SOAP: XML-PERUSTAINEN VIESTINVÄLITYS- JA ETÄKUTSUPROTOKOLLA

Petri Makkonen

20.11.2003

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

TIIVISTELMÄ

Järjestelmien yhteiskäytettävyydellä säästetään kustannuksia, koska jo toteutettuja toiminnallisuuksia ei tarvitse ohjelmoida uudelleen. Yhteiskäytettävyyttä voidaan hyödyntää organisaatioiden sisäisten ja organisaatioiden välisten tietojärjestelmien yhteensovittamisessa. Erilaisissa ympäristöissä toteutettujen järjestelmien yhteiskäytettävyys on mahdollista ainoastaan silloin, kun järjestelmät kykenevät kommunikoimaan toistensa kanssa yhtenäisellä tavalla tai kommunikoinnissa käytetään apuna rajapintoja, joiden avulla muokataan lähettäjän viesti vastaanottajan ymmärtämään muotoon. XML-perustainen SOAP-protokolla tarjoaa laajalti hyväksytyn yhtenäisen käytännön, jonka avulla järjestelmät voivat kommunikoida suoraan toistensa kanssa Internetin välityksellä. Tässä tutkielmassa perehdytään SOAP-protokollaan SOAP-määritysten ja muun kirjallisuuden pohjalta.

ACM-luokat: (ACM Computing Classification System, 1998 version): C.2, H.4.3
Avainsanat: SOAP, Internet, XML

SISÄLLYSLUETTELO

1 JOHDANTO

Hajautetun tietojenkäsittelyn mallissa järjestelmien ohjelmakomponentteja voidaan jakaa eri koneille tai järjestelmät voivat käyttää omien lisäksi myös toisten järjestelmien ohjelmakomponentteja. Eri koneilla sijaitsevien ohjelmakomponenttien yhteistoimintaa varten tarvitaan tietoliikenneyhteyksien lisäksi menetelmä ohjelmakomponenttien väliseen kommunikointiin. Hajautettujen ohjelmien keskinäistä yhteistoimintaa varten on kehitetty menetelmä, jota kutsutaan proseduurien etäkutsuksi (Remote Procedure Call, RPC). Proseduurien etäkutsut mahdollistavat ohjelman käyttävän toisella koneella sijaitsevia ohjelmakomponentteja ikään kuin ne sijaitsisivat samassa koneessa (Shiva & Virmani, 1993).

Proseduurien etäkutsujen suorittamista varten kehitettiin lukuisia eri tekniikoita, joista osa saavutti vahvan aseman ohjelmien kehittäjien keskuudessa. Usein nämä tekniikat olivat tiukasti sidottuja omiin kehitysympäristöihinsä ja niitä tukeviin ohjelmointikieliin. Käytännöllisesti katsoen ainoastaan samoilla ohjelmointikielillä toteutetut ohjelmat pystyivät toimimaan suoraan toistensa kanssa. Tosin myös erilaisilla kielillä toteutettujen järjestelmien keskinäinen kommunikointi oli mahdollista, mutta silloin vaadittiin rajapintoihin tehtäviä konfigurointeja ja dataan kohdistuvia konvertointeja. Näiden konfigurointien ja konvertointien tekeminen oli usein varsin hankala ja aikaa vievä ratkaisu ja niistä aiheutui huomattavia lisäkustannuksia järjestelmien integrointien yhteydessä (Chester, 2001; Winer, 2001).

Yhtenäisen ja yleisesti hyväksytyn protokollan määritteleminen ei ole yksinkertainen tehtävä. Sen edellytyksenä on käyttäjiltä saatava tuki, joka taas useimmiten edellyttää protokollan olevan sekä avoin (ts. kaikkien vapaasti hyödynnettävissä) että alusta- ja ohjelmointiympäristöriippumaton (Govindaraju & al., 2000). Alusta- ja ohjelmointiympäristöriippumattomuus saavutetaan parhaiten siten, että määriteltävää protokollaa ei sidota mihinkään yksittäiseen oliomalliin ja siinä on käytettävissä yhtenäinen tiedon esittämismuoto. Näiden edellä mainittujen ominaisuuksien lisäksi protokollan yksinkertaisuuden tiedetään nopeuttavan sen leviämistä, koska se madaltaa käyttöönottokynnystä järjestelmien kehittäjien keskuudessa.

XML-perustaiset viestinvälitys- ja proseduurien etäkutsuprotokollat tarjoavat ratkaisun moniin ongelmiin, joita eri järjestelmien keskinäinen yhteensopimattomuus on aiheuttanut. XML tarjoaa standardoidun metakielen, jota voidaan hyödyntää rakenteisen tiedon esittämisessä.

Lisäksi XML:n ominaisuuksiin kuuluu, että se kykenee kulkemaan verkossa HTTP-protokollan yhteydessä. Tämän ominaisuuden ansiosta XML-perustainen protokolla kykenee tarvittaessa läpäisemään tietoverkkojen liikennettä valvovat palomuurit, jotka ovat aiheuttaneet suuria ongelmia monille muille proseduurien etäkutsuista huolehtiville mekanismeille (Jepsen, 2001).

Tässä tutkielmassa käsitellään SOAP-protokollaa[1]. SOAP on yksinkertainen ja kevyt XML-perustainen protokolla, joka on suunniteltu rakenteisen ja tyypitetyn tiedon siirtoa ja proseduurien etäkutsujen suorittamista varten keskittämättömissä ja hajautetuissa ympäristöissä (W3C, 2003b). Pohjimmiltaan SOAP on yksisuuntainen viesti, mutta mm. proseduurien etäkutsujen suorittamista varten sille on määritelty menetelmä vastausviestin muodostamiseksi ja lähettämiseksi. Tämä menetelmä tekee SOAP:ista kaksisuuntaisen (Elfwing & al., 2002).

<table>
<tr><td colspan="2">Sovellusohjelmisto</td></tr>
<tr><td>SOAP</td><td>Kehitysympäristö</td></tr>
<tr><td colspan="2">XML</td></tr>
<tr><td colspan="2">HTTP, SMTP, FTP, …</td></tr>
<tr><td colspan="2">TCP/IP</td></tr>
</table>

Kuva 1. SOAP:in toimintaympäristö.

Kuvassa 1 on kuvattu SOAP:in toimintaympäristö. Sovellusohjelmiston ohjelmakomponentit, jotka voisivat olla kirjoitetut esimerkiksi Java-kielellä, vastaavat viestin muodostamisesta sekä lähettämiseen ja vastaanottamiseen liittyvistä kontrolleista. Kukin ohjelmakomponentti voi käyttää kehitysympäristön[2] tarjoamia luokkia ja kirjastoja SOAP-viestin jäsentämiseen ja XML-muotoisen datan käsittelyyn. XML-muodossa oleva data siirretään SOAP-protokollaa ja jotakin tiedonsiirto-protokollaa (yleensä HTTP) hyödyntäen ohjelmistokomponentilta toiselle verkon välityksellä. Kuljetuskerroksena käytetään TCP/IP-protokollaa.

Luvussa 2 käsitellään SOAP-protokollan kehityksen lähtökohdat. Luvussa perehdytään nykyisin SOAP:in kehityksestä vastaavaan W3C-konsortioon, XML-kieleen ja SOAP:in edeltä-

[1] SOAP on alkujaan lyhenne sanoista Simple Object Access Protocol.
[2] Tunnettuja kehitysympäristöjä ovat esimerkiksi JAX-RPC, Axis ja SOAP::Lite.

jään XML-RPC:hen. Luvussa käydään lyhyesti läpi myös proseduurien etäkutsut ja rajapinnat.

Luvussa 3 perehdytään SOAP-protokollaan yleisellä tasolla. Luvussa käydään läpi SOAP-viestin rakenne ja sisältö SOAP:in määritysten pohjalta ja SOAP-viestin viestinvälityksessä käytettävät rinnakkaisprotokollat HTTP ja SMTP. Luvussa käsitellään myös SOAP-protokollan kilpailijat CORBA ja Java-RMI sekä yksi keskeisimmistä uusista standardoiduista tekniikoista, web-palveluiden malli, johon SOAP on kiinnitetty järjestelmien väliseksi viestinvälitysprotokollaksi. Tarkasteltavista teknologioista on sovellettu uusimpia versioita[3]. Kohdassa 3.9 tarkastellaan SOAP-kehitysympäristöjä ja kohdassa 3.10 kirjallisuudessa esitettyjen tutkimustulosten perusteella SOAP:in suorituskykyä. Lopuksi luvussa 4 tehdään yhteenveto tämän tutkielman sisällöstä.

[3] W3C julkaisi kesäkuussa 2003 SOAP-protokollasta version 1.2 (W3C, 2003b). IETF julkaisi HTTP 1.1-protokollan tammikuussa 1997 (Berners-Lee & al., 1997).

2 LÄHTÖKOHDAT

Tässä luvussa perehdytään XML-perustaisten etäkutsuprotokollien taustaan. Nykyisin SOAP-protokollan kehityksestä vastaa maailmanlaajuinen W3C-konsortio, jonka tausta ja toiminta on esitelty kohdassa 2.1. Kohdissa 2.2 ja 2.3 käydään lyhyesti läpi proseduurien etäkutsut ja rajapinnat. Kohdassa 2.4 käsitellään XML:n keskeisimmät ominaisuudet, sekä ne syyt, miksi XML-perustaisuuden katsotaan antavan protokollalle ympäristöriippumattomuuden ja standardinomaisen aseman. Kohdassa 2.5 perehdytään SOAP:in edeltäjään XML-RPC:hen ja käydään läpi XML-RPC:n keskeisimmät ominaisuudet, heikkoudet ja vahvuudet. Kohdassa esitetään myös esimerkin muodossa XML-RPC -kutsu ja -vastaus, sekä käydään läpi XML-RPC:ssä käytössä olevat tietotyypit.

2.1 W3C-konsortio

W3C-konsortio (World Wide Web Consortium) on voittoa tavoittelematon kansainvälinen organisaatio. Sen tavoitteena on yhdenmukaistaa webiin liittyviä teknologioita ja sitä kautta parantaa eri tahojen kehittämien sovellusten ja järjestelmien keskinäistä vuorovaikutusta. Lisäksi tavoitteisiin kuuluu rohkaista eri puolilla maailmaa toimivia ihmisiä keskustelemaan Internetiin liittyvien tekniikoiden ja menetelmien tulevaisuudesta ja kehittämisestä (W3C, 2003d).

2.1.1 Jäsenistö

W3C-konsortio perustettiin lokakuussa vuonna 1994. Sen perustivat MIT/LCS:ssä työskentelevä Tim Berners-Lee yhteistyössä CERN:in kanssa. Perustamista tukivat myös DARPA ja Euroopan komissio. Huhtikuussa 1995 W3C-konsortioon liittyi ensimmäisenä Eurooppalaisena organisaationa ranskalainen INRIA ja seuraavana vuonna 1996 ensimmäisenä Aasialaisena organisaationa Keion yliopisto (W3C, 2003d).

Suurin osa W3C:n kehitystyöstä tapahtuu nykyisin juuri MIT:ssä, Keion yliopistossa ja INRIA:ssa. Kaiken kaikkiaan W3C:hen kuuluu noin 500 jäsenorganisaatiota eri puolilta maailmaa. Mukana olevia web-teknologiaa hyödyntäviä kaupallisia yrityksiä ovat mm. Sun Mi-

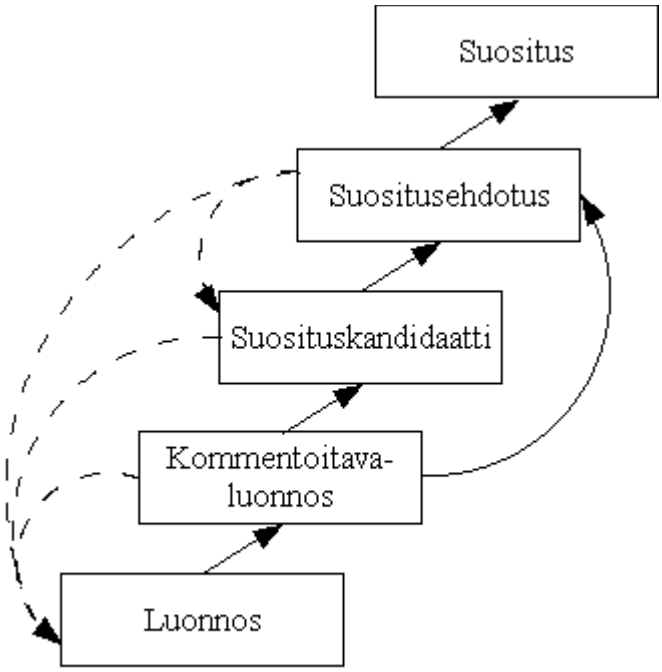
crosystems, IBM ja Microsoft. W3C:tä voidaankin pitää Internetin osalta tapahtuvan kehityksen kattojärjestönä siihen kuuluvien jäsenorganisaatioiden aseman ja laajuuden perusteella (W3C, 2003d).

2.1.2 Suositukset

W3C:n keskeisiin periaatteisiin kuuluu, että sen julkaisemat tekniset raportit ovat vapaasti kaikkien hyödynnettävissä. Teknisiä raportteja on kahden tyyppisiä - muistioita ja eri statuksella olevia suosituksia. *Muistio* (Note) on julkinen raportti, jolla ei ole virallista W3C-konsortion hyväksyntää ja jota W3C:n jäsenet eivät ole velvoitettuja tukemaan. W3C:n käytännön mukaan ennen suosituksen julkistamista teknisen raportin tulee käydä läpi suosituspolun eri vaiheet. Tällä menettelyllä varmistetaan suositukseksi asti edenneen teknisen raportin huolellinen ja systemaattinen käsittely sekä annetaan asiasta kiinnostuneille tahoille mahdollisuus vaikuttaa suosituksen sisältöön. Kommentteja ja ehdotuksia suosituksen sisältöön voivat antaa myös W3C-konsortioon kuulumattomat henkilöt ja organisaatiot. Suosituksen työstämistä varten kootaan työryhmä, jolle nimetään ohjaaja (W3C, 2003d).

Suosituspolku sisältää viisi eri vaihetta. Kunkin vaiheen tuloksena syntyvässä raportissa tulee olla merkintä vaiheeseen liittyvästä statuksesta. Ensimmäisen vaiheen status on *luonnos* (Working Draft), joka sisältää yleisluontoisen esityksen raportin sisällöstä ja prosessin jatkamisesta. Tässä vaiheessa teknisellä raportilla ei tarvitse olla kaikkien W3C:n jäsenten hyväksyntää takanaan. Kun suositusta työstävä työryhmä on saavuttanut tavoitteensa, niin teknisen raportin status muutetaan nimikkeelle kommentoitava luonnos (Last Call Working Draft). Kommentoitava luonnos on julkinen tekninen raportti, jolle raportin laatinut työryhmä pyytää palautetta (W3C, 2003d).

Kun *kommentoitava luonnos* -statuksella olevalle tekniselle raportille on saatu kerätyksi palautetta, niin palautteen perusteella ohjaaja tekee päätöksen raportin statuksen muuttamiseksi suosituspolussa joko eteen- tai taaksepäin. Mikäli raportin ei katsota täyttävän vaatimuksia tai siihen täytyy tehdä suuria muutoksia, niin raportin status voidaan muuttaa takaisin luonnokseksi. Muussa tapauksessa raportin status muutetaan suosituspolussa ylöspäin joko statukselle suosituskandidaatti (candidate recommendation) tai statukselle suositusehdotus (proposed recommendation), jolloin yksi välivaihe jätetään väliin (W3C, 2003d).



Kuva 2. Suosituspolun vaiheet ja mahdolliset siirtymät (W3C, 2003d).

Kuvassa 2 on esitelty W3C:n teknisten raporttien suosituspolku luonnoksesta suositukseksi. Kuvassa olevat kaariviivat kuvaavat teknisen raportin statuksen mahdollisia muutoksia siirryttäessä vaiheesta toiseen. Yhtenäiset kaariviivat kuvaavat vaiheen hyväksymisen jälkeisiä etenemisvaihtoehtoja ja katkoviivat puolestaan vaiheen hylkäämisen jälkeisiä etenemisvaihtoehtoja.

Kun tekninen raportti on statuksella *suosituskandidaatti*, niin tänä aikana työryhmä kerää palautetta raportin määritysten mukaan tehdyistä toteutuksista ja käyttäjien kokemuksista. Tarvittaessa raporttia muokataan saadun palautteen perusteella. Tämän vaiheen päättyessä ohjaaja tekee päätöksen raportin statuksen muuttamisesta. Mikäli raporttia ei ole tarvetta palauttaa suosituspolussa taaksepäin, niin raportti muutetaan statukselle *suositusehdotus,* joka on varsinaista suositusta edeltävä status, ja lähetetään W3C:n neuvoa-antavan komiteaan tarkistettavaksi. Tämän vaiheen aikana teknisen raportin tulee saada W3C-konsortion jäsenten ja hallituksen hyväksyntä. Kun hyväksyntä on saatu, niin W3C antaa raportille lopullisen suosituksen. *Suosituksen* julkaiseminen sitoo W3C-konsortion jäseniä tukemaan raportin sisältöä ja edesauttamaan sen leviämistä yleiseksi käytännöksi. Tunnetuimmat W3C:n julkaisemat suositukset ovat eri versiot HTML:stä ja XML (W3C, 2003d).

2.2 Proseduurien etäkutsut

Proseduurien etäkutsut on mekanismi hajautettujen järjestelmien keskinäiseen kommunikointiin. Ajatus on lähtöisin ohjelmointimalleista, joissa proseduurien kutsuilla voidaan käyttää toisissa moduuleissa sijaitsevia ohjelmakomponentteja (Chu & al., 1997). Proseduurien etäkutsulla tarkoitetaan asiakas/palvelin -suhdetta, jossa kommunikointi järjestelmien välillä tapahtuu palvelupyyntöjen muodossa. Palvelua pyytävää järjestelmää kutsutaan asiakkaaksi (client) ja palvelun tarjoavaa järjestelmää palvelimeksi (server) (Shiva & Virmani, 1993).

Proseduurien etäkutsut -mallissa asiakasjärjestelmässä oleva ohjelma muodostaa ajonaikana pyynnön, jossa on mukana kutsuttavan palvelun nimi ja mahdolliset parametrit arvoineen. Palvelun nimestä käytetään nimitystä proseduuri tai metodi, jos kyseessä on Java-perustainen järjestelmä. Palvelupyyntö lähetetään määrättyyn URL-osoitteeseen, joka sijaitsee palvelinjärjestelmässä. Palvelinjärjestelmä käsittelee pyynnön ja toteuttaa vaaditun operaation mahdollisten mukana seuraavien parametrien arvoilla ja muodostaa sen jälkeen tuloksen, joka palautetaan asiakasjärjestelmälle. Tulos voi olla myös pelkkä ilmoitus, jossa kerrotaan operaation suorituksen onnistuminen, esimerkiksi tietojen tallennuksen yhteydessä (Penttinen, 2001; Petron, 1997).

Olennainen asia proseduurien etäkutsuissa on, että asiakasjärjestelmässä ei itse asiassa tiedetä, kuinka kutsuttava ohjelma on toteutettu palvelinjärjestelmässä. Tämän ansiosta palvelinjärjestelmässä sijaitseva ohjelma voidaan milloin tahansa korvata uudella ohjelmalla, ilman että asiakaskoneella sijaitsevaan kutsuvaan ohjelmaan täytyisi tehdä muutoksia. Yleisesti ottaen ei ole edes välttämätöntä informoida asiakasjärjestelmän ylläpitäjää kyseisistä muutoksista palvelinjärjestelmässä (Penttinen, 2001). Edellytyksenä tälle on, että palvelimella uusi korvaava ohjelma on nimetty samalla tavalla kuin korvattu ohjelma ja se kykenee ottamaan vastaan samat parametrit kuin edeltäjänsä ja muodostamaan halutun tuloksen järkevästi, sekä palauttamaan sen kutsuvaan järjestelmään sen ymmärtämässä muodossa. Sama asia voidaan nähdä myös toisinpäin, sillä asiakaskoneella sijaitseva, palvelupyyntöjä tekevä ohjelma voidaan myös korvata uudella ohjelmalla, ilman että palvelinjärjestelmään täytyisi tehdä muutoksia.
Etäkutsu-mekanismin mahdollistamiseksi on luotu useita erilaisia proseduurien etäkutsutekniikoita, joista tunnetuin lienee CORBA. Kehitettyjen proseduurien etäkutsutekniikoiden ongelmaksi on muodostunut eri ohjelmointikielillä rakennettujen järjestelmien keskinäinen kommunikointi. Kehitetyt tekniikat ovat yleensä sidottuja omaan oliomalliinsa ja kehitysym-

päristöönsä, eikä tarjolla ole ollut helppoa tapaa erilaisilla ohjelmointikielillä rakennettujen järjestelmien yhteensovittamiseksi. Tähän SOAP:in uskotaan tarjoavan ratkaisun (Chester, 2001).

