Henkilökohtainen mobiilikalenteri

Arto Vahvanen

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Pro gradu -tutkielma
Lokakuu 2002

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tekijän Nimi: Arto Vahvanen
Pro gradu -tutkielma, 66 sivua
Lokakuu 2002

Perinteisiä paperikalentereita on käytetty ajankäytön organisoimiseen jo kymmeniä vuosia. Varsinkin toimistotyössä ne ovat olleet yksi tärkeimpiä työvälineitä tapaamisten, kokousten ja muiden menojen suunnittelussa. Paperikalenterien rinnalle on tullut myös elektronisia kalentereita. Tämän päivän langattomat tekniikat ja mobiilien laitteiden ominaisuudet mahdollistavat dynaamisen tavan saada elektronisiin kalentereihin uudenlaista tietoa aivan uudella tavalla. SyncML-protokollan avulla palvelut voidaan lisäksi kehittää yhteensopiviksi eri valmistajien laitteille.

Tutkielmassa suunnitellaan mobiilipalvelu, jonka avulla voidaan synkronoida personoituja kalenterimerkintöjä käyttäjien päätelaitteisiin. Tutkielmassa esitellään mobiilipalveluun liittyviä suunnitteluongelmia ja etsitään ratkaisuja näihin ongelmiin. Ratkaisujen perusteella kuvataan palvelulle yleinen arkkitehtuuri sekä toimintamalli. Lisäksi tutkielmassa esitellään SyncML-protokollan käyttö ja siihen liittyviä ominaisuuksia sekä vaihtoehtoja palvelun mukauttamiselle eri ympäristöihin.

Lopullisesta mallista voidaan todeta, että tehokas personointi on mahdollista toteuttaa käyttämällä oppivia agentteja, jotka seuraavat käyttäjien reaktioita synkronoituihin kalenterimerkintöihin. SyncML-protokolla mahdollistaa laitteen ominaisuuksien huomioon ottamisen personoinnissa.

Avainsanat ja -sanonnat: SyncML, mobiilipalvelu, kalenteri, synkronointi, personointi.

Sisällys

1. Johdanto 1
2. Määritelmiä 4
   2.1. Agentti 4
   2.2. Mobiilit päätelaitteet 4
   2.3. XML 4
   2.4. Synkronointi 5
   2.5. vCalendar 6
   2.6. Push-teknologia 8
3. SyncML 10
   3.1. SyncML:n runko 10
   3.2. Muutosrekisteri ja ID-linkitys 13
   3.3. Sync-ankkurit ja -tyypit 14
   3.4. SyncML-paketit ja -viestit 17
   3.5. SyncML-protokollan edut ja ongelmat 23
4. Palvelun kuvaus 25
   4.1. Palvelun yleiskuvaus 25
   4.2. Palvelun tekniset rajoitteet 27
   4.3. Suunnitteluongelmat 27
   4.4. Arkkitehtuuri 31
5. Personointi 34
   5.1. Personoinnin merkitys 34
   5.2. Palvelun mukauttaminen eri laitteille 35
   5.3. Tietopohjainen tiedonsuodatus 36
   5.4. Oppiva tiedonsuodatus 37
6. Toteutus 42
   6.1. Agenttien sijainti 42
   6.2. Tietosisällön kuvaus 43
   6.3. Tietoturva 47
7. Palvelun toiminta SyncML:n avulla 50
   7.1. Esivaihe 50
   7.2. Synkronoinnin alustus 52
   7.3. Synkronointi ja linkitys 53
   7.4. Vaihtoehtoiset toteutukset 56
   7.5. Lopullinen arkkitehtuuri 56
8. Yhteenveto 59
   8.1. Palvelun arviointi ja rajoitukset 59

8.2. Jatkotutkimusaiheet 61

Viiteluettelo 63

1. Johdanto

Perinteisiä paperikalentereita on käytetty ajankäytön organisoimiseen jo kymmeniä vuosia. Varsinkin toimistotyössä ne ovat olleet yksi tärkeimpiä työvälineitä tapaamisten, kokousten ja muiden menojen suunnittelussa. Erilaisten toimisto-ohjelmistojen yleistyttyä, paperikalenterien rinnalle on tullut myös elektronisia kalentereita. Ne tarjoavat paperikalentereita tehokkaampia ominaisuuksia päivien suunnitellussa ja useampien ihmisten keskinäisten aikataulujen hallinnassa.

Erilaisten PDA-laitteiden (Personal Digital Assistant) ja matkapuhelimien yleistyminen ja niiden toiminnallisuuksien parantuminen on tuonut elektroniset kalenterit myös mobiiliin maailmaan. Elektronisissa kalentereissa on kuitenkin vielä puutteita, jotka saavat ihmiset kantamaan vanhoja paperikalentereita mukanaan. Ihmiset ovat tottuneet siihen, että kalenterista löytyvät mm. nimipäivät, juhlapyhät ja liputuspäivät, joita elektronisista versioista ei yleensä löydy.

Tämän päivän langattomat tekniikat, mobiilit laitteet ja niiden elektroniset kalenterit tarjoavat kuitenkin tapoja, joilla edellä mainitut paperisten kalenterien perusominaisuudet on mahdollista saada jokaiseen kannettavaan laitteeseen. Lisäksi uudet tekniikat mahdollistavat dynaamisen tavan saada kalentereihin uudenlaista tietoa aivan uudella tavalla. Useimmilla puhelinvalmistajilla on omia sovelluksia, joilla puhelimen tiedot voidaan synkronoida PC:llä olevien tietojen kanssa. Ongelmana on, että käytetyt synkronointiprotokollat ovat valmistaja- tai jopa mallikohtaisia sekä usein tukevat vain lyhyenmatkan langatonta tai sarjakaapeliperustaista tiedonsiirtoa. Tämä on rajoittanut sovelluskehittäjien ja palveluntarjoajien mahdollisuuksia sekä lopulta vähentänyt loppukäyttäjien vaihtoehtoja. Ongelman ovat huomioineet nyt myös alan suurimmat laitevalmistajat (mm. Ericsson, Motorola ja Nokia) ja näiden aloitteesta kehitteillä olevan SyncML-protokollan (Synchronization MarkUp Language) tarkoituksena on yhtenäistää eri laitteiden ja laitevalmistajien synkronointiprotokollat. Myös Symbian, yksi suurimmista PDA-laitteiden käyttöjärjestelmien valmistajista, on liittänyt SyncML-protokollan tuen omaan käyttöjärjestelmäänsä. Symbian OS-käyttöjärjestelmä tarjoaa täysin SyncML-yhteensopivan asiakastoteutuksen. Langattomien laitteiden uusien ominaisuuksien ja SyncML-protokollan avulla sovelluskehittäjien on mahdollista kehittää palveluja useille eri valmistajien laitteille ja lisätä loppukäyttäjien palveluiden määrää.

Kiurun [2000] mukaan ihmiset liittävät esimerkiksi seuraavia ominaisuuksia matkapuhelimien kestotilauspalveluihin:

- tuore ja päivittynyt informaatio sekä
- pitkälle personoitu informaatio.

Tämän tutkielman tavoitteena on vastata edellä mainittuihin haasteisiin ja uusia teknologioita hyväksikäyttäen suunnitella mobiilipalvelu, joka mahdollistaa erilaisten tietojen dynaamisen synkronoinnin eri langattomien päätelaitteiden kalentereihin. Tavoite on, että tutkielma toimisi runkona palvelun lopulliselle ja yksityiskohtaiselle suunnittelulle ja helpottaisi näin palvelun toteuttamista. Samalla on myös tarkoituksena miettiä, mitä ongelmia palvelun suunnittelussa ja toteutuksessa on, sekä pyrkiä löytämään näihin ongelmiin ratkaisuja. Palvelun kehittämisessä keskitytään erityisesti palvelun personointiin sekä SyncML:n käyttöön sovelluksessa. Lisäksi tutkielmassa mietitään mobiilipalveluiden perusongelmia sekä suunnitellaan palvelua arkkitehtuuritasolla eteenpäin. Useita pienempiä palveluiden kehittämiseen liittyviä tutkimusongelmia ei tutkielmassa käsitellä, vaan ne rajataan tutkielman ulkopuolelle. Vaikkakaan tutkielmaan ei resurssien puutteessa kuulu myöskään palvelun implementointi, tarkoituksena on suunnitella ja kuvata palvelu niin yksityiskohtaisesti, että varsinaisen toteutuksen tarkempi suunnittelu on mahdollista tämän raportin pohjalta.

Tutkielman perusrakenne seuraa Järvisen ja Järvisen esittelemää konstruktiivisen tutkielman rakennetta [Järvinen ja Järvinen, 2000]. Järvinen ja Järvinen esittelevät mallissaan uutta realisaatiota kuvaavan raportin rakenteen. Tässä luvussa on esitelty taustaa sekä motiiveja palvelun kehittämiselle. Toisessa luvussa esittelen tutkielmalle tärkeitä määritelmiä ja käsitteitä. Koska SyncML-protokolla on perustana koko palvelun toteutukselle, sen käsittely on sijoitettu omaan lukuunsa luvuksi kolme. Lisäksi luvussa kolme selvitetään SyncML:n etuja sekä huonoja puolia. Luvussa neljä esittelen palvelun perusidean yleisellä tasolla. Tähän liittyvät palvelun kuvauksen lisäksi suunnitteluongelmat sekä palvelun yleinen arkkitehtuuri. Luku neljä toimii lähtökohtana palvelun suunnittelulle. Lisäksi luvussa neljä selvitetään palvelun tekniset rajoitteet. Luvussa viisi keskitytään tutkielman tärkeimpään suunnitteluongelmaan, palvelun personointiin. Tässä luvussa selvitetään personoinnin merkitys suunnitellulle palvelulle sekä eri vaihtoehtoja sen toteuttamiseen. Luvussa kuusi tutkitaan pienempien suunnitteluongelmien eri

ratkaisuvaihtoehtoja sekä kuvataan vaihtoehtojen valinta perusteluineen. Luvussa seitsemän kuvataan palvelun toimintaa SyncML-protokollaa hyödyntäen. Lisäksi luvussa kuvataan myös, miten palvelun lopullinen arkkitehtuuri muodostuu. Koska palvelun implementointi ei kuulu tutkielmaan, tässä luvussa palvelun toiminta kuvataan niin yksityiskohtaisesti, että lukijalle jää varmuus, että se on toteutettavissa [Järvinen ja Järvinen, 2000]. Seuraavaksi luvussa kahdeksan arvioidaan palvelua tavoitteiden ja vaatimusten osalta sekä esitellään mahdollisia jatkotutkimusaiheita.

2. Määritelmiä

2.1. Agentti

Agentit ovat joko itsenäisiä ohjelmia tai irrallisia prosesseja, jotka suorittavat niille annettua tehtävää [Nwana, 1996]. Maesin [1995] määrityksen mukaan agentit aistivat ja toimivat monimutkaisessa ja dynaamisessa ympäristössään ja ymmärtävät niille asetettujen tehtävien tavoitteet. Tässä palvelussa agentit suorittavat itsenäisesti tiedon suodatusta.

2.2. Mobiilit päätelaitteet

Mobiilit päätelaitteet voidaan jakaa kolmeen ryhmään: kommunikaattori-tyyppiset laitteet, normaalit matkapuhelimet sekä PDA-laitteet. Neljäs ryhmä voisi periaatteessa olla kannettavat tietokoneet, mutta tutkielman kannalta olennaisimmat ryhmät ovat kolme ensin mainittua. Suunniteltavan palvelun kannalta kolme ensimmäistä ryhmää vastaavat toisiaan samankaltaisilla ominaisuuksillaan. Päätelaitteista puhuessani tarkoitan juuri näitä kolmea ensimmäistä ryhmää.

2.3. XML

XML (Extensible markup language) [XML] on kuvauskieli, jolla voidaan kuvata kaikkea rakenteista (structured) tietoa. Rakenteinen tieto sisältää sekä tiedon sisällön että myös kuvauksen tiedosta (tiedon roolin). Esimerkiksi otsikko kuvaa tiedon roolin (esimerkiksi tämän tutkielman otsikko) ja otsikkoteksti (esimerkiksi ”Henkilökohtainen mobiilikalenteri”) kuvaa tiedon sisällön. XML-dokumentti koostuu hierarkkisesti rakenteisesta tiedosta (kuva 2.1). Käyttäjä voi itse määritellä käytettävät tietokentät sekä tiedon sisällön. Tämän vuoksi XML on joustava ja usein käytetty formaatti varsinkin web-maailmassa, jossa sillä voidaan kuvata tietoa joustavammin kuin esimerkiksi HTML-formaatilla.

```
<?xml version=”1.0”?>
<profiili>
  <profiilinro>2234</profiilinro>
  <yhteystieto>+35850112233</yhteystieto>
  <kategoriat>
     <kategoria>urheilu</kategoria>
     <kategoria>elokuvat</kategoria>
  </kategoriat>
  <dokumentit>
     <dokumentti>./merkinnat/jalkapallo-ottelu1.xml</dokumentti>
     <dokumentti>./merkinnat/jalkapallo-ottelu2.xml</dokumentti>
     <dokumentti>./merkinnat/starwars2.xml</dokumentti>
  </dokumentit>
</profiili>
```

Kuva 2.1 Profiili XML-formaatissa

Yksi XML:n eduista on sen laaja käyttö tekstipohjaisten dokumenttien kuvauskielenä. XML mahdollistaa selkeän dokumenttien luettavuuden sekä tehokkaan tiedon käsittelyn. Lisäksi XML mahdollistaa sisällön hallinnan sekä rakenteen hallinnan käyttämällä skeema- sekä DTD-tiedostoja. DTD:n heikkona puolena on sen ilmaisuvoiman riittämättömyys monimutkaisempien tietotyyppien sekä muuttujien arvojen määrittelyyn. Samoin dokumentin rakenteen määrittely DTD:llä on puutteellinen. Skeeman avulla on mahdollista rajata muuttujien arvoja monipuolisesti sekä määritellä dokumentin rakenne täydellisesti.

2.4. Synkronointi

Tiedon synkronointi yleisesti on terminä itsensäselittävä: operaatiojoukko kahden tietoalkiojoukon yhtenäistämiseksi. Tässä tutkielmassa synkronoinnilla tarkoitetaan päätelaitteiden kalenteritietojen synkronointia SyncML-protokollaa hyväksikäyttäen.

Synkronointiprotokolla on tapa, jolla synkronointi toteutetaan. Se määrittelee operaatiojoukon ja näiden suorittamisjärjestyksen, jotta tiedot saadaan siirrettyä laitteesta toiseen.

Tietoalkiolla (data item) tarkoitan yksittäistä tietoa, esimerkiksi yhtä synkronoitavaa kalenterimerkintää. Tietoalkiot tallennetaan palvelimella tietokantaan.

Tietokanta on johdonmukainen kokoelma tietoa [Elmasri ja Navathe, 2000]. Satunnaisesti lajiteltua ja valittua tietoa ei voi siis kutsua tietokannaksi. Päätelaiteella tietoalkiot on lajiteltu tietotyyppikohtaisesti omiin hakemistoihinsa, esimerkiksi kontaktit yhteen hakemistoon ja kalenterit toiseen. Usein päätelaitteen hakemistoja kutsutaan tietokannoiksi. Tätä tapaa käytetään myös tässä tutkielmassa.

2.5. vCalendar

Yhteisen synkronointiprotokollan lisäksi tarvitaan myös standardoitu esitystapa kalenterimerkinnöille, jotta eri päätelaitteet ymmärtävät niitä. vCalendar on laitteisto- ja tiedonsiirtoriippumaton tiedostoformaatti, jolla kuvataan erilaisia kalenteritietoja. Perinteiset kalenteri– ja aikataulutiedot paperikalentereista on muutettu joustavaan ja elektroniseen formaattiin vCalendar spesifikaatiossa [VCAL, 1997]. Useat toimisto-ohjelmistot (esim. Outlook 98) sekä useat langattomat päätelaitteet (Palm III, Nokia Communicator 9110/9210) tukevat vCalendar-formaattia. Lisäksi SyncML-protokolla mahdollistaa vCalendar-objektien upottamisen suoraan SyncML-viestiin.

vCalendar koostuu ominaisuuksista (properties), joille annetaan haluttuja arvoja. Tässä akohdassa esitellään tälle tutkielmalle oleelliset ominaisuudet. Kuvassa 2.2 esitetyssä esimerkissä on yksinkertainen vCalendar merkintä, jossa kuvataan tapahtuma tulevasta elokuvan ensi-illasta.

```
BEGIN:VCALENDAR
  VERSION:1.0
  BEGIN:VEVENT
     DTSTART:20020516T120000Z
     DTEND:20020516T140000Z
     SUMMARY:Elokuvan StarWars, Episode II ensi-ilta
     LOCATION:Plevna I
     CATEGORIES:Viihde
     DESCRIPTION:Elokuva StarWars, Episode II saa ensi-iltansa Plevna 1:ss.
     Varaa liput: http://www.finnkino.fi. Hinta 8e.
     PRIORITY:3
  END:VEVENT
END:VCALENDAR
```

Kuva 2.2 Kalenterimerkintä vCalendar-formaatissa [VCAL, 1996 (mukaillen)]

Jokainen vCalendar-objekti sijoitetaan BEGIN:VCALENDAR ja END:VCALENDAR esittelyiden väliin. BEGIN:VCALENDAR esitellään ensimmäisenä. Tämän jälkeen kuvataan käytettävän spesifikaation versionumero (tässä tapauksessa 1.0). BEGIN:VEVENT aloittaa yhden kalenterimerkintäkokonaisuuden. Vastaava loppumerkintä on END:VEVENT. VEVENT esittelyn sisään asetetaan varsinaisen merkinnän ominaisuudet: aloitusaika, lopetusaika, otsikko, paikka, kategoria, kuvaus sekä merkinnän prioriteetti. Aloitus– sekä lopetusaika kuvataan muodossa:

<*vuosi*><*kuukausi*><*päivä*>*T*<*tunnit*><*minuutit*><*sekuntit*>

Jos aika halutaan ilmoittaa UTC-aikana (Universal Time Coordinated), loppuun lisätään Z (kuten kuvassa 2.2). Ilman Z-merkintää aikojen oletetaan olevan paikallista aikaa. SUMMARY kuvaa merkinnän otsikon, LOCATION vastaavasti merkinnän tapahtumapaikan ja CATEGORIES merkinnän luokat (esim. Viihde, Kulttuuri, Juhla jne.). DESCRIPTION-ominaisuudella voidaan kuvata merkintää tarkemmin. PRIORITY-ominaisuudella voidaan vaikuttaa merkinnän prioriteettiin käyttäen kokonaislukua: 1 tärkein, 2 toiseksi tärkein jne. Tätä voidaan hyödyntää esimerkiksi päällekkäisten tapahtumien yhteydessä näyttämällä tärkein tapahtuma ensimmäisenä.

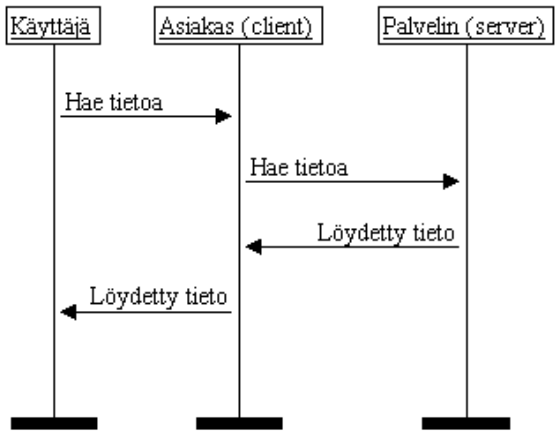
vCalendar-standardi tukee myös useita muita ominaisuuksia, joista kaikkia ei ole pakko toteuttaa. Tämä tarkoittaa sitä, että kalenteriohjelmasta riippuen ohjelmat voivat joko tukea kyseessä olevaa ominaisuutta tai olla tukematta. Tällaisia kalenterille annettavia ominaisuuksia ovat esimerkiksi:

- TRANSP (Time Transparency), jolla voi määrittää kalenterimerkinnän vievän ajanjakson vapaaksi,
- SEQUENCE (Sequence Number), jonka avulla voidaan kuvata, kuinka usein kyseessä olevaa kalenterimerkintää on muutettu,
- LAST-MODIFIED (Last Modified), joka kertoo kalenterimerkinnän viimeisen muutosajankohdan,
- ATTACH, jolla voidaan liittää joko liitteitä suoraan vCalendar-objektiin tai viitata liitteeseen URL:lla,
- ATTENDEE, jonka avulla voidaan kertoa ketä kalenterimerkintä koskee, sekä
- CLASS (Classification), jonka avulla voidaan kuvata, onko kalenterimerkintä yleinen (public), yksityinen (private) vai salainen (confidental).

Kalenteriohjelmien erilaiset toteutukset sekä erilaiset vCalendar-standardin tuet aiheuttavat ongelmia yleisen kalenteripalvelun suunnittelussa. Näihin suunnitteluongelmiin ja niiden ratkaisuihin palataan luvussa viisi.

2.6. Push-teknologia

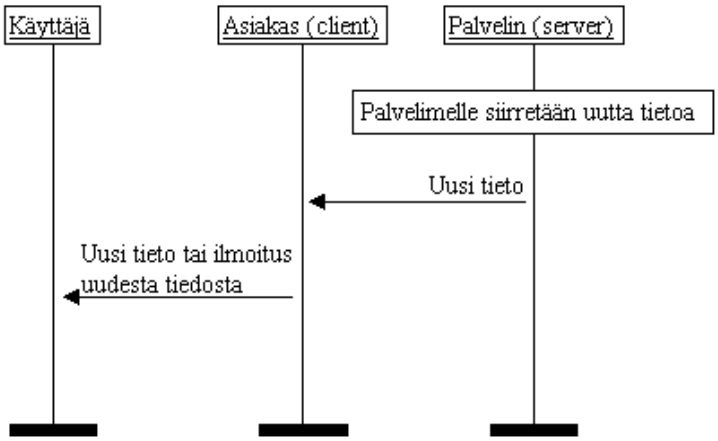
Ihmisen yleisin tapa hakea tietoa on etsiä sitä itsenäisesti esimerkiksi sanomalehdistä tai Internetistä www-sivuja selailemalla tai hakukoneita käyttämällä. Tätä tapaa kutsutaan pull-teknologiaksi[1]. Käyttäjä siis hakee itse tietoa valitusta tietolähteestä. Koko ajan kasvavasta tietomäärästä on kuitenkin vaikea löytää oikeaa ja olennaista tietoa. 90-luvun puolivälin jälkeen tätä ongelmaa ovat pyrkineet ratkomaan ns. push-teknologiat[2] (vrt. pull- ja push-teknologiat [Kendall ja Kendall, 1999]), jotka ovatkin kasvattaneet suosiotaan eri aloilla. Push-teknologia on tiedonvälitysteknologia, jossa valittua tietoa siirretään automaattisesti käyttäjälle [Käpylä et al., 1998]. Esimerkiksi matkapuhelinoperaattorit tarjoavat erilaisia push-palveluja uutisten tai urheilutulosten välitykseen käyttäjälle. Kun haluttu uutinen saapuu operaattorin tietokantaan, se siirretään käyttäjän matkapuhelimeen. Erilaisia push-palveluja on kehitetty uutispalvelimille, sähköpostiin ja myös Internetissä tapahtuvaan tiedonhakuun.



Kuva 2.3 Pull-palvelu

Kuvassa 2.3 on kuvattuna pull-palvelun periaate. Käyttäjä hakee tietoa esimerkiksi www-selaimella (asiakas) hakukonetta käyttäen. Kysely välitetään palvelimelle, joka palauttaa löydetyt sivut asiakkaan kautta käyttäjälle. Käyttäjän vastuulle jää etsiä oleelliset tiedot palautetuista dokumenteista.

