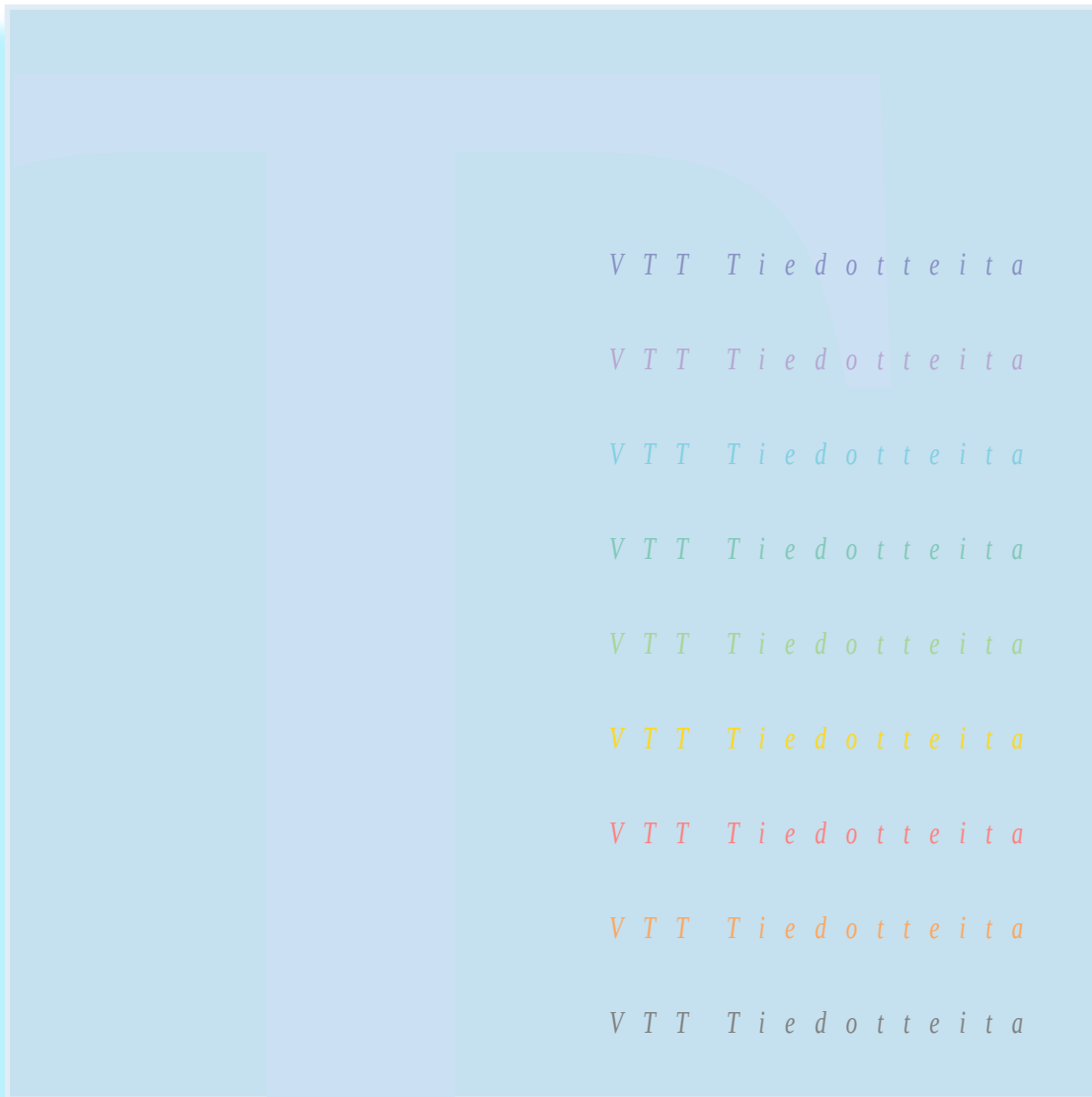
2130

VTT TIEDOTTEITA

VTT TIEDOTTEITA

Aki Tikkala

Jatkuva-aikaisten multimediasovellusten kehitysalusta



VALTION TEKNILLINEN TUTKIMUSKESKUS ESPOO 2002

VTT TIEDOTTEITA – RESEARCH NOTES 2130

Jatkuva-aikaisten multimediasovellusten kehitysalusta

Aki Tikkala
VTT Elektroniikka

VALTION TEKNILLINEN TUTKIMUSKESKUS
ESPOO 2002

ISBN 951–38–5943–6 (URL: http://www.inf.vtt.fi/pdf/)
ISSN 1455–0865 (URL: http://www.inf.vtt.fi/pdf/)