2.3 Rajapinta

Rajapinnaksi kutsustaan niitä palvelin- ja asiakasjärjestelmien välisiä ohjelman osia tai yhteyksiä, joissa tietoa siirretään järjestelmien välillä tai joissa pyydetään palvelua toisesta järjestelmästä (Penttinen, 2001). Rajapinnat siis mahdollistavat kahden eri järjestelmän välisen kommunikoinnin. Yhteyden muodostamisen yhteydessä tapahtuvaa kommunikointia kutsutaan kättelyksi.

Mikäli palvelin- ja asiakasjärjestelmät on toteutettu toisistaan poikkeavissa ohjelmointiympäristöissä, on usein vaarana, että järjestelmät eivät kykene kommunikoimaan suoraan toistensa kanssa. Tähän syynä voi olla esimerkiksi erilainen tiedon esittämis- ja/tai tallennusmuoto. Näissä tapauksissa joudutaan rajapinnassa tekemään konfigurointeja, jotta rajapinnan kautta kulkeva data saataisiin konvertoitua sellaiseen muotoon, että vastaanottava järjestelmä voisi sitä lukea ja käyttää. Tämä suhde on silloin molemminpuolinen, eli myös palvelinjärjestelmän lähettämän vastauksen tietosisältö joudutaan konvertoimaan asiakasjärjestelmän ymmärtämään muotoon (Chester, 2001).

2.4 XML

XML on tekstipohjainen metakieli rakenteisten dokumenttien ja datan esittämiseen, jota käyttämällä voidaan luoda tietoa kuvaava sanasto (Chester, 2001). XML:ää ei pidä sekoittaa ohjelmointikieliin, koska se ei sitä ole. Tietokoneiden on helppo lukea ja generoida XML-dokumentteja, koska oikein muodostettu XML-dokumentti on yksiselitteinen. XML-spesifikaatio on määritelty siten, että sitä voidaan tarvittaessa helposti laajentaa ja se on alustariippumaton (W3C, 2002).

XML:ssä käytetään tiedon tallennuksessa tekstiformaattia, vaikka sen sisältö ei välttämättä ole kokonaisuudessaan tarkoitettu tarkasteltavaksi ihmissilmällä. Binääriformaattiin verrattuna tekstiformaatin etuna on, että sen lähdekoodia voidaan tarvittaessa tarkastella tekstieditoria

käyttäen, eikä näin ollen tarvita erillistä ohjelmaa datan esittämiseen ihmisen ymmärtämässä muodossa (W3C, 2002).

Sekä SOAP että XML-RPC ovat itse asiassa XML-dokumentteja. Samoin kuin webissä yleisesti sivujen kuvauskielenä käytetty HTML, on XML perimmältään kansainvälisen standardin saaneen SGML-dokumentinkuvauskielen johdannainen (Gardner, 2001). HTML:ssä käyttäjällä on käytössään valmiiksi määritellyt tagit tiedon ja ulkoasun kuvaamiseen. XML puolestaan mahdollistaa mekanismin, jolla käyttäjä voi tarvittaessa itse määritellä tiedonkuvauksessa käytettäviä tageja (Bertino & Catania, 2001; Martin, 2001). Gardnerin (2001) mukaan XML on optimaalinen datan lähettämiseen, vastaanottamiseen ja konvertointiin suunniteltu mekanismi. XML:n syntaksi on yksinkertaisempi, mutta tiukempi kuin SGML:ssä. XML kuvaa datan rakenteellista sisältöä, mutta sitä varten on kehitetty myös liitännäistekniikoita, joiden avulla voidaan kuvata XML-dokumentin ulkoasu. SOAP:issa ja XML-RPC:ssä ei ulkoasun kuvaamiseen ole tarvetta, koska niissä välitetään pelkästään dataa järjestelmien välillä.

XML:n käyttö proseduurien etäkutsuissa tekee mekanismeista järjestelmäriippumattomia, koska XML on saavuttanut standardinomaisen aseman järjestelmien kehittäjien keskuudessa (O'Connell & McCrindle, 2001). Järjestelmien vaatimuksena on ainoastaan kyetä tekemään tekstipohjainen XML-tiedosto, joka siirretään jotakin tiedonsiirtoprotokollaa (esim. HTTP tai SMTP) apuna käyttäen toiseen järjestelmään. Vastaanottavan järjestelmän tehtävänä on kyetä vastaanottamaan ja tulkitsemaan siirretty XML-tiedosto, sekä muodostamaan vastausta varten uusi XML-tiedosto ja lähettää se kutsuvaan järjestelmään.

Tiukasti määritellyn syntaksinsa vuoksi XML-dokumentteja/tiedostoja voidaan hyödyntää ohjelmoinnissa. Tämä tiukasti määritelty syntaksi tarkoittaa että XML:ssä ei sallita virheellistä merkkaustapaa. Esimerkki virheellisestä merkkaustavasta voisi olla lopetustagin puuttuminen, mikä tekee koko XML-dokumentista epäkelvon. Sen sijaan tämän tiukan syntaksinsa ansiosta datan varastoiminen XML-muodossa ei ole järkevää, koska tiedostojen koko saattaisi muodostua ongelmaksi. Tämä ongelma aiheutuu siitä, että jokainen yksittäinen tietoalkio tulee sijoittaa vähintään yhden elementin muodostavan tagi-parin sisään (Gardner, 2001).

2.5 XML-RPC

XML-RPC -spesifikaatiolla on pituutta yhteensä seitsemän sivua ja sitä pidetään hyvin yksinkertaisena ja helposti opittavana mekanismina. Yksinkertaisuutensa ja selkeytensä vuoksi sen sisältämää koodia on ihmisenkin helppo lukea. Yksinkertaiseen määritykseen perustuvan ”keveyden” ansiosta XML-RPC tullee pysymään käytössä myös jatkossa, vaikka sen rinnalle on johdettu uudempia, monipuolisempia ja täten laajempia määrityksiä. XML-RPC on XML-metakielen laajennus ja se on ensimmäisenä kehitetty XML-perustainen proseduurien etäkutsuprotokolla. XML-RPC mahdollistaa yksinkertaisen tavan muodostaa proseduurien etäkutsuja sisältäviä viestejä ja näiden vastauksia (O’Connell & McCrindle, 2001).

XML-RPC:ssä on kaksi osapuolta, asiakas ja palvelin. Asiakasjärjestelmässä muodostetaan XML-merkkauskieleen perustuva viesti, joka sisältää kutsuttavan proseduurin nimen, sekä sen mukana lähetettävien parametrien arvot. Viesti vastaanotetaan palvelinjärjestelmässä ja viestin prosessoinnin seurauksena muodostetaan XML-RPC -vastaussanoma, joka lähetetään kutsun tehneeseen asiakasjärjestelmään. XML-RPC:tä voidaan käyttää minkä tahansa ohjelmointikielen kanssa, jolla kyetään generoimaan XML-dokumentti. Käytännössä XML-RPC:tä hyödyntäviä järjestelmiä on toteutettu mm. Java, C++, Perl, Tcl ja PHP -kielillä (Penttinen, 2001).

XML-RPC -protokolla kulkee HTTP-protokollan päällä (tai sen yhteydessä) ja viestin siirtämisessä käytetään hyväksi HTTP-POST -metodia (Girardot & al., 2000). Käytännössä tämä tarkoittaa sitä, että HTTP-viestin runko-osaan upotetaan XML-RPC -protokollan mukainen viesti, joka siis sijaitsee samassa tiedostossa kuin itse HTTP-viestikin (Penttinen, 2001).

2.5.1 XML-RPC -kutsu

XML-RPC -viestissä proseduurin kutsu ja parametrit arvoineen kääritään methodCall-elementin sisälle. XML-merkkauskielen sääntöjen mukaan elementti muodostuu tagi-parista. Elementin tunnus sijoitetaan tagin aloitus- ja lopetusmerkkien väliin. Tagin aloitusmerkki on < (pienempi kuin) ja lopetusmerkki puolestaan on > (suurempi kuin). Elementin lopetustagi on muutoin samanlainen kuin aloitustagikin, mutta tagin aloitusmerkkiä seuraa kauttaviiva, en-

nen elementin tunnusta ja lopetusmerkkiä. Täten esimerkiksi methodCall-tunnuksen omaava elementti kirjoitetaan seuraavasti (Winer, 1999):

<methodCall> ….. </methodCall>.

Kutsuttavan proseduurin nimi sijoitetaan methodName-elementin sisään. Mikäli palvelinjärjestelmässä proseduurit on jaettu erillisiin paketteihin, niin paketin nimi tulee merkitä ennen proseduurin nimeä pisteellä eroteltuna, esimerkiksi (Winer, 1999):

<methodName> paketti1.proseduuri13 </methodName>.

XML-RPC -kutsuun ei merkitä parametrien tunnuksia, vaan ainoastaan niiden tyypit ja arvot. Vastaanottava sovellus erottaa parametrien arvot toisistaan sen perusteella, missä järjestyksessä ne on esitetty XML-RPC -viestissä. XML-RPC -kutsuviesti sisältää params-elementin, jonka sisään kaikki parametrien arvot sijoitetaan. Yksittäinen parametrin arvo kääritään vielä param-, value- ja tyypin ilmaisevan elementin sisään. (Penttinen, 2001).

```
<?xml version="1.0"?>
 <methodCall>
       <methodName> tilitiedot.palautaSaldo </methodName>
       <params>
          <param>
            <value><string>Makkonen Nuno</string></value>
          </param>
          <param>
            <value><string>9k3747a49374a848s937sd489fsf49</string></value>
          </param>
          <param>
            <value><string>445323-624311</string></value>
          </param>
        </params>
 </methodCall>
```

Kuva 3. XML-RPC -kutsu.

Kuvassa 3 olevassa yksinkertaisessa esimerkissä on generoitu XML-RPC -kutsuviesti, jossa kutsutaan vaikkapa jonkin pankin järjestelmää. Kutsuttava proseduuri palautaSaldo sijaitsee palvelinjärjestelmässä tilitiedot nimisessä paketissa. Kyseinen paketti on voitu toteuttaa periaatteessa millä tahansa ohjelmointikielellä - eikä asiakasjärjestelmän tekijällä ole välttämättä edes tietoa sen toteutustavasta. Asiakasjärjestelmän rakentajan tulee kuitenkin tietää URL-osoite, johon kutsu lähetetään ja kutsuttavien proseduurien esittelyt. Proseduurin esittely si-

sältää proseduurin nimen (ja mahdollisesti myös paketin) sekä mahdolliset parametrit. Kuvan esimerkissä parametrien arvoina palvelinjärjestelmään välitetään proseduurin esittelyssä kuvatut tiedot: henkilön suku- ja etunimi, lähetettävien tietojen perusteella generoitu tunnus (esim. MD5-tarkistusavain) ja tilinumero.

2.5.2 XML-RPC -vastaus

Otettuaan vastaan asiakasjärjestelmän lähettämän palvelupyynnön palvelinjärjestelmä muodostaa vastausviestin. Vastausviesti kääritään methodResponse-elementin sisään. XML-RPC -spesifikaation mukaan vastausviesti voi sisältää ainoastaan yhden palautettavan arvon. Tämä ei kuitenkaan aiheuta ongelmia, koska XML-RPC:ssä on määritelty tietotyypit taulukoiden ja tietueiden käyttöä varten (O'Connell & McCrindle, 2001). Samalla tavalla kuin palvelupyynnössä, myös vastaus kääritään params-, param- ja value-elementin sisään. Vastausviestin param-elementti ei eroa millään tavoin pyynnön yhteydessä käytettävästä vastaavasta elementistä (Penttinen, 2001). Kuvassa 4 on esitelty XML-RPC -vastaus.

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
        <value>
          <string> 120.34 </string>
        </value>
    </param>
</params>
</methodResponse>
```

Kuva 4. XML-RPC -vastaus.

Kuvan 4 esimerkissä palvelinjärjestelmässä on muodostettu vastausviesti asiakkaan kuvan 3 palautaSaldo-kyselyyn. Pyyntöviestin tavoin myös vastausviesti on XML-dokumentti. Esimerkin vastausviesti sisältää merkkijonotyyppisen palautusarvon 120.34.

2.5.3 XML-RPC:ssä käytettävät tietotyypit

Parametrin tyypin oletusarvo on merkkijono (String). Jos tyyppiä ei ole eksplisiittisesti ilmaistu, niin parametrin tyyppinä käytetään oletusarvoa. Muita mahdollisia tyyppejä XML-

RPC:ssä ovat kokonaisluku (int), liukuluku (double), totuusarvo (boolean) (arvot 1 (tosi) ja 0 (epätosi)), päivämäärä (date), taulukko (array) ja tietue (struct) (Winer, 1999). XML-RPC -viesteissä voidaan lähettää myös binäärimuotoista dataa käyttämällä Base64-koodausmenetelmää, jolla binäärimuotoinen data konvertoidaan ASCII-muotoiseksi. Tätä menetelmää on käytetty yleisesti binäärimuotoisen datan siirtämisessä sähköpostiviestien mukana (O'Connell & McCrindle, 2001). XML-RPC:n heikkous on siinä, että spesifikaatio ei mahdollista käyttäjien itsensä määrittelemiä tietotyyppejä (Penttinen, 2001).

Taulukko-tietotyyppiä oleva elementti muodostetaan XML-RPC:ssä siten, että siihen kuuluvat muut elementit kääritään array-elementin sisään. Taulukko sisältää data-elementin, jonka sisään sijoitetaan taulukon arvot. Data-elementti voi sisältää 1..*n* kappaletta yksittäisiä arvoja, jotka kääritään value-elementin ja tyypin ilmaiseman elementin sisään. Taulukon sisältämät yksittäiset arvot voivat olla mitä tahansa XML-RPC:n tukemaa tietotyyppiä, joten ne voivat siten olla myös taulukko- tai tietuetyyppiä. Taulukon sisältämät alkiot voivat olla eri tietotyyppiä olevia arvoja (Penttinen, 2001).

```
<?xml version="1.0"?>
<methodResponse>
<params>
 <param>
    <value>
       <array>
          <data>
             <value><int>32</int></value>
             <value><double>28.32</double></value>
             <value><string>Joensuu</string></value>
             <value><boolean>0</boolean></value>
          </data>
       </array>
    </value>
 </param>
</params>
</methodResponse>
```

Kuva 5. Taulukko-tietotyypin käyttö XML-RPC:ssä.

Kuvassa 5 on esitelty XML-RPC -vastaus, jossa palautusarvo on taulukkotyyppiä. Esimerkin taulukon alkiot sisältävät eri tyyppiä olevia arvoja. Koska merkkijono on XML-RPC:ssä oletustyyppi, niin arvoa Joensuu ei välttämättä tarvitsisi sijoittaa tyypin ilmaisevan elementin sisään, vaan yhtä hyvin arvo voisi olla suoraan value-elementin sisällä.

Tietue sijoitetaan struct-elementin sisään. Tietueen yksittäisen kentän ilmaiseva tieto sijoitetaan member-elementin sisään ja niitä voi olla tietueessa 0..*n* kappaletta. Tietueen kentän nimi tulee name-elementin sisään ja arvo ilmaistaan value-elementin sisällä siten, että arvo kääritään tietotyypin ilmaisevan elementin sisään. Tietueen arvo voi olla mitä tahansa XML-RPC:ssä käytössä olevaa tietotyyppiä – arvona voi siis olla myös taulukko tai toinen tietue (Penttinen, 2001).