[1] Eng. sana pull tarkoitaa vetää
[2] Eng. sana push tarkoittaa työntää



Kuva 2.4 Push-palvelu

Push-palvelu on esitelty kuvassa 2.4. Kun palvelimelle siirretään uutta tietoa, palvelin välittää tiedon asiakkaalle. Käyttäjälle voidaan ilmoittaa uudesta tiedosta, jolloin käyttäjä voi hakea sen itselleen sopivana ajankohtana. Toinen ja enemmän käytetty vaihtoehto on lähettää uusi tieto suoraan käyttäjälle, jolloin erillistä hakua ei tarvitse tehdä. Ensimmäistä vaihtoehtoa käytetään esimerkiksi MMS-viestien (Multimedia Messaging Service) välityksessä, jos vastaanottajalla ei ole MMS-viestiä tukevaa puhelinta. Käyttäjä saa ilmoituksen vastaanotetusta viestistä ja voi käydä lukemassa sen Internetistä. Jälkimmäinen tekniikka on käytössä useissa SMS-viestipalveluissa (Short Message Service), joissa uutiset, pörssikurssit ja muut informatiiviset viestit lähetetään suoraan käyttäjälle.

3. SyncML

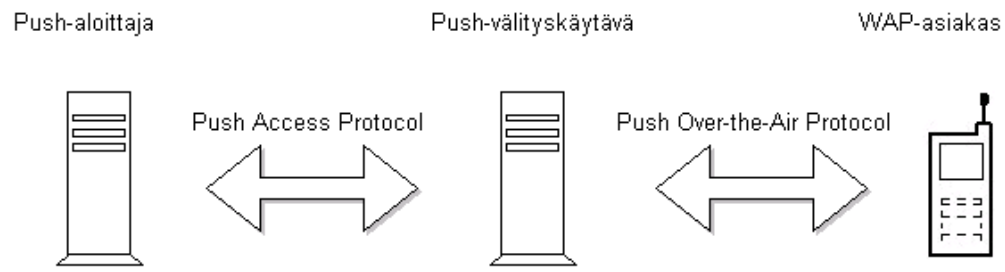
Tässä luvussa esittelen SyncML-protokollan yksityiskohtaisesti. Aluksi käsittelen SyncML:n taustaa sekä runkoa, jonka jälkeen kuvaan protokollan ominaisuuksia. Tämän jälkeen käsittelen protokollan tiedonsiirron kulun ja lopuksi vertaan SyncML-protokollaa muihin synkronointiprotokolliin.

3.1. SyncML:n runko

Edistääkseen paikasta riippumatonta pääsyä millä tahansa laitteella mihin tahansa verkossa olevaan tietoon, Ericsson, IBM, Lotus, Motorola, Nokia, Palm Inc., Psion ja Starfish Software käynnistivät SyncML- aloitteen. Aloitteen tarkoituksena oli kehittää yleinen synkronointiprotokolla, jota voidaan käyttää koko alalla [SyncML White Paper, 2001]. SyncML on siis kehitetty laitevalmistajien aloitteesta yhtenäistää päätelaitteiden synkronointiprotokolla ja edesauttaa näin palveluntarjoajien mahdollisuutta kehittää laitteistoriippumattomampia palveluja.

SyncML-spesifikaatio määrittelee kaksi protokollaa: tiedonsiirtoprotokollan (SyncML Synchronization Protocol) sekä esitysprotokollan (SyncML Representation Protocol). Tiedonsiirtoprotokollassa määritellään operaatiojoukko, jonka avulla tieto saadaan kulkemaan laitteesta toiseen. Esitysprotokolla määrittelee lähetettävien SyncML-viestien esitystavan ja muodon. Viestit esitetään käyttämällä XML-formaattia. Esitysprotokolla määrittelee kaikki tarvittavat XML-elementit ja muuttujat, joita SyncML-viesteissä käytetään.

SyncML-ohjelmointikirjasto (SyncML Reference Toolkit) tarjoaa rajapinnat SyncML-yhteensopivien XML-dokumenttien kasaamiseen sekä purkamiseen. Lisäksi se hoitaa itse tiedonsiirron laitteiden välillä. Laitteet yhdistetään käyttämällä yleisiä tiedonsiirtoprotokollia, kuten HTTP:tä (Hypertext Transfer Protocol), WSP:tä (Wireless Session Protocol) tai OBEX:ia (Object Exchange System). Tutkielman oleellisin tiedonsiirtoprotokolla on WSP, joka on osa WAP-protokollaa (Wireless Application Protocol). WSP tarjoaa mahdollisuuden langattomaan tiedonsiirtoon WAP-protokollaa tukevien laitteiden välillä. Langaton tiedonsiirto on ehdoton vaatimus palvelun toiminnalle. Tällä hetkellä WAP tukee palvelimelta asiakkaalle -tyyppistä tiedonsiirtoa vain HTTP-protokollaa käyttäen. Kuvassa 3.1 on yksinkertaistettu malli WAP:n push-mallista.

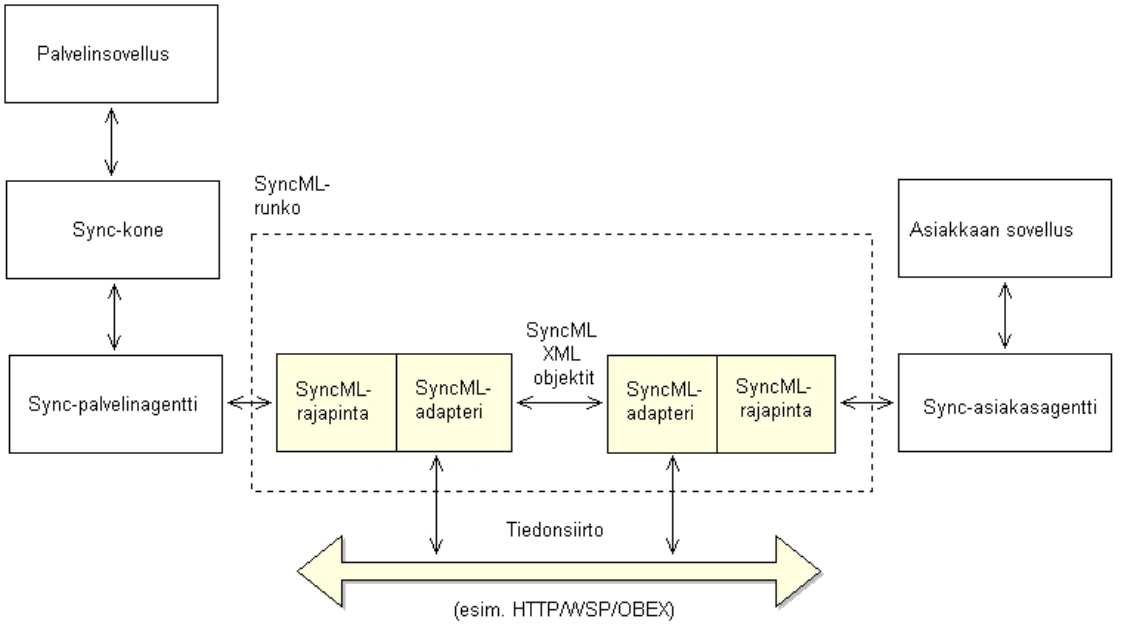


Kuva 3.1 WAP Push-arkkitehtuuri [WAP, 2001]

Push-aloittaja käyttää HTTP POST-pyyntöä ja lähettää XML-dokumentin Push-välityskäytävälle (Push proxy gateway). Push-viesti sisältää ohjausosan ja viestiosan. Ohjausosa on XML-dokumentti, joka sisältää Push-välityskäytävän tarvitsemat asiakkaan välitystiedot. Tällaisia voivat olla esimerkiksi puhelinnumero tai sähköpostiosoite. Viestiosa sisältää asiakkaalle lähetetyn tiedon, suunniteltavan palvelun tapauksessa siis SyncML viestin. Push-välityskäytävä siirtää saadun dokumentin WAP-asiakkaalle käyttäen Push Over-the-Air Protocol -protokollaa (OTA). Push Access Protocol -protokollaa (PAP) käytetään siis tiedonsiirtoon palvelimelta Push-välityspalvelimelle sekä Push Over-the-Air Protocol -protokollaa tiedonsiirtoon päätelaitteelle.

Toimiva SyncML-järjestelmä koostuu seuraavista osista (kuva 3.2):

1. SyncML-palvelin (SyncML server)
2. SyncML-asiakas (SyncML client)
3. SyncML-runko (SyncML framework)



Kuva 3.2 SyncML protokolla [SyncML Protocol, 2002]

SyncML-palvelimelle (yleensä palvelin tai PC) toteutetaan sync-palvelinagentti (sync server agent) sekä sync-kone (sync engine). Sync-palvelinagentti käyttää SyncML-kirjaston tarjoamia rajapintoja SyncML-viestien luomiseen ja purkamiseen. Sync-kone sisältää SyncML-protokollan mukaisen toiminnallisuuden: muutosrekisterin (Change log, esitellään kohdassa 3.2) ja ID-linkityksen (ID mapping, esitellään kohdassa 3.2). Tarvittaessa sync-kone suorittaa myös sync-analyysin saapuneille viesteille ja viestien sisällä oleville tietoalkioille.

SyncML-asiakas (yleensä päätelaite tai toinen PC) toteuttaa sync-asiakasagentin. Tässä tutkielmassa SyncML-asiakkaalla tarkoitan aina päätelaitetta. Samoin kuin sync-palvelinagentti, myös sync-asiakasagentti käyttää SyncML-kirjaston rajapintoja SyncML-viestien kokoamiseen sekä purkamiseen. Lisäksi sync-asiakasagentti toteuttaa myös muutosrekisterin.

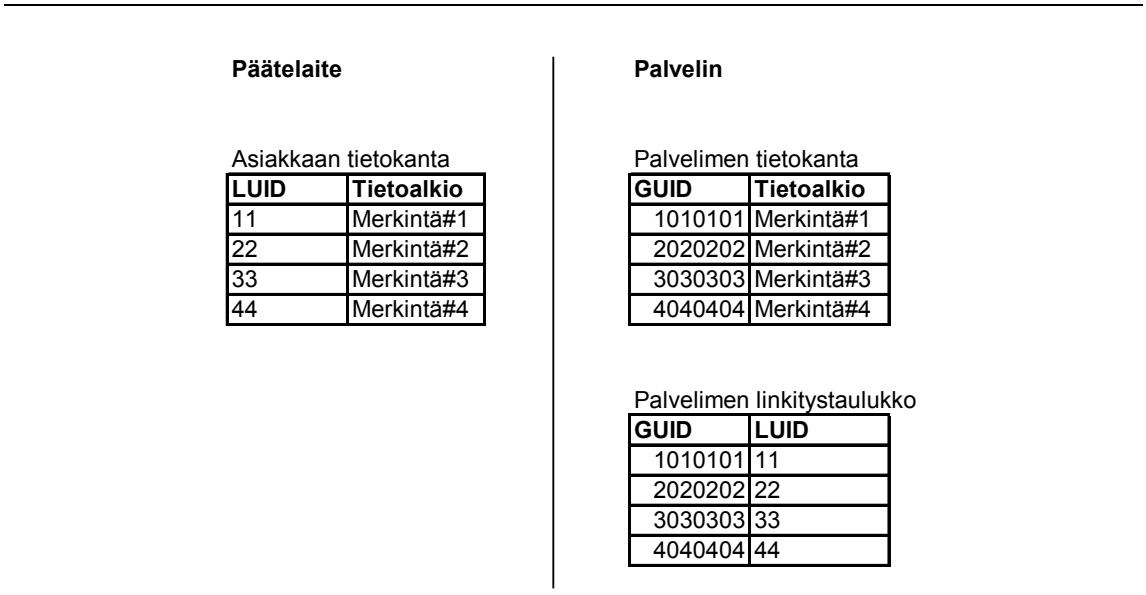
SyncML-runko toteuttaa varsinaisen tiedonsiirron palvelimen sekä päätelaitteen välillä. SyncML-adapterit hoitavat tiedonsiirron halutulla tiedonsiirtoprotokollalla välittämällä luodut viestit adapterilta toiselle. Sekä sync-asiakasagentti että sync-palvelinagentti käyttävät SyncML-rungon tarjoamia rajapintoja kommunikoidakseen keskenään. Tiedonsiirto adapterien välillä tapahtuu käyttäen jo aikaisemmin mainittuja protokollia.

3.2. Muutosrekisteri ja ID-linkitys

SyncML-protokolla mahdollistaa vain muuttuneiden tietojen päivittämisen laitteiden välillä. Tämä vähentää tiedonsiirtoa laitteiden välillä ja sen merkitys korostuu varsinkin langattomassa tiedonsiirrossa. Muutosrekisteri on palvelimelle sekä asiakkaalle toteutettava yksinkertainen tehtäväjono, johon tietoalkioiden muutokset tallennetaan synkronointikertojen välillä. Muutoksia ovat esimerkiksi tietoalkion lisäys, tuhoaminen tai sen sisällön muuttaminen. Protokolla ei määrittele mitään erityistä muotoa muutosrekisterin toteuttamiselle, mutta toteuttajan on kuitenkin tehtävä se siten, että tarvittaessa muuttuneet tiedot saadaan selville.

Tietoalkiot tunnistetaan yksilöllisillä ID-tunnisteilla. Jokaisella asiakkaalla on alkioilleen omat yksilölliset LUID-tunnisteet (Locally Unique Identifier). Palvelimella ne on linkitetty palvelimen yksilöllisiin GUID-tunnisteisiin (Globally Unique Identifier). Palvelimella on lisäksi otettava huomioon, että asiakkaan tunnisteet voivat olla yksilöllisiä tietotyyppikohtaisesti. Toisin sanoen asiakkaan kalenterimerkinnällä ja kontaktilla voi olla sama LUID-tunniste. Tämän vuoksi asiakas lähettää palvelimelle aina myös tietoalkioihin liittyvän tietotyypin.

Palvelimen on oltava koko ajan tietoinen omien GUID-tunnisteidensa sekä asiakkaiden LUID-tunnisteiden vastaavuudesta. Palvelimella on siis tallessa, mitkä asiakkaan tunnisteet vastaavat palvelimella olevia tietoalkiota. Tätä protokollan osaa kutsutaan ID-linkitykseksi (kuva 3.3).



Kuva 3.3 ID-linkitys

Asiakas luo aina LUID-tunnisteet kaikille päätelaitteen tietoalkiolle. Jos palvelimelle luodaan tietoalkio, se välitetään asiakkaalle, joka paketissa viisi palauttaa lisätylle tietoalkiolle LUID-tunnisteen. Palvelimen tehtäväksi jää lisätä uusi alkio ID-linkitykseen sekä lisätä tälle alkiolle GUID-tunniste. Asiakkaan operaatiot kohdistetaan tietoalkioihin aina käyttämällä LUID-tunnistetta. Tämän vuoksi on erittäin tärkeää, että ID-linkitys on aina ajan tasalla.

Useamman laitteen synkronointitapauksessa, jossa asiakasta tai palvelinta synkronoidaan usean eri laitteen kanssa, muutosrekisterin on oltava laitekohtainen. Muutosrekisterin on kyettävä osoittamaan tietoalkion muutokset kaikille synkronoitaville laitteille. Palvelimen tapauksessa myös ID-linkitys on toteutettava laitekohtaisesti. Toisin sanoen palvelimen on linkitettävä jokaisen päätelaitteen alkiot erikseen vastaamaan omia tietoalkioitaan. Asiakkaan puolella synkronointi useamman palvelimen kanssa vaatii laajempaa toteutusta. Päätelaitteeseen on toteuttava palvelinkohtainen muutosrekisteri, jossa toisen palvelimen muutokset päätelaitteeseen, kuten lisäykset tai poistot, on lisättävä muiden palvelinten muutosrekisteriin. Päätelaitteen muutosrekisterin on siis oltava hiukan moniulotteisempi. Ongelma useamman laitteen synkronoinnissa on, että päätelaitteet voivat toteuttaa sen eri tavoin. Yksinkertaisemmissa ratkaisuissa päätelaite ei tue useampaa palvelinta, jolloin lähes jokainen synkronointi päätelaitteen kanssa on ns. hidas synkronointi (Slow sync). Tämä johtuu siitä, että laite ei tue palvelinkohtaisia ankkureita (ankkurit esitellään yksityiskohtaisesti kohdassa 3.3) eikä muutosrekistereitä ja näin aina yhden palvelimen synkronoinnissa ankkurit vaihtuvat. Toisen palvelimen aloittaessa synkronoinnin ankkurit ovat vaihtuneet ja palvelin suorittaa hitaan synkronoinnin. Jälleen ankkurit vaihtuvat ja tilanne toistuu toisen palvelimen aloittaessa synkronoinnin. Tämä sotii SyncML:n mahdollisuuksia vastaan, jossa siis vain muuttuneet tiedot päivitetään. Loppukäyttäjälle tämä näkyy hidastuneina päivityksinä. Toiset päätelaitteet taas tukevat useamman palvelimen käyttöä, jolloin edellä mainittua tilannetta ei synny [SyncML Discussions, 2002].

3.3. Sync-ankkurit ja -tyypit

Sync-ankkurit (Sync anchors) mahdollistavat yksinkertaisen tavan tarkistaa, ovatko SyncML-asiakkaalla sekä SyncML-palvelimella olevat tiedot eheät. Eheällä tarkoitan sitä, että tiedot ovat mielekkäässä muodossa ja näin järkevästi

synkronoitavissa. Eheys voi rikkoutua esimerkiksi laitteen (asiakkaan) muistia nollattaessa tai kun laite synkronoidaan kolmannen osapuolen kanssa (esimerkiksi toinen päätelaite). Tällöin asiakkaan tietoalkiot ja näiden yksilölliset LUID-tunnisteet eivät vastaa palvelimelle linkitettyjä tunnisteita. LUID-tunnisteet voivat olla kokonaan vaihtuneet tai ne viittaavat vääriin tietoalkioihin. Näin ollen järkevää synkronointia ei voida suorittaa ennen kuin tietokannat molemmissa laitteissa on saatettu samaan tilaan.

Sekä asiakkaalla että palvelimella on kaksi ankkuria: viimeinen (Last anchor) sekä seuraava (Next anchor). Viimeinen-ankkuri kertoo, milloin laite on viimeksi synkronoitu ja seuraava-ankkuri kertoo nykyisen synkronointihetken. Asiakas lähettää omat ankkurinsa palvelimelle heti ensimmäisessä paketissa. SyncML-palvelin tallentaa ankkurit omaan tietokantaansa. Ankkurit ovat tietotyyppikohtaisia, toisin sanoen jokaisella tietotyypillä on omat ankkurinsa (esimerkiksi kalenteritiedoilla omansa ja kontakteilla omansa). Tämä mahdollistaa vain yhden tietotyypin synkronoinnin kerrallaan. Kun palvelin saa ankkurit asiakkaalta, se vertaa saamaansa viimeinen-ankkuria edellisellä synkronointikerralla talletettuun seuraava-ankkuriin. Jos ankkurit ovat samat, palvelin voi olla varma, että asiakkaan tietokanta on eheä. Jos ankkurit eriävät, palvelin ei voi tietää mitä tietoja asiakkaalla on. Tällöin palvelin voi pyytää asiakasta hitaaseen synkronointiin tai asiakkaan virkistyssynkronointiin (Refresh Sync from client only) tietokantojen eheyttämiseksi. Protokollan mukaan palvelimen on talletettava asiakkaan ankkurit, jotta edellä kuvailtu vertailu voidaan suorittaa. Asiakkaan vastaava toiminta ei ole pakollista.

SyncML-protokolla esittelee seitsemän erilaista synkronointitapaa eri tilanteisiin. Taulukossa 3.1 on esiteltynä eri synkronointitavat ja niiden käyttötarkoitus.

Sync-tyyppi	Määritelmä
Kaksisuuntainen synkronointi (two-way sync)	Yleisin synkronointimenetelmä, jossa asiakas ja palvelin lähettävät toisilleen muuttuneet tietoalkionsa. Asiakas lähettää muutoksensa ensin.
Hidas synkronointi (slow sync)	Kaksisuuntaisen synkronoinnin erityismuoto, jossa asiakkaan kaikki tietoalkiot lähetetään palvelimelle. Palvelin vertaa jokaista tietoalkioita omiinsa ja suorittaa näille Sync-analyysin.
Asiakkaan yksisuuntainen synkronointi (one-way sync from client only)	Asiakas lähettää muuttuneen tietoalkionsa palvelimelle. Palvelin ei lähetä omia muutoksiaan.
Asiakkaan virkistyssynkronointi (Refresh sync from client only)	Asiakas lähettää kaikki tietoalkionsa palvelimelle. Palvelin korvaa omat tietonsa asiakkaan tiedoilla (Sync-analyysiä ei suoriteta).
Palvelimen yksisuuntainen synkronointi (one-way sync from server only)	Palvelin lähettää muuttuneet tietoalkionsa asiakkaalle. Asiakas ei lähetä omia muutoksiaan.
Palvelimen virkistyssynkronointi (Refresh sync from server only)	Palvelin lähettää kaikki tietoalkionsa asiakkaalle. Asiakas korvaa omat tietonsa palvelimen tiedoilla.

Taulukko 3.1 Sync-tyypit [SyncML Protocol, 2002]

SyncML Sync Protocol [SyncML Protocol, 2002] esittelee lisäksi palvelimen käynnistämän synkronoinnin (server alerted sync) yhtenä sync-tyyppinä. Itse asiassa palvelimen käynnistämä synkronointi on vain tapa, jolla palvelin voi käynnistää synkronoinnin kertomalla asiakkaalle minkälaiseen synkronointiin se haluaa. Asiakas jatkaa synkronointia jollain taulukossa 3.1 esitellyllä tavalla.

Valittu synkronointitapa kerrotaan toiselle osapuolelle ensimmäisen paketin mukana. Tiedon saatuaan toinen osapuoli pystyy suorittamaan oikeanlaisen synkronoinnin ja välittämään oikeata ja oikean määrän tietoa. Normaali SyncML-protokollan mukainen synkronointi seuraisi seuraavaa käytäntöä: ensimmäisellä kerralla suoritetaan hidas tai asiakkaan virkistyssynkronointi

(kaikki tiedot asiakkaalta siirretään palvelimelle) ja tämän jälkeen käytetään kaksisuuntaista synkronointia (vain muuttuneet tiedot siirretään laitteesta toiseen). Ongelmatapauksissa voidaan suorittaa uudestaan hidas synkronointi tai virkistyssynkronointi. Tämän tutkielman toteutukselle tärkein synkronoinnin aloitus on palvelimen käynnistämä synkronointi, jossa siis palvelin aktivoi synkronoinnin pyytämällä asiakkaan haluamaansa synkronointiin. Asiakas jatkaa synkronointia normaalisti palvelimen pyytämän synkronointitavan edellyttämällä tavalla.

3.4. SyncML-paketit ja -viestit

SyncML-protokollassa tiedonsynkronointioperaatiot on sidottu SyncML-paketteihin. SyncML-paketti on käsitteellinen kehys yhdelle tai useammalle SyncML-viestille, jotka vaaditaan synkronoitavan tiedon siirtämiseen [SyncML Representation, 2002]. Kaikki paketit koostuvat siis SyncML-viesteistä, jotka kuvataan XML-formaatissa. Yksi viesti on yksittäinen XML-dokumentti. Kuten määrittelin kohdassa 2.3, XML mahdollistaa rakenteellisen tiedon hallinnan. Tästä on hyötyä tiedon synkronoinnissa, jossa sisällön lisäksi myös viestien rakenteellinen semantiikka on tärkeää [SyncML Representation, 2002]. Koska XML on kuvauskielenä kohtuullisen tilaa vievää ja hidastaa näin tiedonsiirtoa, SyncML-elementit ja muuttujien nimet ovat usein lyhennettyjä. Tämä pienentää XML-dokumentteja jonkin verran. Lisäksi XML-dokumentit on mahdollista pakata binääriseen muotoon (WAP Binary XML) [WBXML, 2001], jolloin viestit vievät selvästi alkuperäistä vähemmän tilaa. Useimmat päätelaitteet tukevat tätä pakkaustapaa.

