tai sovittaakseen sen formaatin clientille helposti esitettävään muotoon. Proxy voi esimerkiksi

muuttaa PostScript-dokumentin ASCII-tiedostoksi säilyttäen datan tekstillisen sisällön, mutta jättämällä formaatit ja kuvat pois.

- Välimuisti (*caching*): Välimuistina toimiva proxy lisää asynkronisuutta tietovirtaan. ”Tarpeen niin vaatiessa” -perusteella toimivaa välimuisti-proxyä voidaan käyttää minimoimaan huonoa yhteyttä pitkin lähetettävän datan määrää. Etukäteen muistiin noutava proxy taasen hyödyntää hyvää yhteyttä ja noutaa dataa muistiin sen tarpeellisuuden ennakoiden.



Kuva 2. Proxy-arkkitehtuuri [GENERA99].

4.2. Sovelluksen sopeutumismahdollisuudet

Yksinkertaisin adaptointitekniikka, jonka sovellus voi tuottaa, on mukauttaa kommunikointivaatimuksensa yhteyden tasoa vastaavaksi. Esimerkiksi multimediatietokannassa sovellus voi valita vastaanottavansa tekstikuvauksen ennemmin kuin huonolaatuisen kuvan, sillä vaikkapa röntgenkuvia siirrettäessä kuvan laadun heikkeneminen voi olla kohtalokkaan harhaanjohtavaa. Toinen esimerkki adaptoitumisesta olisi tilanne, jossa sovellus noutaa tietokannasta useita tietueita selailua ja valintaa varten verkkoyhteyden äkillisesti heiketessä. Tällöin adaptiivinen sovellus siirtää tietueiden poimimisen proxypalvelimelle tai agentille ennen niiden siirtämistä heikkoa yhteyttä pitkin sovellukselle. Kolmanneksi esimerkiksi adaptiivisuudesta valitaan hieman kehittyneempi sovellus, audio-video konferenssi-sovellus, joka pystyy sopeutumaan vähentämällä QoS-vaatimuksiaan parhaillaan siirrettävää mediaa vastaavaksi tai esimerkiksi luopumalla videokuvan siirrosta audioyhteyden hyväksi.

Tiedonsiirtotavan muuttamisen lisäksi sovellus voi uudelleen rakentautumalla pyrkiä sopeutumaan muutoksiin yhteysrakenteissa. Esimerkiksi ryhmäsovellus, joka vaatii ryhmän tilan ylläpitoa, voi monilähetystä (*multicasting*) tukevassa paikallisverkossa hajauttaa ryhmän tilatiedon ylläpitämisen saavutettavuuden ja vikasietoisuuden parantamiseksi. Tilanteessa, jossa osa ryhmän jäsenistä siirtyy langattoman yhteyden päähän, joka ei tue ryhmäkutsuja ja monilähetystä, on sovelluksen helpompi hallita ryhmän tilatietoa, jos hallinta keskitetään tiettyyn pisteeseen. Tällöin sovelluksen voi lisäksi olla mahdollista vähentää yhteydenpitoa mobiileihin ryhmän jäseniin, jotta parempien yhteyksien päässä olevien jäsenten vasteajat eivät kärsisi [MOST99].

4.3. Adaptoitumisen ajoitus

Dynaaminen sopeutuminen tarjoaa sekä haasteita että mahdollisuuksia systeemisuunnittelijoille. Yksi vaihtoehto sopeutumisen ajoittamiselle on sovelluksen lataamis- tai aloitusvaihe. Tämä on helpoin vaihtoehto, sillä tässä vaiheessa sovellus ei ole vielä asettunut minkäänlaiseen tilaan, mutta mikäli olosuhteet muuttuvat suorituksen aikana, ei sovellus pysty sopeutumaan muutoksiin. Esimerkki tästä olisi sovellus, joka kykenisi valitsemaan mitä solmuja se käyttää, eli missä ympäristössä se toimii. WWW-selain voi valita usean samansisältöisen palvelimen tai proxy-palvelimen väliltä. Vaihtoehtoinen tapa on päästää sovellus adaptoitumaan sekä käynnistymisen yhteydessä että suorittumisen aikana. Ajonaikainen sopeutuminen on monimutkaisempi toteuttaa, sillä tällöin sovelluksen täytyy kyetä muokkaamaan itseään uudelleen [HETERO99].