JULKAISIJA – UTGIVARE – PUBLISHER

Valtion teknillinen tutkimuskeskus (VTT), Vuorimiehentie 5, PL 2000, 02044 VTT
puh. vaihde (09) 4561, faksi (09) 456 4374

Statens tekniska forskningscentral (VTT), Bergsmansvägen 5, PB 2000, 02044 VTT
tel. växel (09) 4561, fax (09) 456 4374

Technical Research Centre of Finland (VTT), Vuorimiehentie 5, P.O.Box 2000, FIN–02044 VTT, Finland
phone internat. + 358 9 4561, fax + 358 9 456 4374

VTT Elektroniikka, Sulautetut ohjelmistot, Kaitoväylä 1, PL 1100, 90571 OULU
puh. vaihde (08) 551 2111, faksi (08) 551 2320

VTT Elektronik, Inbyggd programvara, Kaitoväylä 1, PB 1100, 90571 ULEÅBORG
tel. växel (08) 551 2111, fax (08) 551 2320

VTT Electronics, Embedded Software, Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland
phone internat. + 358 8 551 2111, fax + 358 8 551 2320

Toimitus Kerttu Tirronen

Otamedia Oy, Espoo 2002

Tikkanen, Aki. Jatkuva-aikaisten multimediasovellusten kehitysalusta [Development platform for streaming multimedia applications]. Espoo 2002. Valtion teknillinen tutkimuskeskus, VTT Tiedotteita – – Research Notes 2130. 55 s.

Avainsanat multimedia platform, RTP, Real-Time-Protocol, SIP, Session Initiation Protocol

Tiivistelmä

Työn tavoitteena oli kehittää jatkuva-aikaista multimediaa hyödyntävien sovellusten kehitysalusta.

Työssä selvitettiin käytetyimpien multimediajärjestelmien ominaisuudet. Analyysin perusteella valittiin protokollat, jotka välttämättä tarvittiin sovelluksiin. Valintatehtävä osoittautui helpoksi, sillä alalle on muodostunut selkeästi joukko de facto -standardeja, joiden noudattaminen oli elintärkeää yhteensopivuuden kannalta. Lisäksi selvitettiin olemassa olevien toteutusten uudelleenkäyttömahdollisuuksia huomioiden alustalle asetetut vaatimukset.

Alustaan toteutettavat protokollat määriteltiin kahden käyttötapauksen avulla. Ensimmäisessä käyttötapauksessa siirrettiin jatkuva-aikaista videokuvaa palvelimelta mobiiliin päätelaitteeseen. Toisessa käyttötapauksessa siirrettiin jatkuva-aikaista kaksisuuntaista videokuvaa kahden mobiilin päätelaitteen välillä.

Luotu alusta jaettiin neljään alijärjestelmään, joista tässä työssä keskityttiin multimediaprotokollat sisältävään media-alijärjestelmään. Media-alijärjestelmä koostui itsenäisistä mediaprotokollakomponenteista, jotka koostettiin toimivaksi kokonaisuudeksi liitutauluarkkitehtuurityylin mukaisesti yhdellä hallitsevalla komponentilla. Näin saavutettiin modulaarinen järjestelmä, johon oli mahdollista liittää pienillä muutoksilla uusia protokollatoteutuksia.

Tikkanen, Aki. Jatkuva-aikaisten multimediasovellusten kehitysalusta [Development platform for streaming multimedia applications]. Espoo 2002. Technical Research Centre of Finland, VTT Tiedotteita – Research Notes 2130. 55 p.

Keywords multimedia platform, RTP, Real-Time-Protocol, SIP, Session Initiation Protocol

Abstract

The goal of this work was to develop a platform for streaming multimedia applications.

The work included studying the properties of the most used multimedia platforms. They were analysed to decide which ones must be included in the platform in order to achieve proper application functionality. The decision was easy to make because there are a number of de facto standards in this domain and those protocols must be supported in the name of compatibility. The possibility of using existing implementations was also examined.