```
<params>
   <param>
      <value>
         <struct>
            <member>
               <name> maa 1 </name>
               <value><string>Ranska</string></value>
            </member>
            <member>
               <name> maa 2 </name>
               <value><string>Kanada</string></value>
            </member>
         </struct>
    </value>
   </param>
</params>
```

Kuva 6. Tietueen käyttö XML-RPC:ssä.

Kuvassa 6 on esitelty tietueen käyttöä XML-RPC -viestissä. Kuvan tietue sisältää kaksi kenttää, jotka on sijoitettu member-elementtien sisälle. Kentän nimi ilmaistaan name-elementin sisällä.

2.5.4 HTTP-paketointi XML-RPC -kutsussa

XML-RPC -protokollan mukainen viesti kulkee verkossa aina HTTP-protokollan seurana. XML-RPC -viesti sidotaan HTTP-viestiin upottamalla se HTTP-viestin runkoon ja tämän takia täytyy viestissä olla HTTP-protokollaan pakollisena kuuluvat merkinnät. Nämä HTTP-protokollaan kuuluvat merkinnät sisältävät seuraavat rivit (Penttinen, 2001): URI, User-agent, Host, Content-type ja Content-length.

*URI:*lla (Uniform Resource Identifier) tarkoitetaan webissä käytettävää nimeämis- ja tunnistamismenetelmää, jonka avulla kaikki webin muodostavat elementit kyetään yksikäsitteisesti

identifioimaan. Näitä elementtejä ovat mm. sähköiset dokumentit, palvelut, ladattavat tiedostot, kuvat ja webin käyttäjät.

URI koostuu merkkijonosta ja sitä käyttäen päästään käsiksi webissä oleviin resursseihin. URI voidaan luokitella sen alakäsitteiden perusteella URL:ksi tai URN:ksi. Tunnetumpi lyhenne näistä on *URL*, jolla tarkoitetaan fyysistä osoitetta, jossa jokin resurssi sijaitsee. Kyseisessä fyysisessä osoitteessa sijaitseva resurssi voidaan halutessa muuttaa toiseksi resurssiksi. *URN* puolestaan on jollekin tietylle resurssille (esimerkiksi dokumentille tai kuvalle) annettu tunnus, ja vaikka resurssin fyysinen osoite muuttuisikin, niin URN-tunnus pysyy aina samana (Berners-Lee & al. 1998).

User-agent eli käyttäjäagentti on selain tai muu vastaava ohjelma, jolla otetaan yhteyttä web-palveluun. Käyttäjäagentti lähettää itsestään tietoja, joiden avulla voidaan päätellä kyetäänkö kyseisellä käyttäjäagentilla käsittelemään pyydettyä palvelua, esimerkiksi ottamaan vastaan ja näyttämään kutsuttu www-sivu (W3C, 1999a).

Host-kenttä ilmaisee Internet-palvelimen nimen ja mahdollisen portin numeron. Mikäli portin numeroa ei anneta, niin siinä tapauksessa portin numeroksi päätellään käytettävän protokollan mukainen oletusportti (esim. HTTP-protokollaa käytettäessä oletusportin numero on 80). Internet-palvelimen nimi ja portin numero erotetaan toisistaan kaksoispisteellä, esimerkiksi ”Host: www.w3c.org:80” (W3C, 1999a).

Content-type kertoo viestin MIME-tyypin, jonka avulla viestin vastaanottava järjestelmä kykenee tunnistamaan viestin sisällön tyypin. Content-type koostuu tyypistä ja alityypistä sekä mahdollisista parametreista. XML-RPC:tä käytettäessä tyypillisin MIME-tyyppi viesteissä on text/plain. Esimerkiksi tyypillä ”Content-type: text/plain; charset=us-ascii” ilmaistaan, että viestin sisältö on teksti-formaatissa (ylätyyppi ”text”) ja teksti on muotoilematonta (plain). Charset-parametri kertoo, että merkistönä käytetään 7-bittistä us-ascii:ta (W3C, 1999a).

Content-length puolestaan ilmaisee viestin pituuden tavuina ja sen käyttäminen on vapaaehtoista[4] (W3C, 1999a). Viestin sisältämien tavujen määrä ilmaistaan kirjoittamalla tunnuksen Content-length perään kaksoispiste ja tavujen määrä, esimerkiksi ”Content-length: 8274”.

[4] HTTP 1.0 -versiossa Content-length on pakollinen (Chiu & al., 2002).

3 SOAP-PROTOKOLLA

Tässä luvussa perehdytään varsinaiseen tutkielman aiheeseen, eli SOAP-protokollaan. Kohdassa 3.1 käydään läpi SOAP-protokolla yleisellä tasolla. Kohdassa käsitellään mm. SOAP-protokollan suunnittelun päämäärät, SOAP:iin olennaisesti liittyvä nimiavaruuksien viittaamistekniikka ja SOAP-viestin käsittelymalli. Kohdassa 3.2 käsitellään SOAP-viestin rakenne W3C:n julkaiseman suosituksen pohjalta. SOAP-viesti rakentuu elementeistä, attribuuteista ja tekstimuodossa olevasta datasta. Kohdassa perehdytään suosituksessa spesifikaatioon määriteltyihin elementteihin ja attribuutteihin, sekä erityisesti virheiden käsittelyyn ja hallintaan.

SOAP-tiedon mallinnuksessa tietorakenteet johdetaan suunnattua verkkoa ja siihen liittyviä sääntöjä käyttäen. Kohdassa 3.3 keskitytään SOAP-viestissä kuljetettavan tiedon mallinnukseen ja kooditussääntöihin. Kohdassa 3.4 käydään läpi SOAP:ia tukevat viestinvälitysmallit ja kohdassa 3.5 on esitelty suositus proseduurien etäkutsujen suorittamiseksi SOAP-viestejä käyttäen. SOAP-viesti täytyy sitoa johonkin rinnakkaisprotokollaan viestin kuljetuksen ajaksi. Kohdassa 3.6 käsitellään yleisimmät SOAP:ia tukevat rinnakkaisprotokollat ja SOAP:in sidokset näihin käytäntöihin. SOAP-protokollan kilpailijat on esitelty kohdassa 3.7 ja web-palveluiden malli kohdassa 3.8. Kohdassa 3.9 on esitelty joitakin SOAP-kehitysympäristöjä ja kohdassa 3.9 on perehdytty SOAP:in suorituskykyyn.

3.1 Yleistä

W3C:n oman määritelmän mukaan SOAP on kevyt ja yksinkertainen mekanismi, joka on suunniteltu rakenteisen ja vahvasti tyypitetyn informaation vaihtoon hajautetuissa, keskittämättömissä ympäristöissä (W3C, 2000). Proseduurien etäkutsuja varten SOAP:issa kuvataan metodien/proseduurien tunnukset, parametrit, palautettavat arvot tyyppeineen ja virheviestit yleisimmin esiintyville virhetyypeille (Tsenov, 2002).

3.1.1 Suunnittelun päämäärät

SOAP ei sinänsä tarjoa mitään uutta ja mullistavaa tapaa toteuttaa tiedonsiirtoa tai proseduurien etäkutsuja, vaan näiden toimintojen suorittaminen onnistuu myös lukuisilla muilla me-

netelmillä ja tekniikoilla. SOAP:in tuoma lisäarvo perustuu ennen kaikkea palomuurien läpäisykykyyn (yhdessä HTTP-protokollan kanssa käytettynä), yleiskäyttöisyyteen (monet ohjelmointikielet tukevat sitä), tiedon esittämiseen XML:n ymmärtämässä muodossa sekä sen saamaan laajaan tukeen suurten yritysten ja muiden webissä toimivien organisaatioiden taholta (Govindaraju & al., 2000; W3C, 2000).

SOAP:ia suunniteltaessa päämäärinä ovatkin olleet spesifikaation yksinkertaisuus ja laajennettavuus (W3C, 2003b). *Yksinkertaisuudella* pyritään siihen, että spesifikaatio säilyy helposti opittavana ja hallittavana. Tämän taas toivotaan madaltavan kynnystä SOAP:in käyttöönottamiseen järjestelmien suunnittelun ja toteutuksen yhteydessä.

Laajennettavuudella pyritään siihen, että spesifikaatioon voidaan tulevaisuudessa liittää helposti tarpeellisia lisämääritteitä. W3C-konsortio on lisännyt saman laajennettavuuden mahdollistavan vaatimuksen mukaan kaikkiin XML-perheeseen kuuluviin spesifikaatioihinsa. Tämän ominaisuuden tarkoituksena on ennalta ehkäistä käytettävien menetelmien ennenaikaista vanhenemista nopeasti uudistuvassa ja muuttuvassa Internet-maailmassa.

3.1.2 SOAP-viestin lähettäminen ja vastaanottaminen

SOAP ei kykene kulkemaan itsenäisesti verkossa, vaan sitä tulee käyttää yhdessä jonkun muun tiedonsiirtoa tukevan protokollan kanssa. Näistä protokollista tunnetuimmat ja käytetyimmät ovat HTTP ja sähköpostiviestien protokolla SMTP. SOAP on pohjimmiltaan yksisuuntainen viesti, mutta siitä voidaan helposti tehdä kaksisuuntaista kutsu/vastausviestiä simuloiva malli muodostamalla vastaanotetun viestin pohjalta uusi viesti (vastaussanoma), joka lähetetään kutsuvalle osapuolelle. Samalla periaatteella SOAP:in avulla voidaan muodostaa myös monimutkaisempia malleja, esimerkiksi kutsu/monivastausviestit (W3C, 2003b).

Chiu & al. (2002) kuvaavat SOAP-viestin lähettämiselle seuraavat vaiheet:

- viestin sisältämien tietorakenteiden läpikäynti
- konekielisen datan konvertointi ASCII-muotoon
- ASCII-muotoisen datan kirjoittaminen puskurimuistiin
- datan siirtäminen tietoverkkoon.

Viestin vastaanottamiselle kuvataan puolestaan seuraavat vaiheet (Chiu & al., 2002):

- viestin lukeminen tietoverkosta puskurimuistiin
- XML-muotoisen datan jäsennys
- elementtien käsittely
- ASCII-muotoisen datan konvertointi konekieliseen muotoon.

SOAP-viestin koko tavuina voi olla 4-10 kertaa suurempi kuin sen sisältämän tiedon esittäminen konekielisessä muodossa. Suorituskyvyssä tämä aiheuttaa merkittäviä eroja silloin, kun käsitellään suuria taulukkorakenteisia tietotyyppejä (Chiu & al., 2002).

3.1.3 Nimiavaruudet

W3C-konsortio kehitti nimiavaruuksien viittaamistekniikan vastauksena ongelmiin, jotka aiheutuivat samalla tavalla nimetyistä XML-dokumentin elementeistä ja attribuuteista (Karpinski, 1999). W3C:n (1999b) mukaan näiden samalla tavalla nimettyjen elementtien yhteentörmäystä voidaan verrata puhelinnumeroihin, sillä sama puhelinnumero voi olla käytössä useammalla henkilöllä eri verkkoryhmissä. Nämä muuten identtiset puhelinnumerot erotetaan toisistaan suuntanumerojen avulla. Tätä suuntanumeroa vastaa XML-dokumentissa nimiavaruus johon viitataan. Nimiavaruuksiin viittaamalla XML-dokumentin sisältämät elementit ja attribuutit voidaan identifioida yksiselitteisesti (W3C, 1999b).

Nimiavaruus koostuu URL-osoitteesta ja siinä mahdollisesti sijaitsevasta XML-dokumentista. Tämä XML-dokumentti voi sisältää kyseiseen nimiavaruuteen liittyvien elementtien ja attribuuttien dokumentoinnin. Kuitenkaan nimiavaruuksiin viittaamisen ensisijainen tehtävä ei ole käsitellä mahdollista XML-dokumenttia, vaan antaa elementille yksiselitteinen tunnus etuliitteeksi (vrt. samaa URL-osoitetta voi olla käytössä vain yksi kappale).

Nimiavaruuksien viittaamistekniikalla SOAP-viestin sisällön kelpoisuus voidaan todentaa ja samalla ottaa käyttöön viitatussa nimiavaruudessa määritellyt tietotyypit. Mikäli viestin sisältö ei läpäise kelpoisuuden tarkistusta, eli vastaanottava sovellus ei tunnista viestissä viitattuja nimiavaruuksia, tulee määritysten mukaan rakennetun järjestelmän kyetä generoimaan

asiasta virheilmoitus. Tällä todentamisella mahdollistetaan myös erilaisten SOAP-versioiden keskinäinen yhteensopivuus (Tsenov, 2002).

Teknisesti viittaaminen tapahtuu käyttämällä attribuuttia *xmlns*, joka on lyhenne sanoista ’xml namespace’. Attribuutilla xmlns asetetaan nimiavaruuden tunnuksen arvoksi nimiavaruuden URL-osoite. Viittaus kirjoitetaan elementin aloitustagin sisään seuraavasti:

xmlns:<nimiavaruuden tunnus>="<nimiavaruuden URL>".

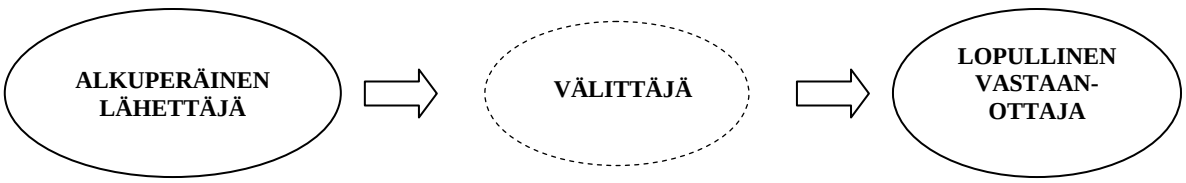
SOAP-viestin elementeissä nimiavaruus otetaan käyttöön liittämällä viitatun nimiavaruuden tunnus elementin tunnuksen etuliitteeksi seuraavalla tavalla:

<nimiavaruuden tunnus:elementin tunnus>.

Jos saman elementin sisällä viitataan useampaan kuin yhteen nimiavaruuteen, niin nimiavaruuden tunnus/URL-osoite yhdistelmät erotellaan toisistaan tyhjeellä (W3C, 2003b).

3.1.4 Käsittelymalli

W3C on mallintanut SOAP-viestin käsittelyn. SOAP-viestin osapuolet ovat viestin *alkuperäinen lähettäjä* (initial sender) ja *lopullinen vastaanottaja* (ultimate receiver). Lisäksi osapuolina voi olla nolla tai useampi viestin *välittäjä* (intermediary), jotka ovat sekä viestin vastaanottajia että viestin lähettäjiä. Näistä eri osapuolista käytetään nimitystä SOAP-solmut (SOAP nodes) (W3C, 2003b).



Kuva 7. SOAP-solmut.

SOAP-solmut (kuva 7) pystytään identifioimaan URI:n perusteella. Vaikka URI:lle ei ole määritelty enimmäispituutta, niin sovellusten tulisi kuitenkin pystyä käsittelemään vähintään

2048 merkkiä pitkät URI:t. Koneiden IP-osoitteiden käyttöä URI:ssa ei suositella ja mikäli sitä käytetään, niin sovelluksen tulisi tukea myös koneiden DNS-nimien käyttöä (W3C, 2003b).

Kullakin SOAP-solmulla voi olla yksi tai useampi rooli. Roolien tarkoituksena on kertoa SOAP-solmulle, mitkä SOAP-viestin sisältämät otsikkolohkoelementit sen tulee käsitellä. Viestin prosessoinnin aikana SOAP-solmun rooli(t) eivät saa vaihdella (W3C, 2003b). Koska SOAP-viestin runko-osaa ei saa käsitellä välittäjäsolmuissa, niin tästä seuraa, että SOAP-viestin lopullinen vastaanottajasolmu ei voi olla samassa viestiketjussa sekä välittäjäsolmuna että lopullisena vastaanottajasolmuna. Sen sijaan käsittelymalli ei rajoita yksittäisen välittäjäsolmun sijoittamista viestiketjun kahteen tai useampaan kohtaan, eikä alkuperäistä lähettäjäsolmua toimimasta myös välittäjäsolmuna.

Käsittelymallin mukaan tulee SOAP:ia hyödyntävää sovellusta toteutettaessa määritellä ne roolit, joissa kunkin SOAP-solmun täytyy käsitellä viestiä. Jokaisessa SOAP-solmussa voidaan käsitellä kyseiselle solmulle kohdistetut otsikkolohkoelementit. Mikäli jokin solmu ei kykene käsittelemään sille pakolliseksi määriteltyä otsikkolohkoelementtiä, tulee sen joka tapauksessa kyetä generoimaan asiasta virheilmoitus, lähettää se eteenpäin ja lopettaa SOAP-viestin käsittely. Välittäjäsolmujen tulee kyetä lähettämään viesti eteenpäin viestipolun seuraavalle solmulle. Reitin lopussa olevan vastaanottajasolmun tulee kyetä käsittelemään sille kuuluvien pakollisten otsikkolohkoelementtien lisäksi myös SOAP-viestin runkoelementin sisältö. Mikäli se ei tähän kykene, tulee sen kuitenkin pystyä generoimaan tästä asianmukainen virheilmoitus. Mikäli välittäjäsolmussa muutetaan vastaanotettua SOAP-viestiä, tulee solmun huolehtia, että viestin rakenne pysyy validina. Yksittäinen SOAP-solmu voi tukea yhtä tai useampaa SOAP:in versiota. Mikäli solmu ei kykene tunnistamaan käytettyä versiota, niin sen tulee generoida asiasta virheilmoitus (W3C, 2003a).

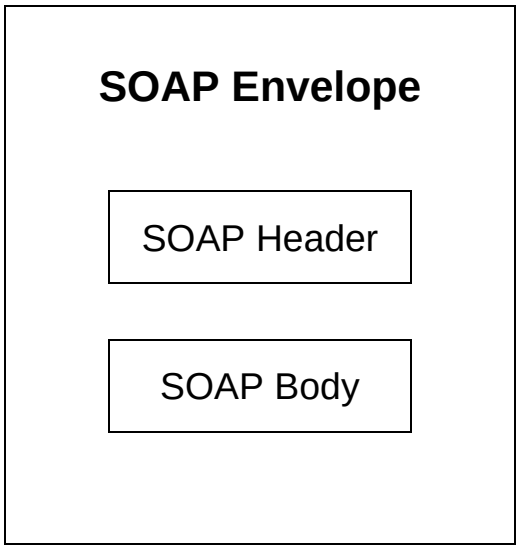
3.2 Viestin rakenne

SOAP-viesti on XML-dokumentti, joka koostuu sisäkkäisistä elementeistä, attribuuteista ja tekstimuodossa olevasta datasta. SOAP-viesti voidaan jakaa kolmeen pääelementtiin: 1) kuorielementtiin, joka toimii SOAP-viestin kehyksenä, 2) otsikkoelementtiin, joka sisältää mekanismin liittää SOAP-viestiin laajennuksia ja 3) runkoelementtiin, jossa sijaitsee varsinainen

siirrettävä tietorakenne. SOAP-viestissä on määritelty pakollisina elementteinä kuori- ja runkoelementit ja valinnaisina elementteinä otsikko- sekä käyttäjän itsensä määrittelemät elementit. Käyttäjän itsensä määrittelemät elementit eivät voi olla kuorielementin suoranaisia lapsielementtejä, vaan ne sijaitsevat joko otsikko- tai runkoelementin sisällä. Elementin pakollisuudella tarkoitetaan sitä, että ilman kyseisen elementin olemassaoloa ei SOAP-dokumentti ole validi (W3C, 2003b).

3.2.1 Kuorielementti

Kuorielementin (envelope) aloitus- ja lopetustagit kätkevät sisäänsä kaikki muut SOAP-viestin elementit. Kuorielementti esiintyy SOAP-viestissä fyysisesti aina ensimmäisenä ja siinä kerrotaan mitä SOAP-viesti sisältää, minkä pitäisi käsitellä kutakin viestin osaa ja onko viestin sisältämien yksittäisten komponenttien käsittely pakollista vai valinnaista (Jepsen, 2001). Kuorielementissä ilmaistaan nimiavaruuden viittauksella, mikä on SOAP-viestissä käytetty SOAP-versio. Kuorielementti sisältää yhden tai kaksi suoranaista lapsielementtiä, riippuen siitä onko SOAP-viestissä käytössä otsikkoelementti (W3C, 2003b).



Kuva 8. Kuorielementin rakenne (W3C, 2003a).

Kuvassa 8 on esitelty kuorielementin rakenne. Kuorielementti kätkee sisäänsä valinnaisen otsikkoelementin ja pakollisen runkoelementin.

SOAP 1.2 -version määrittelyyn sisällytetty kuorielementin nimiavaruus ja XML-dokumentaatio löytyvät URL-osoitteesta http://www.w3.org/2003/05/soap-envelope ja vastaava SOAP 1.1 -version nimiavaruus ja XML-dokumentaatio löytyvät puolestaan URL-osoitteesta http://schemas.xmlsoap.org/soap/envelope/. Viittaamalla näihin nimiavaruuksiin SOAP-viestin vastaanottava sovellus tunnistaa yksiselitteisesti käytetyn SOAP-version (W3C, 2003a).

```
<SOAP-KEHYS:ENVELOPE
 xmlns:SOAP-KEHYS=”http://www.w3.org/2003/05/soap-envelope”>
   <SOAP-KEHYS:BODY>
       <HaeEsimerkkiArvo>
                              <parametri>1</ parametri >
       </HaeEsimerkkiArvo>
   </SOAP-KEHYS:BODY>
</SOAP-KEHYS:ENVELOPE>
```

Kuva 9. Nimiavaruuteen viittaaminen SOAP-viestissä (W3C:n (2003a) esimerkkiä mukaillen).

Kuvassa 9 on esitelty SOAP-viesti, jossa kuorielementin etuliitteeksi on liitetty nimiavaruuden tunnus SOAP-KEHYS. Attribuuttia xmlns käyttäen nimiavaruuden tunnuksen arvoksi on asetettu SOAP 1.2 -versiosta kertova nimiavaruus http://www.w3.org/2003/05/soap-envelope.

Kuorielementti voi sisältää myös ylimääräisiä attribuutteja. Näiden ylimääräisten attribuuttien tulee olla sopivia käytetyn nimiavaruuden kanssa. Samoin kuorielementin sisältämien lapsielementtien tulee esiintyessään olla yhteensopivia käytetyn nimiavaruuden kanssa (Jepsen, 2001). Välittäjäsolmuissa voidaan viestin kulun aikana lisätä kuorielementtiin attribuutteja ja nimiavaruuksien viittauksia (W3C, 2003b).

3.2.2 Otsikkoelementti

Otsikkoelementin (header) käyttö on valinnaista ja jos sitä käytetään, niin se tulee aina kuorielementin sisään ennen runkoelementtiä. Otsikkoelementissä voidaan määritellä viestin syntaksin laajennuksia, esimerkiksi jos halutaan antaa viestin vastaanottavalle sovellukselle tietoja transaktiosta tai valtuuksista. SOAP:in määritys mahdollistaa attribuuttien ja nimiavaruuksien viittausten lisäämisen otsikkoelementtiin viestiketjun läpikäymisen aikana (W3C, 2003b).

3.2.3 Otsikkolohkoelementti

Otsikkoelementin sisältämiä lapsielementtejä kutsutaan nimellä *otsikkolohkoelementit* (header block). Kukin viestin sisältämä otsikkolohkoelementti täytyy validoida käyttäen nimiavaruuksiin viittausta. Otsikkolohkoelementillä voi muiden elementtien tapaan olla myös lapsielementtejä. Näiden lapsielementtien kohdalla nimiavaruuksiin viittaaminen ei kuitenkaan ole pakollista (W3C, 2003b).

Otsikkolohkoelementit on tarkoitettu SOAP-viestin vastaanottavien solmujen (eli välittäjäsolmujen ja lopullisen vastaanottajasolmun) käsiteltäviksi. Käsittelyn tuloksena SOAP-solmussa käsitelty otsikkolohkoelementti poistetaan SOAP-viestistä. Otsikkolohkoelementin käsittelyn seurauksena voidaan tarvittaessa sovelluskohtaisesti määrätä missä järjestyksessä muut otsikkolohkoelementit ja/tai runkoelementti (jos kyseessä lopullinen vastaanottajasolmu) käsitellään. Ilman tällaisen käsittelyjärjestyskontrollin olemassaoloa on SOAP-viestin eri osien käsittelyjärjestys mielivaltaista. SOAP:in määrittelyssä ei oteta kantaa siihen, tulisiko lopullisessa vastaanottajasolmussa käsitellä ensin runkoelementti vai otsikkoelementti (W3C, 2003b).

Otsikkolohkoelementti voi sisältää kahden tyyppisiä attribuutteja: käyttäjän itsensä määrittelemiä sovelluskohtaisia attribuutteja ja versiossa määriteltyjä attribuutteja, joilla on erikoismerkitys prosessoitaessa SOAP-viestiä. Otsikkolohkoelementtejä varten on määritelty seuraavat attribuutit: kooditustyyliattribuutti, rooliattribuutti, otsikkolohkon ymmärtämisattribuutti ja eteenpäinsiirtoattribuutti (W3C, 2003b).

3.2.4 Kooditustyyliattribuutti

Kooditustyyliattribuutti (encodingStyle) viittaa kooditussääntöihin, joita käytetään SOAP-viestin eri osien sarjallistamisessa ja sarjallistamisen purussa. Sarjallistamisella tarkoitetaan datan koodaamista XML-muotoon viestin siirtämisen ajaksi ja sarjallistamisen purulla taas XML-muodossa olevan datan palauttamista takaisin alkuperäiseen muotoonsa. Kooditussäännöillä on olemassa omat nimiavaruutensa, joihin kooditustyyliattribuutilla voidaan viitata asettamalla attribuutin arvoksi säännöt sisältävän XML-tiedoston URL-osoite. W3C-konsortion asettama työryhmä on määritellyt kooditussäännölle nimiavaruuden URL-osoitteessa:

”http://www.w3.org/2003/05/soap-encoding”.

Järjestelmien rakentajat voivat myös itse määritellä omia kooditussääntöjään ja viitata niihin kooditustyyliattribuuttia käyttäen (W3C, 2003c).

Kooditustyyliattribuuttia voidaan käyttää otsikkolohkoelementin, runkoelementin lapsielementtien, yksityiskohtaelementin ja näistä periytyvien elementtien yhteydessä. Kooditussääntöön viittaus periytyy sen elementin jälkeläisille, jossa viittaus on suoritettu, ellei jälkeläisessä itsessään viitata johonkin toiseen kooditussääntöön tai ilmaista kooditustyyliattribuuttia käyttäen, että kooditussääntöön viittausta ei tehdä. Mikäli lapsielementin ja sen jälkeläisten ei haluta kuuluvan viittauksen vaikutusalueeseen tulee kyseisen lapsielementin kooditustyyliattribuutin arvoksi asettaa nimiavaruuden arvo http://www.w3.org/2003/05/soap-envelope/encoding/none. Vikaelementin yhteydessä kooditustyyliattribuutin käyttö ei sen sijaan ole sallittua (W3C, 2003a).