SyncML-viestit koostuvat otsikko– ja runko-osasta. Otsikko-osa sisältää välitystietojen (esimerkiksi kohdelaite) lisäksi versiointitiedot sekä SyncML-viestin yksilölliset tunnisteet viestin tunnistamiseksi kohdelaitteessa. Runko-osa sisältää yhden tai useamman SyncML-komennon (SyncML commands). Komennot sisältävät muita elementtejä, jotka kuvaavat komennon merkityksen sisältäen myös mahdollisen synkronoitavan tiedon. Komentojen merkitys siis vaihtelee riippuen niiden sijainnista muihin elementteihin. Laitteiden on kyettävä tunnistamaan se, jos paketti sisältää useamman kuin yhden viestin. Jokaisen paketin viimeinen viesti sisältää Final-elementin, joka kertoo paketin loppuvan. Jos kyseistä elementtiä ei viestissä ole, laite voi varmistua siitä, että kyseiseen pakettiin on tulossa vielä lisää viestejä.

Alla oleva esimerkki (kuva 3.4) kuvaa yhtä SyncML-viestiä palvelimelta asiakkaalle.

```
<SyncML>
 <SyncHdr>
        <VerDTD>1.0</VerDTD<VerProto>SyncML/1.0</VerProto>
        <SessionID>1</SessionID>
        <MsgID1</MsgÌD>
        <Target><LocURI>minun_puhelin</LocURI></Target>
        <Source><LocURI>sync_palvelin</LocURI></Target>
 </SyncHdr>
 </SyncBody>
        <Status>
            <CmdID>1</CmdID>
            <MsgRef>2</MsgRef>
            <CmdRef>0</CmdRef>
            <Cmd>SyncHdr</Cmd>
            <TargetRef>sync_palvelin</TargetRef>
            <SourceRef>minun_puhelin</SourceRef>
            <Data>200</Data>
        </Status>
        <Sync>
            <CmdID>2</CmdID>
            <Target><LocURI>./kalenteri</LocURI></Target>
            <Source><LocURI>./kalenteri</LocURI></Target>
            <Add>
                <CmdID>3</CmdID>
                <Meta>
                    <Type xmlns=”syncml:metinf”>text/x-vcalendar</Type>
                </Meta>
                <Item>
                    <Source><LocURI>1234</LocURI></Source>
                    <Data>
                     BEGIN:VCALENDAR
                     ...[muu vCalendar-osuus]
                     END:VCALENDAR
                    </Data>
                </Item>
            </Add>
        </Sync>
        <Final/>
 </SyncBody>
</SyncML>
```

Kuva 3.4 SyncML-viesti

Tässä tutkielmassa ei ole oleellista selvittää lukijalle kaikkia SyncML-esitysprotokollan määrittelemiä elementtejä, joten kuvan 3.4 esimerkin

elementtien merkitystä ei kuvata sen tarkemmin. Viestistä voi kuitenkin vahvistaa seuraavia aiemmin esiteltyjä asioita:

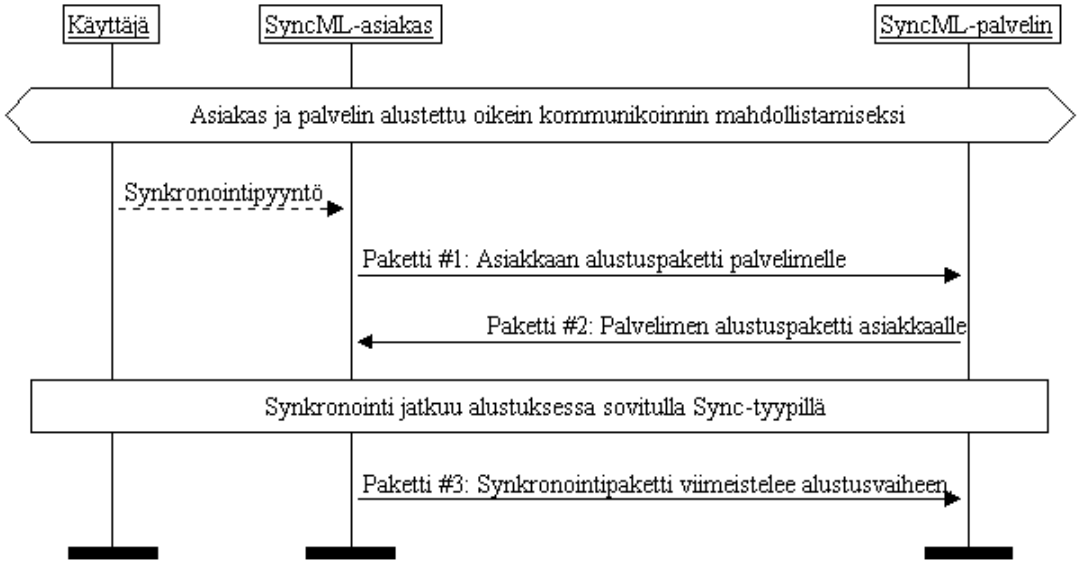
- Rakenteinen kuvauskieli mahdollistaa elementtien riippuvuuden ja selkeän luettavuuden.
- SyncML-viesti koostuu otsikosta (SyncHdr) sekä runko-osasta (SyncBody).
- Elementin merkitys vaihtelee riippuen sen sijainnista muihin elementteihin. Esimerkissä Target- ja Source-elementtien merkitykset ovat erilaiset riippuen niiden paikasta. Otsikko-osassa niillä kuvataan laitteiden osoitetietoja ja Sync-elementin alla ne kuvaavat laitteissa sijaitsevia tietokantoja. Lisäksi elementtien merkitystä voi täsmentää meta-tiedoilla [SyncML Meta-Information, 2002].
- vCalendar-objektit on mahdollista upottaa suoraan SyncML-viesteihin (Data-elementin sisään).
- Final-elementti kuvaa viestin loppumisen.

Yksinkertainen synkronointikerta sisältää kuusi SyncML-pakettia. Yhtä tällaista tiedonsiirtotapahtumaa kutsutaan SyncML-istunnoksi (SyncML session). SyncML-istunto voidaan jakaa kolmeen käsitteelliseen osaan:

1. Synkronoinnin alustus
2. Synkronointi
3. Tietoalkioiden linkitys

Synkronoinnin alustuksessa, joka normaalisti sisältää kaksi viestiä, vaihdetaan laitteiden ominaisuustiedot, sovitaan käytettävästä synkronointitavasta sekä autentikoidaan laitteet. Alustus alkaa päätelaitteen lähettäessä palvelimelle alustusviestin (paketti #1 kuvassa 3.5), jossa se ilmoittaa palvelimelle synkronoitavat tietokannat ja käytettävän synkronointityypin (eri synkronointityypit on kuvattu kohdassa 3.3). Lisäksi päätelaite voi lähettää alustusviestissään autentikointitiedot, joilla palvelin tunnistaa laitteen ja käyttäjän. Laitetietojen lähettäminen ei myöskään ole pakollista. Palvelin vastaa päätelaitteen alustusviestiin omalla alustusviestillään (paketti #2 kuvassa 3.5). Tämä sisältää kuittaustiedot kaikkiin päätelaitteen alustusviestissä olleisiin komentoihin, kuten käytettäviin synkronointitapoihin sekä synkronoitaviin tietokantoihin. Jos palvelin haluaa käyttää jotain muuta synkronointitapaa, se voi ilmoittaa siitä omassa alustuspaketissaan oikealla virhekoodilla. Palvelimen on myös tarkistettava synkronoitavien tietokantojen tila sekä analysoitava päätelaitteen mahdollisesti lähettämät laitetiedot virheiden varalta. Virheet on

jälleen kuitattava päätelaitteelle määritellyillä virhekoodeilla. Jos päätelaite ei lähettänyt laitetietoja, päätelaite voi pyytää niitä seuraavassa paketissa. Tällöin päätelaitteen on lähetettävä ne seuraavassa mahdollisessa paketissa (paketti #3 kuvassa 3.5). Autentikointitietojen puuttuessa alustuspaketista palvelin voi näitä tarvitessaan ”haastaa” päätelaitteen autentikointiin. Tällöin päätelaite lähettää paketin #1 uudelleen autentikointitietojen kanssa. Palvelin tunnistaa laitteen ja kuittaa tämän omassa alustusviestissään (käyttäjän autentikointia ja tietoturva-asiaa käsitellään tarkemmin luvussa kuusi).

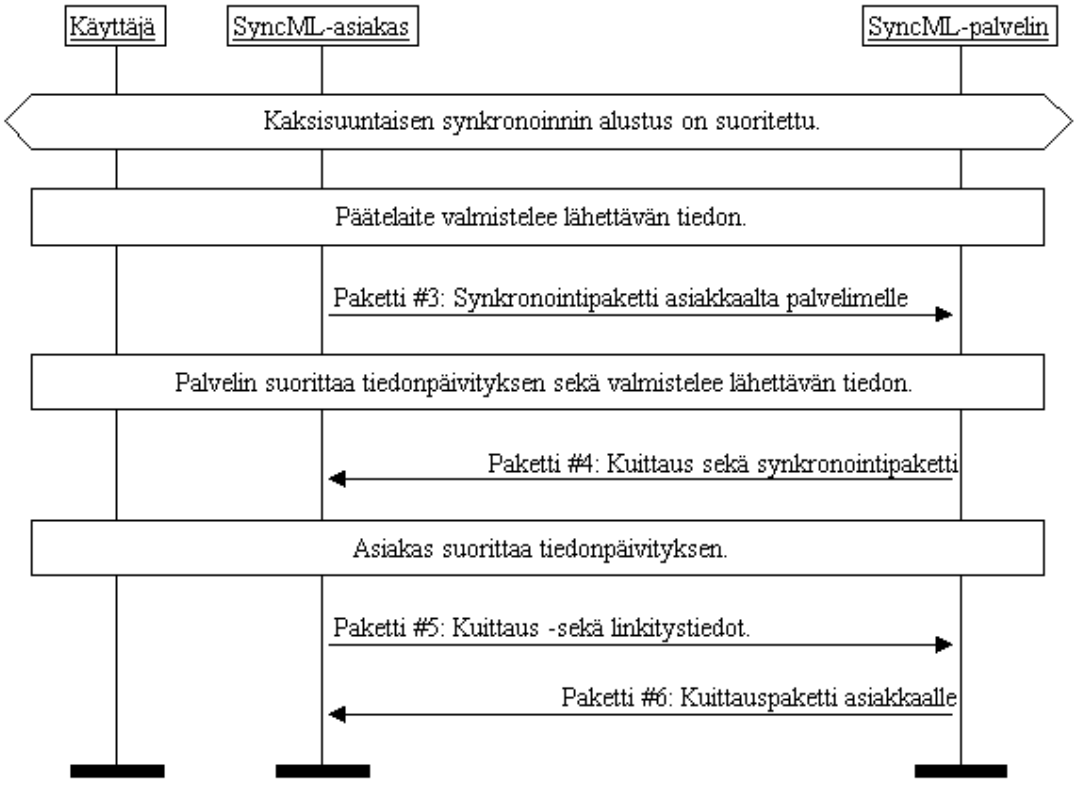


Kuva 3.5 Synkronoinnin alustus [SyncML Protocol, 2002]

On huomattava, että päätelaitteen on kuitattava kaikki palvelimen paketissa kaksi lähettämät komennot. Tämä tapahtuu varsinaisen synkronointivaiheen ensimmäisellä paketilla (paketti #3 kuvassa 3.5).

Kuten edellä mainitsin, alustusvaihe sisältää yleensä kaksi pakettia. Palvelimen käynnistämässä synkronoinnissa kuitenkin alustusvaihe sisältää kolme pakettia. Kolmas paketti lähetetään palvelimelta asiakkaalle ennen kahta esiteltyä alustuspakettia. Tätä pakettia kutsutaan paketiksi nolla. Paketilla nolla palvelimella on mahdollisuus käynnistää synkronointi ja pyytää haluttua tietokantaa synkronoitavaksi haluamallaan sync-tyypillä. Tällöin paketti yksi, eli asiakkaan alustuspaketti, sisältää edellä esiteltyjen tietojen lisäksi myös kuittaustiedot palvelimen pyytämistä synkronoinneista. Palvelin voi tarvitessa jo paketissa nolla haastaa päätelaitteen autentikointiin, jolloin asiakas lähettää tunnistetiedot jo paketissa yksi. Koska push-palvelussa palvelin käynnistää operaatiojoukon, on palvelimen käynnistämä synkronointi suunniteltavan palvelun peruselementti.

Synkronointivaiheessa asiakas ja palvelin vaihtavat tietoalkioitaan. Tämän vaiheen käynnistää aina asiakas lukemalla muuttuneet tietoalkiot omasta muutosrekisteristään ja lähettämällä ne palvelimelle (paketti #3 kuvassa 3.x) (riippuen käytettävästä sync-tyypistä asiakas lähettää joko muuttuneet tietoalkionsa tai tietokantansa kaikki alkiot). Palvelin suorittaa vastaanotetut komennot, kuten alkion lisäykset ja poistot. Palvelimen on myös suoritettava sync-analyysi mahdollisten konfliktien varalta. Sync-analyysissä palvelin vertaa saamiaan tietoalkioita palvelimelle talletettuihin tietoalkioihin vertaillen esimerkiksi tietoalkion muutosaikaleimoja tai sisältöjä. Vertailun perusteella palvelin voi joko suorittaa vastaanotetun komennon loppuun tai hylätä komennon (jos esimerkiksi palvelimen tietoalkion aikaleima on uudempi, eli sitä on muokattu myöhemmin). Tällöin palvelimen tietoalkio siirretään asiakkaalle seuraavassa paketissa. Konfliktin hallinta voidaan toteuttaa myös siten, että päätösvalta ”voittavasta” osapuolesta on käyttäjällä. Palvelimen suoritettua saadut komennot, se välittää palvelimen muutokset asiakkaalle (paketti #4 kuvassa 3.6). Lisäksi palvelin kuittaa kaikki palvelimen komennot paketista kolme.



Kuva 3.6 Synkronointi ja linkitys [SyncML Protocol, 2002]

Tietoalkioiden linkitys viimeistelee sync-istunnon. Kun asiakas vastaanottaa ja prosessoi palvelimen komennot paketista neljä, se suorittaa samalla tietokantojensa päivityksen. Tämän seurauksena asiakas lähettää palvelimelle mahdollisten uusien tietoalkioiden linkitystiedot eli uusien tietoalkioiden LUID-tunnisteet (paketti #5 kuvassa 3.6). Niiden avulla palvelin voi linkittää palvelimella olevat tietoalkiot päätelaitteella oleviin vastaaviin tietoalkioihin. Lisäksi asiakas kuittaa kaikki palvelimen paketin neljä komennot. Palvelin lopettaa sync-istunnon kuittaamalla linkitystiedot saaduksi (paketti #6 kuvassa 3.6).

Tiedonsiirto tapahtuu siis edellä kuvatuissa paketeissa. Sekä SyncML-asiakkaan että SyncML-palvelimen on pystyttävä käsittelemään saapuva tieto sekä lähettämään oikeaa tietoa oikeassa paketissa. Jos vastaanottaja ei pysty käsittelemään saatua tietoa, sen on vastattava esitysprotokollassa määritellyillä virhenumeroilla. Erityisen tärkeässä roolissa ovat kuittaustiedot (status-tiedot). Jos palvelin ei voi jostain syystä tallettaa saamaansa tietoalkiota, sen on oikealla virhenumerolla kerrottava se asiakkaalle seuraavassa lähettämässään paketissa. Näin asiakas voi yrittää uudestaan saman tietoalkion siirtoa seuraavalla synkronointikerralla. Sama koskee myös palvelimen kuittauksia. Kuittaukset liitetään aina tiettyihin tunnistetietoihin. Istunto-, viesti- sekä komentotunnisteet on liitettävä jokaiseen viestiin ja komentoon, jotta kuittaukset saavat merkityksen. Laitteen lähettäessä viestiä se sisällyttää viestiin viesti- ja istuntotunnisteen ja viestin jokaiseen komentoon komentotunnisteen. Vastaanottava osapuoli palauttaa kuittauksen tätä vastaavien vastaanotettujen tunnisteiden kanssa.

Vaikka normaali synkronointi sisältää kuusi (tai seitsemän) pakettia, tarpeen vaatiessa synkronointi voidaan tehdä myös ilman erillistä alustusvaihetta, jolloin synkronointi sisältää kaksi (tai kolme jos synkronointi on palvelimen aloittama synkronointi) pakettia vähemmän. Tällöin paketeissa yksi ja kaksi välitettävät tiedot siirretään paketeissa kolme ja neljä. Vaikka tämä vähentääkin siirrettävien pakettien määrää, tähän liittyy tietoturvaongelmia, joiden takia tapa ei ole suositeltava: autentikointi suoritetaan vasta, kun päätelaitteen tiedot on jo siirretty palvelimelle, joten päätelaite ei voi varmuudella tietää, tapahtuuko synkronointi oikean palvelimen kanssa.

3.5. SyncML-protokollan edut ja ongelmat

SyncML-protokollan etuna valmistajakohtaisiin synkronointiprotokolliin voidaan esittää kuusi pääkohtaa [Stemberger, 2001]:

- SyncML on tehokas sekä langattomassa että langallisessa tietoverkossa. SyncML-protokolla tukee tiedonsiirtoa kaikissa verkoissa, joissa päätelaitteita käytetään. Varsinkin langattomiin verkkoihin liittyy useita ongelmia, kuten rajallinen tiedonsiirtonopeus sekä tiedonsiirron epäluotettavuus, jotka SyncML-protokollan suunnittelussa on otettu huomioon.
- SyncML tukee eri tiedonsiirtoprotokollia, kuten HTTP:tä, WSP:tä, Obex:ia sekä TCP/IP:tä.
- SyncML määrittelee, missä muodossa tieto on oltava, kun sitä siirretään. Samalla SyncML tukee yleisiä päätelaitteissa esiintyviä formaatteja, kuten vCalendar- ja iCalendar-formaatteja sekä vCard-formaattia. Se missä muodossa tieto tallennetaan laitteissa, ei vaikuta protokollaan.
- SyncML on riippumaton ohjelmointikielestä, minkä vuoksi SyncML-sovelluksia voidaan toteuttaa mittavalla määrällä eri kieliä. Tämä lisää lopulta käyttäjän sovellusvalikoimaa.
- SyncML on optimoitu langattomille päätelaitteille.
- SyncML on suunniteltu olemassa olevien Internet-teknologioiden päälle, jotka ovat tunnettuja ja laajalti testattuja.

SyncML-protokolla ratkoo jo itse useita palveluja vaivaavat ongelmat. Luotettavuuden ja nopeuden lisääntyminen kasvattaa asiakkaiden luottamusta palveluun ja palveluntarjoajaan. Lisäksi avoin standardoitu protokolla antaa sovelluskehittäjille mahdollisuuden rakentaa palveluja halvemmalla ja palvelut ovat tarjolla useammalle eri laitteelle. Yhden tuetun protokollan ylläpitäminen on huomattavasti helpompaa, kuin tukea eri valmistajien omia synkronointiprotokollia. Lopullisen hyödyn saavuttavat loppukäyttäjät kasvavassa palveluiden määrässä.

Kuten useissa muissakin protokollissa, myös SyncML:ssä tärkeimpien operaatioiden ja elementtien toteuttaminen on määritelty pakolliseksi ja harvemmin käytettävien, lisäominaisuuksia antavien elementtien tukeminen on vapaavalintaista. Pakollisten ominaisuuksien määrittäminen mahdollistaa tärkeimpien operaatioiden yhteensopivuuden kaikkien SyncML:ää tukevien laitteiden välillä. Kuitenkin monimutkaisempien ja asioita helpottavien

elementtien toteuttaminen on usein vapaaehtoisia ja on lopulta valmistajasta kiinni, mitkä niistä toteutetaan. Tämä voi aiheuttaa yhteensopivuusongelmia eri valmistajien tai laitemallien välillä. Lisäksi useat suuret laitevalmistajat käyttävät yhä omia synkronointiprotokolliaan ja näin vaikeuttavat SyncML-protokollaan perustuvien palveluiden yleistymistä.

4. Palvelun kuvaus

Tässä luvussa kuvaan palvelun toimintamallin sekä esittelen palvelun suunnitteluun ja toteutukseen liittyviä ongelmia. Ongelmien kuvauksessa otetaan kantaa myös mahdollisiin ratkaisuvaihtoehtoihin, mutta vaihtoehtojen tarkempi määrittely, vaihtoehdon valinta sekä perustelut esitellään luvuissa viisi ja kuusi. Luvun lopuksi kuvaan palvelun yksinkertaisen arkkitehtuurin.

4.1. Palvelun yleiskuvaus

Suunniteltava palvelu välittää käyttäjälle erilaisia kalenteritapahtumia langattomaan päätelaitteeseen ilman käyttäjän aktivoimaa tapahtumaa. Palvelu on siis niin sanottu push-palvelu (ks. kohta 2.5), jossa palvelin siirtää tietoa päätelaitteeseen. Tiedon välityksessä palvelimen ja päätelaitteen välillä käytetään SyncML-protokollaa (tarkemmin WSP:tä). Palvelun käytön edellytyksenä on, että käyttäjän päätelaite tukee SyncML-protokollaa, WAP-protokollaa ja että päätelaitteessa on kalenteriohjelma, joka tukee vCalendar-formaattia. Lisäksi päätelaitteen pitää tukea WAP-Push ominaisuutta. SyncML-protokollan avulla pyritään minimoimaan tiedonsiirto päätelaitteen ja palvelimen välillä.

Palvelu voidaan toteuttaa erilaisiin ympäristöihin; riippuen palvelun kohteesta (esimerkiksi toimiiko palvelu yrityksessä sisäisesti vai onko palvelu julkinen) kalenterimerkinnät voivat vaihdella hyvinkin paljon. Yrityksen sisäisessä käytössä kalenterimerkinnät voivat olla esimerkiksi seuraavanlaisia:

- palaverit,
- katselmoinnit,
- määräajat ja toimitusajat, sekä
- ryhmäpäivät ja vastaavat tapahtumat.

vCalendar-formaatti antaa myös mahdollisuuden liitteiden liittämiseen merkintöihin. Päätelaitteiden rajallisen muistikapasiteetin vuoksi on kuitenkin suositeltavampaa vain viitata haluttuun asiakirjaan. Katselmointeihin tai palavereihin voidaan liittää katselmoitavan dokumentin viittaus (URL) tai ryhmäpäiviin karttana toimivan kuvatiedoston sijainti. Katselmointimerkintä voisi näyttää esimerkiksi seuraavanlaiselta (esitystapa vaihtelee kalenterisovelluksesta riippuen):

Aihe: Katselmointi
Paikka: Kokoushuone 1
Aloitusaika: 20.05.2002 09:00
Lopetusaika: 20.05.2002 11:00
Liite: http://www.yritys.fi/dokumentit/dokumentti.doc
Kuvaus: Projektin 1 suunnitteludokumentin katselmointi

Kuva 4.1 Kalenterimerkintä katselmoinnista

Lisäksi vCalendar-formaatti tukee yksityiseen käyttöön hyödyllistä ATTENDEE-ominaisuutta, jolla voidaan kuvata, ketä kyseinen kalenterimerkintä koskee. Näin voidaan helposti rajata, kenelle kalenterimerkintä synkronoidaan.