The protocols implemented into the platform were defined with two use cases. In the first case, a live video was streamed from the server to the mobile terminal. In the second case, a dual-way live video was streamed between two mobile terminals.

The created platform was divided into four subsystems; of these four, this study was focused on the multimedia subsystem. The multimedia subsystem consists of independent protocol components which were assembled into the functional unit by using the repository defined by the blackboard architecture. This resulted in a modular system with a capability to easily add or change protocols.

Alkusanat

Tämädiplomityö on tehty Oulun yliopistolle VTT Elektroniikan Oulun toimipisteessä. Työ on ollut osa MUPPET-projektia, jonka tutkimuskohteena oli jatkuva-aikaista multimediaa mobiilissa ympäristössä hyödyntävien sovellusten kehitysalusta. Projektin rahoittajana toimi Hantro Products Oy.

Haluan kiittää Hantro Products Oy:n Otto Pulkkista ja Jarkko Nisulaa, joiden tarjoaman sujuvan yhteistyön tuloksena tämä diplomityö syntyi. Lisäksi haluan kiittää Ari Parkkilaa uusia näkemyksiä antaneista keskusteluista sekä erityisesti yhteistyöstä ja ohjauksesta projektin aikana. Käsikirjoituksen kommentoinnista ja työn ohjaajana toimisesta VTT:ssä haluan kiittää Juhani Latvakoskea. Lisäksi kiitän valvojaani professori Juha Röningiä Oulun yliopistosta työni tarkastamisesta ja hyödyllisistä kommenteista. Kiitokset kuuluvat myös professori Jukka Riekille toisena valvojana toimimisesta.

Lopuksi haluan kiittää vanhempiani, jotka ovat olleet tukenani koko opiskeluni ajan.

Oulussa 4.1.2002

Aki Tikkala

Sisällysluettelo

Symboliluettelo

3GPP	3rd Generation Partnership Project – Organisaatio, joka määrittelee kolmannen sukupolven matkapuhelinjärjestelmän.
ABNF	Augmented Backus-Naur Form – Viestien notaation kuvaamiseen käytetty kieli.
ALF	Application Level Framing – Menetelmä, jolla tieto ositetaan lähettämistä varten.
ASN	Abstract Syntax Notation – Viestien notaation kuvaamiseen käytetty kieli.
DS	Differanted Services – Palvelun laadun takaamiseksi kehitetty malli.
GPL	GNU General Public License – Lisenssi, jota käytetään avoimen lähdekoodin oikeuksien hallintaan.
HTTP	Hypertext Transfer Protocol – Pääasiassa WWW-sivujen lataamiseen käytetty protokolla.
IETF	Internet Engineering Task Force – Organisaatio, jonka tehtävänä on määritellä Internetissä käytettävät protokollat.
ILP	Integrated Layer Processing – Periaate, jolla tiedon muokkaus toteutetaan.
IP	Internet Protocol – Internetin perustana oleva tiedonsiirtoprotokolla.
LAN	Local Area Network – Paikallisverkko.
MGCP	Media Gateway Control Protocol – Protokolla, jolla hallitaan IP- ja puhelinverkkoja yhdistäviä yhdyskäytäviä.
MPEG	Motion Picture Experts Group – Videokuvan pakkausstandardeja kehittävä yhteisö.
MTU	Maximum Transfer Unit – Suurin mahdollinen lähetysyksikön koko.
PER	Packet Encoding Rules – Viestien kuvaustapa.

RFC	Request For Comments – IETF:n määritysdokumentti.
RSVP	Resource Reservation Protocol – Protokolla, jota käytetään resurssien varaamiseen Internet-liikenteelle.
RTCP	RTP Control Protocol – Protokolla, jolla kontrolloidaan RTP-protokollan liikennettä.
RTP	Real-Time Transfer Protocol – Protokolla, jota käytetään reaali-aikaisen tiedon siirtämiseen.
RTSP	Real Time Streaming Protocol – Protokolla, jolla hallitaan jatkuva-aikaisia mediavirtoja.
SAP	Session Announcement Protocol – Multimedia konferenssien ilmoitusprotokolla.
SDP	Session Description Protocol – Protokolla, jota käytetään yhteyden parametrien kuvaamiseen.
SIP	Session Initiation Protocol – Internetissä käytettävä signalointiprotokolla.
SMIL	Synchronized Multimedia Integration Language – Kuvauskieli, jota käytetään multimedian synkronointiin.
SMTP	Simple Mail Transfer Protocol – Sähköpostin välittämiseen käytetty protokolla.
ST2	Streams 2 – Varhainen reaali-aikaisen tiedon siirtämiseen käytetty protokolla.
TCP	Transfer Control Protocol – Internetissä käytettävä yhteydellinen tiedonsiirtoprotokolla.
UDP	User Datagram Protocol – Internetissä käytettävä yhteydetön tiedonsiirtoprotokolla.
URI	Uniform Resource Identifier – Resurssien tunnistus menetelmä.
WLAN	Wireless LAN – Langaton LAN.