Ajonaikaista sopeutumista voi tapahtua joko periodisesti, jolloin järjestelmä tasapainottaa kuorimituksen uudelleen aina t aikayksikön välein tai vaihtoehtoisesti periaatteilla tarpeen niin vaatiessa tai sopivan tilaisuuden tullen. Keskimmäisessä tapauksessa järjestelmä pyrkii sopeutumaan aina kun jonkin parametrin arvo putoaa tietyn kynnyksen alapuolelle ja jälkimmäisessä tapauksessa järjestelmä yrittää aina opportunistisesti käyttää tilaisuuden hyödykseen kun resursseja vapautuu.

Ajonaikainen sopeutuminen lisää huomattavasti ohjelmoinnin ja sopeutumisprosessin monimutkaisuutta, mutta se on välttämätöntä, jotta saavutettaisiin kelvollinen suorituskyky ajettaessa pitkäkestoisia sovelluksia vaihtelevissa olosuhteissa.

4.4. Verkkoympäristön tila

Sopeutuakseen vallitsevaan verkkoympäristön tilaan tarvitsee sovellus tietoa siitä. Yhteysprotokollat tuottavat tietoa verkon tilasta ja jakavat tietouttaan myös sovelluksille. Implisiittiseen palautteeseen perustuvat protokollat tarkkailevat sisääntulevaa tietovirtaa ja tekevät sen perusteella päätelmiä verkko-olosuhteista. TCP on hyvä esimerkki tällaisesta protokollasta. TCP tulkitsee hylätyt paketit ruuhkan merkiksi verkossa, jolloin lähettäjä voi pienentää pakettien lähetystahtia. Eksplisiittisen palautteen ollessa kyseessä jokin verkon sisäinen itsenäinen osio tuottaa tietoa verkon olosuhteista suoraan lähettäjille. Hyvä esimerkki on ATM:n (*Asynchronous Transfer Mode*) ABR –laatuluokka (*Available Bit Rate –class*), jossa lähettäjät saavat säännöllisin väliajoin tietoa verkon ruuhkatilanteesta tai niille voidaan jopa ilmoittaa suurin mahdollinen nopeus millä lähettäminen sallitaan.

Implisiittinen palautteen tuottaminen on aina toteutettavissa, sillä se ei vaadi verkkotukea, mutta sillä on myös puutteensa. Implisiittinen palaute kertoo ainoastaan muutoksen lisäyksen, eli esimerkiksi kahden palvelimen

kommunikoidessa, välittyy tieto siitä, miten niiden välinen kaistanleveys kehittyy. Lisäksi implisiittistä tietoa saattaa olla vaikeaa tulkita. Pakettien häviäminen kielii selvästi ruuhkaisesta verkosta, mutta se ei kerro pitäisikö sovelluksen keskeyttää lähetys hetkeksi, lähettää paketteja uudelleen tai toimia jotenkin toisin.

Implisiittinen informaatio perustuu kokemukseen, joten sovellus pystyy keräämään tietoa verkon tilasta ja palvelujen laadusta ainoastaan sellaisten verkon polkujen varrelta, joilla se operoi. Lisäksi palautetta kertyy ainoastaan silloin kun sovellus aktiivisesti käyttää verkkoa, eli sovelluksen käynnistyessä tai ollessa joutilaana ei tietoa ole saatavilla. Selviytyäkseen näistä rajoitteista sovellus voisi säännöllisin väliajoin kokeilla kaikkia eri palveluluokkia selvittääkseen verkon suorituskyvyn käytettävissä olevien solmujen välillä. Joissain tapauksissa edellä mainitun kaltainen ratkaisu voi olla sopiva, mutta käytännössä tehokkaiden verkontestausrutiinien toteuttaminen on vaikeaa. Lisäksi ylenmääräinen kokeileminen voi olla kallista, sillä se kuluttaa sekä sovelluksen aikaa että verkon resursseja.

Eksplisiittinen palaute on helppokäyttöistä sovelluksille, mutta sen tuottaminen vaatii tukea verkkokerrokselta. Eksplisiittistä palautetta voidaan tuottaa kahdella tavalla. Verkko voi tuottaa palautetta jatkuvasti, kuten esimerkiksi edellä mainitussa ABR-liikenteessä, jossa erillistä nopeudenhallinnointiin käytettävää solua vaihdetaan verkon kanssa 32 solun välein. Vaihtoehtoisesti sovellukselle voidaan antaa ilmoitus aina kun tapahtuu jotakin erityistä, kuten verkon kaistanleveys laskee alle tietyn arvon tai yhteys vaihtuu verkkotyypistä toiseen.