```
<env:Header>
   <o:omaElementti xmlns:o="http://esimerkkikone.fi/pakollinen_nimiavaruus/"
       env:encodingStyle="http://esimerkkikone.fi/omat_tyylit/">
      <o:ekaElementti> … tietoa … </o:ekaElementti >
      <o:tokaElementti
         env:encodingStyle="http://www.w3.org/2003/05/soap-envelope/encoding/none">
                  … tietoa …
      </o:tokaElementti>
   </o:omaElementti>
</env:Header>
```

Kuva 10. Viittaus kooditussäännön nimiavaruuteen (W3C:n (2003b) esimerkin pohjalta).

Kuvassa 10 on SOAP-viestin otsikkoelementti, joka sisältää yhden otsikkolohkoelementin. Otsikkolohkoelementissä on viitattu kooditustyyliattribuutilla sovelluskohtaisesti määriteltyyn kooditussäännön arvoon http://esimerkkikone.fi/omat_tyylit/. Viitattu kooditussääntö on voimassa otsikkolohkoelementin ensimmäisessä lapsielementissä ekaElementti, mutta sen sisarelementissä tokaElementti on kooditustyyliattribuutti asetettu viittaamaan nimiavaruuteen http://www.w3.org/2003/05/soap-envelope/encoding/none, jolla kumotaan kooditussäännön vaikutus tämän elementin sisällä.

3.2.5 Solmun rooliattribuutti

Solmun rooliattribuuttia (role) käytetään otsikkolohkoelementtien yhteydessä ja sillä ilmaistaan minkä roolin omaavien solmujen tulee käsitellä kyseistä elementtiä. Attribuutin arvona on URI-osoite. Mikäli rooliattribuuttia ei käytetä otsikkolohkoelementin yhteydessä, niin ainoastaan viestin lopullinen vastaanottaja voi käsitellä kyseistä otsikkolohkoelementtiä. SOAP:issa on määritelty rooliattribuutille seuraavat kolme arvoa, jotka sisältävät roolit next, none ja ultimateReceiver:

- ”http://w3.org/2003/05/soap-envelope/role/next”
- ”http://w3.org/2003/05/soap-envelope/role/none”
- ”http://w3.org/2003/05/soap-envelope/role/ultimateReceiver”

Kun otsikkolohkoelementin rooliattribuutille asetetaan rooliksi *next*, niin sillä ilmaistaan, että SOAP-viestin vastaanottavan solmun tulee käsitellä kyseinen otsikkolohkoelementti. Mikäli lähettäjäsolmussa muodostetusta SOAP-viestistä löytyy otsikkolohkoelementti, jonka rooliattribuutin roolina on next ja kyseissä viestiketjussa on mukana välittäjäsolmuja, niin siinä tapauksessa kyseinen otsikkolohkoelementti tulee käsitellä ensimmäisessä välittäjäsolmussa (joka siis on myös viestin ensimmäinen vastaanottaja). Käsiteltyään itselleen kuuluvan otsikkolohkoelementin, tulee SOAP-solmun poistaa kyseinen elementti viestistä ennen sen eteenpäin lähettämistä. SOAP-solmussa voidaan myös lisätä uusia otsikkolohkoelementtejä SOAP-viestiin. Tarvittaessa SOAP-viestiin voidaan lisätä kopio jo käsitellystä otsikkolohkoelementistä, jos halutaan että vastaavanlainen otsikkolohkoelementti käsitellään myös viestiketjun seuraavassa solmussa (W3C, 2003b).

Rooliattribuutin roolilla *none* ilmaistaan, että kyseistä otsikkolohkoelementtiä ei käsitellä ainoassakaan viestiketjun solmussa. Kyseisen otsikkolohkoelementin tarkoituksena voi olla sellaisen tiedon kuljettaminen, jota käytetään muiden otsikkolohkoelementtien käsittelemiseen. Kun otsikkolohkoelementin rooliattribuutin rooliksi on asetettu *ultimateReceiver*, tulee kyseinen otsikkolohkoelementti käsitellä vasta SOAP-viestin lopullisessa vastaanottajasolmussa. Otsikkolohkoelementin käsittelysääntö on tällöin sama kuin rooliattribuuttia ei olisi käytetty lainkaan (W3C, 2003b).

SOAP:ia käyttävän sovelluksen suunnittelija voi myös itse määritellä SOAP-solmuille rooleja. SOAP-solmujen roolien tunnistaminen tulee hoitaa sovelluskohtaisin menetelmin. Rooliattribuutin käyttö ei ole sallittu otsikkolohkoelementtien ulkopuolella. Tarvittaessa välittäjäsolmuissa voidaan myös muuttaa rooliattribuuttien arvoja (W3C, 2003b).

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <p:lohko1 xmlns:p="http://example.com"
        env:role="http://example.com/Log"
          env:mustUnderstand="true">
    ...
    </p: lohko1>
    <q:lohko2 xmlns:q="http://example.com"
     env:role=”http://www.w3.org/2003/05/soap-envelope/role/next”>
    ...
    </q: lohko2>
    <r:lohko3 xmlns:r="http://example.com">
    ...
    </r: lohko3>
  </env:Header>
  <env:Body >
    ...
</env:Envelope>
```

Kuva 11. Rooliattribuuttien käyttö otsikkolohkoelementeissä (W3C, 2003b).

Kuvassa 11 on kuvattu SOAP-viesti, joka sisältää kolme otsikkolohkoelementtiä. Ensimmäisen otsikkolohkoelementin rooliattribuutin arvoksi on asetettu sovelluskohtaisesti määritelty arvo http://example.com/Log. Kyseisen otsikkolohkoelementin käsittelee ensimmäinen viestiketjun solmu, jonka roolille arvo on osoitettu. Toisen otsikkolohkoelementin rooliattribuutin arvo määrää, että elementin käsittely tulee tapahtua viestiketjun seuraavassa solmussa. Kuvan kolmannessa otsikkolohkoelementissä ei ole käytetty rooliattribuuttia lainkaan. Tällöin kyseisen otsikkolohkoelementin käsittely tapahtuu viestin lopullisessa vastaanottajasolmussa.

Rooliattribuuttiin liittyy läheisesti *eteenpäinsiirtoattribuutti* (relay), joka on xml-kaavassa määriteltyä boolean tyyppiä. Eteenpäinsiirtoattribuutti otetaan otsikkolohkoelementissä käyttöön seuraavasti: ”nimiavaruuden tunnus:relay=true”. Eteenpäinsiirtoattribuuttia käytetään kun halutaan, että välittäjäsolmulle kohdennettu otsikkolohkoelementti lähetetään viestipolussa eteenpäin otsikkolohkoelementin käsittelyn epäonnistuttua. Ilman eteenpäinsiirtoattribuutin käyttöä otsikkolohkoelementti poistetaan viestistä, vaikka käsittely epäonnistuisikin. Tämän seurauksena poistettua otsikkolohkoelementtiä ei käsitellä SOAP-viestin siirron aikana missään vaiheessa. Eteenpäinsiirtoattribuutin arvon asettaminen epätodeksi (false), tarkoittaa

samaa kuin attribuuttia ei käytettäisi lainkaan. Jos viestin vastaanottaja on viestin lopullinen vastaanottajasolmu, niin prosessoinnin yhteydessä eteenpäinsiirtoattribuutti jätetään huomioimatta. Eteenpäinsiirtoattribuutin arvo voidaan muuttaa viestin siirron aikana (W3C, 2003b).

```
<env:Envelope xmlns:env=”http://www.w3.org/2003/05/soap-envelope”>
  <env:Header>
    ...
    <q:lohko2 xmlns:q="http://example.com"
      env:role=”http://www.w3.org/2003/05/soap-envelope/role/next”
      env:relay=”true”>
    ...
    </q: lohko2>
  </env:Header>
    ...
```

Kuva 12. Eteenpäinsiirtoattribuutin käyttö otsikkolohkoelementeissä (W3C, 2003b).

Kuvan 12 esimerkissä otsikkolohkoelementissä lohko2 on asetettu eteenpäinsiirtoattribuutin arvoksi true. SOAP-solmu poistaa käsittelemänsä otsikkolohkoelementin viestistä huolimatta siitä onnistuiko käsittely vai ei. Koska kuvan 12 esimerkissä on käytetty otsikkolohkoelementissä eteenpäinsiirtoattribuuttia, niin poistoa ei tehdä, mikäli käsittely epäonnistuu.

3.2.6 Otsikkolohkon ymmärtämisattribuutti

Otsikkolohkon ymmärtämisattribuutti on XML-kaavassa määriteltyä boolean tyyppiä ja täten sen arvo voi olla joko true tai false. Vaihtoehtoisesti voidaan käyttää myös arvoja 1 tai 0. Ymmärtämisattribuutti otetaan käyttöön seuraavasti:

”nimiavaruuden tunnus:mustUnderstand=true".

Kun solmun rooli osoittaa, että solmun tulee käsitellä otsikkolohkoelementtiä, jonka ymmärtämisattribuutille on asetettu arvoksi true, täytyy solmun kyetä käsittelemään kyseinen otsikkolohkoelementti oikein. Mikäli solmu ei tähän kykene, sen täytyy generoida asiasta virheilmoitus ja lopettaa SOAP-viestin käsittely. Mikäli ymmärtämisattribuutin arvo on false, niin otsikkolohkoelementin käsittely ei ole välttämätöntä prosessin jatkumisen kannalta. Semanttisuudeltaan false vastaisi samaa kuin ymmärtämisattribuuttia ei käytettäisi lainkaan (W3C, 2003b).

Mikäli otsikkolohkoelementissä käytetään ymmärtämisattribuuttia ja eteenpäinsiirtoattribuuttia siten, että molempien arvoina on true, niin silloin ymmärtämisattribuutti on vahvempi, eli sen vaikutus jää voimaan ja otsikkolohkoelementtiä ei siirrettä viestiketjussa eteenpäin. Jos SOAP-viestiä käsittelevä sovellus tunnistaa ymmärtämisattribuutin muualla kuin otsikkolohkoelementin sisällä, niin se jätetään huomioimatta, eikä asiasta generoida virheilmoitusta. Välittäjäsolmuissa voidaan tarvittaessa muuttaa SOAP-viestissä olevien ymmärtämisattribuuttien arvoja (W3C, 2003b).

```
<SOAP-KEHYS:Header>
   <o:omaElementti xmlns:o=”http://esimerkkikone.fi/pakollinen_nimiavaruus/”
      SOAP-KEHYS:encodingStyle=“http://esimerkkikone.fi/omat_tyylit/ ”
      SOAP-KEHYS:role=”http://www.w3.org/2003/05/soap-envelope/role/next”
      SOAP-KEHYS:mustUnderstand=”true”
      SOAP-KEHYS:relay=”true”>
            … tietoa …
   </o:omaElementti>
 </ SOAP-KEHYS:Header>
```

Kuva 13. Otsikkolohkon ymmärtämisattribuutin käyttöönottaminen.

Kuvassa 13 otsikkolohkoelementille omaElementti on asetettu ymmärtämisattribuutin arvoksi true. Otsikkolohkoelementin rooliattribuutin rooli on next, joka puolestaan aktivoi seuraavan SOAP-solmun, joka ottaa viestin vastaan. Ymmärtämisattribuutin käytöllä taataan, että jos kyseisellä roolilla toimiva solmu ei kykene ymmärrettävästi käsittelemään otsikkolohkoelementtiä omaElementti, niin viestin prosessointi lopetetaan ja generoidaan SOAP-virheviesti. Kuudennella rivillä olevalla eteenpäinsiirtoattribuutin käytöllä ei ole tässä tapauksessa vaikutusta, koska ymmärtämisattribuutin arvona on true.

3.2.7 Runkoelementti

Runkoelementti (body) sisältää SOAP-viestissä lähetettävän varsinaisen datan, joka on esitetty XML-muodossa. Runkoelementtiä saa SOAP-viestin lähettämisen jälkeen käsitellä ainoastaan SOAP-solmussa, joka on viestin lopullinen vastaanottaja. Runkoelementin käyttö on määritelty SOAP-viestissä pakolliseksi, joskin se voi olla myös ilman sisältöä. Jos SOAP-viestin prosessoinnin yhteydessä on generoitu vikaelementti, niin sen tulee olla runkoelementin sisällä (W3C, 2003b). Vikaelementtiä käsitellään tarkemmin alakohdassa 3.2.8.

Runkoelementin lapsielementit voivat sisältää muiden elementtien tavoin nimiavaruuksien viittauksia. Nimiavaruuksien käyttöä suositellaan runkoelementin lapsielementtien kohdalla, koska nimiavaruuksien avulla tehtävien mahdollisten tarkastuksien ansiosta elementtien tulkinta helpottuu (W3C, 2003b).

SOAP-viestin runkoelementtiin sisältyville lapsielementeille voivat toteuttajat itse määritellä nimiavaruuksia tai ottaa vaihtoehtoisesti käyttöön jonkun toisen määrittelemän nimiavaruuden. Lapsielementeissä tapahtuville nimiavaruuksien viittauksille pätevät samat säännöt kuin kuorielementissä tapahtuville nimiavaruuksien viittauksille (W3C, 2000).

3.2.8 Vikaelementti

Vikaelementti (fault) on erikoiselementti, joka generoidaan silloin, kun SOAP-solmu ei kykene käsittelemään ymmärrettävästi vastaanottamaansa SOAP-viestiä. Vika voi aiheutua mm. viestin puutteellisesta rakenteesta, pakollisesta otsikkolohkoelementistä, puuttuvista arvoista, joita vaaditaan viestin prosessoinnissa tai käytetystä SOAP-versiosta, jolle SOAP-solmussa ei löydy tukea. Vikaelementti sisältää tiedon virheestä ja sen laadusta sekä mahdollisesta virheen aiheuttajasta. Esiintyessään vikaelementti on runkoelementin ainoa suoranainen lapsielementti. Vikaelementille on määritelty kaksi pakollista lapsielementtiä - koodielementti ja syyelementti (W3C, 2003b).

Koodielementille (code) on määritelty kaksi lapsielementtiä. Nämä lapsielementit ovat pakollinen *arvoelementti* (value) ja valinnainen alikoodielementti (subcode). Kun arvoelementtiä käytetään koodielementin suoranaisena lapsielementtinä, niin se voi pitää sisällään ainoastaan SOAP:in virhekoodeja. *Virhekoodit* ovat faultCodeEnum-tyyppisiä, lueteltua tyyppiä olevia arvoja, joiden avulla voidaan luokitella SOAP-viestin käsittelyn aikana tapahtuneet virheet (W3C, 2003b). Tietotyypin faultCodeEnum mahdolliset arvot selitteineen on lueteltu taulukossa 1.

Taulukko 1. Tietotyypin faultCodeEnum arvot (W3C, 2003b).

Arvo	*Selite*
VersionMismatch	SOAP-viestiä käsittelevä solmu ei tue käytettyä SOAP-versiota tai kuorielementin tunnus ei ole vaatimuksen mukainen.
MustUnderstand	Viestiä käsittelevä SOAP-solmu ei pysty ymmärrettävästi käsittelemään sille kohdistettua pakollista otsikkolohkoelementtiä.
DataEncodingUnknown	Viestiä käsittelevä SOAP-solmu ei tue kooditustyyliattribuutilla osoitettua nimiavaruutta.
Sender	SOAP-viesti on väärin muodostettu tai siitä puuttuu viestin prosessoinnin kannalta välttämättömiä arvoja tai ne ovat väärässä järjestyksessä. Kun lähettäjäsolmussa vastaanotetaan Sender-tyyppiä oleva virhe, niin sovelluksen tulee huolehtia siitä, että täsmälleen samaa viestiä ei lähetetä vastaanottavaan solmuun uudelleen ennen kuin siihen on tehty korjaus.
Receiver	SOAP-viestiketjun lopullinen vastaanottajasolmu ei kykene käsittelemään viestiä. Tämän tyyppinen virhe ei aseta vaatimusta alkuperäisen viestin uudelleen muokkaamisesta ennen uutta lähetystä, sillä virhe on voinut johtua myös siitä, että lopullinen vastaanottajasolmu ei ole saanut alkuperäistä viestiä kokonaan vastaanotetuksi.

Alikoodielementin avulla voidaan ilmaista tarkempaa tietoa virheen laadusta. Elementin rakenne on samanlainen kuin koodielementin, eli se sisältää pakollisen arvoelementin ja valinnaisen alikoodielementin. Alikoodielementin arvoelementti sisältää tarkennettua tietoa (tai itse asiassa tarkennetun arvon) koodielementin tai edeltävän alikoodielementin arvoelementin arvolle. Alikoodielementin arvoelementin arvot ovat lueteltua tyyppiä, jotka on määritelty sovelluskohtaisesti tai SOAP-laajennuksen yhteydessä. Rakenne sallii alikoodielementille äärettömän määrän itsensä kaltaisia perillisiä, tosin sillä rajoituksella, että yhtä sukupolvea kohden voi olla vain yksi lapsielementti (W3C, 2003b). Kuvan 14 esimerkissä on kuvattu SOAP:in virheviesti, kun virheen on aiheuttanut viestin alkuperäinen lähettäjä.

```
<env:Envelope xmlns:env=”http://www.w3c.org/2003/05/soap-envelope”
              xmlns:rpc=”http://www.w3.org/2003/05/soap-rpc”
              xmlns:xml=”http://www.w3.org/XML/1998/namespace”>
<env:Body>
  <env:Fault>
    <env:Code>
      <env:Value>env:Sender</env:Value>
        <env:Subcode>
            <env:Value>rpc:BadArguments</env:Value>
        </env:Subcode>
    </env:Code>
    <env:Reason>
        <env:Text xml:lang=”en-US”>Processing error</env:Text>
        <env:Text xml:lang=”FI”>Virhe viestin käsittelyssä</env:Text>
    </env:Reason>
    <env:Detail>
        <v:virheenTiedot
          xmlns:v=”http://esimerkkikone.fi/virheet”>
            <v:selite> Muuttuja sisältää ei-numeerisen arvon</v:selite>
            <v:virhekoodi>123</v:virhekoodi>
        </ v:virheenTiedot>
    </env:Detail>
  </env:Fault>
</env:Body>
</env:Envelope>
```

Kuva 14. SOAP-virheviesti (W3C:n (2003b) esimerkin pohjalta).

Kuvan 14 esimerkissä kuorielementin yhteydessä tehdään viittaukset kolmeen nimiavaruuteen. Virheen on aiheuttanut pyynnön mukana tullut väärän tyyppinen arvo, joka on estänyt ohjelman suorituksen. Esimerkissä koodielementin arvoelementin arvo ilmaisee virheen aiheuttajan. *Syyelementti* ilmaisee sanallisen selityksen virheestä. *Tekstielementin* (text) ja *kieliattribuutin* (lang) käytöllä voidaan syyelementin sisältämä tieto kieliversioida. Kieliattribuutin arvo on lyhenne käytetystä kielestä ja esimerkiksi arvo en-US tarkoittaa amerikanenglantia ja arvo FI puolestaan suomea. Tekstielementin yhteydessä kieliattribuutin käyttäminen on pakollista. Kuvan esimerkissä on käytetty myös valinnaista yksityiskohtaelementtiä, jossa on annettu sovelluskohtaista lisätietoa virheen laadusta.