1. Johdanto

Multimedia on nykyään kiinteä osa elämäämme riippumatta siitä, huomaammeko sitä vaiko emme. Randall Packer pohtii multimedian historiaa artikkelissaan "Just What Is Multimedia, anyway?"[1]. Hänen mukaansa multimedian historia ulottuu aina eteläisen Ranskan esihistoriallisen ajan ritualistisiin kokoontumisiin. Packer määrittelee multimedian olennaiseksi ominaisuudeksi syventymisen (engl. immersion). Wagner huomasi saman seikan 1800-luvun lopulla uudistaessaan oopperan esitystavan. Tuloksena oopperasta syntyi media, joka "piirsi katsojan silmään kuvan virtuaalisesta maailmasta". Nykyään sama kuva piirtyy silmiimme monista eri lähteistä, niin televisiosta, tietokoneen ruuduilta kuin kannettavista päätelaitteistakin.

Yksi tulevaisuuden viestintäjärjestelmien suurimmista palvelukokonaisuuksista on erilaiset multimediaan perustuvat palvelut. On arvioitu, että vuoteen 2010 mennessä kolmannes matkapuhelinoperaattoreiden tuloista tulee multimedia- palveluista [2]. Multimediajärjestelmiä on olemassa kiinteisiin ympäristöihin, mutta toistaiseksi ei vielä juurikaan langattomiin ja mobiileihin ympäristöihin.

Multimedian siirtäminen voidaan toteuttaa kahdella eri tavalla, joko lataamalla (engl. download) tai jatkuva-aikaisesti (engl. streaming). Lataamalla tapahtuvassa siirrossa koko medialeike siirretään päätelaitteeseen, ennen kuin käyttäjä voi sitä katsella. Tällöin verkkoyhteyttä tarvitaan vain siirron aikana ja korkearesoluutioinen medialeike voidaan siirtää taustalla ja katsella sitten, kun on aikaa. Jatkuva-aikaisessa siirrossa käyttäjän pitää katsella leikettä samalla, kun sitä siirretään. Tällä tekniikalla mahdollistetaan minkä tahansa pituisten leikkeiden katselu, kuten myös suoran lähetyksen seuraaminen [3]. Jatkuva-aikaista siirtoa käytettäessä ei laitteistolta vaadita niin suuria muistiresursseja kuin lataamalla tapahtuvassa siirrossa. Siksi jatkuva-aikainen siirto on erityisen sopiva mobiiliin ympäristöön, jossa laitteiden resurssit ovat tavallisesti vähäiset.

Tämän työn tavoitteena oli kehittää jatkuva-aikaista multimediaa hyödyntävien sovellusten kehitysalusta. Käytännön tehtävänä oli suunnitella kehitysalustan arkkitehtuuri ja toteuttaa se.

Työn aluksi tutustuttiin olemassa oleviin multimediaprotokolliin ja -formaatteihin sekä eri organisaatioiden asettamiin standardeihin ja suosituksiin. Tällä pyrittiin saavuttamaan mahdollisimman hyvä yhteensopivuus olemassa olevien toteutusten kanssa. Lisäksi selvitettiin, onko saatavilla valmiita ohjelmistoja tai ohjelmiston osia, joita olisi mahdollista hyödyntää sulautetussa ympäristössä toimivassa multimediajärjestelmässä.

Alalla on muutamia organisaatioita, joiden käsitykset tekniikasta ohjaavat koko alan toimintaa. Internet-protokollien ja -standardien kanssa työskentelevän IETF[1]:n (The Internet Engineering Task Force) protokollat ovat tällä hetkellä vahvimpia ehdokkaita käytettäväksi protokolliksi. Esimerkiksi matkapuhelinjärjestelmiä määrittelevä 3GPP[2] (Third Generation Partnership Project) on ottanut IETF:n protokollat multimedia-alijärjestelmän pohjaksi ja myös jatkuva-aikaiseen siirtoon tarkoitettuun alijärjestelmään [4, 5].