Paljon laskentaa vaativat hajautetut tieteelliset simulaatiot pitävät yleensä yllä tietoa verkon tilasta joko sisäisesti mittaamalla jonkin työkalun avulla verkon tilaa tai ulkoisesti mittaamalla eri solmuissa ja verkon eri osissa suorittuvien töiden edistymistä. Simulaatiot myös usein tasoittavat kuormitusta siirtämällä laskentaa pois solmuilta jotka edistyvät muita hitaammin. Yleensäkin adaptoitumismalli, jossa sovellus tarkkailee omaa suorituskykyään ja sopeutuu havaitessaan muutoksia on erittäin käyttökelpoinen [HETERO99].

Yleensä sopeutumista ei voida tehdä ilman ulkoisia mittauksia ympäristön tilasta. Solmujen valinta suoritusta aloitettaessa täytyy perustaa ulkoisille mittauksille, sillä ilman ei voida tietää mitkä solmuista ovat käytettävissä. Dynaaminen vaihto solmujoukosta toiseen, kuten myös solmujen lukumäärän kasvattaminen sovelluksen suorittumisen aikana, vaatii ulkoista tietoa solmuista, sillä sovelluksen sisäinen informaatio rajoittuu niiden solmujen tilaan, joita sovellus parhaillaan käyttää.

Heterogeeniset ympäristöt ovat haasteellisia systeemisuunnittelijoille, joten on tärkeää, että verkkoyhteydestä tietoa tuottava järjestelmä on yksinkertainen ja mobiili. Yhtenäinen runko, sekä erilaisten verkkoarkkitehtuurien resurssien monitorointijärjestelmä, ovat välttämätön aines adaptiivisten sovellusten kehittämiselle [HETERO99].

5. ESIMERKKINÄ MOST-PROJEKTI

MOST = Mobile Open Systems Technology for the Utilities Industry

Adaptiivisuutta sovellus-, sovellusalusta- tai käyttöjärjestelmätasolla tukevia järjestelmiä on kehitetty useita, mutta sellaisia järjestelmiä, jotka tukisivat adaptiivisuutta kaikilla näillä tasoilla on vain muutamia. Lancasterin Yliopisto ja EA Technology kehittivät yhteistyössä langattoman sovelluksen tukemaan sähköntuotantoteollisuuden kenttätyöntekijöiden toimintaa. Tässä luvussa kuvaillaan MOST-projektin kehitystyön tuloksena syntynyttä langatonta yhteistyösovellusta, joka tukee adaptiivisuutta niin käyttäjä-, sovellus-, sovellusalusta- kuin käyttöjärjestelmätasolla.

Kenttätyöntekijät ovat vastuussa sähkönjakeluverkon luomisesta ja ylläpidosta, joten heidän työnsä koostuu enimmäkseen rutiininomaisista rakennus- ja ylläpitotehtävistä. Äkilliset häiriöt sähkönjakeluverkossa, kuten täydelliset sähkökatkokset, voivat aiheuttaa huomattavia vahinkoja esimerkiksi sairaaloiden tai raskaan teollisuuden toiminnalle. Virhetilanteen sattuessa jakeluverkon toiminnan palauttaminen nopeasti ja turvallisesti vaatii paljon resursseja ja koordinoitua yhteistyötä vian ja paikan määrittämiseen ja eristämiseen sekä virheen korjaamiseen.

MOST-projektissa otettiin erityisesti huomioon seuraavat järjestelmälle asetettavat vaatimukset:

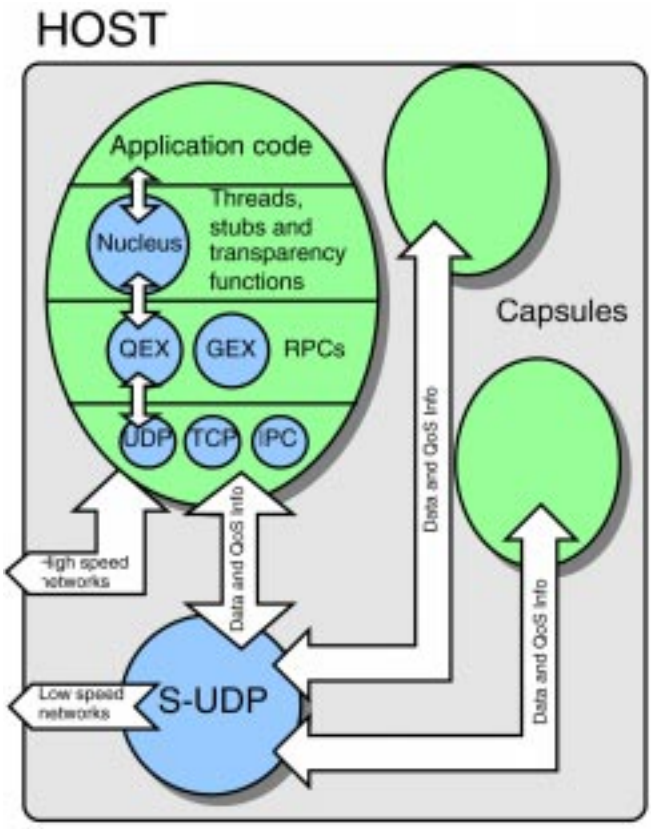
- Liikkuvuus: Kenttäinsinöörit liikkuvat paikasta toiseen ja tarvitsevat yhteyden sekä kiinteisiin että langattomiin kohteisiin.
- Paikkatieto: Kenttäinsinöörit käsittelevät paljon kaavioita ja muuta graafista tietoa, kuten karttoja, konepiirustuksia ja yhteysdiagrammeja.
- Multimedia: Kartat ja niiden viitetiedot sisältävät sekä rasteri– että vektorimuotoista dataa ja mahdollisesti myös käsin tehtyjä merkintöjä, joten on tarpeellista ylläpitää myös audio-yhteyttä insinöörien välillä.
- Yhteiskäyttöisyys: Järjestelmän tulee mahdollistaa fyysisesti eri paikoissa sijaitsevien insinöörien kommunikointi puheen avulla sekä yhteisten jaettujen karttojen, kaavioiden ja osoitustyökalujen käyttö.
- Yhteistyö (ulkopuolisten tahojen kanssa): Järjestelmältä vaaditaan yhteistyökykyisyyttä muiden (julkisten) laitosten tietojärjestelmien kanssa.

Sovellus kehitettiin toimimaan testiympäristössä, joka koostuu ethernet-verkolla yhdistetyistä työasemista sekä kannettavista koneista modeemiyhteydellä, GSM-verkon välityksellä.

5.1. Systeemiarkkitehtuuri

Sovellus koostuu kolmesta komponentista: sovelluskoodista, tukea antavasta hajautetun järjestelmän sovellusalustasta ja modeemiyhteyden (*dial-up*)

verkkoajurista (*S-UDP, Serial- User Datagramm Protocol*). Kuvassa 2 on esitetty järjestelmän yleisarkkitehtuuri.



Kuva 2. MOSTin systeemiarkkitehtuuri [MOST99].

Järjestelmäarkkitehtuuri perustuu ANSAware–nimiseen hajautettuun järjestelmäalustaan, jolla on ollut voimakas vaikutus ISO/ITU-T RM-ODP –standardin (*Reference Model for Open Distributed Processing*) kehittämisessä. ANSAware- järjestelmäalusta tukee suoraan operatiivisuutta muiden järjestelmien kanssa. MOST-projektin järjestelmäalustaan on lisäksi liitetty adaptiivisuutta tukevia piirteitä. Järjestelmä koostuu komponenteista, eli objekteista. Jokainen objekti käsittää koteloidun sisäisen tilan ja yhden tai useamman rajapinnan, jotka määrittelevät miten muut objektit voivat kommunikoida ko. objektin kanssa. Rajapinta on joukko nimettyjä operaatioita, joita voidaan kutsua joko synkronisesti tai asynkronisesti. Toiminnalliset rajapinnat ylläpidetään välityspalvelun avulla. Sovelluksen alimmalla tasolla on S-UDP -ajuri, joka toimii lähes vastaavasti kuin SLIP (*Serial Line Internet Protocol*) tai PPP (*Point-to-Point Protocol*), eli se paketoi ja lähettää tiedon eteenpäin sarjaliikenneyhteyttä tai modeemilinjaa pitkin. S-UPD eroaa perinteisistä SLIPistä ja PPPstä siten, että sen QoS-rajapinta tarjoaa sovelluksille tietoa linjojen tilasta, yhteydettömistä palvelimista ja odottavien pakettien määrästä ja tilasta.