Julkiselle sektorille suunniteltaessa kalenterimerkinnät voivat olla esimerkiksi seuraavanlaisia:

- nimipäivät,
- juhlapäivät ja pyhät,
- liputuspäivät,
- elokuvien ensi-illat,
- teatterien ensi-illat,
- DVD- elokuvien / CD:iden julkaisupäivämäärät,
- urheilutapahtumat,
- näyttelyt sekä museot.

Lisäksi käyttäjällä on mahdollisuus muokata omaa palveluaan siten, että edellä mainittujen kalenterimerkintöjen lisäksi voidaan synkronoida myös käyttäjän itse lisäämiä merkintöjä muistutuksen tapaan. Tällaisia voisivat olla esimerkiksi syntymäpäivät ja muut vuosittaiset ja viikoittaiset tapahtumat. Periaatteena on, että kalenterimerkinnät voivat liittyä mihin tahansa alueeseen palvelun kohteesta riippuen. Se, mitä palveluntarjoajan tietokantaan syötetään, on kiinni nimenomaan palveluntarjoajasta.

Kalenterimerkintöjä voi olla kahdenlaisia: kokopäiväisiä tai normaaleja yksittäisiä tapahtumia. Kun merkintä on kokopäiväinen, se näkyy päivän yläpuolella päivään yleisesti liittyvänä tapahtumana (vertaa esimerkiksi perinteisen paperikalenterin nimipäivät ja liputuspäivät). Yksittäinen tapahtuma näkyy kalenterissa normaalina merkintänä (esimerkiksi konsertin

aika ja paikka). Kuten aikaisemmin jo mainitsin, vCalendar-formaatti tukee myös liitteitä. Kaikkiin kalenteritapahtumiin on mahdollista liittää kuvien tai asiakirjojen sijainteja, www-osoitteita tai esimerkiksi tuotteiden hintoja.

4.2. Palvelun tekniset rajoitteet

Palvelun lähtökohtana on käyttää uusia avoimia standardeja, joista useat ovat vielä laitteiden valmistajilla käyttöönottovaiheessa. Esimerkiksi WAP 1.2-spesifikaatio määrittelee WAP Push-tekniikan, jonka avulla mahdollistetaan tiedonsiirto palvelimelta asiakkaalle (kuvattu tarkemmin kohdassa 3.1). WAP-Push ominaisuutta ei kuitenkaan tällä hetkellä tueta yleisesti päätelaitteissa vaan se on valmistajilla vasta käyttöönottovaiheessa. Protokollien mukana tuomien ominaisuuksien puuttuminen tarkoittaa sitä, että tällä hetkellä markkinoilla olevilla laitteilla ei tutkielmassa suunniteltavaa palvelua pystytä käyttämään. Uusien laitemallien mukana tuomat tarvittavat ominaisuudet mahdollistavat palvelun laajan käytön.

Lisäksi useat standardit määrittelevät ominaisuuksien toteuttamisen joko pakollisiksi tai valinnaisiksi. Toisin sanoen suuri joukko standardien ominaisuuksista ja niiden toteuttamisesta päätelaitteeseen on kiinni laitevalmistajan tarpeista ja motiiveista. Esimerkiksi vCalendar-formaatti tarjoaa ominaisuuksia, joista useat tämän palvelun kannalta tarpeelliset ovat valinnaisia. SyncML-protokollaan liittyi sama ongelma. Koska SyncML on standardi eikä sovellus, SyncML-sovellussuunnittelijoiden uhkakuvana on laitevalmistajien osittainen SyncML-protokollan tukeminen laitteissa. Useat laitevalmistajat voivat tukea SyncML-protokollaan vain osittain ja useita tärkeitä ominaisuuksia voidaan jättää pois. Tämä tietenkin vaikeuttaa sovellussuunnittelijoiden mahdollisuutta tehdä täysin laiteriippumattomia palveluita. SyncML:ää tukevien laitteiden puuttuessa vielä markkinoilta, jää tämän uhkakuvan toteutuminen nähtäväksi.

4.3. Suunnitteluongelmat

Chen ja Suda [1997] tutkivat hajautettuja sovelluksia mobiililaitteissa. Samoja periaatteita voidaan käyttää myös asiakas/palvelintyyppistä mobiilipalvelua suunniteltaessa. Mobiilipalvelun suunnittelussa on otettava huomioon seuraavia teknisiä asioita [Chen ja Suda, 1997]:

1. Palvelun on varauduttava useisiin yhteyskatkoksiin: langattoman verkon sekä mobiililaitteiden kohdalla yhteyskatkot ovat kiinteää verkkoa yleisempiä.
2. Tiedonsiirtonopeus on rajallinen.
3. Asiakaslaitteiden rajallinen muistikapasiteetti ja pc-koneita vähäisempi prosessoriteho.

Koska palvelusta pyritään suunnittelemaan mahdollisimman monikäyttöinen, tarkastellaan myös seuraavia sisällöllisiä vaatimuksia palvelun monipuoliselle mukautukselle [Autere et al., 2001]:

4. Palvelu on mukautettavissa eri sisällöille.
5. Palvelu mukautuu käyttäjän tarpeiden mukaan.
6. Palvelu mukautuu asiakaslaitteen ominaisuuksien mukaan (palvelua voi käyttää erilaisilla asiakaslaitteilla).
7. Palvelu voidaan mukauttaa eri kulttuurien mukaan (kieli, kansallisuus).

Palveluntarjoajan (sekä suunnittelijan) näkökulmasta palvelun mukauttaminen tarkoittaa sitä, että palvelu on muokattavissa mahdollisimman pienellä työmäärällä toiseen ympäristöön [Autere et al., 2001]. Käyttäjälle tämä näkyy palvelun käyttäjäkohtaisena mukautumisena niin laitteen ominaisuuksien kuin käyttäjän mielenkiinnon kohteidenkin mukaan.

Palvelu on pääasiallisesti tarkoitettu julkisen sektorin käyttöön. Tämän vuoksi palvelun käytön ja käyttöönoton oltava myös muiden kuin alan ammattilaisten hallittavissa. Edellä listattujen vaatimusten lisäksi lisään palvelun suunnittelulle vaatimuksen:

8. Palvelun käyttöönoton on oltava käyttäjälle vaivatonta.

Vaatimukset 1 ja 2 täyttyvät osittain palvelussa käytettävien standardien ominaisuuksilla. Mahdolliset toistuvat yhteyskatkokset tarkoittavat sitä (vaatimus numero 1), että palvelun on pystyttävä jatkamaan mahdollisesti käynnissä ollutta operaatiota yhteyden jälleen palatessa. SyncML-protokollan muutosrekisteri toimii keskeneräisten tapahtumien listana: tapahtuma poistetaan muutosrekisteristä vasta, kun saadaan vahvistus onnistuneesta siirrosta (SyncML-paketissa numero viisi). Jos yhteys katkeaa, kun tiedonsiirto on ollut käynnissä, yritetään muutosrekisteriin jääneitä tapahtumia siirtää uudelleen yhteyden jälleen muodostuttua. Näin kalenterimerkintöjä ei jää synkronoimatta, vaikka mahdollisia yhteyskatkoksia tapahtuisi. Vaatimus 2 on otettu huomioon jo SyncML-protokollan suunnitteluvaiheessa: protokolla on

suunniteltu mobiililaitteille ja näin optimoitu epäluotettaviin ja hitaisiin verkkoihin. SyncML-viestit voidaan halutessa siirtää binäärisessä XML-formaattissa, jolloin viestien kooksi saadaan murto-osa tekstipohjaisesta XML-viestistä. Lisäksi muutosrekisteri sekä id-linkitys mahdollistaa vain synkronointikertojen välillä muuttuneiden tietojen synkronoinnin. Lisäksi protokolla mahdollistaa tietotyyppikohtaisen synkronoinnin (synkronoidaan vain esimerkiksi kalenteritiedot). Näillä toimenpiteillä voidaan vähentää huomattavasti siirrettävää tietoa.

Vaatimusten 4, 5, 6 ja 7 täyttämiseen vaaditaan palvelun asiasisällön muokkaamista tarpeiden mukaan. Palvelun mukauttaminen eri sisällölle sekä mukauttaminen kielen ja kansallisuuden perusteella on periaatteessa sama asia. Koska vCalendar-formaatin käyttäminen ei edellytä tietyn kielen käyttöä eikä rajoita kalenterimerkinnän sisältöä, riippuu palvelun sisältö palveluntarjoajasta. Palvelun sisältö ja käytettävä kieli voi siis vaihdella palvelun kohteen mukaan. Käyttäjien kansallisuus voidaan ottaa huomioon tarjoamalla esimerkiksi maakohtaisia merkkipäiviä tai nimipäiviä. Palvelun mukauttaminen eri käyttäjäryhmille on siis lähes täysin riippumaton toteutuksesta.

Keskeisin suunnitteluongelma on se, että palvelu mukautuu käyttäjän tarpeiden mukaan. Käyttäjän laitteen ominaisuuksien mukaan tehtävä mukauttaminen on hoidettava erikseen. Sisällöllistä mukauttamista noudatetaan ottamalla huomioon käyttäjän mielenkiinnon kohteet ja tarpeet. Vain käyttäjää kiinnostavat ja käyttäjälle tarpeelliset kalenterimerkinnät synkronoidaan käyttäjän laitteeseen. Tätä operaatiota kutsutaan personoinniksi. Personoinnilla vähennetään käyttäjään kohdistuvaa tietotulvaa ja vältetään käyttäjän turhautumista liiallisista merkinnöistä. Personointi liitetäänkin usein push-palveluihin, jossa siis tietoa siirretään käyttäjän toimista riippumatta palveluntarjoajalta käyttäjälle. Push-palveluissa personointi on siis erityisen tärkeää, koska käyttäjä ei itse suoranaisesti valitse siirrettävää tietoa, alkuperäinen tietomäärä voi olla suuri ja palvelu ei saa täyttää päätelaitteen rajallista muistikapasiteettia.

Personoinnista vastaavat sitä varten suunnitellut itsenäiset agentit, jotka suodattavat halutut tiedot käyttäen profiileja. Sen lisäksi, että profiileihin tallennetaan käyttäjän yhteystietoja, esimerkiksi sähköpostiosoite tai matkapuhelinnumero, sekä Device Information -dokumentti, profiilit kuvaavat myös käyttäjän mielenkiinnon kohteita (mistä halutaan tietoa) tai mahdollisesti

myös sitä tietoa, mistä käyttäjä ei ole kiinnostunut (esimerkkinä sähköpostin suodatus, jossa käytetään otsikko- sekä lähettäjäkenttiä suodatuksen perusteina) [Belkin ja Croft, 1992]. Profiiliin voidaan tallentaa esimerkiksi avainsanoja halutusta aiheesta tai avainsanoja siitä mitä halutaan suodattaa pois. Profiiliin voidaan tallentaa suoraan myös erilaisia kyselyitä (esimerkiksi sql-kyselyitä), joita voidaan ajaa uutta tietoa vastaan.

Käyttäjäkohtainen profiili voidaan luoda periaatteessa kahdella eri tavalla: käyttäjä määrittelee profiilin itse (tietopohjaiset agentit) tai agentit itse pyrkivät luomaan profiilin seuraamalla käyttäjän toimintoja (oppivat agentit) [Maes, 1994] [Antikainen et al., 1998]. Tietopohjaisessa mallissa agentit eivät siis muokkaa profiileja, vaan käyttäjä itse alustaa ne ja muokkaa niitä käyttäen esimerkiksi www- tai wap-lomaketta. Oppivassa mallissa agentit luovat ja ylläpitävät profiileja seuraamalla käyttäjän toimintoja.

Koska profiilien käyttö on jatkuvaa (ei kertaluontoista), ne on tallennettava levylle. Agenttien tehokkuuden kannalta parasta olisi tallentaa ne mahdollisimman lähelle agentteja itseään. Device Information -dokumentti saadaan laitteista valmiiksi XML-muodossa [SyncML Device Information, 2002], joten yksi vaihtoehto on tallentaa muukin profiileihin sijoitettava tieto XML-pohjaisiin tiedostoihin [Cingil, 2000]. XML-pohjaista lähestymistapaa on alettu tutkia yhä enemmän sen vuoksi, että se on laitteisto- ja alustariippumaton. Toinen vaihtoehto on tallentaa tiedot esimerkiksi relaatiotietokantaan tai käyttää muita tiedostopohjaisia ratkaisuja XML:n sijaan. Myös synkronoitavat kalenterimerkinnät sekä SyncML-protokollan muutosrekisteri ja tunnistelinkitys on talletettava levylle. Kuten profiilien tallennuksessa, vaihtoehtoja on useita. Kalenterimerkintöjen kohdalla on otettava huomioon myös se mahdollisuus, että merkinnät tallennetaan suoraan vCalendar-formaatissa levylle.

Profiilien paikkaan vaikuttaa siis agenttien sijainti. Agentin sijaintiin vaikuttavat taas vaatimukset 2, 3 ja 6. Kuvan 2.6 mallissa agentit (kuvassa suodattimet) on sijoitettu palvelun puolelle eli palvelimelle. Vaihtoehtoisesti agentit voidaan sijoittaa päätelaitteeseen tai muualle verkkoon, esimerkiksi jollekin välityspalvelimelle. Jos halutaan minimoida verkon kuormittamista, on agentit ja näin tiedon suodatus sijoitettava palvelimelle. Toisaalta jos asiakkaita on useita tuhansia, palvelin voi kuormittua tiedon suodatuksen ja profiilien ylläpidon aiheuttamista toimista. Vaatimuksen numero kolme mukaan

päätelaitteiden suorituskyky sekä muistikapasiteetti on myös otettava huomioon.

Edellä mainittujen suunnitteluongelmien lisäksi mobiilipalveluissa on aina otettava huomioon käyttäjän tietoturva. Palvelussa siirretään käyttäjän omia kalenterimerkintöjä tai vähintäänkin niiden ajankohdat verkon yli laitteelta palvelimelle. Lisäksi palvelu tallettaa käyttäjälle henkilökohtaisia asioita personointia varten profiiliin. Näihin liittyen voidaan eritellä kaksi tätä palvelua koskevaa tietoturvaongelmaa: käyttäjäprofiilin suojaus sekä laitteen ja palvelimen välillä siirtyvän tiedon suojaus [Loeb, 1992]. Palveluntarjoajan on pystyttävä takaamaan asiakkaalle tietojen täysi luottamuksellisuus ja tietoturva.

Jos tarkastellaan edellä listattuja vaatimuksia, voidaan selvästi todeta, että vain vaatimukset yhdestä kolmeen johtuvat palvelun mobiilisuudesta. Muut vaatimukset pätevät yleisesti kaikkiin sähköisiin palveluihin. Kuitenkin on huomattava, että palvelun mobiilius yhdistettynä tässä tutkielmassa määriteltyihin päätelaitteisiin asettaa lisävaatimuksia sekä personoinnille että tietoturvalle. Näihin lisävaatimuksiin palaan tutkielman myöhemmissä kohdissa aina kutakin ongelmaa käsiteltäessä.

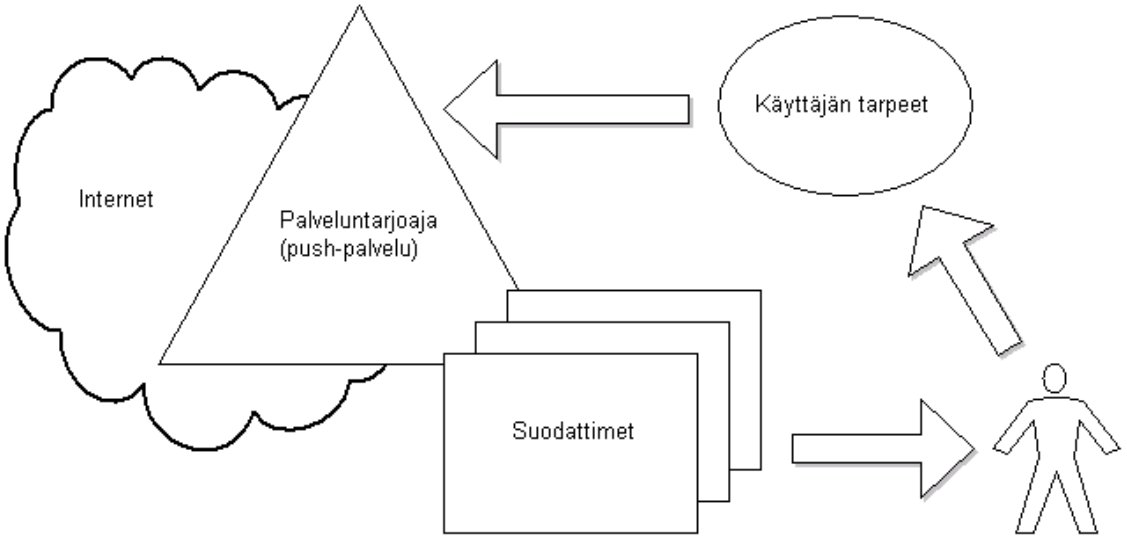
Palvelun toteutukseen liittyy lisäksi useita pienempiä ongelmakohtia, joita ei resurssien puutteen vuoksi tässä raportissa tutkita. Ne ovatkin mahdollisia jatkotutkimusten aiheita. Tutkielman keskeisimpänä suunnitteluongelmana on pyrkiä löytämään paras vaihtoehto agentin toiminnallisuudelle ja tehokkaalle tiedon suodatukselle. Kuitenkaan agenttien tarkkaan toteutukseen, siis siihen, miten tiedon suodatus algoritmitasolla tapahtuu, ei oteta kantaa.

4.4. Arkkitehtuuri

Koska useat suunnittelu– sekä toteutusongelmat, kuten agenttien lopullinen sijainti, vaikuttavat palvelun kokonaisarkkitehtuuriin, kuvaan tässä vaiheessa palvelun mallin yleisellä tasolla. Tarkoituksena on antaa lukijalle kuva, mistä palvelussa on arkkitehtuurillisesti kyse. Tässä kuvattavaa mallia voi myös pitää lähtökohtana palvelun suunnittelulle. Luvussa viisi kuvaan palvelun arkkitehtuuria tarkemmin ratkaisujen perusteella.

Kendall ja Kendall [1999] kuvaavat push-palvelun (kuva 4.2) seuraavasti: palveluntarjoaja suodattaa sopivaa informaatiota käyttäjien tarpeiden ja

vaatimusten perusteella verkosta tai palvelustaan ja välittää näin saadut tulokset käyttäjälle.

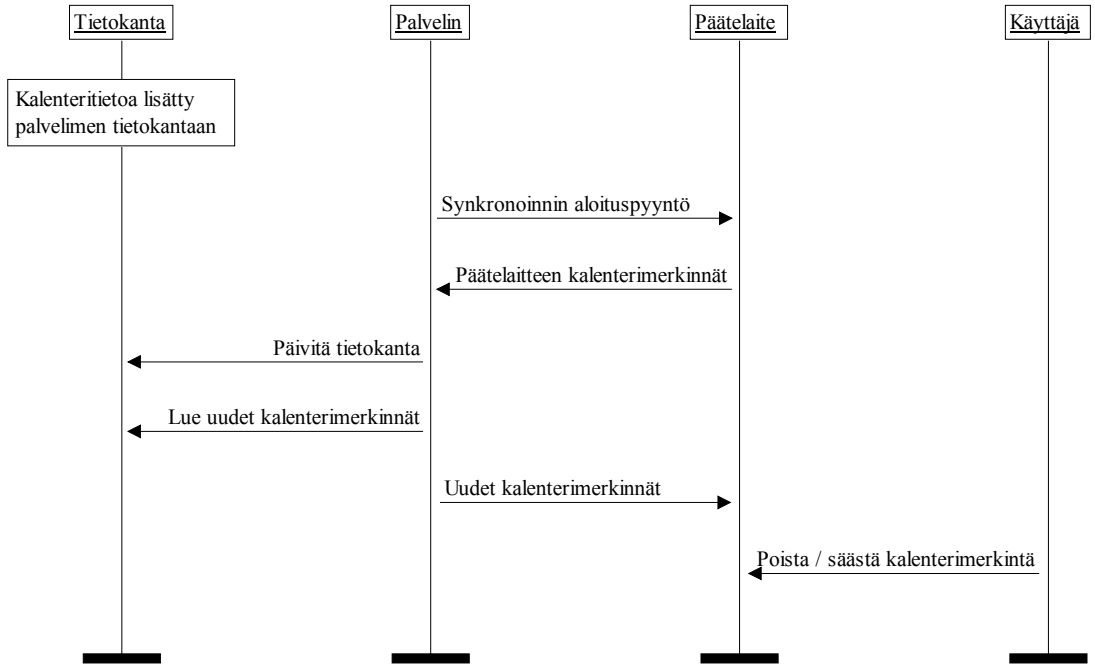


Kuva 4.2 Push-malli [Kendall ja Kendall, 1999]

Kuvassa 4.2 esitellään myös ”Käyttäjän tarpeet”. Tästä lähtien tutkielmassa käytetään sanaa käyttäjäprofiili (user profile) tai profiili kuvaamaan kyseistä elementtiä. Tällä tarkoitetaan siis niitä käyttäjäkohtaisia sääntöjä, joilla tietoa suodatetaan käyttäjälle sopivaksi. Tutkielmassa suunniteltava palvelu sisältää samat elementit kuin Kendallin ja Kendallin [1999] malli: tietokannan (kuvassa Internet), palvelun tarjoajan, agentit (kuvassa suodattimet), profiilit (kuvassa käyttäjän tarpeet) sekä asiakkaan päätelaitteen. Näiden lisäksi palveluun kuuluu SyncML-agentit päätelaitteeseen sekä palvelimelle, SyncML-protokollan mukaiset muutosrekisterit sekä ID-linkitykset. Elementtien lopullinen sijainti ja niiden suhteet toisiinsa määritellään luvussa viisi.

Palvelun yksinkertainen toimintaskenaario on kuvattuna kuvassa 4.3. Skenaariossa ei vielä ole kuvattuna yksityiskohtaisia elementtejä, kuten SyncML-protokollan mukaisia ominaisuuksia, vaan skenaarion tarkoituksena on kuvata palvelun perustoimintaa. Kun palvelimen tietokantaan lisätään kalenterimerkintöjä, palvelin aktivoi synkronoinnin päätelaitteen kanssa pyytämällä haluamaansa synkronointitapaa (skenaarion tapauksessa kaksisuuntaista synkronointia). Tämä tapahtuu SyncML-paketilla 0 (katso kohta 3.4). Aluksi palvelin vastaanottaa päätelaitteella muokatut kalenterimerkinnät (käyttäjän itse lisäämät tai muokkaamat) ja tiedon sieltä poistetuista merkinnöistä sekä päivittää saadut tiedot käyttäjäkohtaiseen tietokantaan. Tämän jälkeen palvelin lukee uudet merkinnät tietokannasta ja

lähettää ne päätelaittelle. Lopulta uudet merkinnät on synkronoitu käyttäjän laitteeseen, minkä jälkeen merkinnät voidaan joko säästää tai poistaa.



Kuva 4.3 Palvelun toimintaskenaario

5. Personointi

Tässä luvussa perehdyn tutkielman tärkeimpään tutkimusongelmaan, palvelun personointiin. Aluksi pohdin personoinnin merkitystä, minkä jälkeen esittelen personoinnin toteutustapoja.