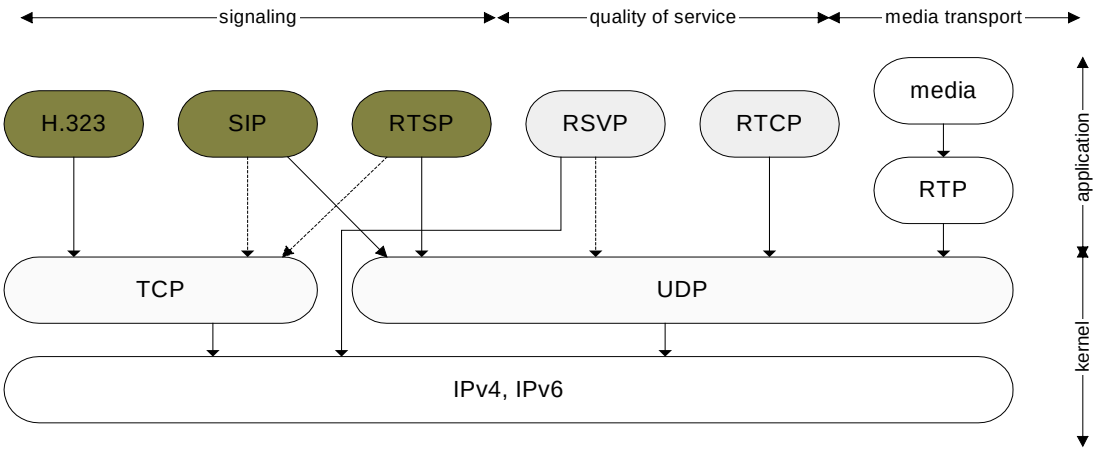
[1] http://www.ietf.org
[2] http://www.3gpp.org

2. Multimediastandardit

2.1 Protokollat

Jatkuva-aikaista multimediaa hyödyntävän sovelluksen toteuttaminen vaatii useiden eri protokollien käyttämistä ja niiden välistä sujuvaa yhteistyötä. Kuvassa 1 [6] on esitetty protokollapino, joka kuvaa jatkuva-aikaisen multimedian hyödyntämiseen vaadittavien protokollien suhdetta toisiinsa. Kaikki kuvassa esitetyt protokollat eivät ole välttämättömiä, vaan siinä on mukana myös vaihtoehtoisia protokollia.

Kuvan 1 protokollat voidaan jakaa media- ja kontrolliprotokolliin. RTP [7] on ainut kuvassa esitetty mediaprotokolla. Mediaprotokollan tehtävänä on median siirtäminen. H.323[3], SIP [8] ja RTSP [9] ovat signalointiprotokollia, joiden päätehtävänä on muodostaa yhteys ja neuvotella mediaprotokollayhteydelle sopivat yhteysparametrit. Palvelun laatua hallitsevia signalointiprotokollia kuvassa edustavat RSVP [10] ja RTCP [7]. Niiden tehtävänä on tarjota eri keinoin palvelulle mahdollisuus parantaa sen laatua.



Kuva 1. IP-puheluihin käytettävä protokollapino.

Multimediaprotokollat eivät suhtaudu toisiinsa perinteisesti pinona, jossa toiminnallisuutta lisätään joka kerroksella, vaan ne ovat enemminkin sijoittuneet horisontaalisesti siten, että jokainen hoitaa itsenäisesti oman tehtävänsä. Tämän kokonaisuuden päällä on sovellus, joka koordinoi alla olevien protokollien yhteistoimintaa. Horisontaalisen järjestelyn etuna on se, että näin on mahdollisuus käyttää hyväksi vaihtoehtoisia protokollia ilman, että koko pino täytyy toteuttaa uudestaan. Näin saavutetaan joustavuutta multimediasovellusten toteuttamiseen.

[3] International Telecommuncation Union (1998) Packet based multimedia communication systems. Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland.