Järjestelmäalustan objektiorientoitunutta mallia mukaillaan myös itse sovelluksen toteutuksessa. Sovellus on itse asiassa kokoelma moduleja, joista jokainen vastaa yhden elementin toiminnallisuudesta sovelluksessa. Päämodulit ja niiden toiminnat ovat seuraavat:

- GIS–moduli (*Geographic Information System*): Tämän modulin avulla insinöörit voivat katsella ja tehdä merkintöjä karttoihin ja kaavioihin. Moduli tukee myös tosiaikaista synkronista ryhmätyöskentelyä konferenssin hallinta –modulin avulla.
- Konferenssin hallinta –moduli: Tämä moduli ylläpitää käyttäjien välisiä neuvotteluja, jotka sisältävät audio-yhteyksiä ja yhteiskäyttöisiä sovelluksia, kuten GIS-moduli.
- Hajautetun tietokannan hallinta –moduli: Tämän modulin avulla päästään keskustietokannassa sijaitsevaan asiakasrekisteriin.
- Rakenteinen sähköposti: Tämä moduli tukee rakenteisia sähköposteja, joissa voidaan lähettää esimerkiksi GIS-modulilla tuotettuja karttoja.

Jokainen moduli on toteutettu yhdellä tai useammalla sovellusalustan komponentilla. Konferenssin hallinta –moduli vastaa kaikista ryhmäaktiviteettien koordinnoinneista ja tarjoaa muille sovelluksen moduleille rajapinnan ryhmän koostumuksen tarkasteluun.

Järjestelmään on lisätty monia QoS-piirteitä, joiden avulla sovelluksen modulien on mahdollista tarkkailla ja kontrolloida vuorovaikutustaan muiden ohjelman komponenttien kanssa. Sovellusalusta ja sovellukset voivat sopeutua muutoksiin alemmissa kerroksissa läpi koko arkkitehtuurin kulkevan QoS-informaation avulla.

5.2. Sopeutuminen järjestelmätasolla

Kuten jo aikaisemminkin on todettu, voivat sovellukset käyttää resursseja tehokkaammin hyväkseen, jos niiden on mahdollista reagoida allaolevissa kerroksissa tapahtuviin muutoksiin. MOST-prototyypissä adaptoitumista on helpotettu ylläpitämällä QoS–arkkitehtuuria, johon järjestelmän eri tasot voivat rekisteröidä QoS–vaatimuksensa ja jolta ne myös saavat ilmoituksen palvelun laadun muuttuessa.

QoS–arkkitehtuuri tukee seuraavia parametreja:

- Suoritusteho
- Etenemisviive
- Joutoaika
- Tavoitettavuus

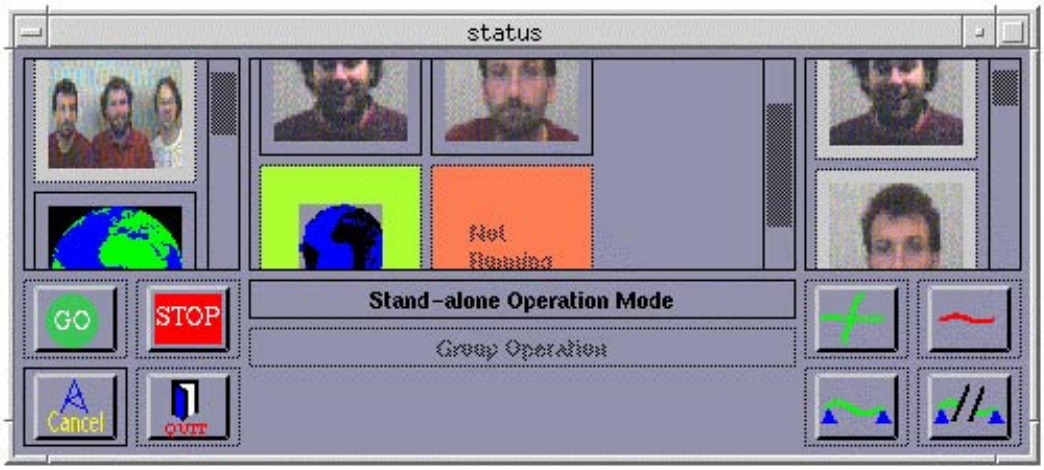
Järjestelmän RPC–mekanismi (*Remote Procedure Call*), nimeltään QEX (*Quality-of-Service driven remote EXecution*), kerää statistiikkaa kahta ensimmäistä parametria varten. Protokolla käyttää statistiikaa uudelleenlähettämisen ajastamiseen ruuhkan välttämiseksi ja kontrolloimiseksi sekä tuottaa informaatiota