Kuvan 14 esimerkissä kieliattribuutti otetaan käyttöön viittaamalla XML:n nimiavaruuden arvoon http://www.w3.org/XML/1998/namespace/. XML:n määrityksissä on sidottu tämän nimiavaruuden arvon tunnukseksi lyhenne xml. Tämä arvo eroaa muista mahdollisista nimiavaruuksien arvoista, joille voidaan määritellä tunnus sovelluskohtaisesti (W3C, 2003c).

Pakollisten koodi- ja syyelementin lisäksi vikaelementille on määritelty kolme muuta mahdollista lapsielementtiä, joiden käyttäminen on valinnaista. Nämä elementit ovat solmuele-

mentti, roolielementti ja yksityiskohtaelementti. *Solmuelementin* (node) käyttö on pakollista niissä tapauksissa, jolloin SOAP-virheviesti generoidaan välittäjäsolmuissa. Solmuelementti toimii apuna virheen jäljittämisessä, sillä sen avulla paikallistetaan viestiketjusta paikka, jossa virhe tapahtui. Solmuelementti sisältää arvona virheen generoineen SOAP-solmun URI-osoitteen (W3C, 2003b).

Roolielementillä (role) puolestaan ilmaistaan virheen generoineen SOAP-solmun rooli tai yksi rooleista, jos SOAP-solmulla on niitä useampi. Roolielementti on XML-kaavassa määriteltyä tyyppiä anyURI ja sen arvona voivat olla samat arvot kuin mitkä rooliattribuuteille on annettu (W3C, 2003b). *Yksityiskohtaelementin* (detail) tarkoituksena on antaa mahdollisia lisätietoja koodielementissä kuvatulle virheelle. Yksityiskohtaelementin sisältämät tiedot generoidaan aina sovelluskohtaisesti (W3C, 2003b).

Mikäli viestin vastaanottava SOAP-solmu ei tue käytettyä SOAP-viestin versiota, niin SOAP-virheviestiin tulisi sisällyttää tieto tuetuista versioista. Tätä tarkoitusta varten on määritelty korjauselementti (upgrade), joka sijaitsee otsikkoelementin lapsielementtinä. *Korjauselementti* sisältää lapsielementtinään yhden tai useamman tukielementin (SupportedEnvelope). Mikäli virheviestin generoinut SOAP-solmu tukee useita SOAP:in versioita, niin *tukielementtien* keskinäisellä järjestyksellä ilmaistaan versioiden paremmuusjärjestys virheen generoineen SOAP-solmun kannalta. Tukielementti sisältää qname-attribuutin, jossa kuorielementin tunnukselle annetaan sisäinen nimiavaruuden tunnus ja tämän tunnuksen arvoksi asetetaan nimiavaruuden viittauksella tuetun SOAP-version nimiavaruuden URL-osoite (W3C, 2003b).

```
<env:Header>
  <env:Upgrade>
   <env:SupportedEnvelope qname="ns1:Envelope"
          xmlns:ns1="http://www.w3.org/2003/05/soap-envelope"/>
   <env:SupportedEnvelope qname="ns2:Envelope"
          xmlns:ns2="http://schemas.xmlsoap.org/soap/envelope/"/>
  </env:Upgrade>
 </env:Header>
```

Kuva 15. Korjauselementin käyttö virheviestin otsikkoelementissä (W3C, 2003b).

Kuvassa 15 on esimerkki korjauselementin käytöstä osana SOAP-virheviestin otsikkoelementtiä. Korjauselementti sisältää tiedon siitä, että virheen generoinut SOAP-solmu tukee kahta SOAP-versiota. Ensimmäinen tukielementti ilmaisee, että ensisijaisesti virheen generoineeseen solmuun tulisi lähettää SOAP 1.2 -spesifikaation mukaan muodostettuja SOAP-

viestejä, sillä paikallisen kuorielementin tunnuksen etuliitteessä on viitattu nimiavaruuteen http://www.w3.org/2003/05/soap-envelope/, joka siis on SOAP 1.2 -versiolle määritelty arvo. Toisessa tukielementissä on paikallisen kuorielementin tunnuksen etuliite laitettu viittaamaan nimiavaruuteen http://schemas.xmlsoap.org/soap/envelope/, joka on SOAP 1.1 -version vastaava arvo. Tällä ilmaistaan, että myös SOAP 1.1 -version mukaan muodostettuja SOAP-viestejä tuetaan tässä solmussa, vaikkakin implisiittisesti suositellaan käyttämään ensimmäisen tukielementin ilmaisemaa versiota.

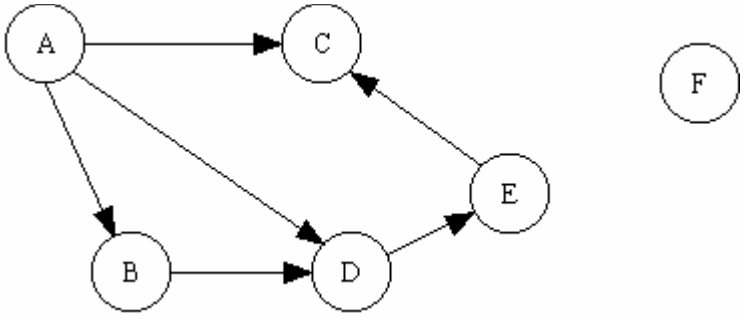
Ymmärtämättömyyselementtiä (NotUnderstood) käytetään silloin kun SOAP-virheviestin koodielementin arvoelementti saa arvokseen MustUnderstand taulukon 1 mukaisesti. Esiintyessään ymmärtämättömyyselementti on virheviestissä otsikkolohkoelementtinä. Ymmärtämättömyyselementillä ilmaistaan, että SOAP-virheviestin generoinut SOAP-solmu ei kykene käsittelemään sille pakolliseksi määriteltyä otsikkolohkoelementtiä. Ymmärtämättömyyselementti sisältää qname-attribuutin, jonka arvoksi asetetaan paikallinen nimiavaruuden tunnus ja virheen aiheuttaneen otsikkolohkoelementin tunnus kaksoispisteellä eroteltuina. Lisäksi ymmärtämättömyyselementissä on nimiavaruuden tunnus asetettu viittaamaan virheen aiheuttaneen otsikkolohkoelementin viittaamaan nimiavaruuteen. SOAP-virheviestiin voidaan generoida useita ymmärtämättömyyselementtejä mikäli solmu, jossa virhe aiheutui, ei kykene käsittelemään useampia sille pakolliseksi määriteltyjä otsikkolohkoelementtejä (W3C, 2003b).

3.3 SOAP-tiedon mallinnus

SOAP-tiedon mallinnuksella määritellään tapa esittää sovelluskohtaisesti määritellyt tietorakenteet suunnattuna verkkona, joka voi sisältää nimiöidyt kaaret. SOAP-tiedon mallinnuksen käyttö on vapaaehtoista, eikä sen käyttäminen ole tarpeen jos sovelluksessa jo käytetään jotakin XML-mallinnusta tiedon esittämisessä. SOAP-tiedon mallinnus tukee SOAP-kooditussääntöä, joka otetaan käyttöön SOAP-viestin elementissä asettamalla kooditustyyliattribuutin arvoksi nimiavaruus http://www.w3.org/2003/05/soap-encoding. SOAP-tiedon mallinnuksessa esitettyjen sääntöjen perusteella voidaan tietorakenne sarjallistaa SOAP-viestin siirtämisen ajaksi ja purkaa lopullisessa vastaanottajasolmussa (W3C, 2003c).

SOAP-tiedon mallinnuksessa käytetään kahta sääntöä. Ensimmäisen säännön mukaan jokainen elementti esittää kaarta ja mikäli kyseinen elementti ei sisällä viittausattribuutin (ref) käyttöä, niin sama elementti esittää myös solmua. Tämän elementin esittämä kaari puolestaan osoittaa elementissä esitettyä solmua. Kyseessä on siten implisiittisesti kuvattu tapaus, kun solmusta lähtevä kaari osoittaa samaan solmuun. Solmu kyetään tunnistamaan käyttäen sen esittämän elementin kanssa *tunnisteattribuuttia* (id). Tunnisteattribuutilla tulee olla kuorielementin sisällä uniikki arvo.

Toisen säännön mukaan elementti sisältää viittausattribuutin, jolloin elementti esittää ainoastaan kaarta. *Viittausattribuutin* arvoksi asetetaan jokin arvo, jolle löytyy vastine SOAP-viestissä esitettyjen tunnisteattribuuttien joukosta. Elementissä, jossa käytetään viittausattribuuttia, ei saa käyttää tunnisteattribuuttia. Solmusta voi lähteä kaari, joka ei osoita mihinkään solmuun. Samoin kaari voi tulla solmuun, ilman että se lähtee mistään solmusta (W3C, 2003c).



Kuva 16. Suunnattu verkko.

Kuvassa 16 on kuvattu suunnattu verkko, jossa on näkyvissä kuusi solmua ja kuusi kaarta. SOAP-tiedon mallinnuksen sääntöjen mukaan jokaisesta solmusta lähtee lisäksi kaari itseensä, joten näkyvien kaarien lisäksi verkko sisältäisikin tässä tapauksessa kuusi muuta kaarta, eli kaarien lukumäärä olisi yhteensä 12.

```
<v:verkko xmlns:v=”http://http://esimerkkikone.fi/verkko”
          xmlns:enc=”http://www.w3.org/2003/05/soap-encoding”>
  <v:Kaari enc:id=’A’>
    <v:Kaari enc:ref=’C’/>
    <v:Kaari enc:ref=’D’/>
    <v:Kaari enc:ref=’B’/>
  </v:Kaari>
  <v:Kaari enc:id=’B’>
    <v:Kaari enc:ref=’D’/>
  </v:Kaari>
  <v:Kaari enc:id=’C’/>
  <v:Kaari enc:id=’D’>
    <v:Kaari enc:ref=’E’/>
  </v:Kaari>
  <v:Kaari enc:id=’E’>
    <v:Kaari enc:ref=’C’/>
  </v:Kaari>
  <v:Kaari enc:id=’F’/>
</v:verkko>
```

Kuva 17. SOAP-tiedon mallinnuksen mukaan kuvattu verkko.

Kun lisäämme SOAP-tiedon mallinnuksen sääntöihin uuden säännön, jonka mukaan solmun kuvaavan elementin lapsielementit ovat solmusta lähteviä kaaria, voimme esittää kuvassa 16 kuvatun verkon XML-muodossa kuvassa 17 esitellyllä tavalla.

SOAP-koodituss ääntöjen mukaan solmuja kuvaavien elementtien kanssa voidaan käyttää *solmutyyppiattribuuttia* (nodeType). Jos attribuuttia käytetään, niin sen arvona voi olla yksi seuraavista: simple, array tai struct. Attribuutin arvolla ilmaistaan, sisältääkö solmu yksinkertaista tietotyyppiä olevan arvon, taulukon vai tietueen (W3C, 2003c).

Taulukkotyyppiä olevan solmun esittämistä varten on määritelty alkiotyyppiattribuutti (itemType) ja taulukonkokoattribuutti (arraySize). Näiden attribuuttien käyttäminen on vapaaehtoista. *Alkiotyyppiattribuutilla* ilmaistaan taulukon alkioiden arvoina käytetty tietotyyppi. EBNF-notaatiota käyttäen taulukonkokoattribuutin arvo ilmaistaan seuraavien sääntöjen mukaisesti (W3C, 2003c).

[1] taulukon koon arvo ::= (”*” | Koko) seuraavaKoko*
[2] seuraavaKoko ::= Tyhje Koko
[3] Koko ::= [0-9] +
[4] Tyhje ::= (#x20 | #x9 | #xD | #xA) +

Taulukonkokoattribuutin arvo ilmaisee taulukon sisältämien alkioiden määrän ja arvona voidaan kokonaisluvun lisäksi käyttää merkkiä ’*’ (asteriski), jolla ilmaistaan, että taulukko voi sisältää 0..*n* kappaletta alkioita. Taulukonkokoattribuutti voi sisältää useamman aliarvon, jotka erotellaan toisistaan tyhjeellä (tai tyhjeillä). Tyhjettä ei voida käyttää arvon ensimmäisenä, eikä viimeisenä merkkinä. Jokainen attribuutin sisältämä aliarvo ilmaisee yksittäisen ulottuvuuden sisältämien alkioiden lukumäärän, joten esimerkiksi attribuutin arvo ”4 8 2” tarkoittaa, että kyseessä on 3-ulotteinen taulukko, jonka sisältämien alkioiden lukumäärä on yhteensä 4*8*2=64. Asteriski-arvoa voidaan käyttää ainoastaan ensimmäisen ulottuvuuden aliarvona (W3C, 2003c).

```
<t:taulukko xmlns:enc=http://www.w3.org/2003/05/soap-encoding enc:arraySize="3 2 2">
    <t:alkio> 111 </t:alkio>
    <t:alkio> 112 </t:alkio>
    <t:alkio> 121 </t:alkio>
    <t:alkio> 122 </t:alkio>
    <t:alkio> 211 </t:alkio>
    <t:alkio> 212 </t:alkio>
    <t:alkio> 221 </t:alkio>
    <t:alkio> 222 </t:alkio>
    <t:alkio> 311 </t:alkio>
    <t:alkio> 312 </t:alkio>
    <t:alkio> 321 </t:alkio>
    <t:alkio> 322 </t:alkio>
</t:taulukko>
```

Kuva 18. SOAP-kooditussäännön mukainen kolmiulotteinen taulukko.

Kuvassa 18 on esitetty kolmiulotteinen taulukko. Koska esimerkissä ei ole attribuutilla kerrottu taulukon alkioiden arvojen tyyppiä, niin arvot ovat oletustyyppiä, eli merkkijonoja. Alkioiden arvojen tyyppi olisi voitu eksplisiittisesti ilmaista kahdella tavalla. Ensimmäinen tapa olisi ollut käyttää taulukkoelementissä alkiotyyppiattribuuttia ja toinen tapa olisi ollut käyttää jokaisessa alkioelementissä tyyppiattribuuttia, jolloin esimerkiksi taulukon kolmas alkioelementti olisi merkitty seuraavasti: <t:alkio na1:type=”na2:String”>121</t:alkio>, missä na1 ja na2 ovat joidenkin määriteltyjen nimiavaruuksien tunnuksia. Taulukossa kaikkien alkioiden arvoja sisältävien elementtien tunnukset ovat samoja ja elementit järjestetään position mukaan nousevassa järjestyksessä alkaen vasemmalta oikealle (W3C, 2003c).

SOAP-kooditussäännöissä on määritelty myös tietue. Tietueessa solmua esittävän elementin tunnus ilmaisee tietueen nimen ja tämän elementin/solmun lapsielementit/kaaret ovat tietueen kenttiä. Kullekin lapsielementille tulee antaa toisistaan poikkeava tunnus (W3C, 2003c).

SOAP-kooditussäännöissä on määritelty tarkennusarvot kolmelle virhetyypille, jotka voivat aiheutua sarjallistamisen purun yhteydessä. Esiintyessään nämä tarkennusarvot sijoitetaan koodielementin lapsielementtinä olevan alikoodielementin arvoksi. Jokaisessa näissä virhetapauksessa virheviestin koodielementti saa arvokseen sender (W3C, 2003c). Tarkennukset ja selitteet on esitelty taulukossa 2.

Taulukko 2. SOAP-kooditussäännössä määritellyt virheen tarkennusarvot (W3C, 2003c).

Alikoodielementin arvo	*Selite*
MissingID	SOAP-viesti sisältää viittausattribuutin, jonka arvolle ei löydy vastinetta.
DuplicateID	SOAP-viesti sisältää kaksi tai useamman tunnisteattribuutin, joille on annettu sama arvo.
UntypedValue	Solmussa on käytetty tyyppiä, jota SOAP-viestiä käsittelevä SOAP-solmu ei tunnista.

3.4 SOAP-viestinvälitysmallit

Yhden loogisen tapahtuman suorittamiseksi SOAP-solmut välittävät sähköisiä viestejä toistensa kanssa. Yhteen tapahtumaan liittyy yksi tai useampi sähköinen viesti, jotka voivat olla SOAP-viestejä tai joitakin muita sähköisiä viestejä. *SOAP-viestinvälitysmalleissa* määritellään yhteen tapahtumaan liittyvät sähköiset viestinvälitykset SOAP-viestien näkökulmasta katsottuna. Yksinkertaisin esimerkki tapahtumasta on tiedonsiirto kahden SOAP-solmun välillä, jolloin viestin lähettävä SOAP-solmu muodostaa SOAP-viestin, joka siirretään mahdollisten välittäjäsolmujen kautta toiseen solmuun. Tällöin käytetään *SOAP-kutsu* nimistä viestinvälitysmallia. Jos edellä mainitun mukaisesta tiedonsiirrosta halutaan lähettää kuittausviesti alkuperäisen viestin lähettäneeseen solmuun, niin kyseessä on *SOAP-kutsu/vastaus* viestinvälitysmalli (W3C, 2003c).

Joskus voi tulla eteen tilanne, että tapahtumasta halutaan lähettää välituloksia kutsun esittäneelle SOAP-solmulle ja tällöin voidaan soveltaa *SOAP-kutsu/monivastaus* viestinvälitysmallia. Toisinaan taas yhden tapahtuman suorittamiseksi täytyy tehdä useampia kutsuja, ennen kuin vastaus voidaan muodostaa. Tällöin voidaan puhua *SOAP-monikutsu/vastaus* viestinvälitysmallista. Jos edellisessä tapauksessa ei haluta vastausviestiä, niin kyseessä olisi *SOAP-monikutsu* viestinvälitysmalli.

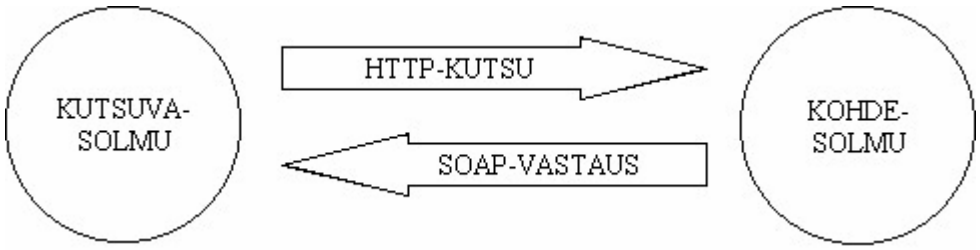
SOAP-viestinvälitysmalleissa huomioidaan ainoastaan SOAP-viestien kulku. Koska vastausta tulee aina edeltää kutsu, niin puhtaasti SOAP-viestejä käyttäen ei SOAP-vastaus viestinvälitysmallin käyttäminen ole mahdollista. Tätä mallia voidaan soveltaa esimerkiksi silloin, kun SOAP-solmusta lähetetään kutsu jollakin muulla tavoin kuin SOAP-viestiä käyttäen, esimerkiksi HTTP-kutsuna (W3C, 2003c).

3.5 SOAP RPC -malli

Yksi keskeisimmistä päämääristä SOAP:ia määriteltäessä oli mallintaa proseduurien etäkutsujen suorittaminen SOAP:ia hyödyntäen. Määrittelyn tarkoituksena oli tehdä mallista sellainen, että se mukailisi yleisimmin käytössä olevien ohjelmointikielien tapaa suorittaa proseduurien kutsuja. SOAP RPC -mallin käytön tukeminen SOAP-sovelluksissa on valinnaista, joten välttämättä kaikki SOAP-solmut eivät kykene käsittelemään SOAP RPC-viestejä mallin edellyttämällä tavalla. Sovelluskehittäjillä on vapaat kädet määritellä myös omia sovelluskohtaisia tapoja proseduurien etäkutsujen suorittamiseksi, kunhan ne eivät ole ristiriidassa SOAP:in määrityksiin nähden (W3C, 2003c).

SOAP RPC -malli mukautuu kohdassa 3.3 käsiteltyihin SOAP-tiedon mallinnukseen ja SOAP-kooditussääntöihin, joita hyödynnetään mallin mukaisten tietorakenteiden sarjallistamisessa ja sarjallistamisen purkamisessa. SOAP RPC -mallissa käytetyt SOAP-kutsu- ja SOAP-vastausviestit ovat yhteensopivia HTTP-protokollan HTTP-kutsu- ja HTTP-vastausviestien kanssa. Kuitenkaan mallia ei ole kiinnitetty yksin HTTP-protokollaan kanssa käytettäväksi, vaan sitä voidaan tarvittaessa käyttää myös muiden asiaan soveltuvien protokollien kanssa. Soveltuvuuden vaatimuksena on, että protokolla pystyy käyttämään HTTP-GET ja/tai HTTP-POST web-metodeita tai niitä vastaavia (mahdollisesti protokollan omia) web-metodeita (W3C, 2003c).