5.1. Personoinnin merkitys

Personointi liittyy nykyään yleisesti sähköisiin palveluihin ja sen suosio on kasvanut viime vuosina selvästi. Yhdysvalloissa julkaistun tutkimuksen mukaan 30% (18.8 miljoonaa) internetiä käyttävästä aikuisväestöstä on personoinut www-sivuja [Mabley, 1999] ja vuonna 2000 vastaava luku oli kasvanut jo 28.8 miljoonaan [Mabley, 2000]. Personointi www-sivuilla on yleisesti käyttäjän asetusten ”muistamista” sekä kohdennetun mainonnan käyttämistä. Yhä useammin käyttäjän toiminnot www-sivulla tallennetaan ja seuraavalla käyntikerralla palvelu yrittää täsmentää mainontaa käyttäjän mieltymyksille sopivammaksi. Personointi ei siis sinällään ole mikään mobiilipalvelun erityispiirre, vaan yleisesti käytetty ominaisuus nykyisissä sähköisissä palveluissa. Kuitenkin on otettava huomioon, että personoinnin merkitys mobiilimaailmassa on laaja-alaisempaa verrattuna web-maailmaan. Mobiilipalveluissa on otettava huomioon myös käyttäjien erilaiset päätelaitteet ja näiden ominaisuudet, koska päätelaitteiden muistikapasiteetit sekä prosessointitehot vaihtelevat mallien ja valmistajien mukaan (suunnitteluvaatimus 6). Tiedonsuodatuksen merkitys korostuu, kun päätelaitteeseen mahtuu vain murto-osa siitä, mitä PC-koneisiin voidaan tallentaa.

Käyttäjien personointi ja samalla sisällön personointi onkin palvelussa keskeisenä osana. Tekesin tutkimukseen tehdyn empiirisen tutkimuksen aineiston mukaan [Kiuru, 2000] matkapuhelimiin halutaan pitkälle personoitua tietoa, joka on myös ajan tasalla. Personoinnissa tärkeimpänä koetaan tarpeettomien informaatiosisältöjen poissulkeminen. Informaatiosisällöstä halutaan rajata pois sellaiset osat, jotka eivät missään tapauksessa missään olosuhteissa tunnu nyt eivätkä lähitulevaisuudessa kiinnostavilta [Kiuru, 2000]. Käyttäjille on siis pystyttävä synkronoimaan vain heitä kiinnostavia kalenterimerkintöjä.

Aikaisemman määrittelyn mukaan tiedonsuodatuksen voi jakaa tietopohjaiseen ja oppivaan tiedonsuodatukseen. Tietopohjaisessa mallissa agentit eivät itse muuta profiilia, vaan muutokset tulevat käyttäjältä. Usein profiilia ei muuteta

ollenkaan. Oppivassa mallissa agentit muokkaavat profiilia aina käyttäjien toimintojen tai mahdollisen gps-paikkannuksen avulla saamiensa tietojen mukaan. Näin profiili muuttuu koko ajan tarkemmaksi kuvaukseksi käyttäjän mieltymyksistä. Riippuen profiileihin tallennetusta tiedosta, tietopohjaisen ja oppivan mallin voi jakaa useampaan alakohtaan. Seuraavaksi esittelen tietopohjaisen ja oppivan mallin toteutusvaihtoehtoja. Näitä ennen käsitellään palvelun mukauttamista käyttäjän laitteen mukaan.

5.2. Palvelun mukauttaminen eri laitteille

Kuten edellisessä kohdassa mainitsin, mobiilipalveluissa laitekohtainen personointi on erittäin tärkeää. SyncML-protokolla mahdollistaa laitteiden ominaisuuksien huomioonottamisen personoinnissa. Keskustelevat päätelaitteet voivat protokollan ensimmäisten pakettien aikana vaihtaa Device Information -dokumentin [SyncML Device Information, 2002]. Device Information -dokumentti sisältää yksityiskohtaista tietoa käytettävästä laitteesta ja sen ominaisuuksista (esimerkiksi päätelaitteen muistikapasiteetin sekä laitteen tukemat kalenterimerkinnän eli vCalendar:in ominaisuudet). Tietojen avulla palvelua voidaan mukauttaa enemmän käyttäjän laitteelle sopivaksi, esimerkiksi tehokkaamman laitteen ollessa kyseessä mahdollinen liitetiedosto voidaan liittää suoraan kalenterimerkintään, kun taas ominaisuuksiltaan heikompaan laitteeseen voidaan synkronoida merkintä, jossa on vain viite kyseiseen liitteeseen. Tuettavilla kalenterimerkinnän ominaisuuksilla voidaan lisäksi lisätä tai vähentää merkintään liitettävää tietomäärää. Tällä tavoin voidaan myös tiputtaa vCalendar-objektista sellaiset kentät pois, mitä laite ei tue. Lisäksi Device Information- dokumentista saadaan tieto päätelaitteen tukemista SyncML-protokollan mukaisista synkronisointityypeistä. Palvelin voi tämän tiedon avulla vaihtaa haluttua synkronointitapaa ja tukea näin useampaa eri tyyppistä laitetta.

Jos mobiililaitteet tuovat lisäongelmia personointiin, niin toisaalta ne mahdollistavat merkittävän edun aiemmin viitattuun staattiseen web-maailmaan verrattuna: gps-paikannuksen (Global Positioning System) avulla personointi voidaan ulottaa riippumaan myös käyttäjän maantieteellisestä sijainnista (vaatimus 4, 5 ja 7). Käyttäjän paikan ja liikkeiden avulla käyttäjälle voidaan synkronoida entistä personoidumpia kalenterimerkintöjä. Kun käyttäjä matkustaa toiselle paikkakunnalle, palvelin voi synkronoida käyttäjälle tarkennettua tietoa uuden paikkakunnan tapahtumista. Koska gps-paikannus on maailmanlaajuinen järjestelmä, personointi olisi mahdollista ulottaa myös

kotimaan rajojen ulkopuolelle. Tällöin käyttäjälle voitaisiin synkronoida maakohtaisia tapahtumia hänen olleessa ulkomailla.

5.3. Tietopohjainen tiedonsuodatus

Yleisimmät tavat tietopohjaisen mallin toteuttamisessa ovat tiedon luokitus ja avainsanojen käyttö. Näistä ensimmäinen, tiedon luokitus, on loppukäyttäjälle yksinkertainen ja helppo. Käyttäjä valitsee annetuista kategoriavaihtoehdoista itseään kiinnostavimmat käyttäen esimerkiksi www-sivun lomakkeita. Valitut kategoriat tallennetaan käyttäjän profiiliin. Tämän jälkeen jokaiseen uuteen palvelun tietokantaan tulevaan kalenterimerkintään liitetään jokin kategoria. Tätä kategoriaa verrataan käyttäjän profiilissa oleviin luokkiin ja merkintä joko suodatetaan pois käyttäjälle synkronoitavista merkinnöistä tai merkitään synkronoitavaksi. Vaihtoehto on varsin helppo käyttäjälle, yksinkertainen toteuttaa ja suodatusprosessi on tehokas ja nopea.

Luokituksen käyttäminen toimii tehokkaasti, jos kategorioita on vain muutama ja ne ovat hyvin spesifisiä. Tehokkaalla tarkoitan tässä kahta asiaa: kuinka nopeasti suodatus tapahtuu sekä kuinka oikeita tuloksia suodatuksella saadaan aikaan. Suunniteltavassa palvelussa on kuitenkin varauduttava siihen, että kalenterimerkintöjä voi tulla miltä aihealueelta tahansa. Jos kategoria on liian yleinen, esimeriksi urheilu, käyttäjän saama kalenterimerkintämäärä on aivan liian suuri. Käyttäjä ei välttämättä myöskään ole kiinnostunut kuin jonkin tyyppisistä urheilutapahtumista. Kategoriat onkin rajattava spesifisemmiksi, esimerkiksi jalkapallo ja jääkiekko, mikä taas johtaa kategorioiden määrän nousuun. Luvun kolme alussa esittelin mahdollisia kalenterimerkintöjä ja mainitsin, että palvelun kannalta ei ole merkitystä mitä kalenterimerkinnät sisältävät. Tämän vuoksi kategoriamäärä voi nousta liian suureksi, eikä käyttäjän voi olettaa valitsevan mielenkiinnon kohteitansa sadan kategorian joukosta.

Avainsanat tuovat pelkkiin kategorioihin verrattuna lisää mahdollisuuksia ja täsmentävät käyttäjän todellista mielenkiinnon kohdetta. Ihmiset käyttävät avainsanoja lähes päivittäin hakiessaan tietoa Internetistä esimerkiksi käyttäen hakukoneita. Palvelussa käyttäjä kuvaa mielenkiinnon kohteitaan avainsanoilla ja annetut avainsanat tallennetaan profiiliin. Avainsanoja, jotka voivat olla myös kokonaisia lauseita, verrataan uusiin kalenterimerkintöihin.

Avainsanoihin perustuvaan malliin saa lisää ulottuvuutta, jos sanoihin liittää lisätietoa, esimerkiksi erilaisia prioriteettiasteita. Prioriteettiasteilla käyttäjä voi määritellä jotkin sanat merkityksellisemmiksi kuin toiset. Toinen tehokkuutta lisäävä tapa on antaa käyttäjälle mahdollisuus lisätä avainsanoihin boolean operaatioita (AND, OR, NOT). Tämä tapa on käytössä myös useissa hakukoneissa. Boolean operaatioilla voidaan kuvata eri sanojen riippuvuutta ja saadaan huomattavasti tarkemmin kuvattua käyttäjän mielenkiinnon kohteet. Käyttäjä voi esimerkiksi määritellä mielenkiinnokseen ”urheilu AND jääkiekko NOT jalkapallo”. Esimerkin käyttäjä olisi kiinnostunut urheilusta yleensä sekä tarkemmin jääkiekosta, muttei jalkapallosta. Näiden lisäksi on mahdollista määritellä muitakin operaattoreita, kuten NEAR, ”villit merkit” * ja ? sekä käyttää sulkuja suhteiden tarkentamiseksi. Ongelmana erilaisissa boolean operaatiossa on kuitenkin niiden hankaluus peruskäyttäjälle. Hyvän määrittelyn tekeminen voi kokeneellekin käyttäjälle osoittautua liian hankalaksi. Pahimmassa tapauksessa se johtaa oleellisen tiedon pois suodatukseen ja vääränlaisen tiedon synkronointiin. Luvussa kolme lisäsin myös vaatimuksen kahdeksan, jonka mukaan palvelu on oltava helposti käyttöönotettavissa. Erilaisten prioriteettien lisääminen ja boolean operaatioiden käyttäminen voi tehostaa suodatusta, mutta hankaloittaa ratkaisevasti profiilin luontia.

Tietopohjaisessa mallissa käyttäjä on aina vastuussa siitä, mitä hänelle synkronoidaan. Käyttäjän mielenkiinnon muuttuessa hänen on itse päivitettävä profiiliaan manuaalisesti, jotta oikeanlaista tietoa pystytään synkronoimaan. Oppivassa tiedonsuodatuksessa profiilin muokkautuminen on agenttien vastuulla.

5.4. Oppiva tiedonsuodatus

Kuten tietopohjaisen mallin, myös oppivan mallin voi toteuttaa usealla eri tavalla. Yksi yleisistä tavoista on pitää yllä tietoa käyttäjän hyväksi havaitsemista tietoalkioista. Käyttäjää voi pyytää arvioimaan löydettyjä tietoalkioita ja käyttäjän hyväksi arvioimat tietoalkiot tallennetaan käyttäjän henkilökohtaiseen profiiliin. Talletettuja merkintöjä verrataan uusiin tietoalkioihin. Tämän palvelun tapauksessa agentit seuraisivat, poistaako käyttäjä synkronoidun kalenterimerkinnän tietyn ajan sisällä vai ei. Säilytetyt kalenterimerkinnät tallennetaan profiiliin (tai tallennetaan pelkkä viittaus haluttuun kalenterimerkintään) ja niitä verrataan uusiin kalenterimerkintöihin. Tietenkin profiiliin voidaan tallentaa myös sellaiset kalenterimerkinnät, jotka

käyttäjä on poistanut ja vertailu tehdään niiden perusteella. Näin etsitään ne merkinnät, joita ei haluta synkronoida käyttäjälle.

Jo edellisen kappaleen perusteella voitaisiin asettaa seuraavia olettamuksia oppivien agenttien toimivuudelle. Maes [1994] määrittelee kaksi perustekijää, joiden on toteuduttava oppivien agenttien kohdalla:

1. sovelluksen (palvelun) käytön on tuotettava kohtuullinen määrä toistuvia toimintoja
2. ja toimintojen on oltava (osittain) erilaisia eri käyttäjien kesken.

Jos jälkimmäinen vaatimus ei toteudu, tietopohjainen toteutus saavuttaisi nopeampia tuloksia (todennäköiset käyttäjän toiminnot olisivat tiedossa jo käytön alkuvaiheessa, jolloin hitaampi oppimisprosessi oli turha). Ensimmäisen vaatimuksen täyttyminen on edellytys agenttien oppimiselle. Jos palvelun käytöstä ei seuraa tarpeeksi informatiivisia toimintoja, agenteilla ei ole tietoa, mistä oppia. Suunniteltavan palvelun kohdalla tämä tarkoittaisi sitä, että käyttäjät eivät koskaan poistaisi synkronoituja merkintöjä tai poistaisivat ne aina. Jälkimmäisen vaatimuksen toteutuminen on palvelussa todennäköistä. Käyttäjien kiinnostuksen kohteet varmasti vaihtelevat riittävästi ja näin erilaisten toimintojen saaminen eri kalenterimerkintöihin on todennäköistä. Ensimmäiseen vaatimukseen liittyy riskejä, koska käyttäjät voivat hyvin hitaasti tarkastaa, mitä kalenteriin on synkronoitu. Personoinnin optimoimiseksi käyttäjiä olisikin hyvä rohkaista heti poistamaan kalenteristaan sellaiset merkinnät, jotka eivät kiinnosta heitä.

Foltz ja Dumais [1992] esittelevät tutkimuksessaan avainsanoihin ja dokumentteihin perustuvat profiilit ja tutkivat niiden tehokkuutta tiedon suodatuksessa. Foltz ja Dumais [1992] päätyvät dokumentteihin perustuvan profiilin paremmuuteen: jo muutama käyttäjän valitsema mielenkiintoinen dokumentti osoittautui yhtä tehokkaaksi kuin pitkä lista avainsanoja. Dokumentteihin perustuva profiili on myös yksinkertaisempi käyttäjän kannalta, sillä käyttäjän ei tarvitse luoda avainsanoja, vaan vain osoittaa, mitkä dokumenteista ovat mielenkiintoisia ja mitkä eivät.

Dokumentteihin perustuvaan profiiliin ja yleisesti oppiviin agentteihin perustuvaan suodatukseen liittyy myös ongelmia. Alkutilanteessa käyttäjän toiminnasta ei ole saatu vielä mitään tietoa ja agenteilla ei siis ole mahdollisuutta uusien kalenterimerkintöjen arvioimiseen. Yksi vaihtoehto on alkutilanteessa synkronoida merkintöjä muutama kerrallaan, jolloin

minkäänlaista tietotulvaa käyttäjän kalenteriin ei pääsisi syntymään. Kun käyttäjältä saadaan tarpeeksi palautetta kalenterimerkinnöistä, voidaan synkronointitiheyttä vähentää ja synkronoida useampi kalenterimerkintä kerrallaan. Toinen vaihtoehto on yhdistää dokumentteihin perustuva profiilin luonti sekä jokin tietopohjainen malli. Tietopohjaista mallia käytetään alkutilanteessa kunnes saadaan tarpeeksi informaatiota käyttäjän toimista. Tämän jälkeen siirrytään oppivaan malliin. Ensimmäiseen vaihtoehtoon verrattuna näin vältytään ns. huonojen merkintöjen synkronoinnilta, ja käyttäjä saa alusta asti vain haluamiaan merkintöjä.

Toinen merkittävä ongelma liittyy käyttäjän mielenkiinnon kohteen muuttumiseen. Tietopohjaisessa mallissa käyttäjä käy tarpeen vaatiessa itse muokkaamassa profiiliaan, joten agenteilla on aina tieto käyttäjälle sopivista kalenterimerkinnöistä. Oppivassa mallissa käyttäjälle synkronoidaan koko ajan vain tarkemmin personoituja kalenterimerkintöjä. Jos käyttäjän mielenkiinnon kohde muuttuu, agenttien on vaikea saada siitä tietoa. Tähän ei varsinaisesti ole mitään automatisoitua ja varmaa keinoa. Yksi mielenkiintoinen ja jatkotutkimuksia kaipaava ratkaisu voisi olla satunnaisesti valittujen ei-kiinnostavien merkintöjen synkronointi käyttäjälle. Kutsun näitä tarkoituksella synkronoitavia ”vääriä” merkintöjä vertausmerkinnöiksi. Käyttäjälle voidaan synkronoida vertausmerkintöjä samalla, kun synkronoidaan varsinaisia suodatettuja merkintöjä. Vertausmerkintöjen pitää olla sellaisista aihealueista, joita käyttäjälle ei muuten synkronoitaisi. Vertausmerkintöjen perusteella voidaan koko ajan tarkastella käyttäjän mielenkiinnon kohteita ja muuttumista. Jos käyttäjä ei tuhoa vertausmerkintöä, tämä lisätään käyttäjän profiiliin. Samalla palvelu saa tietoa käyttäjän mielenkiinnon muuttumisesta ja profiilia saadaan muuttumaan käyttäjän mieltymyksien mukaan. Haittapuolena tietenkin käyttäjä saa kalenteriinsa merkintöjä, joista ei alunperin ollut kiinnostunut.

Yleisesti tekstin vertailussa (avainsanat sekä dokumentit) ongelmana on hyvyysfunktion määrittäminen. Hyvyysfunktio määrittelee milloin vertailun tulos on hyväksyttävä (merkintä synkronoidaan). Esimerkiksi dokumenttien vertailussa hyvyysfunktio kertoo paljonko dokumenteissa pitää olla yhteneväisyyttä, jotta merkintä synkronoidaan. Koska jonkinlaisia heuristisia sääntöjä on pakko käyttää, on rajatapauksissa aina vaarana menettää käyttäjän kannalta mielenkiintoinen kalenterimerkintä tai toisaalta synkronoida käyttäjän ei-toivoma merkintä. Väärän arvion mahdollisuutta voi tietenkin vähentää parantamalla evaluoinnin tekevää operaatiota.

Foltz ja Dumais [1992] testasivat suodatusta tieteellisten artikkeleiden tiivistelmiin. Verrattavat tiedot olivat siis puhdasta tekstiä. Pelkkää tekstiä kutsutaan yleisesti ei-rakenteiseksi (unstructured) tiedoksi [Belkin ja Croft, 1992]. Vastaavasti puolirakenteinen (semistructured) tieto on tietoa, jossa on ennalta määriteltyjä kenttiä sekä kenttiä, jotka sisältävät ei-rakenteista tietoa [Malone et al., 1987]. Kalenterimerkinnät ovat puolirakenteista tietoa, joissa mm. aika ja paikka ovat määriteltyjä kenttiä, kun taas merkinnän kuvaus on ei-rakenteinen tekstikenttä. Ei-rakenteisen tiedon tulkkaamisessa tarvitaan luonnollisen kielen tunnistamista ja jäsentämistä. Verrattuna ei-rakenteiseen tietoon, puolirakenteinen tieto mahdollistaa yksinkertaisemman ja tehokkaamman tavan tiedon jäsentämiseen. vCalendar-formaatilla on ominaisuuksia, kuten merkinnän kategoria, jotka tukevat puolirakenteista lähestymistapaa. Tarkoituksena on siis välttää suoraa tekstin analysointia ja pyrkiä vertaamaan ensin rakenteisten kenttien arvoja (mikä voi olla suora peruste merkinnän suodatukseen) ja tämän jälkeen avoimia kenttiä. Ratkaisuvaihtoehto kalenterimerkintöjen tehokkaalle suodatukselle voisi siis olla seuraava; käyttäjä valitsee yleisestä listasta mielenkiintoaan vastaavat kategoriat, joita käytetään alkutilanteessa, kun käyttäjältä ei vielä ole saatu minkäänlaista palautetta. Tässä tapauksessa kategoriat voivat olla yleisempiä, esimerkiksi musiikki tai urheilu, kuin jos käytettäisiin vain tietopohjaista profiilin luontia. Kun agentti saa tarpeeksi palautetta käyttäjän valitsemista kalenterimerkinnöistä, käytetään dokumentteihin perustuvaa profiilia. Dokumenttien vertailussa verrataan ensin ennalta määriteltyjä kenttiä, kuten merkinnän kategoriaa ja tämän jälkeen analysoidaan merkinnän kuvausta ja otsikkoa sekä mahdollisesti myös aikaa ja paikkaa. Jos vertailun tulos täyttää ehdot, kalenterimerkintä lähetetään käyttäjälle.

Paljon tutkittu ja mielenkiintoinen vaihtoehto on käyttää sosiaalista suodatusta (collaborative filtering tai social filtering). Sosiaalisen suodatuksen tarkoituksena on hyödyntää muiden samantyyppisten käyttäjien profiileja suodatuksessa [Golderberg et al., 1992] [Maes, 1994] [Claypool et al., 1999]. Esimerkkinä voimme kuvata tilanteen, jossa henkilöt A ja B ovat synkronoineet samankaltaisia merkintöjä aikaisemmin. Palvelulla on uusi merkintä, jonka henkilö B on synkronoinut. Nyt henkilön A agentti voi kysyä henkilön B agentilta neuvoa merkinnän suhteen ja näin synkronoida merkinnän myös henkilölle A. Sosiaalinen suodatus parantaa oppivien agenttien tiedon suodatuksen tarkkuutta, kun oman profiilin lisäksi voidaan käyttää hyväksi myös muiden saman tyyppisten ihmisten profiileja. Varsinkin rajatapauksissa,

joissa ei olla aivan varmoja kannattaako kalenterimerkintä synkronoida vai ei, evaluoinnin tulos paranee selvästi. Sosiaalisen synkronoinnin ongelma tähän palveluun liittyen on kalenterimerkintöjen aikasidonnaisuus: merkinnät on synkronoitava mahdollisimman nopeasti kaikille käyttäjille (ennen kuin tapahtuman ajankohta on alkanut). Periaatteessa tämä merkitsee sitä, että kalenterimerkinnät synkronoidaan kaikille käyttäjille lähes samanaikaisesti. Sosiaalisen suodatuksen käyttäminen taas vaatisi palvelulta sitä, että kalenterimerkinnät synkronoitaisiin asiakkaille porrastetusti. Näin agentit pystyisivät pyytämään toisiltaan mahdollisia ”neuvoja”. Sosiaalinen suodatus parantaisi kyllä suodatuksen laatua, mutta palvelun aikasidonnaisuus estää sen mielekkään käytön.

Tekesin tutkimusta [Kiuru, 2000] varten haastatellut suhtautuivat periaatteessa positiivisesti ajatukseen siitä, että palveluntarjoajalta saa automaattisesti personoituja sisältöjä joko käyttäjätietojen perusteella tai omien aikaisempien valintojen ja hakujen pohjalta. Vaikka oppivaan malliin suhtauduttiin positiivisesti, sen toimivuutta epäiltiin. Itse toteutettua (tietopohjaista) personointia pidettiin selvästi parempana vaihtoehtona. Edellä on kuvattu ratkaisuvaihtoehtoja tietopohjaisista ja oppivista malleista sekä niiden yhdistetyistä ratkaisuista ja esitelty niiden hyviä ja huonoja puolia. Palvelun suunnitteluun liitettyjen vaatimusten viisi ja kahdeksan täyttämiseen oppiva malli sopii parhaiten. Lisäksi tietopohjaisen mallin yhdistäminen oppivaan malliin oppivan mallin alkutilanteen ongelman vuoksi parantaa suodatuksen tehokkuutta myös palvelun käyttöönoton alkuvaiheessa. Vaikkakin edellä mainitussa Tekesin tutkimuksessa [Kiuru, 2000] haastatellut pitävät tietopohjaista mallia parempana, on oppivassa mallissa huomattavia etuja tietopohjaiseen verrattuna. Oppiva malli on käyttäjälle vaivattomin vaihtoehto sekä tehokkain varsinkin käyttäjän tarpeiden huomioon ottamisessa. Mitä enemmän käyttäjälle synkronoidaan kalenterimerkintöjä, sitä tarkemmin agentit pystyvät suodattamaan oleellisen tiedon [Maes, 1994]. Lisäksi ehdotetussa yhdistetyssä mallissa käyttäjällä on alkutilanteessa mahdollisuus vaikuttaa mitä tietoa ylipäätään synkronoidaan. Kuitenkin jos palvelun synkronoitavien kalenterimerkintöjen skaala on kovin suppea, voi puhtaan tietopohjaisen personoinnin käyttäminen olla järkevää sen helpomman implementoinnin vuoksi. Lisäksi oppivan mallin lisääminen myöhemmin palveluun on täysin mahdollista. Tietopohjaisessa mallissa on kuitenkin muistettava, että personointi on tällöin täysin käyttäjän vastuulla.