Kuvassa 1 on esitetty protokollien sijoittuminen suhteessa käyttöjärjestelmään (engl. kernel) ja sovellukseen (engl. application). Kaikki varsinaisesti median käsittelyyn liittyvät protokollat toteutetaan tyypillisesti sovellustasolla, kun taas tiedon siirtämiseen liittyvät protokollat on perinteisesti toteutettu käyttöjärjestelmään. Siksi mediaprotokollien yhteydessä puhutaankin usein sovellustason protokollista (engl. application level protocol). Tässä yhteydessä sovellustasolla ei tarkoiteta täsmälleen samaa kuin OSI-mallin[4] 7. kerros, vaan sillä pyritään ilmaisemaan sitä, että tavallisesti sovelluskehittäjän on toteutettava tai ainakin ymmärrettävä multimediaprotokollien sisäinen toiminta.

2.1.1 Real-Time Protocol ja RTP Control Protocol

Ensimmäinen protokolla, joka oli tarkoitettu reaaliaikaisen tiedon siirtämiseen IP-verkossa, oli Streams 2 (ST2) [11]. Se tunnettiin myös nimellä IPv5, sillä sitä käytettiin jatkuva-aikaisen tiedon siirtämiseen IPv4:n rinnalla. ST2 oli periaatteessa hyvä protokolla, mutta sisälsi myös ongelmia – sekä teknisiä että poliittisia. Loppujen lopuksi IETF:n työryhmä jakoi ST2:n toiminnallisuuden kahteen eri protokollaan: RSVP:hen ja RTP:hen. [12]

Real-Time Protocol (RTP) ja RTP Control Protocol (RTCP) on määritelty IETF:n dokumentissa RFC 1889 [7]. RTP:n tehtävänä on tarjota päästä-päähän -kuljetuspalvelu reaaliaikaista tietoa vaativille sovelluksille. RTCP puolestaan tarjoaa kontrollimekanismin RTP:lle.

Real-Time Protocol (RTP)

RTP:n tarjoamat palvelut ovat tyypin tunnistus, järjestysnumerointi, aikaleimat ja liikenteen tarkkailu [13]. RTP ei kuitenkaan itsessääntarjoa mekanismia, joka takaisi pakettien toimituksen ajallaan vastaanottajalle, eikä muitakaan takeita palvelun laadusta. Jos halutaan taata pakettien luotettava perillemeno, on rinnalla käytettävä muita menetelmiä, jotka pyrkivät takaamaan palvelun laadun. Näitä tehtäviä hoitavat esimerkiksi Resource Reservatation Setup -protokolla (RSVP) [10] ja Differentiated Services (DS) -malli [14].

RTP voidaan nähdä enemmän arkkitehtuurina kuin protokollana [12]. Spesifikaation mukaan RTP on protokollakehys (engl. framework), joka määrittelee vain viestiformaatin, mutta toteutusarkkitehtuuri ja -tekniikka ovat täysin riippuvaisia kehittäjästä. Tällä

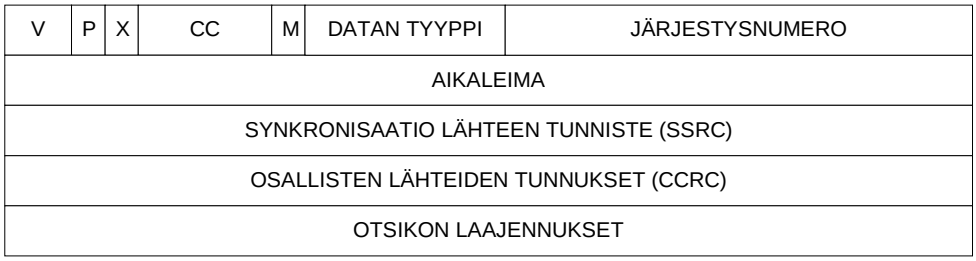
[4] Information Processing Systems – Open Systems Interconnection - Basic Reference Model (1984). ISO-7498, International Organization for Standardization.

tavoin saavutetaan yhteensopivuus eri sovellusten välillä, muttei kuitenkaan rajoiteta sovelluksen sisäistä arkkitehtuuria.