SOAP RPC -malli tukee SOAP-vastaus (kuva 19) ja SOAP-kutsu/vastaus (kuva 20) viestinvälitysmallien käyttöä. Myös muiden viestinvälitysmallien käyttäminen voi olla mahdollista, mutta niitä ei ole sisällytetty SOAP:in määritykseen. Solmua, johon proseduurin etäkutsupyyntö lähetetään, kutsutaan SOAP RPC -mallissa kohdesolmuksi. SOAP-kutsu/vastaus viestinvälitysmallia käytettäessä kohdesolmun URI voidaan kuljettaa osana otsikkolohkoelementtiä, jolloin kohdesolmun URI esitetään otsikkolohkoelementissä viitattavan nimiavaruuden arvona. Kohdesolmun URI voidaan kuljettaa myös SOAP-viestin ulkopuolella osana rinnakkaisprotokollan mukaista viestiä. Esimerkiksi HTTP-POST -metodia käytettäessä URI sijoitetaan HTTP-viestin alkuun. Kohdesolmun URI:n lisäksi SOAP RPC -kutsussa tulee luonnollisesti olla mukana kutsuttavan proseduurin/metodin nimi ja vaadittavat parametrit arvoineen (W3C, 2003c).



Kuva 19. SOAP-vastaus viestinvälitysmalli, käytettäessä HTTP-kutsua.

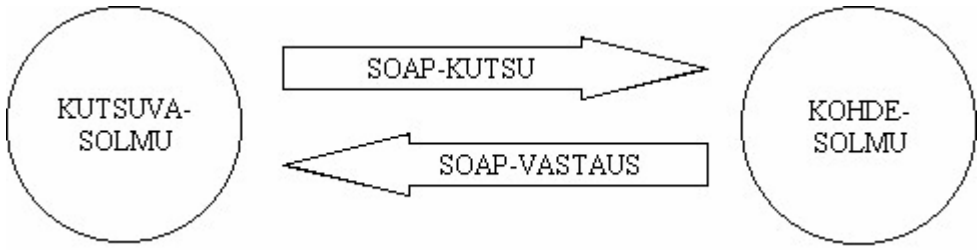
SOAP-vastaus viestinvälitysmallissa käytetään GET-metodia proseduurin etäkutsun toteuttamiseksi. Tällöin proseduuria/metodia kutsuvasta SOAP-solmusta ei lähetetä SOAP-viestiä kohdesolmuun, vaan kutsuttavan proseduurin/metodin nimi, parametrit arvoineen ja mahdolliset muut välitettävät tiedot liitetään kutsuttaessa kohdesolmun URI:iin (W3C, 2003c). Näiden tietojen liittäminen URI:iin riippuu käytetystä protokollasta ja esimerkiksi käytettäessä HTTP-protokollaa GET-metodin kanssa, kohdesolmun URI:n ilmaiseva merkkijono ja liitettävät tiedot erotetaan toisistaan ’?’-merkillä. Liitettävät tiedot esitetään tunnus/arvo-pareina, jotka erotetaan toisista tunnus/arvo -pareista ’&’-merkillä. Alla on esimerkki kohdesolmun URI:n muodosta:

”http://www.kohdesolmu.fi/applikaatio/haut.jsp?method=haeSaldo&Laji=625&Surr=09232”

Esimerkissä kohdesolmun URI päättyy kutsuttavan servletin tiedostonimeen ’haut.jsp’. Esimerkissä voidaan olettaa tiedoston nimestä (haut), joka on monikossa, että kyseinen servletti

sisältää useamman haku-tyyppisen metodin sisältävän rajapinnan ja method-tunnuksen arvo haeSaldo on myös kutsuttavan metodin tunnus. Laji- ja Surr-tunnukset ja niiden arvot vastaavat metodin haeSaldo parametreja ja arvoja.

Kohdesolmussa otetaan vastaan välitettävät tunnus/arvo -parit ja niiden perusteella muodostetaan SOAP-viesti, joka lähetetään vastauksena kutsun esittäneeseen SOAP-solmuun. Vastaus voi sisältää joko metodin suorituksen tuloksen tai ilmoituksen virheestä. Kuten SOAP-viestin luonteeseen kuuluu, niin vastausviesti on sidottu rinnakkaisprotokollaan, johon SOAP-viesti on upotettu. SOAP-vastaus viestinvälitysmallia voidaan käyttää turvallisten hakujen yhteydessä, mutta esimerkiksi siirrettäessä tallennusta varten suuria tietomääriä, sen käyttäminen ei ole perusteltua, koska URI:n ja liitettävien tietojen muodostaman merkkijonon pituus on rajallinen (W3C, 2003c).



Kuva 20. SOAP-kutsu/vastaus viestinvälitysmalli.

Käytettäessä SOAP-kutsu/vastaus viestinvälitysmallia (kuva 20) kutsuttavan proseduurin/metodin nimi esitetään runkoelementin lapsielementin tunnuksena. Periaatteiltaan tämä eroaa XML-RPC:stä, jossa kutsutun proseduurin/metodin nimi on ollut tätä tarkoitusta varten määritellyn elementin arvona. SOAP-kutsu/vastaus viestinvälitysmallin kanssa käytetään HTTP-POST tai sitä vastaavaa web-metodia ja kohdesolmun URI kuljetetaan joko SOAP-viestin ulkopuolella rinnakkaisprotokollan mukaisessa viestissä tai SOAP-viestin otsikkolohkoelementissä. Mikäli kohdesolmun URI sisällytetään otsikkolohkoelementtiin, niin URI esitetään siinä nimiavaruuden viittauksen arvona (W3C, 2003c).

SOAP-kutsu/vastaus viestinvälitysmallin käyttäminen on perusteltua silloin, kun kutsu sisältää laajan määrän tallennettavaa tietoa, välittäjäsolmuissa vaaditaan otsikkolohkoelementtien sisältämää tietoa tai kun kyseessä on kriittinen haku, jolloin haun suorittamisessa käytettäviä tunnus/arvo-pareja ei haluta kuljettaa kohdesolmun URI:in liitettynä (W3C, 2003c).

SOAP-kutsu/vastaus viestinvälitysmallissa kutsu esitetään tietue-elementtinä, jonka tulee olla runkoelementin ainoa lapsielementti. Tietue-elementin tunnus on kutsuttavan proseduurin/metodin nimi ja parametrit esitetään tietue-elementin lapsielementteinä, jolloin parametrien tunnukset ovat myös kyseisten lapsielementtien tunnuksia ja lapsielementit sisältävät parametrin arvon. Mikäli kutsun (ja myös vastauksen) mukana halutaan lähettää jotakin ylimääräistä tietoa, niin kyseiset tiedot tulee sisällyttää otsikkolohkoelementteihin (W3C, 2003c).

SOAP RPC -mallin mukaisessa vastausviestissä (kuvat 19 ja 20) vastaus esitetään runkoelementin lapsielementissä, josta voidaan tässä yhteydessä käyttää nimitystä vastauselementti ja joka on tyypiltään tietue. Vastauselementin tulee olla runkoelementin ainoa lapsielementti. Jos kutsun tulos on muuta kuin void-tyyppiä, niin se esitetään tuloselementin (result) arvona. Tällöin tuloselementin tunnukseen eteen tulee liittää nimiavaruuden arvon http://www.w3.org/2003/ 05/soap-rpc tunnus. Void-tyyppisen tuloksen kanssa ei saa käyttää kyseisen nimiavaruuden viittausta, eikä myöskään tuloselementtiä (W3C, 2003c).

SOAP RPC -mallin virheenkäsittelyssä on määritelty joukko mahdollisia virheitä, jotka voivat aiheutua RPC-viestiketjun suorituksen yhteydessä (W3C,2003c). Näissä virhetilanteissa käytetyt virhekoodit, virheen tarkennuskoodit ja niiden selitteet on esitelty taulukossa 3.

Taulukko 3. SOAP RPC-virheiden tarkennukset (W3C, 2003c).

Koodielementin arvo	*Alikoodielementin arvo*	*Selite*
Receiver	-	Kohdesolmu ei kykene käsittelemään viestiä jonkin tilapäisen virheen takia. Esimerkiksi muistia ei ole tarpeeksi.
DataEncodingUnknown	-	Kohdesolmu ei tue viestin sarjallistamisen purussa käytettävää kooditussääntöä.
Sender	ProcedureNotPresent	Kohdesolmu ei tunnista kutsussa käytettyä proseduuria. Alikoodielementin arvon etuliitteenä tulee käyttää nimiavaruuden http://w3.org/2003/05/soap-rpc tunnusta.

Taulukko 3. SOAP RPC-virheiden tarkennukset (W3C, 2003c) (jatk.).

Koodielementin arvo	*Alikoodielementin arvo*	*Selite*
Sender	BadArguments	Virhe aiheutuu kutsussa käytetyistä parametreista tai niiden arvoista. Parametreja voi puuttua, niiden tunnukset voivat olla vääriä tai arvot voivat olla sopimatonta tyyppiä. Alikoodielementin arvon etuliitteenä tulee käyttää nimiavaruuden http://w3.org/2003/05/soap-rpc tunnusta.

3.6 Rinnakkaisprotokollat

SOAP ei kykene kulkemaan verkossa itsenäisesti, vaan se tarvitsee viestien siirtoa varten jonkin tiedonsiirtoprotokollan tukea. Vaikka SOAP on suunniteltu ensisijaisesti kulkemaan HTTP-protokollan yhteydessä, niin sitä voidaan käyttää myös yhdessä muiden tiedonsiirtoprotokollien kanssa, kuten SMTP, POP ja FTP (Tsenov, 2002; Govindaraju & al., 2000). Tätä kahden protokollan käyttämistä rinnakkain kutsutaan sitomiseksi (binding).

3.6.1 HTTP

HTTP on sovellustason tiedonsiirtoprotokolla, jota käytetään www-ympäristöissä asiakkaan ja palvelimen välisten pyyntö- ja vastausviestien siirtämisessä. HTTP on nykyisin selvästi käytetyin tiedonsiirtoprotokolla Internetissä (Berners-Lee & al., 1997; Ibiblio.org, 2003).

W3C:n asettama tutkijaryhmä on määritellyt *SOAP-HTTP -sidoksen*. Nimensä mukaisesti sidos määrittelee tavan käyttää SOAP-protokollaa yhdessä HTTP-protokollan kanssa. Määritelty sidos ei ole ainoa tapa sitoa SOAP- ja HTTP-protokollia toisiinsa, eikä määritys sulje pois vaihtoehtoisia tapoja SOAP-protokollan sitomiseksi HTTP:hen tai johonkin muuhun sopivaan protokollaan. SOAP-solmua, joka noudattaa täydellisesti SOAP-HTTP -sidosta, kutsutaan SOAP-HTTP -sidokseen mukautuneeksi solmuksi (W3C, 2003c).

SOAP-HTTP -sidoksessa käytetään base-URI -elementtiä HTTP:n sääntöjen mukaisesti. Base-URI -elementissä on arvona HTTP-pyynnön URI-osoite tai vaihtoehtoisesti HTTP Content-location -otsikkokentän arvo. Base-URI kertoo mistä osoitteesta pyyntö saapuu kutsuttavaan solmuun (W3C, 2003c; Berners-Lee & al., 1997).

Sidoksen ansiosta SOAP-viestissä voidaan hyödyntää HTTP:ssä määriteltyjä vastauksen tilakoodistoja. Tilakoodiston palautusarvo 200 merkitsee, että pyyntö onnistui, ja 400 ja 500 -alkuiset palautusarvot kertovat puolestaan pyynnön epäonnistumisesta (W3C, 2003c).

Taulukossa 4 on lueteltu tilakoodiston mahdollisia arvoja ja niiden selitteitä. SOAP-HTTP -sidokseen mukautuvan solmun tulee vähintäänkin tunnistaa, mihin ryhmään tilakoodin arvo kuuluu. Ryhmät erottaa toisistaan kolminumeroisen luvun ensimmäisestä luvusta.

Taulukko 4. Tilakoodiston arvot ja selitteet (W3C, 2003c).

Tilakoodin arvo	*Selite*
200-alkuiset	Onnistunut pyyntö. Vastaanottava solmu on löytynyt ja se kykenee palauttamaan vastauksen.
300-alkuiset	Uudelleenohjaus. Pyynnön vastaanottaman solmun alkuperäinen sisältö on siirretty toiseen osoitteeseen. Pyyntö välitetään tämän alkuperäisen solmun kautta eteenpäin, jotta se löytää korvaavan solmun, jossa sijaitsee pyydetty sisältö.
400-alkuiset	Virhe pyynnössä tai pyyntö ei toteuta kaikkia tarvittavia ehtoja.
400	Virhe pyynnössä.
401	Pyynnön onnistunut suoritus vaatii autentikoinnin. Autentikointia ei ole suoritettu
405	HTTP-palvelin ei tue pyynnössä käytettyä web-metodia.
415	HTTP-palvelin ei tue pyynnössä käytettyä Content-type kentän arvoa.
500-alkuiset	Virhe HTTP-palvelimella. Palvelin ei kykene käsittelemään pyyntöä tai käsittelyn aikana tapahtuu virhe.

SOAP-HTTP -sidoksessa ei hyödynnetä kaikkia HTTP:n tarjoamia ominaisuuksia, sillä sidoksen tarkoituksena on tukea SOAP-solmujen keskinäistä kommunikointia. Jotta sidosta

voidaan käyttää täydellisesti ja voidaan olla varmoja, että SOAP-viestin sisältö pysyy muuttumattomana, tulee kaikkien tapahtumaketjuun osallistuvien solmujen (myös välittäjän roolissa olevien solmujen) tukea sidosta. Jos vaikka vain yhdessä välittäjäsolmussa käytetään HTTP-protokollan eri versioita kuin muissa solmuissa, niin sidoksen kaikkien ominaisuuksien hyödyntäminen ei välttämättä ole mahdollista (W3C, 2003c).

SOAP-HTTP -sidokseen mukautuvien järjestelmien tulee kyetä lähettämään ja vastaanottamaan viestejä, jotka on sarjallistettu mediatyyppiä ’application/soap+xml’ käyttäen. Järjestelmät voivat myös lähettää tai vastaanottaa viestejä, joissa käytetään jotakin muuta kuin ’application/soap+xml’ mediatyyppiä, mutta näiden vaihtoehtoisten mediatyyppien tukeminen ei ole pakollinen ominaisuus SOAP-HTTP -sidokseen mukautuneilla solmuilla (W3C, 2003c).

SOAP-HTTP -sidos tukee kahta HTTP:ssä määriteltyä web-metodia. Nämä tuetut web-metodit ovat POST ja GET. Kyseiset web-metodit vastaavat käyttötarkoitukseltaan vastaavia HTTP-POST ja HTTP-GET -metodeja. Lähettävässä SOAP-HTTP -sidokseen mukautuneessa solmussa tulee käyttää jompaa kumpaa näistä web-metodeista (W3C, 2003c).

SOAP-solmuille on määritelty eri tiloja suorituksen aikana. Pyynnön lähettävän solmun tilat on esitelty taulukossa 5 ja pyynnön vastaanottavan solmun tilat puolestaan taulukossa 6. Huomiota tulee kiinnittää siihen, että SOAP-HTTP -sidos tukee virtauskäsittelyä (streaming) ja tämän seurauksena solmun tila voi olla yhtäaikaisesti sekä vastaanottava että lähettävä, joka onkin määritelty omaksi tilaksi.

Taulukko 5. Pyytävän solmun tilat suorituksen eri vaiheissa (W3C, 2003c).

Tila	*Selite*
Alustava	Tässä tilassa oleva solmu on muodostanut HTTP-pyynnön ja tiedonsiirto on aloitettu. HTTP-pyynnön kentissä on kerrottu käytetty metodi ja vastauksessa hyväksyttävät mediatyypit.
Pyytävä	Pyynnön sisältävää viestiä lähetetään vastaanottavaan solmuun ja odotetaan vastausviestin tuloa.
Lähettävä + vastaanottava	Tässä tilassa solmu on siihen asti, kunnes pyynnön sisältämä viesti on kokonaisuudessaan lähetetty ja vastausviesti on saapunut.

Taulukko 5. Pyytävän solmun tilat suorituksen eri vaiheissa (W3C, 2003c) (jatk.).

Tila	*Selite*
Vastaanottava	Tila on vastaanottava kunnes vastausviesti on vastaanotettu kokonaisuudessaan.
Onnistunut tai epäonnistunut	Lopetustilat. Suorituksen jälkeen solmun tila jää asianmukaiseen lopetustilaan.

Taulukko 6. Pyynnön vastaanottavan solmun tilat suorituksen eri vaiheissa (W3C, 2003c).

Tila	*Selite*
Alustava	Vastaanottava solmu odottaa pyyntöä.
Vastaanottava	Pyynnön vastaanotto on aloitettu ja odotetaan, että vastaussanoma on muodostettu.
Vastaanottava + lähettävä	Solmun tila, kunnes pyyntö on käsitelty ja vastausviesti on lähetetty.
Onnistunut tai epäonnistunut	Päätetila, joka kertoo suorituksen onnistumisesta.

3.6.2 SMTP

SMTP (Simple Mail Transfer Protocol) on kansainvälisen IETF-organisaation vuonna 1982 julkaisema standardi, jossa on määritelty sähköisen viestin siirtäminen kahden palvelimen välillä siten, että vastaanottava palvelin tallentaa saapuneen viestin. SMTP-protokollaa käytetään yleisesti sähköpostiviestien lähettämisessä ja vastaanottamisessa (Postel, 1982).