6. Toteutus

Edellisessä luvussa tutkin tutkielman keskeisintä suunnitteluongelmaa, personointia. Tässä luvussa tarkastelen suuruusluokaltaan pienempiä suunnitteluongelmia ja niiden vaihtoehtoisia ratkaisuja sekä pyrin valitsemaan niistä parhaimman ratkaisuvaihtoehdon.

6.1. Agenttien sijainti

Yksi oleellinen tutkimusongelma on agenttien sijainti. Personointi ei vaikuta agenttien sijaintiin, vaan kaikki luvussa viisi esitetyt ratkaisuvaihtoehdot voidaan toteuttaa riippumatta siitä, mihin personoinnin toteuttavat agentit sijoitetaan. Agentit voidaan sijoittaa käytännössä kahteen paikkaan: joko itse päätelaitteeseen tai palvelimelle. Loeb [1992] sekä La Porta et al. [1998] ehdottavat agentin sijoittamista sovelluksen suorituskyvyn vaatimusten mukaan. Loebin [1992] sekä La Portan et al:in [1998] tutkimuksissa siirrettävä tieto jaetaan jatkuvaan tiedonsiirtoon (stream-based) ja tapahtumaperustaiseen tiedonsiirtoon (transaction-based). Jatkuvaa tiedonsiirtoa ovat esimerkiksi video– tai äänilähetykset ja tapahtumaperustaista taas esimerkiksi yksittäinen kalenterin synkronointi. Koska tapahtumaperustainen tiedonsiirto on osittain kertaluontoista (vain tietty määrä tietoa siirretään kerrallaan), sitä käyttävät sovellukset eivät ole kovinkaan kriittisiä suorituskyvyn suhteen. Tällaisille sovelluksille voidaan toteuttaa monimutkaisempia toimintoja, kuten tiedon suodatusta ja profilointia. Jatkuva tiedonsiirto vaatii nopeaa sekä yksinkertaista tiedonkäsittelyä ja tehokasta suorituskykyä hyvän kuvan– tai äänenlaadun takaamiseksi. Päätelaitteen rajalliset ominaisuudet (muistin määrä ja prosessoriteho) sekä tiedonsiirtonopeus (vaatimukset 2 ja 3) vaikuttavat oleellisesti agenttien sijaintiin. Tiedon suodatus prosessina on suhteellisen raskas ja nopeutta vaativa. Tämän päivän päätelaitteista vain PDA-laitteilla sekä kommunikaattorityyppisillä laitteilla on edellytyksiä muistin ja prosessoritehon puolesta raskaampiin operaatioihin. Langattomien verkkojen kehittyessä tiedonsiirtonopeuden sekä verkon kuormittumisen aiheuttamat ongelmat myös pienentyvät, mutta vielä tänä päivänä tiedonsiirtonopeudet ovat oleellinen kriteeri agenttien palvelimelle sijoittamiseen. Agenttien sijoittaminen palvelimelle pienentää tiedonsiirron tarvetta päätelaitteen ja palvelimen välillä murto-osaan. La Porta et al. [1998] antaa seuraavan suosituksen agenttien sijainnille suhteessa tiedonsiirtotapaan:

1. Agentit, jotka eivät käsittele jatkuvaa tiedonsiirtoa, sijoitetaan sopivaan paikkaan verkossa (itse palvelin tai jokin välityspalvelin tms.).

2. Agentit, jotka käsittelevät jatkuvaa tiedonsiirtoa, sijoitetaan mahdollisimman lähelle päätelaitetta.

Palvelun suunnittelun kannalta palvelimelle sijoittaminen helpottaa suunnittelua sekä kehitystä, koska sovellusta ei toteuteta itse päätelaitteeseen. Sovelluksen toteuttaminen päätelaitteeseen voisi rajoittaa myös tuettavien päätelaitteiden määrää.

Agenttien sijoittamisen lisäksi on myös ajateltava agenttien määrää. Jos agentit toteutetaan siten, että jokaista käyttäjää kohti on yksi omassa prosessissaan toimiva agentti, agenttien ja prosessien määrä palvelimella voi kasvaa palvelun kohteesta riippuen liian suureksi. Jos palvelu toimii pienen yrityksen tai ryhmän kalenterisynkronointina, niin palvelin pystyy suorittamaan agenttien toiminnallisuuden riittävän tehokkaasti. Jos palvelua käytetään yleisenä palveluna julkisella sektorilla, agenttien lukumäärä palvelimella voi nousta useisiin tuhansiin. Kun jokainen näistä agenteista suorittaa tiedonsuodatusta, on mahdollista että palvelin ylikuormittuu. Agenttien sijoittaminen päätelaitteeseen vähentäisi tai poistaisi kokonaan palvelimen ylikuormittumisen mahdollisuutta, koska jokaisen henkilökohtainen agentti sijaitsisi käyttäjän päätelaitteessa. Toisaalta agentteja ei ole välttämätöntä toteuttaa yksi asiakasta kohden periaatteella, vaan agentteja voi olla esimerkiksi yksi kymmentä tai sataa asiakasta kohden. Tällä tavoin toteutettuna agenttien tehokkuus kärsisi, mutta toisaalta se vähentäisi palvelimen kuormitusta.

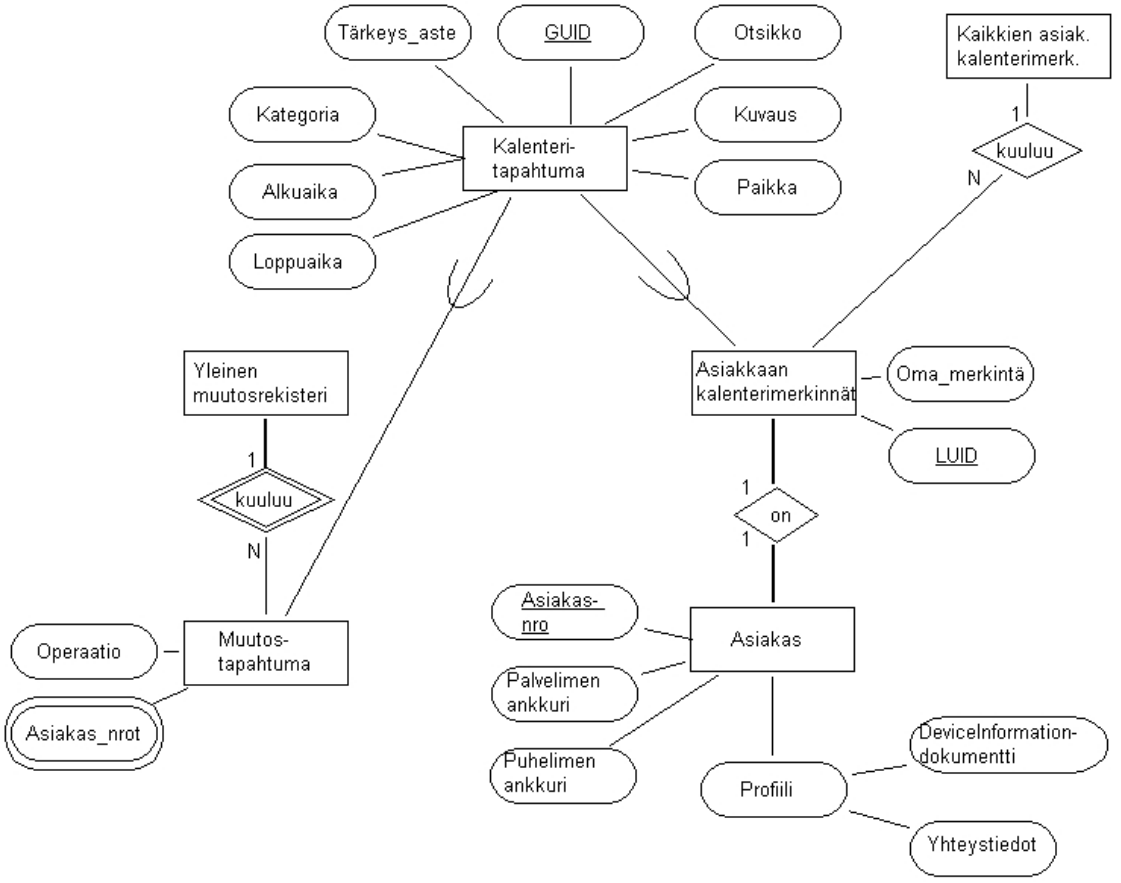
Palvelun toteutuksen kannalta tärkeitä ominaisuuksia ovat toteutuksen riippumattomuus laitevalmistajista ja tiedonsiirron minimointi laitteiden välillä. Päätelaitteiden erilaiset ominaisuudet vaikeuttavat ohjelmointia päätelaiteiden ympäristössä ja lisäävät riippuvuutta valmistajiin. Loebin [1992] sekä La Portan et al:in [1998] tutkimustulosten, palvelun toteutuksen helpottamisen sekä verkon kuormittamisen minimoinnin perusteella agenttien palvelimelle sijoittaminen on tehokkain ratkaisu.

6.2. Tietosisällön kuvaus

Tietokanta on keskeinen osa kaikkia sähköisiä palveluita. Palveluiden on talletettava tietokantaan asiakkaidensa tietoja, mahdollisia profiileja palvelun personointia varten sekä palvelun omaa tietoa. Myös suunniteltavassa palvelussa tietokantaan talletetaan käyttäjien profiilit ja niiden lisäksi käyttäjien henkilökohtaiset merkinnät. Suunniteltavassa palvelussa palvelun omat tiedot

ovat synkronoitavia kalenterimerkintöjä sekä SyncML-protokollan edellyttämiä elementtejä (muutosrekisteri ja tunnistelinkitys).

Riippumatta siitä, mitä tietokantaratkaisua käytetään, itse talletettava tieto ja tietojen suhteet eivät saa muuttua. Kuvassa 6.1 olen kuvannut palvelun tietosisältöä ER-kaavion avulla. Käytän kaaviossa samaa notaatiota kuin Elmasri ja Navathe [2000]. Kuvan on tarkoitus antaa lukijalle kuva siitä mitä tietoa suunniteltavan palvelun tulee sisältää ja mitä relaatioita kohteiden välillä on oltava.



Kuva 6.1 ER-kaavio palvelun tietosisällöstä

Kaaviota luettaessa on syytä huomata, että se on vain ohjeellinen. Kaaviosta on tarkoituksella jätetty pois kohteiden epäoleellisia ominaisuuksia, kuten asiakaskohtaisia tietoja.

Muutosrekisteri ja tunnistelinkitys ovat SyncML-protokollan keskeiset elementit. Kuvassa 6.1 muutosrekisteriä kuvaa Yleinen muutosrekisteri-kohde, joka sisältää muutostapahtumia. Muutostapahtuma-kohteen Operaatio-ominaisuus kuvaa onko tapahtuma lisätty, poistettu vai onko sen tietoja

muokattu. Kun tapahtuman järjestäjä lisää, poistaa tai muokkaa olemassa olevaa tapahtumaa, muutosoperaatio ja tapahtumamerkintä lisätään muutosrekisteriin. Samalla agentit aloittavat suodatuksen käyttäjien profiilien mukaan. Jos agentit päättävät synkronoida tapahtuman käyttäjälle, ne lisäävät kyseisen asiakkaan asiakasnumeron Muutostapahtuma-kohteen moniarvoiseen Asiakas_nrot-ominaisuuteen. Tunnistelinkitys on toteutettu kuvassa 6.1 Asiakkaan kalenterimerkinnät-kohteen avulla. Palvelimenhan on kyettävä linkittämään omat GUID-tunnisteensa päätelaitteen LUID-tunnisteisiin. Asiakkaan kalenterimerkinnät-kohteessa LUID- ja GUID-tunnisteet on tallennettuna omiin ominaisuuksiinsa.

Tärkeimpänä elementtinä tietokannassa ovat synkronoitavat kalenterimerkinnät. Jokaiselle palvelimella olevalle merkinnälle onkin luotava GUID-tunniste, jota siis käytetään merkintöjen tunnistamiseen. Myös vanhentuneita merkintöjä (siis tapahtumia, joiden ajankohta on jo mennyt) on pidettävä tallessa, koska tietopohjaiset agentit käyttävät niitä personoinnissa. Tietenkin agenttien saadessa uusia merkintöjä käyttäjän profiiliin voidaan vanhimpia merkintöjä tarvittaessa poistaa. Käyttäjän henkilökohtaisista merkinnöistä ei tarvitse tallettaa muuta kuin niiden ajankohdat, koska niiden sisällöllä ei ole palvelun toiminnan kannalta merkitystä. Ajankohtia voidaan kuitenkin käyttää päällekkäisyyksien hallintaan. Asiakkaan kalenterimerkintä-kohteen Oma_merkintä-ominaisuus kertoo onko kyseinen merkintä asiakkaan itsensä luoma vai ei.

Koska kohdassa 6.1 päädyin siihen, että agentit sijoitetaan palvelimelle, on luonnollista, että myös käyttäjien profiilit tallennetaan samaan paikkaan. Käyttäjien profiileissa ylläpidetään käyttäjän yhteystietoja ja käyttäjän päätelaitteesta saatavaa Device Information –dokumenttia. Asiakkaan kalenterimerkintä-kohde kuuluu myös tavallaan käyttäjän profiiliin, koska agentit käyttävät juuri asiakkaan laitteessa olevia merkintöjä sisällön personointiin. Jos palvelussa käytettäisiin jotain muuta kuin oppivaa tiedonsuodatusta, esimerkiksi tietopohjaista tiedonsuodatusta, myös suodatukseen tarvittavat säännöt pitäisi tallettaa käyttäjän profiiliin.

Kalenterimerkintöjä lukuun ottamatta kaikki ER-kaavion kohteet ovat varsin yksinkertaisia toteuttaa. Kalenterimerkintöjen kohdalla taas toteutusvaihtoehtoja on enemmän. Merkinnät voidaan tallentaa suoraan vCalendar-formaatissa tai jokainen merkinnän kenttä voidaan tallettaa erikseen omaksi ominaisuudekseen. ER-kaaviossa (kuva 6.1) on käytetty jälkimmäistä

tapaa. Tiedonsuodatuksen kannalta merkinnän hajauttaminen yksittäisiin ominaisuuksiin on tehokkaampaa, koska tällöin halutun kentän arvot saadaan suoraan yhdellä kyselyllä. Jos merkintä talletetaan kokonaisena esimerkiksi vCalendar-muodossa, on tarvittava tieto etsittävä käymällä läpi koko merkintä. Tiedonhaku valmiiksi rakenteellisesta tiedosta on huomattavasti helpompaa ja tehokkaampaa verrattuna tiedonhakuun puhtaasta tekstistä. Haittapuolena hajautetussa vaihtoehdossa on se, että merkintä on ennen synkronointia muutettava joka tapauksessa vCalendar-formaattiin. Nopea tiedonhaku on kuitenkin edellytyksenä nopealle tiedonsuodatukselle ja koska vCalendar-tiedoston generointi on kohtuullisen nopea operaatio, on ominaisuuksien hajauttaminen kannattavampaa.

Edellä selvitin ER-kaavion kohteiden välisiä suhteita sekä sitä, mitä informaatiota tietokantaan tallennetaan. Suurin tietokantoihin liittyvä suunnitteluongelma on kuitenkin käytettävä tietokantaratkaisu. Usein palveluissa käytetään tietokannanhallintajärjestelmiä, kuten DB2, Oracle, PostgreSQL ja MySQL. Tietokannanhallintajärjestelmien hyviä puolia ovat mm. [Bourret, 2002]:

- perusoperaatioiden hallinta (tallennus, haku, päivitys)
- indeksit
- tehokkuus ja hyvä suorituskyky
- tiedon turvaaminen
- tiedon eheys
- samanaikaisuuden hallinta

Edellä mainituista varsinkin samanaikaisuuden hallinta on keskeinen ominaisuus. Suunniteltavan palvelun tapauksessa agentit toimivat tietokannan samanaikaisina käyttäjinä, minkä vuoksi samanaikaisuuden hallinta on tarpeellista. Tietokannan pysyminen eheänä edellyttää juuri samanaikaisuuden hallintaa. Tiedonhaun tehokkuus on myös oleellista agenttien useiden hakujen takia. Selvimpinä tietokannanhallintajärjestelmien huonoina puolina ovat niiden monimutkaisuus, hinta (muutoskustannukset ja laitteistokustannukset), koko ja vaikea siirrettävyys ympäristöstä toiseen.

Helppo siirrettävyys eri ympäristöihin onkin tiedostopohjaisen tiedontallennuksen yksi parhaista ominaisuuksista. Suunniteltavassa palvelussa erilliset tiedostot muodostaisivat käytettävän tietokannan. Esimerkiksi kalenterimerkinnät olisi mahdollista tallentaa yhteen tiedostoon ja

asiakkaat toiseen. Varsinkin XML-pohjainen tiedontallennus on yleistynyt ympäristöriippumattomuutensa sekä joustavuutensa vuoksi. XML käyttää Unicode-koodausta [Unicode, 2002], jonka avulla tieto voidaan esittää eri kirjasimilla mukaan lukien kyrilliset kirjasimet. Kaikki XML-dokumenttien sisältämät merkit ovat oletusarvoisesti Unicode-koodattuja. Unicode-tukensa sekä luettavan ja yksinkertaisen rakenteensa vuoksi XML-pohjainen tiedostokokonaisuus on helppo siirtää ympäristöstä toiseen. Kuten tietokannanhallintajärjestelmät, myös XML tarjoaa valmiit rajapinnat perusoperaatioiden hallintaan. Tällaisia ohjelmointirajapintoja ovat esimerkiksi DOM (Document Object Model) ja SAX (The Simple API for XML). Lisäksi XML on itsensä selittävä (kuvauskieli selittää itsessään tiedon rakenteen sekä tiedon merkityksen) [Bourret, 2002]. XML:stä puuttuu kuitenkin tärkeitä ominaisuuksia, kuten samanaikaisuuden hallinta sekä tiedon turvaaminen ja sen hakurutiinit ovat hitaita tiedon jäsennyksen sekä tekstimuunnosten takia. Nämä puutteet tekevät siitä ongelmallisen suurten palveluiden käytössä. Suunniteltavassa palvelussa XML-ratkaisu toimisikin parhaiten esimerkiksi yrityksen sisäisessä palvelussa, jossa käsiteltävä tietomäärä pysyy kohtuullisen pienenä. Tietokannanhallintajärjestelmä puolustaa edelleen paikkaansa julkisessa palvelussa, jossa sekä asiakas- että tietomäärä on suurempi.

Palvelun mukauttamisen kannalta olisi kuitenkin erittäin tärkeää, että käytettävästä tietokantaratkaisusta huolimatta palvelu olisi helppo siirtää eri kohteisiin. Koska tätä on vaikea saavuttaa yhdellä ratkaisulla, paras vaihtoehto on kapseloida tietokantapalveluiden käyttö omaan loogiseen yksikköönsä, esimerkiksi dynaamiseen kirjastoon, ja implementoida eri toteutukset tietokantajärjestelmälle ja tiedostopohjaiselle ratkaisulle. Tällöin palvelun jakelu eri käyttötarkoituksia varten olisi kiinni ainoastaan yhdestä komponentista sekä tietenkin jaeltavasta tietokantaratkaisusta.

6.3. Tietoturva

Tietoturva on yksi sähköisten palveluiden tärkeimpiä osa-alueita. Langattomien verkkojen ollessa kyseessä tietoturva-asioihin on kiinnitettävä jopa enemmän huomiota, kuin kiinteiden verkkojen kanssa. Mobiileissa päätelaitteissa voi olla eroavaisuuksia esimerkiksi käytettävien salaustapojen ja niiden tasojen kanssa. SyncML pyrkii omalta osaltaan vastaamaan mobiilien laitteiden tiedonsiirtoon liittyviin ongelmiin. Tässä tutkielmassa pyrin tutkimaan palvelun tietoturvaa SyncML-protokollan kannalta. Yleisten tietoturvaan liittyvien ongelmien määrittely jää tutkielman ulkopuolelle.

Aiemmin jaoin palvelun tietoturvaongelmat kahteen osaan: tietoturva profiilien suojauksessa sekä tietoturva tiedonsiirrossa palvelimen ja päätelaitteen välillä. SyncML:n tavoitteena on tarjota puitteet turvalliselle tiedon synkronoinnille. Se ei itsessään määrittele uusia suojausmalleja, vaan sen sijaan mahdollistaa autentikoinnin, autentikointiin haastamisen, hyväksymisen sekä suojatun tiedon lähettämisen SyncML-paketeissa [SyncML Representation, 2002].

Autentikointi eli käyttäjän ja laitteen tunnistaminen on oleellista tietojen suojaamisessa. SyncML tukee molemminpuolista autentikointia, eli sekä palvelin että päätelaite voivat haastaa toisensa autentikointiin. Yleisessä skenaariossa päätelaite autentikoi itsensä palvelimelle ja synkronoi tietonsa sen kanssa. Sama skenaario pätee myös tämän palvelun tapauksessa. Palvelin tarjoaa asiakkaalle tarvittavat käyttäjätunnukset ja salasanat autentikointia varten. Autentikointi tapahtuu SyncML-paketeissa nolla, yksi ja kaksi (ks. kohta 3.4). Koska palvelussa palvelin käynnistää synkronoinnin, se voi haastaa laitteen autentikointiin jo paketissa 0. Tällöin päätelaite palauttaa paketissa 1 autentikointitiedot, ja jos palvelin tunnistaa laitteen, synkronointia jatketaan normaalisti. Jos autentikointi epäonnistuu, palvelin vastaa paketissa 2 epäonnistumista vastaavilla virhekoodeilla ja tunnistamista yritetään uudelleen päätelaitteen toimesta lähettämällä paketti 1 uudestaan. Palvelin vastaanottaa uudet autentikointitiedot ja tulkitsee ne. Tarvittaessa pyyntö voidaan vielä uusia, mutta synkronointi voidaan myös virhetilanteessa keskeyttää.

Epäsuoraa autentikointia tapahtuu myös automaattisesti, kun palvelin ottaa yhteyden laitteeseen puhelinnumeron avulla, joka lähes varmasti ohjaa yhteyden oikeaan laitteeseen. Tällöin palvelin on siis jo osittain autentikoinut laitetta. Itse asiassa puhelinnumero yhdistetään laitteessa olevaan sim-korttiin, kun taas laitteen autentikointitiedot ovat laitekohtaisia. Tämä johtaa siihen, että käyttäjän on päätelaitetta vaihtaessaan ilmoitettava palveluntarjoajalle laitteen vaihdosta. Tällöin palvelu tietää autentikoidessaan laitetta, että autentikointitiedot ovat vaihtuneet.

Yksi laitteen autentikointimenetelmä perustuu laitteen yksilölliselle laitenumerolle. Palvelin saa tämän tiedon päätelaitteen lähettämästä Device Info- objektista. Palvelin voi tunnisteen perusteella analysoida, onko laite oikea. Käyttäjän ja laitteen autentikointi voidaan jakaa siis kolmeen osaan: käyttäjätunnukseen ja salasanaan, sim-kortin puhelinnumeroon sekä laitteen

laitenumeroon. Näiden avulla voidaan sekä laite että käyttäjä tunnistaa ja varmistua siitä, että palvelin synkronoi oikeaa tietoa oikean laitteen kanssa.