RTP on määritelty käyttäen Clarkin ja Tennenhousen esittämää Application Level Framing -nimistä suunnitteluperiaatetta. ALF:n ajatuksena on jakaa alemmille verkkokerroksille annettava tieto ylemmän kerroksen tarpeiden mukaisiin yksiköihin. Näin ylemmällä eli sovellustasolla voidaan paremmin hallita kadonneista tai epäjärjestyksessä saapuneista paketeista aiheutuneita virheitä. Tällä tavoin voidaan esimerkiksi välttyä turhilta uudelleenlähetyksiltä. RTP:ssä on myös käytetty samaisessa artikkelissa esitettyä Integrated Layer Processing -tekniikkaa. Sen perimmäinen ajatus on toteuttaa protokolla siten, että tiedon muokkaus tapahtuu kokonaisuudessaan yhdessä tai kahdessa silmukassa sen sijaan, että muokkaukset tehtäisiin peräkkäisesti. Etuna ILP:stä voidaan nähdä mm. parantunut suorituskyky. [15]

Kaikki RTP mediavirrassa kulkevat paketit sisältävät yleisen otsikkokentän ja hyötykuorman. Hyötykuorman muoto on määritelty eri tavoin eri mediaformaateille. RFC 1890 [16] määrittelee hyötykuormatyyppien yleiset muodostusperiaatteet ja ryhmittelyt. IETF:n avt-työryhmän (audio video transport) www-sivuille[5] on koottu tällä hetkellä määritellyt ja kehitteillä olevat hyötykuormatyypit.

Kuvassa 2 on esitetty RTP-paketin otsikkokentän rakenne. Ensimmäiset 12 oktettia ovat mukana kaikissa RTP-paketeissa. CSRC-kentät ovat sisällytettyinä vain, jos pakettien yhdistämisen tehnyt palvelin (engl. mixer) on ne lisännyt. Taulukossa 1 on esitetty otsikkokenttien kuvaukset.



Kuva 2. RTP-otsikkorakenne.

[5] http://www.ietf.org/html.charters/avt-charter.html

Taulukko 1. RTP-otsikkokenttien kuvaukset.

Kenttä	Kentän koko (bittejä)	Kuvaus
versio(V)	2	Ilmaisee RTP:n versio numeron. Nykyisin on käytössä versio 2.
täyte (P)	1	Jos täytebitti on asetettu, paketti sisältää paketin lopussa yhden tai useamman lisäoktetin, joka kuuluu hyötykuormaan.
laajennus (X)	1	Jos laajennus-bitti on asetettu, kiinteää otsikkoa seuraa otsikkolaajennus.
CSRC-laskuri (CC)	4	CSRC-laskuri ilmaisee otsikkorakenteessa seuraavien CSRC-tunnisteiden lukumäärän.
merkkibitti (M)	1	Merkitys riippuu hyötykuorman tyypistä, mutta on tarkoitettu ilmaisemaan median rajoja, esim. video- tai audiokehystä.
hyötykuorman tyyppi (PT)	7	Hyötykuorman tyyppikenttä ilmaisee RTP-paketin sisältämän hyötykuorman tyypin, jonka perusteella sovellus voi päätellä, kuinka tietoa käsitellään.
järjestysnumero	16	Järjestysnumero ilmaisee kyseisen paketin järjestysnumeron, jota lisätään yhdellä per lähetetty paketti. Ensimmäisen paketin järjestysnumero on satunnainen, mutta sitä seuraavia numeroita sovellus voi käyttää kadonneiden pakettien havainnointiin.
aikaleima	32	Aikaleima sisältää RTP-paketin ensimmäisen oktetin näytteistyshetken. Näytteistyshetki tulee ottaa kellosta, joka kasvattaa aikaa lineaarisesti ja on riittävän tarkka, jotta sen perusteella pakettien synkronisointi on mahdollista.
synkronisointi-lähteen tunniste (SSRC)	32	SSRC-kenttä ilmaisee synkronisointilähteen tunnuksen.
osallisten lähteiden tunnisteet (CSRC)	0-15 * 32	CSRC-lähteiden lista sisältää paketin hyötykuorman tuottamiseen osallistuneiden lähteiden tunnukset. Lista koostetaan palvelimessa, joka kokoaa useasta eri lähteestä tulevien mediavirtojen paketit yhteen RTP-pakettiin.

RTP Control Protocol (RTCP)

RTP Control Protocol (RTCP) tarjoaa mekanismin kontrolloida RTP:tä. Sen päätehtävänä on tarjota informaatiota tietoliikenteen palvelun laadusta. Tätä palautetta voidaan mm. käyttää mukautuvien koodekkien hallintaan [13]. Näin mediapalvelin voi mukautua suuriinkin käyttäjämääriin ja vaihteleviin verkko-oloihin. RTCP:n käyttö on erityisen tärkeää konferenssipuheluissa ja muissa sovelluksissa, joissa on suuri joukko osallistujia.