SOAP:in määrityksessä ei oteta kantaa SOAP-viestin lähettämiseksi osana sähköpostiviestiä. Kuitenkin W3C-konsortio on julkaissut muistio-asteella olevan hahmotelman luodakseen suosituksen SOAP:in käytöstä sähköpostijärjestelmän yhteydessä. Hahmotelman mukaan sitominen voidaan toteuttaa kahdella vaihtoehtoisella tavalla. Ensimmäinen tapa on liittää SOAP-viesti osaksi sähköpostiviestin tekstiosaa ja toinen tapa on siirtää SOAP-viesti sähköpostiviestin liitetiedostona (W3C, 2003a).

```
From: a.oyvind@mycompany.example.com
To: reservations@travelcompany.example.org
Subject: Travel to LA
Date: Thu, 29 Nov 2001 13:20:00 EST
Message-Id: <EE492E16A090090276D208424960C0C@mycompany.example.com>
Content-Type: application/soap+xml

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary
     xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2001-12-20</p:departureDate>
     <p:departureTime>mid morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging
     xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```

Kuva 21. SOAP-viestin liittäminen sähköpostiviestin runko-osaa (W3C, 2003a).

Kuvan 21 esimerkissä on esitelty eräs tapa SOAP-viestin liittämiseksi osaksi sähköpostiviestin runko-osaa. Esimerkissä on kuvattu edestakaisen lentomatkan varauspyyntö, jonka varaaja on lähettänyt matkatoimistoon. Esimerkin kuusi ensimmäistä riviä ovat SMTP-standardin mukaisia kenttiä, joissa ilmaistaan tiedot viestin lähettäjästä, vastaanottajasta, aiheesta, lähetyspäivämäärästä, viestin yksilöivästä tunnuksesta ja viestin sisällön MIME-tyypistä. Seuraavalla rivillä on XML-dokumentin esittelyosa, jossa ilmaistaan käytetty XML-versio (W3C,

2003a). XML-esittelyosan jälkeen tulee kuorielementti, jossa on viitattu SOAP 1.2 -version nimiavaruuteen. Viesti sisältää kaksi otsikkolohkoelementtiä, joissa kuljetetaan tiedot varausnumerosta ja varaajasta (tässä tapauksessa pelkästään nimitieto). Kyseiset tiedot on osoitettu viestiketjun seuraavalle SOAP-solmulle, mutta esimerkistä ei selviä onko kyseinen solmu myös viestin lopullinen vastaanottajasolmu. Voidaan kuitenkin olettaa, että tiedot varaajasta ja varausnumerosta lisätään tarvittaessa viestiin uudelleen, mikäli viesti kulkee välittäjäsolmujen kautta. Edelleen voidaan olettaa, että viestin lopullinen vastaanottajasolmu kykenee prosessoimaan viestin automaattisesti ja lähettämään vastausviestin SMTP-osiossa ilmaistuun sähköpostiosoitteeseen.

3.7 Kilpailevat Tekniikat

Proseduurien etäkutsujen toteuttamiselle on olemassa lukuisia eri tekniikoita. Tässä kohdassa esitellään lyhyesti SOAP:in tunnetuimmat kilpailijat CORBA ja Java-RMI.

3.7.1 CORBA

OMG (The Object Management Group) on vuonna 1989 perustettu konsortio, jonka jäsenistö koostuu n. 800 ohjelmistoalan yrityksestä, yliopistosta ja ohjelmistojen loppukäyttäjistä. OMG-konsortion pääasiallisena tehtävä on tuottaa avoimia, kaikkien hyödynnettävissä olevia määrityksiä olioperustaisen sovelluskehityksen tueksi. CORBA:n lisäksi tunnetuin OMG:n julkaisema määritys on olioperustaisten järjestelmien mallinnuskieli UML (Unified Modeling Language) (OMG, 2002; Hoque, 1998).

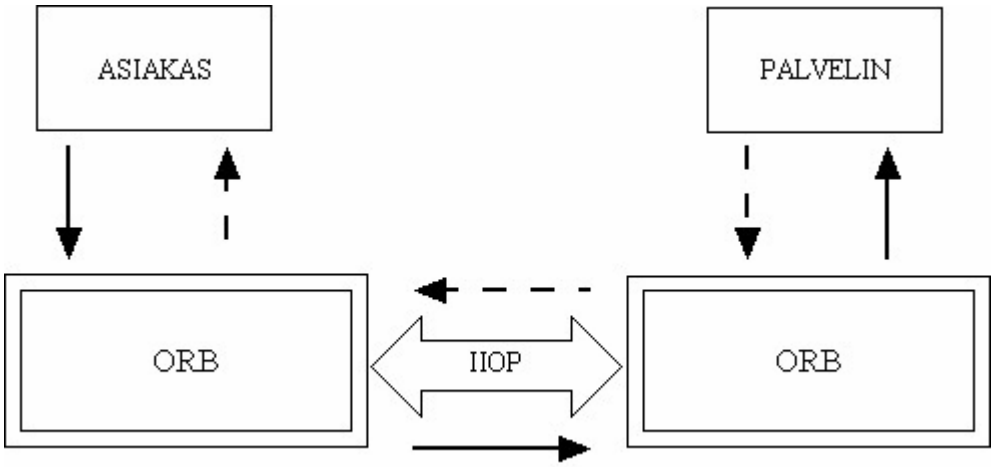
CORBA (Common Object Request Broker Architecture) on laajalti hyväksytty avoin arkkitehtuuri järjestelmien integrointia varten, ja se perustuu olioiden hajautusta tukevaan teknologiaan (Malek & al., 1999). CORBA:lla mahdollistetaan ohjelmistojen keskinäinen vuorovaikutus, ja sen avulla voidaan ylläpitää uudelleenkäytettäviä komponentteja (Chu & al.,1997).

CORBA koostuu useista spesifikaatioista, joissa määritellään olioiden yhteistoimivuus hajautetun tietojenkäsittelyn mallissa. Hoquen (1998) mukaan CORBA-arkkitehtuurissa palvelimet toimivat olioiden tuottajina ja asiakkaat puolestaan olioiden kuluttajina. CORBA-arkkitehtuurin keskeisin yksittäinen tekijä on ORB (Object Request Broker), joka huolehtii olioiden vä-

littämisestä asiakkaan ja palvelimen välillä. Perinteisen RPC-mallin tavoin myös CORBA:ssa tarjottavien palveluiden yksityiskohdat piilotetaan käyttäjiltä (Hoque, 1998).

ORB on suunniteltu ohjelmointikieli- ja alustariippumattomaksi väyläksi olioliikenteelle ja sitä käyttäen oliot voivat tehdä kutsuja ja saada vastauksia toisilta olioilta. Arkkitehtuurissa kukin oliopalvelu on yksilöity uniikilla nimellä ja palvelua voidaan kutsua tätä nimeä käyttäen. Asiakkaan ei tarvitse tietää kutsutun oliopalvelun sijaintia, sillä kaikki kutsut esitetään ORB-komponentille, joka huolehtii kutsutun palvelun paikallistamisesta, tietojen konfiguroimisesta palvelimen ymmärtämään muotoon ja kutsun edelleenlähettämisestä palvelimelle. Kun palvelin on toteuttanut kutsun edellyttämät operaatiot, niin vastaus lähetetään takaisin asiakkaalle ORB-komponentin välityksellä, ja tarvittaessa vastaus muunnetaan kutsujan ymmärtämään muotoon (Hoque, 1998).

CORBA:ssa määritelty IIOP-protokolla (Internet Inter ORB Protocol) vastaa SOAP:in rinnakkaisprotokollia. IIOP-protokollaa käytetään siirtämään olioita eri ORB-komponenttien välillä. HTTP:n tavoin IIOP-protokollan kuljetuskerroksena käytetään TCP/IP-protokollaa (Hoque, 1998).



Kuva 22. CORBA-arkkitehtuurin mukainen kutsu/vastaus –viestinvälitysmalli.

Kuvassa 22 on kuvattu CORBA-arkkitehtuurin kutsu/vastaus-viestinvälitysmalli. Kuvassa kutsun kulku on esitetty yhtenäisillä nuolilla ja vastauksen kulku puolestaan vastaavilla katkoviivoilla. Asiakas lähettää kutsun ORB-komponenttiin, joka paikallistaa kutsun kohteen sijainnin. Koska kohteen sijainti ei sijaitse tämän ORB-komponentin alueella, ORB-kompo-

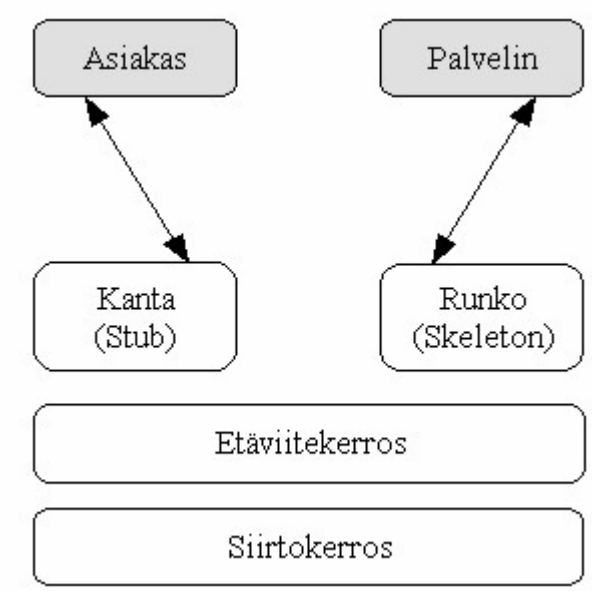
nentti lähettää kutsun eteenpäin toiseen ORB-komponenttiin IIOP-protokollan toimiessa rajapintana. Tämä ORB-komponentti lähettää kutsun edelleen palvelimelle, joka käsiteltyään kutsun lähettää takaisin päin vastauksen. Vastaus kulkee lähettäjälle samaa reittiä, mitä kutsukin, mutta päinvastaisessa järjestyksessä. Asiakas ja palvelin eivät ole missään vaiheessa suoraan tekemisessä toistensa kanssa, eikä heillä ole tietoa toistensa sijainnista.

SOAP:illa on samoja ominaisuuksia kuin tunnetuimmalla proseduurien etäkutsumekanismilla CORBA:lla. Lisäarvoa CORBA:an verrattuna SOAP tarjoaa siinä, että SOAP:issa käytetään tiedon siirtämisessä XML-dataa ja se kykenee siirtymään verkon välityksellä HTTP-protokollan avulla. CORBA-mekanismin ongelma on, että se on sidottu IIOP-protokollaan ja usein palomuurit on konfiguroitu siten, että ne päästävät läpi ainoastaan HTTP-protokollalla kulkevan liikenteen. Tämän takia CORBA:n käyttö webissä on hankalaa. Tämä CORBA:ssa oleva puute on korostunut viime aikoina, koska laajojen väärinkäytösten vuoksi lähes kaikki ylläpitäjät ovat asentaneet palvelintensa suojaksi palomuurin. Lisäksi SOAP:ia on helpompi käyttää kuin monimutkaista CORBA:a (Martin, 2001).

3.7.2 Java-RMI

Java-RMI on Sun Microsystemsin kehittämä tekniikka hajautettujen järjestelmien tuottamiseen. Java-RMI mahdollistaa eri verkkoasemilla sijaitsevien Java-perustaisten ohjelmien käyttävän toistensa oliokomponentteja. Tätä käyttöä varten palvelua pyytävä ohjelma tarvitsee viitteen kyseiseen oliokomponenttiin. Viite voidaan saada joko RMI:ssä käytössä olevasta nimeämispalvelusta tai kyselyn paluuarvona (Sun Microsystems, 2001). Kuvassa 23 on esitelty Java-RMI -järjestelmän arkkitehtuuri.

Java-RMI on tiukasti sidottu Java-ympäristöön eikä se tue yhteiskäyttöä eri ohjelmointikielillä rakennettujen järjestelmien välillä. Nämä tekijät johtavat siihen, että Java-RMI:tä ei voida pitää ratkaisuna etsittäessä yleistä mallia proseduurien etäkutsujen suorittamiselle. Sen sijaan Java-RMI soveltuu CORBA:a paremmin hajautettujen ratkaisujen malliksi, mikäli kyseessä ovat yksinkertaiset ja pienet järjestelmät. Monimutkaisissa, useiden asiakassovellusten käyttämissä ja suuria datan siirtomääriä vaativissa järjestelmissä CORBA puolestaan osoittautuu Java-RMI:tä tehokkaammaksi (Juric & al., 2000).



Kuva 23. Java-RMI -arkkitehtuuri (Juric & al., 2000).

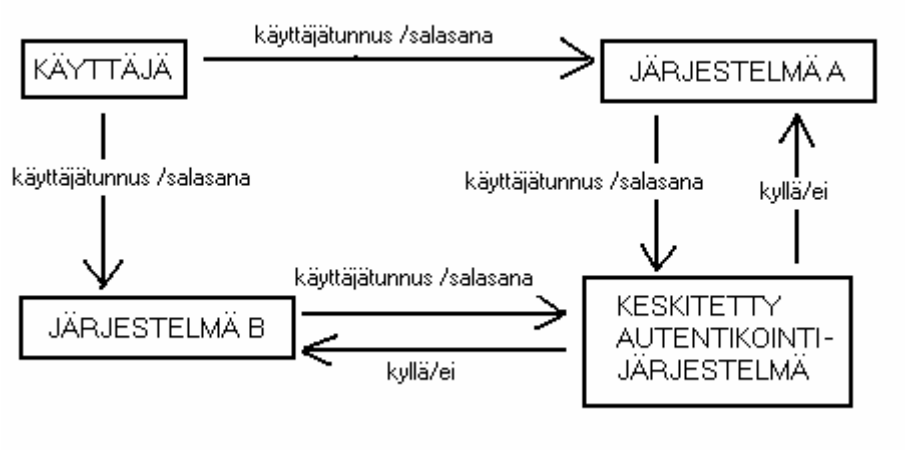
Java-RMI koostuu kolmesta toisistaan riippumattomasta kerroksesta. Kanta/runkokerros toimii rajapintana sovellustason kerroksen ja muun Java-RMI -järjestelmän välillä. Etäviitekerros (remote reference layer) vastaa etäkutsujen semantiikasta ja sisältää asiakas- ja palvelinpuolen komponentit. Siirtokerroksen (transport layer) tehtävänä on varmistaa yhteyksien hallinta ja pyyntöjen välittäminen (Juric & al., 2000).

3.8 Web-palveluiden malli

Tietotekniikan liiton sanastotoimikunnan (2003) mukaan *web-palveluilla* tarkoitetaan ohjelmakomponentteja, jotka hyödyntävät SOAP:ia ja HTTP:tä XML-pohjaisten viestien lähettämisessä. Topley (2003) puolestaan määrittelee web-palvelun Internet-protokollia hyödyntäväksi ohjelmistoksi, joka voidaan identifioida URI:n perusteella ja jonka rajapinta ja sidokset on kuvattu XML:ää käyttäen. Kuitenkin termiä web-palvelu on käytetty jo ennen SOAP- ja XML-teknologioiden kehittämistä. Esimerkiksi Devarakonda & al. (1995) kuvaavat web-palveluksi minkä tahansa webissä olevan palvelun, jota palvelin tarjoaa asiakkaille.

Web-palveluiden malli (Web Services Model) on kehitetty tarjoamaan systemaattinen ja laajennettavissa oleva kehys web-palveluita tarjoavien ja käyttävien järjestelmien väliseen vuorovaikutukseen. Web-palveluiden mallissa määritellään yhtenäinen mekanismi web-palveluita tarjoavien järjestelmien kuvaamiseen ja paikantamiseen, sekä palveluja tarjoavien ja käyttä-

vien järjestelmien väliseen kommunikointiin. Web-palveluiden kuvaamisessa käytetään WSDL-kieltä ja paikantamisessa UDDI-rekisteriä. WSDL-kieltä ja UDDI-rekisteriä käsitellään tarkemmin alakohdissa 3.8.1 ja 3.8.2. Mallin tarkoituksena on taata eri tahojen toteuttamien järjestelmien yhteistoimivuus web-palveluita käytettäessä (Curbera & al., 2002).



Kuva 24. Esimerkki web-palvelusta.

Kuvassa 24 on esimerkki B2B-mallin (business-to-business) mukaisesta web-palvelusta, joka tarjoaa autentikointipalvelua mm. järjestelmille A ja B. Käyttäjän autentikointitiedot on tallennettu tietokantaan, jota keskitetty autentikointijärjestelmä kykenee käyttämään. Kun käyttäjä haluaa kirjautua sisään järjestelmään A, niin hänen autentikointitunnuksensa lähetetään käyttäjäagentin välityksellä järjestelmälle A. Järjestelmä A ottaa vastaan saamansa tunnukset ja pyytää keskitetyltä autentikointijärjestelmältä autentikointipalvelua. Lähetettävän palvelupyynnön mukana järjestelmä A lähettää autentikointijärjestelmälle parametreina käyttäjän käyttäjätunnuksen ja salasanan. Autentikointijärjestelmä suorittaa palvelupyynnön vertailemalla tietokannassa olevia käyttäjätunnus/salasana -yhdistelmiä toisiinsa ja lähettää palautusarvona tiedon autentikoinnin onnistumisesta järjestelmälle A. Saamansa palautusarvon (kuvan 24 esimerkissä: kyllä/ei) perusteella järjestelmä A tekee päätöksen päästetäänkö käyttäjä palveluun sisään. Samaa menetelmää autentikoinnissa käyttää myös järjestelmä B, jolla ei ole mitään yhteyttä järjestelmään A. Varsinainen autentikointi tapahtuu käyttäjältä piilossa, eikä hän ole missään vaiheessa yhteydessä autentikoinnista huolehtivaan järjestelmään.

Web-palveluiden malli on määritelty tukemaan olemassaolevia Internet-protokollia ja se pohjautuu avoimeen XML-standardiin, joka tarjoaa mallille alusta- ja ympäristöriippumattomuu-

den. Alusta- ja ympäristöriippumattomuus mahdollistaa sen, että mallin mukaisia web-palveluita voidaan toteuttaa miltei missä tahansa ohjelmointiympäristössä, kunhan siitä löytyy tuki XML-kielen käytölle (Curbera & al., 2002).

3.8.1 WSDL

SOAP:in tavoin *WSDL-kieli* (Web Service Description Language) on XML-perustainen tekniikka. WSDL-kieli koostuu XML-sanastosta, jota käyttäen voidaan kuvata web-palveluiden rajapinta. Rajapinnan kuvaus sisältää määritelmät siitä, millaisia XML-perustaisia viestejä kuvattuun web-palveluun voidaan lähettää ja millaisia viestejä web-palvelu lähettää vastauksena sitä kutsuvaan asiakasjärjestelmään. Rajapinnan kuvauksen avulla voidaan rakentaa web-palveluiden mallia hyödyntäviä järjestelmiä (Topley, 2003; W3C, 2001).

WSDL-määrityksen tarkoituksena on luoda yleisesti hyväksytty ja noudatettava malli web-palveluiden tuottamiseksi. Päämääränä on lisätä eri tahojen tuottamien järjestelmien vuorovaikutusta luomalla yhteiset pelisäännöt käytettävälle rajapinnalle. Ideana on, että palvelun tarjoaja julkistaa ja ylläpitää järjestelmänsä WSDL-dokumenttia, jonka avulla palvelua käyttävien järjestelmien toteuttajat kykenevät rakentamaan oman järjestelmänsä. Useat ohjelmistoalan yritykset (mm. Borland) ovat rakentaneet työvälineitä, joilla voidaan generoida palvelun tarjoajan WSDL-dokumentista runko Java-ohjelmalle, jonka päälle voidaan helposti rakennetaan kyseistä web-palvelua käyttävä sovellus (Penttinen, 2002).

3.8.2 UDDI

UDDI (The Universal Description, Discovery and Integration) on web-palveluiden malliin kiinteästi liittyvä protokolla, jonka määrittelystä vastaa OASIS–konsortio[5]. UDDI:ssa määritellään ympäristöriippumaton alusta, johon palveluiden kehittäjät voivat rekisteröidä tarjoamiaan web-palveluita. Näitä rekisteröityjä web-palveluita voivat puolestaan toiset kehittäjät etsiä ja ottaa käyttöönsä keskitetyn UDDI-rekisterin kautta (UDDI, 2003).

[5] Tunnettiin aikaisemmin nimellä SGML-Open (UDDI, 2003).

UDDI:ssa ylläpidettävää kaupallista rekisteriä kutsutaan lyhenteellä UBR (UDDI Business Registry). OASIS-konsortion jäsenten ylläpitämien UBR:ien lisäksi monet yritykset pitävät yllä vastaavia omia rekistereitään sisäistä käyttöä varten (Curbera & al., 2002).

UDDI:n rekisteriin tallennetaan kolmen tyyppistä tietoa rekisteröidystä web-palvelusta. Nämä tiedot jaetaan niin kutsuttuihin valkoisiin, keltaisiin ja vihreisiin sivuihin. Valkoiset sivut sisältävät mm. rekisteröidyn palvelun kehittäjien nimi- ja yhteystiedot sekä lyhyen kuvauksen palvelun tarjoajasta. Keltaisilla sivuilla palvelut kategorisoidaan käyttötarkoituksen mukaan ja sinne myös tallennetaan kuvaus palvelusta. Vihreät sivut puolestaan sisältävät palvelun teknisen kuvauksen yksityiskohtineen (Curbera & al., 2002).

Näiden edellä mainittujen sivujen sisältämät tiedot tallennetaan XML-dokumenteiksi rekisteriin. UDDI määrittelee tallennuksessa käytettävien elementtien nimet, jotta palvelun tarjoajien tiedot olisivat yhdenmukaisia. Yhdenmukaisuuden ansiosta rekisteristä on helppo hakea tarvittavia tietoja ja palveluiden kategorisointi yksinkertaistuu (Curbera & al., 2002).

3.9 SOAP:in kehitysympäristöt

SOAP:ia hyödyntävien asiakas- ja palvelinjärjestelmien toteutusta varten on olemassa lukuisa määrä erilaisia SOAP-kehitysympäristöjä. Kehitysympäristöt on rakennettu SOAP:in määritysten pohjalta (tai määrityksiä mukaillen) ja ne sisältävät valmiita ohjelmakomponentteja, joiden avulla SOAP-viesti voidaan automaattisesti muodostaa.

JAX-RPC (Java API for XML-based RPC) on J2EE-arkkitehtuuriin liittyvä kehitysympäristö, joka on suunniteltu RPC-perustaisten web-palveluiden toteutukseen. Topleyn (2003) mukaan JAX-RPC tarjoaa kokeneille Java-toteuttajille yksinkertaisen tavan toteuttaa Java-perustaisia asiakas- ja palvelinjärjestelmiä ilman että heidän tarvitsisi liiemmin tuntea XML- ja SOAP -teknologioita. JAX-RPC tukee SOAP-viestien lähettämistä HTTP-, HTTPS-, SMTP- ja FTP-protokollia hyödyntäen ja siitä löytyy valmis tuki web-palveluiden mallin kuvauskielenä käytetylle WSDL-kielelle (Topley, 2003).

Apache Axis, jonka varhaisempi versio oli nimeltään Apache SOAP, on Apache Software Foundation (ASF) -organisaation SOAP-kehitysympäristö. *Apache SOAP* puolestaan perustui

IBM:n SOAP for Java-projektissa määriteltyyn SOAP4J-kehitysympäristöön. ASF:n missiona on tuottaa avoimeen lähdekoodiin perustuvia ratkaisuja, joista siis Apache Axis on yksi esimerkki. Apache Axis on kirjoitettu Java-kielellä ja se sisältää luokkia, joiden avulla voidaan luoda ja jäsentää SOAP-viestejä. Apache Axis sisältää niinikään työkalut, joiden avulla asiakas- ja palvelinjärjestelmät voivat lähettää ja ottaa vastaan SOAP-viestejä. Tuettuja sovelluspalvelimia ovat ainakin TomCat ja WebSphere. Myöskin Apache Axis tukee WSDL-kieltä (Axis, 2003).