Vaikka autentikoinnin perusteella laite ja käyttäjä voidaan tunnistaa ja tiedot synkronoida oikeaan laitteeseen, tieto on aina siirrettäessä suojattava mahdollisen tiedonkaappauksen varalta. Eri suojausmenetelmiä on useita ja tärkeää on, että päätelaite ja palvelin ymmärtävät toistensa salauksen. SyncML-protokollan mukaisesti laitteiden on tuettava ainakin Basic- [Freed ja Bronstein, 1996] ja MD5 Digest [Rivest, 1992]- menetelmiä autentikointia varten. Tällä pyritään varmistumaan siitä, että autentikointiin tarkoitettavat tunnukset ovat turvassa. Muitakin menetelmiä on mahdollista käyttää, mutta tällöin molempien synkronoivien laitteiden on tuettava niitä. Käytettävä suojausalgoritmi sovitaan autentikoinnin yhteydessä. Muu siirrettävä tieto, eli varsinaiset SyncML-paketit, voidaan myös salata esimerkiksi WAP-protokollan toimesta. Tämän tutkielman puitteissa ei kuitenkaan tutkita tiedon salausta yksityiskohtaisesti, joten näitä menetelmiä ei käsitellä tämän enempää.

Tietoturvan rinnalle voidaan nostaa kysymys palvelun eettisistä ongelmista. Palveluntarjoaja synkronoi käyttäjälle tarkoin profiloitua tietoa, mitä voitaisiin myös kutsua täsmämarkkinoinniksi. Niin kauan kun palveluntarjoajan motiivit ovat oikeat, oleellista ja vain oleellista tietoa synkronoidaan käyttäjälle. Jos motiivit ovat väärät, käyttäjä voi saada vääränlaista informaatiota [Kendall ja Kendall, 1999]. Voidaan myös miettiä, onko käyttäjän mielipiteen muodostuminen palvelun aikaansaannosta ja onko tämä eettisesti oikein? Tämä voisi olla yksi mielenkiintoinen jatkotutkimuksen aihe.

7. Palvelun toiminta SyncML:n avulla

Aiemmissa luvuissa olen miettinyt yksittäisten suunnitteluongelmien parhaita ratkaisuvaihtoehtoja. Tässä luvussa kuvaan palvelun näiden ratkaisujen pohjalta ja lisään palvelun kulkuun SyncML:n käytön. Tarkoituksena on kuvata palvelu niin yksityiskohtaisesti, että lukijalle muodostuu selvä kuva palvelun toteutusmahdollisuuksista. Käsittelen aluksi synkronoinnin vaihe vaiheelta ja lopuksi kuvaan arkkitehtuurin, johon olen tutkielmassani päätynyt.

Palvelun vaiheet uusien kalenterimerkintöjen tietokantaan lisäämisestä merkintöjen synkronointiin voidaan jakaa synkronoinnin alustukseen, synkronointiin sekä tietoalkioiden linkitykseen. Siis samoihin vaiheisiin, joihin SyncML-protokollan mukainen synkronointi jaettiin kohdassa 3.4. Palvelun vaiheisiin lisätään neljäs vaihe, jossa tapahtuu itse personointi. Kutsun tätä vaihetta esivaiheeksi.

7.1. Esivaihe

Esivaihe sisältää kaikki toiminnot, jotka tapahtuvat palvelimella ennen varsinaista synkronointia. Siihen kuuluvat siis uusien kalenterimerkintöjen lisäys palvelun tietokantaan, vanhojen merkintöjen muokkaus, esimerkiksi tapahtumapaikan tai tapahtuman päivämäärän vaihtaminen, ja tiedon suodatus, eli personointi.

Kalenterimerkintöjen lisäystavalla ei varsinaisesti ole merkitystä palvelun toimintaan, mutta sitäkin on syytä miettiä ja etsiä siihen erilaisia toteutusvaihtoehtoja. Palveluntarjoajan kannalta paras vaihtoehto on automatisoida merkintöjen lisäys ja siirtää vastuu itse tapahtumien järjestäjille. Tämä voisi tapahtua esimerkiksi www-lomakkeen avulla, johon vain sopimuksen omaavilla yhteistyökumppaneilla olisi oikeudet. Tapahtuman järjestäjä täyttäisi kaikki tarpeelliset tiedot tapahtumasta, kuten tapahtumapaikan ja -ajan, mahdolliset liitetiedostot tai niiden sijainnin verkossa, hintatiedot, mahdolliset muut www-osoitteet sekä tapahtuman kuvauksen. Tämän jälkeen merkintä lisättäisiin palvelun tietokantaan. Muokkausta varten järjestäjällä on oltava mahdollisuus hakea omia merkintöjään ja muokata niiden sisältöä. Sekä lisäyksistä että muokkauksista merkitään tieto palvelimen omaan muutosrekisteriin. Merkinnän poistoa ei olisi syytä toteuttaa, koska tämä voisi aiheuttaa sekaannusta siitä, onko tapahtunut jonkinlainen virhe palvelun toiminnassa vai onko tapahtuma

oikeasti peruttu. Parempi vaihtoehto olisi muokata tapahtumaa esimerkiksi kuvauksen osalta ja lisätä tapahtumatietoihin merkintä peruuntumisesta. Tällöin merkintää ei poisteta käyttäjältä, ja hän saa varmemmin tiedon tapahtuman muutoksista.

Palvelimen muutosrekisterin muokkaus käynnistää palvelun seuraavan ja tärkeimmän osan eli personoinnin. Agentit lukevat muutosrekisteristä lisätyt merkinnät ja jokaisen käyttäjäkohtaisen profiilin. Suodatus tapahtuu profiileihin tallennettujen tietojen perusteella. Laitteen ominaisuuksien, siis SyncML-protokollan Device Info -dokumentin mukaan tehtävän mukauttamisen, voi periaatteessa suorittaa kahdessa eri vaiheessa. Suositeltava vaihtoehto on suorittaa se esivaiheessa muun personoinnin yhteydessä. Tällöin Device Info -dokumentti siirretään vain kerran laitteiden ensimmäisen synkronointikerran yhteydessä. SyncML-protokollan mukaan Device Info-dokumentin lähetystä kaikkien synkronointikertojen välillä tulisi välttää, koska se on kohtalaisen iso objekti [SyncML Protocol, 2002]. Agentit siis suorittavat laitekohtaisen mukauttamisen tallennetun Device Info -dokumentin mukaan. Toinen vaihtoehto on siirtää Device Info -dokumentti protokollan suositusten vastaisesti joka kerta, jolloin laitekohtainen mukauttaminen olisi mahdollista suorittaa synkronointivaiheessa juuri ennen kalenterimerkintöjen lähettämistä. Etuna tästä olisi se, että jos laitteen ominaisuudet muuttuisivat, palvelin personoisi aina uusimpien tietojen mukaan. Ensimmäisessä vaihtoehdossa laitteen ominaisuuksien muuttuminen vaikuttaisi vasta seuraavaan synkronointikertaan. Tämä pätee myös ensimmäiseen synkronointikertaan, jolloin palvelimella ei ole vielä mitään tietoa laitteesta. Tämän vuoksi palvelimen olisikin syytä ensimmäisellä kerralla lähettää vCalendar-objektit mahdollisimman pelkistettyinä, jotta turhilta yhteensopivuusongelmilta säästyttäisiin.

Personoinnin tulokset, eli agenttien käyttäjille suodattamat kalenterimerkinnät, lisätään käyttäjäkohtaiseen muutosrekisteriin, ja samalla niille luodaan käyttäjäkohtaisesti oma GUID-tunniste. Tunniste lisätään tämän jälkeen ID-linkitykseen. On huomattava, että personoitua merkintää ei voi vielä käyttää muiden merkintöjen personointiin. Tämä siitä syystä, että käyttäjän reaktio (poistaako vai jättääkö käyttäjä merkinnän kalenteriinsa) personoituun ja synkronoituun merkintään saadaan vasta seuraavalla synkronointikerralla.

Palvelimella olevien merkintöjen muokkaukset on käsiteltävä poikkeuksellisesti. On otettava huomioon, että jos palvelimella olevaa

merkintää muokataan, esimerkiksi sen ajankohtaa tai paikkaa vaihdetaan, agenttien on otettava huomioon, onko alkuperäistä merkintää synkronoitu käyttäjälle. Jos merkintä on synkronoitu ja käyttäjä ei ole poistanut merkintää päätelaitteestaan, uusi muokattu merkintä on automaattisesti synkronoitava, jotta asiakas saa uudistuneet tiedot tapahtumasta. Tällöin toimitaan siis täysin SyncML-protokollan mukaisesti eli kaikki muutokset välitetään käyttäjälle. Toisaalta jos merkintää ei ole synkronoitu (alkuperäinen on siis suodatettu pois) tai käyttäjä on poistanut sen kalenteristaan, palvelimella muokattua tai poistettua merkintää ei synkronoida.

Toinen personoinnin käynnistävä tekijä on uusien paikannustietojen saanti gps-paikannuksen avulla. Jos käyttäjä vaihtaa paikkakuntaa, esimerkiksi Tampereelta Helsinkiin, voidaan tämän käyttäjän kohdalla käynnistää henkilökohtainen personointi. Tällöin suodatuksessa otetaan painottavana tekijänä huomioon henkilön paikkakunnan vaihdos ja palvelun tietokanta tutkitaan sopivien merkintöjen varalta. Jos tietokannasta löytyy Helsinkiin sijoittuvia tapahtumia, voidaan ne suodattaa käyttäjälle.

7.2. Synkronoinnin alustus

Synkronoinnin käynnistämisen toteutustapoja on useita. Yksi vaihtoehto on käynnistää synkronointi heti esivaiheen jälkeen, jolloin merkinnät synkronoidaan käyttäjälle heti, kun tapahtumanjärjestäjä on ne tietokantaan lisännyt. Tässä vaihtoehdossa etuna on se, että mitään erityistä ajastusta ei tarvita, vaan aloittaminen on yksinkertaista ja merkinnät saadaan luotettavasti ja nopeasti käyttäjille. Haittapuolena on, että synkronointikertojen lukumäärä voi nousta liian suureksi. Parempi vaihtoehto on hoitaa synkronointi esimerkiksi kerran kahdessa tai kolmessa päivässä, jolloin kaikki tällä aikavälillä tehdyt muutokset synkronoidaan käyttäjille. Etuna on synkronointikertojen rajoitettu ja hallittu määrä. Toisaalta jos synkronoitavaa ei ole, synkronointi voidaan jättää väliin. Lisäksi synkronointi voidaan kytkeä sille määriteltyyn aikaan. Jos merkintä lisätään palveluun myöhässä, eli sille määritetty aika on esimerkiksi kolmen päivän sisällä, se voidaan synkronoida heti. Muita tällaisia poikkeustilanteita olisivat gps-paikannuksen raportoimat käyttäjän siirtymiset. Gps-paikannuksen aktivoimissa tapauksissa synkronointi on tärkeää suorittaa heti, kun henkilö on ollut päivän jossakin muualla kuin kotipaikkakunnassaan.

Synkronoinnin alustusvaihe suoritetaan kuten kohdassa 3.4 määriteltiin. Synkronointi aloitetaan protokollan mukaisesti paketilla nolla ja palvelin pyytää asiakasta kaksisuuntaiseen synkronointiin. Synkronointi jatkuu asiakkaan ja palvelimen alustuspaketeilla (kuva 3.5) ja tarvittaessa asiakas voi lähettää oman Device Info- objektinsa. Jos palvelimella on uudelle objektille tarvetta eikä asiakas sitä ole lähettänyt, se voidaan tässä vaiheessa pyytää. Tällöin objekti saapuu ensimmäisessä synkronointipaketissa.

Aiemmin määritellyllä tavalla myös laitteen ja käyttäjän autentikointi hoidetaan synkronoinnin alustusvaiheessa. Autentikointi on kuvattu tietoturvaa käsittelevässä kohdassa (kohta 6.3). Onnistuneen autentikoinnin jälkeen siirrytään synkronointivaiheeseen.

7.3. Synkronointi ja linkitys

Synkronointivaihe alkaa asiakkaan muutoksien vastaanottamisella. Jos käyttäjä on lisännyt uusia merkintöjä tai tehnyt muutoksia (muokannut tai poistanut) omiin tai palvelun synkronoimiin merkintöihin päätelaitteessaan, ne synkronoidaan palvelimelle. Mahdolliset muutokset voivat siis olla kalenterimerkinnän lisäys, muokkaus tai poisto. Tarkasti SyncML-protokollaa noudattaen kaikille vastaanotetuille muutoksille suoritettaisiin sync-analyysi. Suunniteltavassa palvelussa voidaan kuitenkin hiukan protokollan vastaisesti käsitellä palvelimen muutokset kahdella eri tavalla:

- Kaikille käyttäjän muokkaamille (muokatuille tai poistetuille) palvelun merkinnöille suoritetaan sync-analyysi.
- Käyttäjän omille henkilökohtaisille merkinnöille (lisäys, muokkaus tai poisto) ei suoriteta sync-analyysiä. Palvelin luo uusille merkinnöille GUID-tunnisteen ja tallettaa sen ja saadun LUID-tunnisteen ID-linkitykseen. Merkinnöistä talletetaan vain tapahtuma-aika. Poistot tapahtuvat lähetetyn LUID-tunnisteen avulla, ja palvelin poistaa merkinnän ja vastaavat viittaukset ID-linkityksestä (GUID ja LUID).

Jälkimmäinen on seurausta siitä, ettei palvelulla ole tarvetta tietää käyttäjän omista henkilökohtaisista merkinnöistä muuta kuin niiden aikatiedot. Lisäksi palvelin ei missään tapauksessa muokkaa näitä merkintöjä. Näistä syistä kaikki käyttäjän omiin merkintöihin liittyvät operaatiot suoritetaan aina ilman tarvetta suorittaa erillistä sync-analyysiä. Ensimmäinen tapaus voidaan jakaa kahteen alatapaukseen: käyttäjä muokkaa palvelun merkintää tai poistaa tällaisen. Näille siis suoritetaan tapauskohtainen sync-analyysi.

Sync-analyysissä tutkitaan konfliktien mahdollisuus. Kuten kohdassa 3.2 esitettiin, konflikti voi ilmetä, kun synkronointikertojen välillä sekä päätelaitteessa että palvelimella on muokattu samaa tietoalkiota. Tässä palvelussa konfliktien hallinta eroaa hiukan normaalista, koska ainoastaan käyttäjän lisäykset ja poistot ovat palvelulle olennaisia. Konfliktien ratkomisessa noudatetaan seuraavia sääntöjä:

1. Jos palvelimen merkintää muokataan, se on päivitettävä käyttäjälle uusien päivitettyjen tietojen perille saamiseksi.
2. Toisaalta, jos sekä päätelaitteessa että palvelimella on muokattu samaa merkintää, palvelimen on voitettava konflikti.
3. Kuitenkin, jos käyttäjä on aiheuttanut konfliktin poistamalla merkinnän kalenteristaan (ja siihen on tullut siis palvelimella muutoksia), noudatetaan esivaihetta käsitelleessä kohdassa esiteltyä tapaa eli merkintää ei synkronoida uudelleen.

Säännöt siis ylimäärittävät toisiaan, eli ensimmäistä sääntöä käytetään aina palvelimen muokkauksissa. Toisaalta palvelimen on aina voitettava konfliktit. Kolmas sääntö kuitenkin ylimäärittelee aiemmat säännöt ja synkronointia ei siis suoriteta, jos käyttäjä on jo poistanut merkinnän aiemmin omasta kalenteristaan. Säännöstä kaksi seuraa myös se, että käyttäjän muutokset (käyttäjä voi lisätä merkintöihin esimerkiksi omia kommenttejaan) konfliktitilanteissa palvelun aiemmin synkronoimiin merkintöihin menetetään. Käyttäjän henkilökohtaisiin merkintöihin tällä ei ole vaikutusta.

Toinen mahdollinen konfliktityyppi, joka ei ole aivan SyncML-protokollan mukainen, ilmenee, kun palvelun kalenterimerkinnän ajankohta ja asiakkaan oma yksityinen merkintä ovat päällekkäisiä. Koska käyttäjän omista kalenterimerkinnöistä talletetaan palvelimelle merkintöjen tapahtuma-aika, saadaan tällaiset konfliktit selville. Jos käyttäjän merkintää ei ole vielä synkronoitu palvelimelle, se synkronoidaan viimeistään siis ennen kuin palvelimelta lähdetään siirtämään mitään päätelaitteeseen. Tällaisessa konfliktitapauksessa asiakkaan merkintää ei tietenkään voida poistaa ja palvelun merkintä on jätettävä synkronoimatta. Palvelun kannalta on kuitenkin hyvä, että useat eri kalenteriohjelmat tukevat päällekkäisiä merkintöjä siten, että ne näkyvät rinnakkaisina menoina (esimerkiksi Outlook 98) tai listana (useat matkapuhelimet ja PDA-laitteet). Koska tämän tyyppisestä konfliktista ei lopulta ole haittaa, se voidaan joko jättää huomioimatta tai jättää käyttäjän

päätettäväksi mitä näissä tapauksissa tehdään. Käyttäjä voisi tällöin asettaa profiiliinsa, synkronoidaanko konfliktisia merkintöjä.

Palvelun toiminnan kannalta oleellisimpia merkintöjä ovat synkronoidut merkinnät, jotka käyttäjä on poistanut. Jos käyttäjä poistaa merkinnän kalenteristaan, pitää se poistaa myös käyttäjän profiilista. Agentit siis olettavat, ettei käyttäjä ole kiinnostunut kyseisestä tapahtumasta. Samalla poistetaan myös kaikki viittaukset poistettuun merkintään ID-linkityksestä. Toisaalta, jos käyttäjä ei ole poistanut edellisellä synkronointikerralla siirrettyä merkintää, agentit olettavat, että käyttäjä on kiinnostunut tapahtumasta. Tällöin se voidaan lisätä profiiliin ja sitä käytetään seuraavassa personoinnissa.

Synkronointi jatkuu palvelimen muutosten lähettämisellä asiakkaalle. Palvelin lukee muutosrekisteristä agentin suodattamat merkinnät ja lähettää ne asiakkaalle niille luotujen GUID-tunnisteiden kanssa. On oleellista huomata, ettei muutoksia poisteta muutosrekisteristä, ennen kuin niiden onnistuneelle lisäykselle päätelaitteessa saadaan varmistus. Tämä tapahtuu tietoalkioiden linkitysvaiheen ensimmäisessä paketissa.

Linkitysvaiheen alussa päätelaite lähettää kuittauksen edellä lähetettyjen merkintöjen onnistuneesta tallettamisesta. Jos tallennus on onnistunut, palvelin voi poistaa muutosmerkinnät muutosrekisteristä. Jos lisäys epäonnistuu, merkintä jää muutosrekisteriin ja sitä yritetään seuraavalla synkronointikerralla uudestaan muiden lisäysten ohella. Itse linkitysvaihe toteutetaan tämän jälkeen täysin SyncML-protokollaa noudattaen. Päätelaite luo uuden GUID-tunnisteen kaikille uusille palvelimen lähettämille merkinnöille ja lähettää sen linkitysviestissä palvelimen lähettämän GUID-tunnisteen kanssa. Palvelin päivittää LUID-tunnisteen oikeaan taulukkoon tietokannassa GUID-tunnisteen avulla. Lopuksi palvelin kuittaa viimeisessä paketissa vastaanottamansa linkitystiedot.

Kun päätelaite kuittaa merkinnät saaduksi ja palauttaa niille LUID-tunnisteen, palvelin poistaa synkronoidut merkinnät muutosrekisteristä ja lisää LUID-tunnisteet ID-linkitykseen. Näin ID-linkityksessä on siis palvelimen antamat GUID-tunnisteet, näitä vastaavat päätelaitteen luomat uudet LUID-tunnisteet ja kalenterimerkinnän todellinen sijainti tietokannassa. Tämä suoritetaan kaikille asiakkaille erikseen.

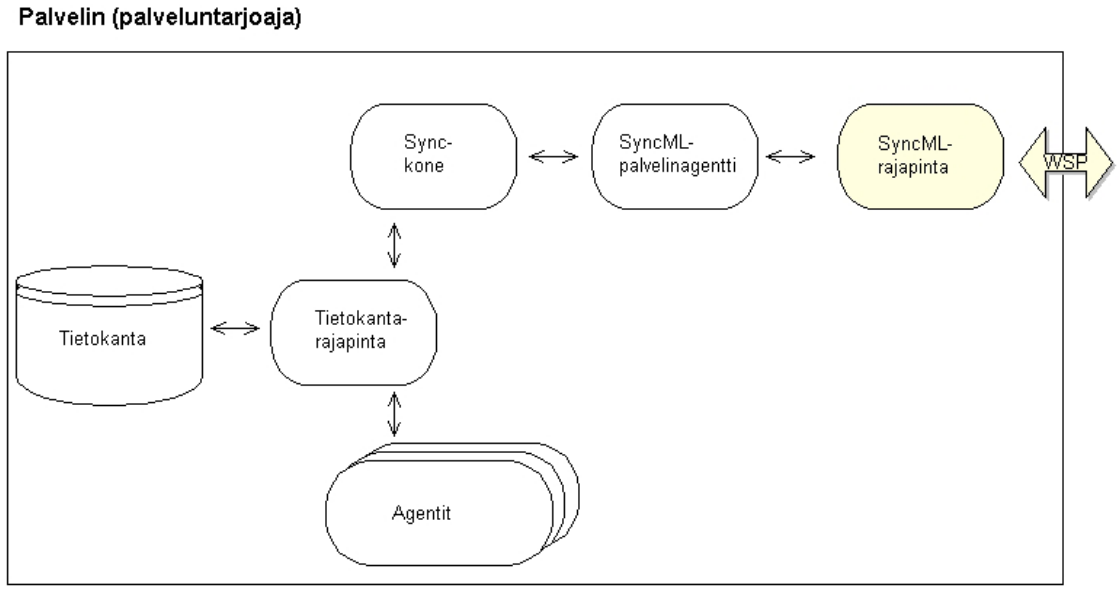
7.4. Vaihtoehtoiset toteutukset

Kuten kohdassa 3.5 SyncML:n eduissa ja huonoissa puolissa mainitsin, on täysin mahdollista, että laitevalmistajat tukevat protokollaa vain osittain. Suunnitellun toteutuksen kannalta on ongelmallista jos päätelaite ei tue palvelimen aloittamaa synkronointia. Vaihtoehtoinen toteutustapa olisi toteuttaa käyttäjän aloittama synkronointi: käyttäjä itse käynnistäisi synkronoinnin palvelun kanssa päätelaitteestaan ja agenttien valmiiksi suodattamat kalenterimerkinnät synkronoitaisiin käyttäjälle. Palvelimen puolella tämä ei juurikaan aiheuttaisi muutospaineita, koska toimintatapa jatkuu paketin nolla jälkeen käytännössä samalla tavalla, mutta päätelaitteeseen pitäisi toteuttaa synkronoinnin aloitus. Tämä voisi olla esimerkiksi palveluntarjoajan pieni sovellus (esim. Java-sovellus), jonka käyttäjä aina halutessaan aktivoisi. Yhä useammat laitteet tukevat Java-sovelluksia, jolloin myös kolmas osapuoli, eli tässä tapauksessa palveluntarjoaja, pystyisi lisäämään omia sovelluksiaan käyttäjän laitteeseen.