sovelluksia varten. Sovelluksen on mahdollista määritellä sallitut poikkeamat yllä mainituille yhteyden parametreille ja se voi pyytää tulla informoiduksi, jos parametrien raja-arvot rikkoutuvat. Sovellus voi myös neuvotella uudet rajat poikkeamille saatuaan ilmoituksen rajojen rikkoutumisesta. Mobiileille sovelluksille tämä piirre on erityisen hyödyllinen, sillä ne voivat päätellä onko viestien lähettäminen ja vastaanottaminen jotakin tiettyä yhteyttä pitkin mahdollista lähettämättä erityisiä testiviestejä. Joutoaika ja tavoitettavuusparametrit ovat arvokkaita ympäristöissä, joissa yhteydet ovat heikkoja. Perinteinen RPC tarjoaa yleensä mekanismit poikkeukselliseen yhteyden katkaisemiseen jos palvelimeen ei saada yhteyttä. Tilanteissa, joissa sovellus hyödyntää takaisinkutsuja tai periodisia päivityksiä, on usein myös palvelimen tarpeellista pystyä päättelemään onko clientti ollut kykenevä ottamaan siihen yhteyttä. Palvelin voi määritellä yhteyden joutoajalle maksimin, jonka jälkeen se lähettää takaisinkutsun, jos yhteyttä ei ole käytetty. Tavoitettavuusparametrin avulla voidaan vahvistaa, ettei joutoaikaparametrin ylittyminen johdu esimerkiksi yhteyden katkeamisesta.

Järjestelmän tuottama QoS-parametrien tarkkailu-, palaute- ja kontrollointimekanismi mahdollistaa sovellusten sopeutumisen. Nykyisessä MOST-prototyypissä ohjelmamodulit voivat luoda täsmällisiä yhteyksiä muissa palvelimissa oleviin kolleegoihinsa ja tarkkailla palvelimien välillä vallitsevia olosuhteita. Esimerkiksi hajautetun tietokannan hallinta –moduuli käyttää QoS-informaatiota sovittaakseen lähettämänsä datan volyymin saatavilla olevaan siirtokapasiteettiin.

5.3. Käyttäjän sopeutuminen

Useimmiten käyttäjä on loppujen lopuksi vastuussa verkkoliikenteen syntymisestä, joten käyttäjän työskentelytapojen ja valintojen sopeuttamisella käytettäviä sovelluksia ja verkkoyhteyksiä parhaiten hyödyntäväksi on suuri vaikutus koko järjestelmän toiminnan tehokkuuteen. Kuvassa 3, seuraavalla sivulla, on esitetty MOST-sovelluksen ryhmänjohtajan käyttöliittymä, joka kuvastaa ryhmätyöskentelyn tilaa. Liittymän avulla voidaan lisätä ja poistaa jäseniä ryhmästä ja kontrolloida sovelluksen moduuleja. Kuva esittää ryhmätyötilannetta kolmen osapuolen kesken. Jokaisen käyttäjän alla on rivistö pienempiä ikoneja, jotka kuvaavat käyttäjän käytössä olevia ohjelmakomponentteja. Ikonien ilmentymän perusteella voidaan päätellä mitkä komponenteista ovat ryhmätyötilassa ja lisäksi mitkä ryhmän jäsenistä ovat yhteydessä kyseiseen komponenttin.



Kuva 3. MOST-sovelluksen käyttöliittymä [MOST99].