Microsoft SOAP Toolkit on Microsoftin kehitysympäristö SOAP:ia käyttävien web-palveluiden toteuttamiseksi. MS SOAP Toolkitiä käyttäen voidaan luoda web-palveluiden mallin mukaiset toiminnallisuudet COM-sovelluksiin. MS SOAP Toolkitissä SOAP:ia käyttävät COM-oliot voidaan kirjoittaa Visual C++ tai Visual Basic -kielillä (Davis & Parashar, 2002; Microsoft, 2003). Kuten useimmat muutkin Microsoftin ohjelmat, MS SOAP Toolkit tukee ainoastaan Microsoftin omia käyttöjärjestelmiä ja palvelimia.

SOAP::Lite on kirjoitettu Perl-kieltä käyttäen ja se on yksi Perlin moduuleista. Moduuli tarjoaa rajapinnan SOAP-protokollaa käsitteleville asiakas- ja palvelinjärjestelmille. SOAP::Lite -moduulille löytyy tuki sekä Unix- että Windows-ympäristöihin ja se tukee suurimmalta osiltaan SOAP:in määrittelyä (Kulchenko, 2003).

XSOAP on Indianan yliopistossa määritelty SOAP-kehitysympäristö proseduurien etäkutsujen toteutukseen. XSOAP:ista on olemassa erilliset kehitysympäristöt sekä Java- että C++ -ympäristöihin. XSOAP:ista löytyy API-rajapinta, joka mahdollistaa sen käytön yhdessä Java-RMI:n kanssa (XSOAP, 2002).

3.10 SOAP:in suorituskyky

Hajautetun tietojenkäsittelyn mallissa järjestelmien suorituskyky on usein yksi kriittisimmistä tekijöistä. Kuitenkin SOAP-protokollan kehittämisessä on suorituskyvyn huomiointi jäänyt taka-alalle, koska muut vaatimukset, kuten yhteiskäytettävyys, on asetettu etusijalle. Kun suorituskyvyltään tehokkaissa teknologioissa, kuten esimerkiksi CORBA:ssa, siirrettään dataa järjestelmien välillä binäärimuodossa, niin vastaavasti SOAP:issa siirrettävä data on koodattu XML-muotoon (Davis & Zhang, 2003).

XML-muodossa olevassa datassa tietoalkiot kääritään tietotyypin ilmaisevan aloitus- ja lopetustagien sisälle. Monissa tapauksissa näiden tietotyyppien kuvaamiseen käytetyt tagit vievät yli puolet viestin kokonaistavumäärästä (Chiu & al., 2002). Tämän seurauksena generoitavien tiedostojen koot kasvavat varsin suuriksi, eikä SOAP tällä saralla kykene kilpailemaan binäärimuotoista dataa käyttävien teknologioiden kanssa (Martin, 2001).

Monissa tutkimuksissa on vertailtu SOAP:in suorituskykyä suhteessa muihin teknologioihin sekä kehitysympäristön vaikutusta suorituskykyyn. Tutkimuksissa on tullut esille, että SOAP:in suorituskykyyn vaikuttavat toteutuksessa käytetyn kehitysympäristön ja siihen kuuluvan XML-jäsentimen lisäksi tiedonsiirrossa käytetyn HTTP-protokollan versio, siirrettävän tietorakenteen tyyppi ja siirrettävän datan määrä.

Kun viesteihin ei sisällytetty kuormaa, niin Elfwing & al. (2002) mittasivat SOAP:ia ja CORBA-arkkitehtuuria hyödyntävien asiakas/palvelin -parien välisten pyyntö/vastaus -viestien vasteajoissa 400-kertaisia eroja CORBA:n hyväksi. Tutkimuksessa kehitysympäristönä oli Apache Axis. Davis ja Parashar (2002) puolestaan saivat vastaavissa omissa mittauksissaan 10-160 kertaisia eroja CORBA:n hyväksi, riippuen käytetystä SOAP-kehitysympäristöstä. Mielenkiintoisena yksityiskohtana voidaan todeta, että Davis ja Parashar mittasivat pienimmän eron vasteajoissa, kun SOAP:ia hyödyntävä asiakas/palvelin -pari oli toteutettu Apache Axisia käyttäen.

Elfwingin & al. (2002) mukaan SOAP:ia hyödyntävän pyyntö/vastaus -viestin vasteaikaa voidaan pienentää selvästi, jos SOAP-viesti jäsennetään viestiketjun aikana vain kaksi kertaa ja jäsentämisessä hyödynnetään tehokasta XML-jäsennintä. Tämä heidän päätelmänsä mukainen teoreettinen minimi SOAP:in vasteajaksi olisi seitsemän kertainen verrattuna CORBA:n vasteaikaan.

Kuormitetuissa viesteissä SOAP:in ja CORBA:n suorituskykyjen väliset erot olivat pienempiä. Davis ja Parashar (2002) mittasivat 800-merkin kokoisella kuormalla SOAP:in vasteajat 7-100 kertaa suuremmiksi kuin CORBA:n vastaava vasteaika. Suurimmat erot mitattiin kun kuormana oli 800 alkiota sisältävä taulukko, jolloin erot vasteajoissa olivat jopa 1000-kertaiset. Tällöin SOAP:ia hyödyntävä asiakas/palvelin -pari oli toteutettu SOAP::Lite kehitysympäristössä. Myös SOAP-kehitysympäristöjen väliltä löytyi merkittäviä eroja. Esimerkiksi kun kuormana käytettiin 800 alkiota sisältävää taulukkoa, niin SOAP::Litellä toteutettu asia-

kas/palvelin -pari vaati 56 kertaa suuremman vasteajan kuin vastaava XSOAP:illa toteutettu asiakas/palvelin -pari.

Taulukko 7. Teknologioiden vertailu.

Teknologia	*Ilman kuormaa*	*Kuormana 800 alkiota sisältävä taulukko.*	*Kuormana 800 merkin mittainen merkkijono.*
Java-RMI	1	1	1
CORBA	1,6	1,1	1,2
XSOAP	47	18	18
MS SOAP Toolkit	251	43	87
SOAP::Lite	250	1006	87
Apache SOAP	257	106	103
Apache Axis	16	127	6,8

Taulukossa 7 on vertailtu Davisin ja Parasharin (2002) tutkimuksessa mitattuja vasteaikoja. Taulukossa esitetyt arvot ovat suhteutettuja siten, että paras vasteaika kussakin tapauksessa on merkitty ykkösellä ja muut arvot esittävät kuinka monta kertaa enemmän vei vastaava toiminta aikaa käytetyllä teknologialla. Kaikki taulukon 7 perustana olevat mittaukset on suoritettu siten, että asiakas- ja palvelin sijaitsevat eri verkkoasemilla.

Davisin ja Parasharin (2002) tutkimuksessa CORBA:lla ja Java-RMI:llä toteutettujen asiakas/palvelin -parien välillä ei esiintynyt merkittäviä eroja, mutta molemmat olivat suorituskyvyltään selvästi SOAP:ia hyödyntäviä asiakas/palvelin -pareja tehokkaampia.

SOAP:in huono suorituskyky aiheutuu monista seikoista. Eräs syy on nk. Nagle-algoritmi, jota käytetään vähentämään verkon ylimääräistä kuormitusta. Nagle-algoritmin käytön seurauksena asiakasjärjestelmästä lähetetään HTTP-protokollaan mukaisesta viestistä ensin otsikko-osa palvelinjärjestelmään ja mikäli palvelinjärjestelmä hyväksyy pyynnön, niin vasta sen jälkeen lähetetään viestin runko-osa (Elfwing & al. 2002; Davisin & Parashar, 2002).

Kutsu/vastaus viestinvälitysmallia käytettäessä CORBA:lla toteutetun asiakas/palvelin -parin välillä kulkee ainoastaan kaksi viestiä – pyyntö ja vastaus. Sen sijaan SOAP-kutsu aiheuttaa TCP-liikenteen, joka koostuu 12 viestistä HTTP-pyynnön ja –vastauksen suorittamiseksi. Elfwing & al. (2002) esittävät suorituskyvyn parantamiseksi TCP-liikenteen analysointia turhien viiveiden poistamiseksi sekä SOAP-kutsuille sopivien jäsentimien valintaa.

Chiu & al. (2002) puolestaan ehdottavat kehittämään XML:n jäsentämiseen liittyviä tekniikoita, joiden avulla voitaisi jäsentää data yhdellä läpikäynnillä, käsitellä tehokkaammin SOAP:in taulukkorakenteita ja XML:n syntaksiin liittyviä tageja. Samaan päätelmään tulivat myös Davis ja Zhang (2002). Chiun & al. (2002) mukaan myös SOAP:in spesifikaatiota tulisi muokata/laajentaa, mikäli SOAP:ia halutaan hyödyntää tieteellisessä laskennassa.

4 YHTEENVETO

SOAP-protokolla kehitettiin alun perin tarjoamaan mekanismi proseduurien etäkutsujen suorittamiseen. Kuitenkin ajan myötä protokollan kehittämisen painopiste muuttui ja nykyisen SOAP:in perustehtävänä on toimia viestinvälitysprotokollana siirrettäessä XML-muotoista dataa. XML-perustaisuus tuo SOAP-protokollalle tarvittavan riippumattomuuden käytetyistä alustoista ja ohjelmointiympäristöistä. XML:n suosiosta kertoo jotain se, että käytetyimpiin ohjelmointikieliin on lisätty XML-tuki. XML toimii rajapintana eri ympäristöissä rakennettujen sovellusten välillä ja näin mahdollistetaan eri sovellusten keskinäinen yhteiskäytettävyys, ilman että rajapintaan tai rajapinnassa kulkevaan tietoon tarvitsisi tehdä konvertointeja.

Yhteiskäytettävyyden kriteerit täyttävä SOAP on saamassa laajan jalansijan proseduurien etäkutsujen suorittamisessa ja viestinvälityksessä käytettävänä protokollana. Protokolla on saanut taakseen suuret ohjelmistoalan yritykset ja W3C-konsortio julkaisi SOAP-suosituksen kesällä 2003. Erilaisten web-arkkitehtuurien yhteistoiminnallisuuden mahdollistavassa web-palveluiden mallissa SOAP on kiinnitetty viestinvälitysprotokollaksi. Tämä kiinnitys on kenties merkittävin yksittäinen askel, joka laajentaa SOAP-protokollan leviämistä, koska web-palveluiden malli on muodostumassa vallitsevaksi menetelmäksi web-palveluiden toteuttamisessa.

SOAP:in ongelmana on pidetty viestien suurta kokoa, joka luonnollisesti huonontaa suorituskykyä siirrettäessä suuria tietomääriä. Ratkaisuksi on ehdotettu moniprotokollajärjestelmää, jossa viestin sisältö konvertoitaisi siirron ajaksi tekstiformaatista esimerkiksi binääriformaattiin. Tämä ei kuitenkaan ole toimiva ratkaisu, jos halutaan säilyttää SOAP:in tarjoama lisäarvo protokollan helppokäyttöisyydestä. Lisäksi konvertoinnit tiedostoformaatista toiseen ja takaisin vaikuttaisivat omalta osaltaan myös suorituskykyyn. Todennäköistä kuitenkin on, että suorituskyvyltään kriittiset tapaukset, joissa siirretään suuria määriä tietoa järjestelmien välillä, tehdään myös jatkossa tehokkaammilla menetelmillä, kuten esimerkiksi Java-RMI tai CORBA. SOAP on pyritty suunnittelemaan dynaamiseksi protokollaksi, johon voidaan tarvittaessa lisätä laajennuksia. Tämä idea on ollut myös koko XML-kielen perustana, joten SOAP-protokollan tulevaisuus vaikuttaa lupaavalta.

VIITELUETTELO

Axis (2003) *The Apache XML Project*, WWW-sivusto, Axis Development Team, http://ws.apache.org/axis/index.html (2.11.2003).

Berners-Lee, T., Fielding, R., Irvine, U.C., Gettys, J., Mogul, J., Frystyk, H. (1997) *Hypertext Transfer Protocol - HTTP/1.1*. http://www.ietf.org/rfc/rfc2068.txt (28.10.2003).

Berners-Lee, T., Fielding, R., Masinter, L. (1998) *Uniform Resource Identifiers (URI): Generic Syntax*. http://www.ietf.org/rfc/rfc2396.txt (18.1.2003).

Bertino, E., Catania, B. (2001) Integrating XML and databases. *IEEE Internet Computing* **5**(4), 84-88.

Chester, T. M. (2001) Cross-platform Integration with XML and SOAP. *IT Professional* **3**(5), 26-34.

Chiu, K., Govindaraju, M., Bramley, R. (2002) Investigating the limits of SOAP performance for scientific computing. *The Eleventh IEEE International Symposium on High Performance Distributed Computing*, (toim. Jacobs, A.), IEEE Technical Committee on Distributed Processing, USA, 246-254.

Chu, W.C., Kai-Chih, L., Lo, W., Shyan-Ming, Y. (1997) From legacy RPC services to distributed objects. *The Sixth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, (toim. Storms, P.), IEEE Technical Committee on Distributed Processing, USA, 2-7.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S. (2002) Unraveling the Web Services Web : An Introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing* **6**(2), 86-93.

Davis, A., Zhang, D. (2002) A Comparative Study of DCOM and SOAP. *Fourth International Multimedia Software Engineering*, (toim. Werner, B.), IEEE Computer Society, USA, 48-55.

Davis, D., Parashar, M. (2002) Latency performance of SOAP implementations. *The Second IEEE/ACM International Symposium on Cluster Computing and the Grid*, (toim. Kawada, S.), IEEE Computer Society, USA, 407-412.

Devarakonda, M., Mukherjee, A., Kish, B.(1995) Meta-scripts as a mechanism for complex web services. *Fifth Workshop on Hot Topics in Operating Systems*, IEEE Press, USA, 46-50.

Elfwing, R., Paulsson, U., Lundberg, L. (2002) Performance of SOAP in Web Service Environment Compared to CORBA. *9th Asia-Pacific Software Engineering Conference*, 84-93.

Gardner, J.R. (2001) Information architecture planning with XML. *Library Hi Tech* **19**(3), 231-241.

Girardot, M., Sundaresan, N. (2000) Millau: an encoding format for efficient representation and exchange of XML over the Web. *Computer Networks* **33**(1-6), 747-765.

Govindaraju, M., Slominski, A., Choppella, V., Bramley, R., Gannon, D. (2000) Requirements for and evaluation of RMI protocols for scientific computing. *Proceedings of the 2000 ACM/IEEE conference on Supercomputing*, IEEE Computer Society, Washington, DC, artikkeli nro 61 (cd-rom).

Hoque, R. (1998) *CORBA 3*. IDG Books Worldwide, Inc, Foster City.

Ibiblio.org (2003) *Internet Pioneers - Tim Berners-Lee*. WWW-sivusto, http://www.ibiblio.org/pioneers/lee.html (12.4.2003)

Jepsen, T. (2001) SOAP cleans up interoperability problems on the Web. *IT Professional* **3**(1) , 52 –55.

Juric, M., Rozman, I., Hericko, M. (2000) Performance comparison of CORBA and RMI. *Information and Software Technology* **42**(11), 915-933.

Karpinski, R. (1999) XML Name Spec To Help Avoid Ambiguity. *InternetWeek* 749, 17.

Kulchenko, P. (2003) *SOAP::Lite for Perl*. WWW-sivusto, http://www.soaplite.com/ (4.5.2003)

Malek, M., Polze, A., Richling, J., Schwarz, J. (1999) Towards predictable CORBA-based Web-services. *Second IEEE International Symposium on Object-Oriented Real-Time Distributed Computing,* (toim. Palagi, L.), IEEE Computer Society Technical Committee on Distributed Processing, USA, 182 –191.

Martin, A. (2001) Can We Integrate Bioinformatics Data on the Internet?. *Trends In Biotechnology* **19**(9), 327-328.

Microsoft (2003) *SOAP*. WWW-sivusto, http://msdn.microsoft.com/soap/ (4.5.2003).

O'Connell, P. McCrindle, R. (2001) Using SOAP to clean up configuration management. *25th Annual International Computer Software and Applications Conference*, (toim. Williams, D.), IEEE Computer Society, USA, 555-560.

OMG (2002) *Object Management Group*. WWW-sivusto, http://www.omg.org/ (29.12.2002).

Penttinen, T. (2001) *XML-rajapinnat*. TEKES-projekti XML-rajapintojen kehittäminen - XRAKE.

Penttinen, T. (2002) *WSDL (Web Services Description Language).* TEKES-projekti XML-rajapintojen kehittäminen - XRAKE.

Petron, E. (1997) Remote Procedure Calls in Linux. *Linux Journal* **4**(4), Artikkeli nro 2.

Postel, Jonathan (1982) *Simple Mail Transfer Protocol*. WWW-sivusto, http://www.ietf.org/rfc/rfc0821.txt (9.12.2002).

Shiva, S., Virmani, R. (1993) Implementation of reliable and efficient remote procedure calls. *IEEE Southeastcon '93,* USA.

Sun Microsystems (2001) *Java Remote Method Invocation.* WWW-sivusto, http://java.sun.com/j2se/1.4.1/docs/guide/rmi/ (21.4.2003).

Tietotekniikan liiton sanastotoimikunta (2003) *ATK-Sanakirja.* Talentum Oy, Helsinki.

Topley, K. (2003) *Java Web Services in a Nutshell.* O'Reilly & Associates, Inc., Sebastopol.

Tsenov, M. (2002) Application of SOAP protocol in e-commerce solution. *First IEEE International Symposium on Intelligent Systems,* IEEE Control Systems Society, Varna, Bulgaria, 59-62.

UDDI (2003) *The Universal Description, Discovery and Integration of Web Services.* WWW-sivusto, http://www.uddi.org/ (12.1.2003).

W3C (1999a) *Hypertext Transfer Protocol - HTTP/1.1*. WWW-sivusto, World Wide Web Consortium, http://www.w3.org/Protocols/rfc2616/rfc2616.html (30.9.2003).

W3C (1999b) *The World Wide Web Consortium Issues "Namespaces in XML" as a W3C Recommendation* WWW-sivusto, World Wide Web Consortium, http://www.w3.org/Press/1999/XML-Namespaces-REC (5.10.2003).

W3C (2000) *Simple Object Access Protocol (SOAP) 1.1*. WWW-sivusto, World Wide Web Consortium, http://www.w3c.org/TR/SOAP (30.5.2003).

W3C (2001) *Web Services Description Language (WSDL) 1.1*. WWW-sivusto, World Wide Web Consortium, http://www.w3c.org/TR/WSDL (14.10.2003).

W3C (2002) *Extensible Markup Language*. WWW-sivusto, World Wide Web Consortium, http://www.w3c.org/XML (8.12.2002).

W3C (2003a) *SOAP Version 1.2 Part 0: Primer*. WWW-sivusto, World Wide Web Consortium, http://www.w3.org/TR/2003/REC-soap12-part0-20030624/ (4.10.2003).

W3C (2003b) *SOAP Version 1.2 Part 1: Messaging Framework.* WWW-sivusto, World Wide Web Consortium, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/ (4.10.2003).

W3C (2003c) *SOAP Version 1.2 Part 2: Adjuncts.* WWW-sivusto, World Wide Web Consortium, http://www.w3.org/TR/2003/REC-soap12-part2-20030624/ (4.10.2003).

W3C (2003d) *About the World Wide Web Consortium*. WWW-sivusto, World Wide Web Consortium, http://www.w3.org/Consortium/ (4.10.2003).

Winer, D. (1999) *XML-RPC Specification*. WWW-sivusto, http://www.xmlrpc.com/spec (10.5.2003)

Winer, D. (2001) *Foreword for the XML-RPC book.* WWW-sivusto, http://www.xmlrpc.com/stories/storyReader$1726 (10.5.2003)

XSOAP (2002) *XSOAP toolkit (a.k.a. SoapRMI).* WWW-sivusto, http://www.extreme.indiana.edu/xgws/xsoap/ (10.05.2003).