Ongelmana tässä toteutuksessa olisi, ettei käyttäjä voi tietää milloin uusia tapahtumia on lisätty palveluun. Palvelu on suunniteltu tämän vuoksi nimenomaan push-palveluksi. Pahimmassa tapauksessa uudet merkinnät voisivat kokonaan jäädä synkronoimatta, jos käyttäjä synkronoisi palvelimen kanssa vain harvoin (tapahtuma on jo mennyt). Nykyisiä palveluita mukaillen on tietenkin mahdollista lähettää synkronoinnista käyttäjälle muistutus aina tarpeen tullen, esimerkiksi SMS-viestillä. Palvelimen aloittama synkronointi on kuitenkin yksi keskeisiä SyncML-protokollan ominaisuuksia, joten on todennäköistä, että SyncML:ää tukevat laitteet tulevaisuudessa tukevat tätä ominaisuutta. Mahdolliseen ominaisuuden puuttumiseen on kuitenkin hyvä varautua.

7.5. Lopullinen arkkitehtuuri

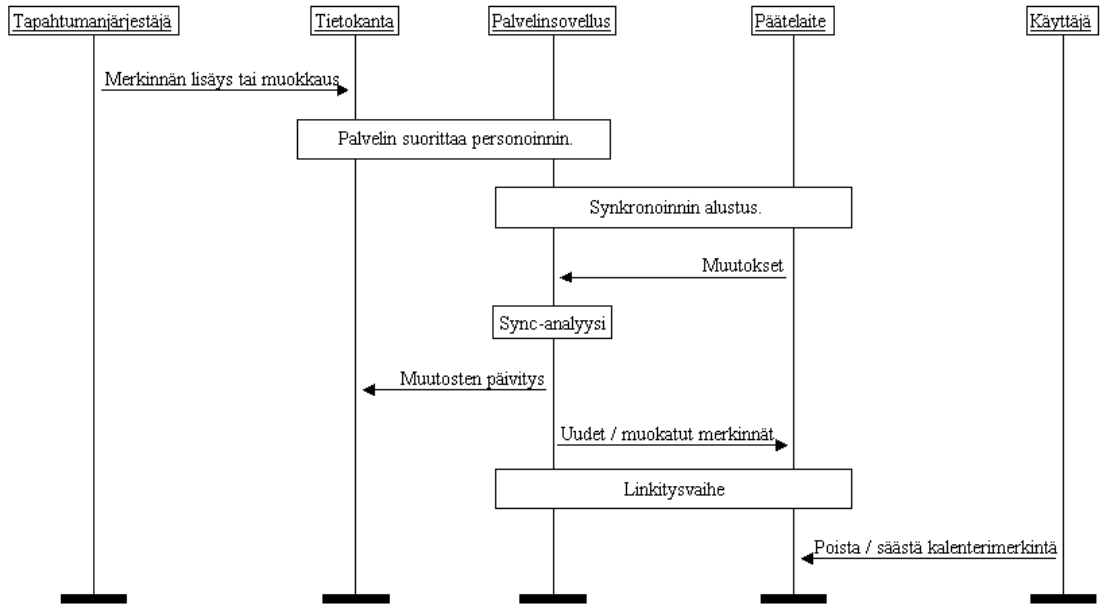
Luvussa neljä esittelin Kendallin ja Kendallin [1999] yleisen mallin push-palvelulle. Tässä mallissa ei otettu kantaa agenttien sijaintiin tai profiilien sijaintiin palvelussa. Mallista puuttuivat myös luonnollisesti SyncML-protokollan mukaiset elementit. Kuvassa 7.1 olen tarkentanut palvelimen arkkitehtuuria tutkielmassa esiteltyjen ratkaisujen mukaan.



Kuva 7.1 Lopullinen arkkitehtuuri

SyncML-palvelinagentti kuvaa yhteyttä ylläpitävää komponenttia. Komponentti käyttää SyncML-protokollan tarjoamia rajapintoja varsinaiseen tiedonsiirtoon sekä SyncML-viestien kasaamiseen. Mahdolliset konfliktit selvitetään Sync-koneessa samoin kuin SyncML-operaatioiden (lisäys, poisto) suorittaminen. Sync-kone siis käyttää tietokantarajapinta-komponenttien tarjoamia tietokantapalveluja oikeiden merkintöjen lähettämiseen ja poistamiseen. Kuten tietokantaa käsittelevässä kohdassa selvitin, tietokantarajapinta-komponentti voi kapseloida käytettävän tietokannan, oli se sitten XML-pohjainen tai jokin muu ratkaisu. Agentit toimivat itsenäisinä komponentteinaan ja hoitavat personoinnin tietokannasta saamiensa tietojen perusteella. Itse tietokanta sisältää käyttäjien profiilit sekä kalenterimerkinnät, palvelun kalenterimerkinnät, muutosrekisterit ja ID-linkitykset.

Samoin luvun neljä skenaariota voidaan myös tarkentaa selvitettyjen ominaisuuksien osalta. Kuvassa 7.2 on kuvattu palvelun toimintamalli esiteltyjen ratkaisujen pohjalta. Skenaarion vaiheet on kuvattu yksityiskohtaisesti tutkielmani aiemmissa kohdissa. Yleisellä tasolla palvelun toiminta voidaan kiteyttää seuraavasti: Tapahtumanjärjestäjä käynnistää agenttien personoinnin lisäämällä tai muokkaamalla palvelun tietokannassa olevia kalenterimerkintöjä. Palvelinsovellus käynnistää synkronoinnin ja suorittaa sync-analyysin vastaanotetuille käyttäjän muokkaamille merkinnöille. Muutokset päivitetään palvelun tietokantaan, jonka jälkeen lisätyt tai muokatut merkinnät lähetetään käyttäjän päätelaitteeseen. Lopuksi suoritetaan vielä linkitysvaihe, jonka jälkeen käyttäjä voi toimia laitteellaan normaalisti.



Kuva 7.2 Lopullinen toimintaskenaario

Arkkitehtuuria (kuva 7.1) sekä skenaariota (kuva 7.2) voidaan pitää palvelun jatkosuunnittelun lähtökohtana. Lisäksi aiempien lukujen päätelmiä ja lopputuloksia voidaan käyttää hyväksi eri osa-alueiden tarkemmassa suunnittelussa. Niin skenaarioista kuin myös arkkitehtuurista on hyvä kuitenkin huomata, että eri osakokonaisuudet ovat varsin irrallisia toimintayksiköitä, ja ne voidaan toteuttaa myös tutkielmassa esitetyistä ratkaisuista eriävällä tavalla. Esimerkiksi agenttien toiminta voidaan toteuttaa usealla eri tavalla sen vaikuttamatta palvelun yleiseen toimintaperiaatteeseen millään lailla. Tällä tavoin saavutetaan myös palvelun helpompi mukauttaminen eri ympäristöihin ja erilaisiin tarpeisiin.

8. Yhteenveto

Tutkielmassa esiteltiin uusia teknologioita käyttävä mobiilipalvelu kalenterimerkintöjen synkronointiin ja suunniteltiin palvelulle yleinen arkkitehtuuri ja toimintamalli. Samalla esiteltiin mobiilin palvelun suunnitteluun ja toteutukseen liittyviä ongelmia sekä etsittiin ratkaisuja näihin perusongelmiin. Seuraavaksi arvioin alustavasti kehittämääni mallia ja esittelen siihen liittyviä rajoituksia. Lopuksi esittelen mahdollisia jatkotutkimusaiheita.

8.1. Palvelun arviointi ja rajoitukset

Luvun neljä vaatimuksista tärkeimpiä olivat palvelun mukauttaminen eri sisällöille sekä palvelun mukauttaminen päätelaitteiden mukaan. Lisäksi oleellisia vaatimuksia olivat mobiilipalvelua koskevat tekniset suunnitteluperiaatteet. Näitä olivat varautuminen useisiin yhteyskatkoihin, rajalliseen tiedonsiirtonopeuteen ja rajallisiin laitteen ominaisuuksiin.

Sisällön mukauttamisen ratkaisemiseksi tutkin personointia. Päädyin oppivan tiedonsuodatuksen sekä tietopohjaisen tiedonsuodatuksen yhdistelmään. Pääpainona toteutuksessa on kuitenkin oppiva tiedonsuodatus, koska tietopohjaista suodatusta käytetään vain alkutilanteen ongelmien takia. Agenttien käyttäminen suodatuksessa varsinkin oppivina agentteina voi ainakin periaatteessa ratkaista tietotulvaongelman, joka tällaista palvelua saattaa vaivata. Suurimman ongelman esittelemäni malli sekä personointi yleensäkin kohtaa käyttäjien mielipiteiden muuttuessa. Oppivassa tiedonsuodatuksessa Tekesin [Kiuru, 2000] tutkimuksessa haastatellut näkivät uhkana toimivuuden arkielämän käytön tasolla: ”lokeroiko” automaattinen personointi informaatiosta käyttäjälle vain osan ja onko tämä informaatio senhetkisten käyttötarpeiden mukaista. Tässä mielessä haastateltavat tiedostivat tarpeidensa muuttuvuuden ja niiden ennustettavuuden vaikeuden hyvin. Kyseiseen ongelmaan kuvasin ratkaisun, jossa käyttäjälle synkronoidaan jokaisella kerralla satunnaisesti valittuja vertausmerkintöjä muista aihealueista kuin itse personoidut merkinnät. Käyttäjän reaktiot juuri näitä merkintöjä kohtaan ovat erityisen kiinnostavia, koska niiden avulla palvelun on siis mahdollisuus havaita muutoksia käyttäjän mielenkiinnonkohteissa. Tekesin [Kiuru, 2000] tutkimuksessa haastateltujen kokema uhka on kuitenkin todellinen, sillä vertausmerkinnät eivät muuta profiilia välttämättä kovin nopeasti. Käyttäjien mielipiteet taas voivat vaihtua hyvinkin nopeasti, jolloin vaarana on juuri vääränlaisen informaation poissuodattaminen.

Palvelussa voidaan ottaa huomioon myös mahdolliset kielivaihtoehdot sekä kulttuurilliset erot. Näiden edellytyksenä on tietenkin se, että palveluun syötetään oikeanlaista informaatiota, jota voidaan näidenkin kriteereiden mukaan suodattaa. Erityisesti kieli– ja kulttuurierojen käsittely mahdollistaa palvelun siirrettävyyden muihin maihin mahdollisimman pienillä muutoksilla. Lisäksi gps-paikannus mahdollistaa jopa paikkakuntakohtaisten kalenterimerkintöjen synkronoinnin.

SyncML-protokollan ominaisuudet mahdollistavat palvelun helpohkon mukauttamisen laitteen ominaisuuksien mukaan. Synkronoivat laitteet vaihtavat laitekohtaisen Device Info- objektin, joka sisältää esimerkiksi tietoa laitteen muistimäärästä sekä laitteen tukemista kalenterin kentistä. Tämä tarkoittaa sitä, että palvelu ja agentit voivat varsin tehokkaasti mukauttaa sisältöä laitteen ominaisuuksien mukaan.

SyncML tarjoaa ratkaisuja myös mobiilipalveluita vaivaaviin ongelmiin, kuten yhteyskatkoihin sekä rajalliseen tiedonsiirtonopeuteen. Yhteyskatkot johtavat siihen, että käynnissä olevat operaatiot on pystyttävä suorittamaan loppuun yhteyden jälleen muodostuessa. Muutosrekisteri toimii suorittamattomien operaatioiden jonona, josta operaatiot poistetaan vasta kohdelaitteen kuitattua ne onnistuneeksi. Tämä tarkoittaa sitä, että yhteyskatkon keskeyttämä operaatio suoritetaan uudestaan seuraavalla synkronointikerralla, eikä tietoja näin huku. Agenttien sijainti palvelimella vähentää siirrettävän tiedon määrää jo oleellisesti, mutta lisäksi binäärisen XML:n avulla siirrettävä tieto voidaan koodata pienempään tilaan. Näiden ominaisuuksien avulla suunniteltu malli pyrkii vastaamaan myös mobiilipalvelun perussuunnitteluongelmiin.

Jos tarkastellaan mukauttamisesta seuraavia vaatimuksia, voidaan todeta seuraavaa: kieleen, kulttuuriin, paikannukseen sekä laitteen ominaisuuksiin sidottu personointi on selvästi tietopohjaista suodatusta. Näiden osalta malli toimii vähintäänkin kohtalaisesti. Suurimmat haasteet kohdataan selvästi oppivassa tiedonsuodatuksessa ja tämä on tiedostettava myös palvelua toteutettaessa. Suunniteltu oppivan agentin malli pystyy kuitenkin varsin tehokkaasti vastaamaan mobiilille palvelulle asetettuihin vaatimuksiin. Varsinkin laitteen ominaisuuksien huomioon ottaminen parantaa palvelun käytettävyyttä erilaisissa laitteissa. Lisäksi esitelty ehdotus käyttää vertausmerkintöjä mahdollistaa käyttäjien mielenkiinnonkohteiden muuttumisen havaitsemisen.

Jos verrataan palvelulle asetettuja vaatimuksia ja lopullisen mallin ominaisuuksia, voi todeta, että malli on tavoitteiden osalta kohtuullisen onnistunut. Kuvattu toimintamalli sekä yleinen arkkitehtuuri antavat lähtökohdan mobiilipalvelun kehittämiselle ja samalla useita keskeisiä suunnitteluongelmia on jo ratkaistu. Tutkielmassa esiteltyjä ratkaisuja sekä kehitettyä mallia voidaan käyttää myös muiden saman tyyppisten palveluiden kehittelyssä.

Oppivien agenttien toimivuuden suhteen voi tietenkin aina esittää jonkin verran kritiikkiä, koska niiden tehokkuus ja toimivuus on kiinni monesta tekijästä, esimerkiksi käyttäjän toiminnoista, synkronoitavista merkinnöistä sekä loppujen lopuksi käytettävistä algoritmeista. Tutkielmassa käytettäviä personointimekanismeja ei ole pyritty ratkaisemaan algoritmitasolla, vaan ne on kuvattu yleisellä tasolla. Niiden toiminnallisuuteen on siis hyvä suhtautua varauksella.

Palvelun toteutuksen kannalta on lisäksi syytä huomata, että useat mallissa käytettävät tekniikat ja protokollat sekä esittelemäni laitevaatimukset ovat vasta laitevalmistajilla käyttöönottovaiheessa. Tämän vuoksi varsinaista toteutusta ei ole mahdollista vielä tehdä eikä palvelun toimivuutta voida todentaa. Lisäksi vaarana on laitteiden osittainen tuki palvelussa käytettäville keskeisille protokollille, mikä voi vaikeuttaa lopullisen palvelun käyttöä eri valmistajien laitteissa.

8.2. Jatkotutkimusaiheet

Palveluun liittyy avoimeksi jätettyjä suunnitteluongelmia, jotka sopisivat paremminkin loppukäyttäjiä tutkivan tieteenalan tutkimusaiheiksi. Tämän tyyppisissä palveluissa on pyrittävä ymmärtämään käyttäjän tarpeita ja ohjattava palvelun toimintaa tämän mukaan. Esimerkiksi synkronoitavien merkintöjen määrä on asia, joka vaikuttaa oleellisesti käyttäjien suhtautumiseen suunniteltuun palveluun. Toinen mielenkiintoinen kysymys, jonka esitin jo tietoturvaa käsittelevässä kohdassa, on palvelun eettiset ongelmat. Palveluntarjoaja synkronoi käyttäjälle tarkoin profiloitua tietoa. Tämän perusteella voidaan kysyä, onko käyttäjän mielipiteen muodostuminen tällaisissa palveluissa palvelun aikaansaannosta ja onko tämä eettisesti oikein.

Tutkielman keskeisimmät tekniset ongelmakohdat liittyvät agentteihin. Suunniteltu personointi kokonaisuudessaan, sen tehokkuus ja toimivuus, olisi mahdollista todentaa empiiristen kokeiden avulla. Vertausmerkintöjen käytön tutkiminen ja jatkokehitys olisi lisäksi palvelun kehittämisen kannalta keskeistä. Esimerkiksi vertausmerkintöjen tehokkuutta voisi yrittää lisätä asettamalla niille suurempaa painoarvoa personoinnissa. Empiirisillä kokeilla olisi mahdollista selvittää, kuinka nopeasti profiili seuraa käyttäjän mielenkiinnon kohteiden muutosta erilaisilla asetuksilla ja kuinka paljon vertausmerkintöjä on synkronoitava suhteessa ”oikeisiin” merkintöihin.

Viiteluettelo

[Antikainen et al., 1998] Hannele Antikainen, Kaisa Kostiainen ja Caj Södergård, New content for mobile terminals, November 13, 1998. Saatavana: *http://www.vtt.fi/tte/projects/MobNews/Mobnews.pdf* (Tarkastettu 07.02.2002)

[Autere et al., 2001] Sirpa Autere, Juha Kjalm, Kari Lehtinen, Vesa Luokkanen, Lasse Pajunen, Wolfgang Milszus, Alessandro Rigallo, Tore Syversen, Bjorn Thorstensen, Tron Walseth, Luis Corteasao and Carlos Amaro, The Third Dimensio – Human-centred approach to designing new mobile services for different terminal equipment: Technologies for adaptive mobile service development – use cases and technology survey, December 2001. Saatavana: *http://www.eurescom.de/public/projectresults/P1100-series/1119D1.asp* (Tarkastettu 06.05.2002)

[Belkin ja Croft, 1992] Nicholas Belkin and W. Bruce Croft, Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM* **35**, 12 (Dec. 1992), 29-38.

[Bourret, 2002] Ronald Bourret, XML and databases. Saatavana: *http://www.rpbourret.com/xml/XMLAndDatabases.htm#isxmladatabase* (Tarkastettu 05.07.2002)

[Chen ja Suda, 1997] Larry T. Chen ja Tatsuya Suda, Designing Distributed Object Systems for Mobile Computing. *IEEE Communications Magazine* **35**, 2 (Feb. 1997).

[Cingil, 2000] Ibrahim Cingil, A broader approach to personalization. *Commun. ACM* **43**, 8 (Aug. 2000), 136-141.

[Claypool et al., 1999] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes and Matthew Sartin, Combining content-based and collaborative filters in an online newspaper. In: *Proc. of Customised Information Delivery Workshop* (Berkeley, California, June 1999), 19-27.

[Elmasri ja Navathe, 2000] Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of database systems*. Addison-Wesley, 2000.

[Foltz ja Dumais, 1992] Peter Foltz and Susan Dumais, Personalized information delivery: An analysis of information filtering methods. *Commun. ACM* **35**, 12 (Dec. 1992), 51-60.

[Freed ja Bronstein, 1996] N. Freed ja N. Bronstein, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. (November 1996) Saatavana: *http://www.ietf.org/rfc/rfc2045.txt* (Tarkastettu 04.08.2002)

[Golderberg et al., 1992] Davin Goldber, David Nichols, Brian M. Oki and Douglas Terry, Using collaborative filtering to weave an information Tapestry. *Commun. ACM* **35**, 12 (Dec. 1992), 61-70.

[Järvinen ja Järvinen, 2000] Pertti Järvinen ja Annikki Järvinen, *Tutkimustyön metodeista*. Opinpaja Oy, Tampere, 2000.

[Kendall ja Kendall, 1999] *Information Delivery Systems: An Exploration of Web, Pull and Push Technologies.* Commun. of AIS Volume 1, Article 14, 1999.

[Kiuru, 2000] Jussi Kiuru, WAP palvelujen käyttäminen ja kokeminen. *Tekes – Kohti yksilöllistä mediamaisemaa. Kuluttajatutkimukset-hanke, teknologiakatsaus 98* (2000).

[Käpylä et al., 1998] Tuula Käpylä, Isto Niemi ja Aarno Lehtola, Towards an Accessible Web by Applying PUSH Technology. In: *Proc. of 4th ERCIM Workshop on User Interface for AII* (Stockholm, Oct. 1998), 133-147.

[La Porta et al., 1998] Thomas La Porta, Ramachandran Ramjee, Thomas Woo and Kirshan K. Sabnani – Experiences with network-based user agents for mobile applications. *Mobile Networks and Applications* **3** (1998), 123-141.

[Loeb, 1992] Shoshana Loeb, Architecting personalized delivery of multimedia information. *Commun. ACM* **35**, 12 (Dec. 1992), 39-48.

[Maes, 1994] Pattie Maes, Agents that reduce work and information overload. *Commun. ACM* **37**, 7 (July 1994), 31-40.

[Maes, 1995] Pattie Maes, Artificial Life meets Entertainment: Interacting with Lifelike Autonomous Agents. *Special Issue on New Horizons of Commercial and Industrial AI* **38**, 11 (November 1995), 108-114.

[Malone et al., 1987] Thomas Malone, Kenneth Grant, Kum-Yew Lai, Ramana Rao and David Rosenblitt, Semistructured messages are surprisingly useful for computer-supported coordination. *ACM Trans. Off. Syst.* **5**, 2 (April 1987), 115-131.

[Mabley, 1999] Kevin Mabley, Privacy vs. Personalization, a delicate balance (1999). Saatavana: *http://www.cyberdialogue.com/library* (Tarkastettu 01.09.2002)

[Mabley, 2000] Kevin Mabley, Privacy vs. Personalization (2000). Saatavana: *http://www.cyberdialogue.com/library* (Tarkastettu 01.09.2002)

[Nwana, 1996] Hyacinth S. Nwana, Software Agents: An Overview. *Knowledge Engineering Review* **11**, 3 (Sept 1996), 1-40.

[Rivest, 1992] R. Rivest, The MD5 Message-Digest Algorithm. (April 1992) Saatavana: *http:// www.ietf.org/rfc/rfc1321.txt* (Tarkastettu 04.08.2002)

[Stemberger, 2001] Scott Stemberger, An introduction to SyncML. (October 2001) Saatavana: *http://www-106.ibm.com/developerworks/library/wi-syncml/* (Tarkastettu 22.07.2002)

[SyncML Device Information, 2002] SyncML Device Informatio DTD, version 1.1, 2002. Saatavana *http://www.syncml.org/download* (Tarkastettu 19.06.2002)

[SyncML Meta-Information, 2002] SyncML Meta-Informatio DTD, version 1.1, 2002. Saatavana *http://www.syncml.org/download* (Tarkastettu 27.08.2002)

[SyncML WSP Binding, 2002] SyncML WSP Binding, version 1.1, 2002. Saatavana: *http://www.syncml.org/download (*Tarkastettu 19.06.2002)

[SyncML Discussions, 2002] SyncML Protocol Development Discussions (04.08.2002), *http://groups.yahoo.com/groups/syncml* (Tarkastettu 04.08.2002)

[SyncML Representation, 2002] SyncML Representation Protocol, Data Synchronization Usage, version 1.1, 2002. Saatavana: *http://www.syncml.org/download* (Tarkastettu 27.08.2002)

[SyncML Protocol, 2002] SyncML Sync Protocol, version 1.1, 2002. Saatavana: *http://www.syncml.org/download* (Tarkastettu 19.06.2002)

[SyncML White Paper, 2001] SyncML White Paper, version 1.0, 2001. Saatavana: *http://www.syncml.org/download* (Tarkastettu 27.08.2002)

[Unicode, 2002] Unicode 3.2.0, 2002. Saatavana: *http://www.unicode.org* (Tarkastettu 15.09.2002)

[VCAL, 1996] vCalendar, The Electronic Calendaring And Scheduling Exchange Format version 1.0, 1996. Saatavana: *http://www.imc.org/pdi* (Tarkastettu 24.02.2002)

[VCAL, 1997] vCalendar White Paper: The Personal Calendaring and Scheduling Exchange Format, 1997. Saatavana: *http://www.imc.org/pdi* (Tarkastettu 24.02.2002)

[WAP, 2001] WAP Push Architectural Overview, version 03-Jul-2001, 2001. Saatavana: *http://www.wapforum.org* (Tarkastettu 02.05.2002)

[WBXML, 2001] Binary XML Content Format Specification version 1.3 25 July 2001, Saatavana: *http://www.wapforum.org* (Tarkastettu 31.08.2002)

[XML] Extensible MarkUp Language. Saatavana: *http://www.w3c.org* (Tarkastettu 22.07.2002)