Ikonien perusteella pystytään myös päättelemään käyttäjän yhteysrakenteen laatu. Jokainen aktiivinen moduli ottaa eksplisiittisen yhteyden hajautetun järjestelmän muissa koneissa sijaitseviin moduleihin. Käytössä olevan yhteyden tehokkuutta kuvaavien QoS-parametrien rekisteröimisen hoitaa erityinen yhteyden kontrollointi –alusta (*binding control interface*). Näiden QoS-parametrien arvojen avulla päivitetään ikonien taustaväri, joka kuvaa käyttäjän (verkko)yhteyden voimakkuutta. Jos ryhmän tehokkuus laskee normaalin työskentelyn aikana, voidaan värien perusteella päätellä kuka käyttäjistä on järjestelmän pullonkaulana. Ryhmän työskentelyn tehokkuuden lisäämiseksi tämä käyttäjä voidaan tilapäisesti jäävätä ryhmän operaatioista. Tilanteessa, jossa synkroninen yhteydenpito käy mahdottomaksi, voidaan sovelluksen rakenteista sähköpostia käyttää kiireettömän tiedon levittämiseen.

6. MITÄ TÄSTÄ OPIMME?

Edellä on tarkasteltu mitä sovelluksen adaptiivisuus oikein tarkoittaa ja mitä kaikkea adaptiivisuuden tukeminen sovellustasolla vaatii sovellukselta ja sitä ympäröivältä järjestelmältä yleensä. Dokumentissa kuvailtuja erilaisia vaihtoehtoja sovelluksen sopeutumiseksi on tutkittu lukemattomissa projekteissa, joiden tuloksena on syntynyt monen nimisiä järjestelmiä adaptiivisuuden toteuttamiseksi. Julkaistun kirjallisuuden perusteella projektit voisi ryhmitellä kolmeen kategoriaan sen mukaan toteutetaanko adaptiivisuutta järjestelmä-, sovellusalusta- vai sovellustasolla. Lähtökohta lähes kaikissa julkaisuissa oli sama: Nykyaikaisessa verkkoympäristössä resurssien saatavuus vaikuttaa sovelluksen suorituskykyyn, jolloin sovelluksen on välttämätöntä kyetä sopeutumaan ympäristön muutoksiin, jotta se suorittuisi tehokkaasti järkevässä vasteajassa. Loppujen lopuksi lähes poikkeuksetta projektien yhteenvedoissakin tultiin samaan tulokseen: Tehokkaasti toimiva adaptiivinen järjestelmä on kokonaisuus, jossa käyttöjärjestelmä tarjoaa mekanismit adaptiivisuuteen, esimerkiksi tiedon vallitsevista QoS–parametrien arvoista, ja sovellusalusta sekä sovellukset varustetaan näitä mekanismeja hyväksi käyttävillä adaptiivisilla menettelytavoilla.

LÄHTEET

[GENERA99] Experiences of Using Generative Communications to Support Adaptive Mobile Applications; Distributed Multimedia Research Group, Lancaster University, 1999.
http://www.comp.lancs.ac.uk/computing/research/mpg/reports_1999.html

[HETERO99] Adaptive Distributed Applications on Heterogeneous Networks, Th. Gross, P. Steenkiste, and J. Subhlok. In Proc. 8th Heterogeneous Computing Workshop (HCW'99)
http://www.cs.cmu.edu/afs/cs.cmu.edu/project/cmcl/archive/Remulac-papers/hcw99.ps

[MOST99] Developing Adaptive Applications: The MOST Experience; A. Friday, N. Davies, G.S. Blair and K. W. J. Cheverst
http://www.comp.lancs.ac.uk/computing/research/mpg/reports_1999.html

[MOWGLI96]Enhanced Services for World-Wide Web in Mobile WAN Environment; Liljeberg, M., Helin, H., Kojo, M., Raatikainen, K. University of Helsinki, Department of Computer Science, April 1996.

[PROTO99] On Providing Support for Protocol Adaptation in Mobile Wireless Networks; Pradeep Sudame and B. R. Badrinath.
ftp://www.cs.rutgers.edu/pub/technical-reports/dcs-tr-333.ps.Z

[STEMM99] An Network Measurement Architecture for Adaptive Applications; Mark Richard Stemm
http://www.cs.berkeley.edu/~stemm/publications/thesis/thesis.html

[STREAM99] A Survey of Application Layer Techniques for Adaptive Streaming of Multimedia; Bobby Vandalore, Wu-chi Feng, Raj Jain and Sonia Fahmy. Submitted to the Journal of Real Time Systems (Special Issue on Adaptive Multimedia), April 1999.
http://www.cis.ohio-state.edu/~jain/papers/amjrts99.htm

Kaikkiin lähteisiin on viitattu 6.11.2000.