RTCP määrittelee seuraavat viisi viestityyppiä, joihin sen toiminta perustuu [12, 17]:

1. Lähettäjän raportit

Sisältävät tietovirran tilasta tietoa, jonka perusteella pystytään arvioimaan palvelun laatua ja mahdollisuuksien mukaan parantamaan sitä. Tilatietoja ovat mm. korkein vastaanotettu paketin sekvenssinumero, pakettien välinen synkronointivirhe (engl. jitter), kadonneiden pakettien suhteellinen osuus edellisen raportin jälkeen ja kumulatiivinen kadonneiden pakettien määrä. Lisäksi paketti sisältää lähettäjän aikaleimat ja lähetettyjen tavujen ja pakettien määrän.

2. Vastaanottajan raportit

Vastaanottajan raportit ovat sisällöltään vastaavia lähettäjän raporttien kanssa, paitsi , että lähettäjäinformaatio (lähetyt paketit ja tavut, aikaleimat) on jätetty pois. Lisäksi paketin tyyppi on määritetty toiseksi.

3. Lähteen määrittelypaketti

Lähteen määrittelypaketti kuvaa lähteen tarkemmin. Nämä tiedot voivat sisältää esimerkiksi käyttäjän nimen, sähköpostiosoitteen tai maantieteellisen sijainnin.

4. Lopetuspaketti

Lopetuspaketti ilmaisee, että käyttäjä on poistumassa konferenssista.

5. Sovelluskohtaiset paketit

2.1.2 Real-time Streaming Protocol

RTSP (Real-Time Streaming Protocol) [9] muodostaa ja kontrolloi yhtä tai useampaa jatkuva-aikaista aika-synkronisoitua mediavirtaa. RTSP voidaan nähdä "videonauhuri-protokollana", sillä se tarjoaa videonauhurin kaltaisen toiminnallisuuden mediavirroille. Yleensä RTSP:n kontrolloimat mediavirrat siirretään käyttäen RTP:tä, mutta RTSP itsessään ei ole riippuvainen käytettävästä siirtomekanismista.

RTSP on suunniteltu tarkoituksella syntaksiltaan ja toiminnaltaan HTTP/1.1:n (Hyper Text Transfer Protocol) [19] kaltaiseksi. Näin on mahdollista lisätä HTTP:n laajennuksia myös RTSP:hen. RTSP poikkeaa seuraavilta osin HTTP:stä [9]:

- RTSP esittelee lukuisia uusia metodeita ja sillä on eri protokollatunniste.
- RTSP-palvelimen täytyy ylläpitää tilatietoa lähes kaikissa tapauksissa.
- RTSP:ssä sekä asiakas että palvelin voivat lähettää palvelupyyntöjä.
- RTSP:ssä tieto siirretään toista protokollaa käyttäen (tähän on kuitenkin poikkeuksia).
- RTSP on määritelty käyttämään merkistönä ISO 10646 (UTF-8) ISO 8859-1:n sijaan, siten se yhtenäinen HTML:n tämänhetkisen kansainvälistymisen kanssa.
- Request-URI sisältää aina absoluuttisen URI:n. HTTP/1.1. käyttää yhteensopivuuden takia taaksepäin absoluuttista polkua pyynnöissä ja lisää palvelimen nimen erilliselle otsikkoriville.

2.1.3 Session Initiation Protocol

SIP (Session Initiation Protocol) [8] on IETF:n määrittelemä signalointiprotokolla. Signalointiprotokollan tehtäviksi voidaan määritellä seuraavat [18]:

- **Käyttäjän paikantaminen.** Selvitetään, missä fyysisessä osoitteessa puhelun vastaanottaja on.
- **Selvitetään vastaanottajan halukkuus yhteyden muodostamiselle.**
- **Ominaisuusparametrien vaihtaminen.** Selvitetään, millaisin parametrein yhteys on mahdollista muodostaa.

- **Muokataan olemassa olevan yhteyden parametreja.**
- **Lopetetaan olemassa oleva yhteys.**

SIP:n suunnittelussa on yhdistelty elementtejä muista IETF:n määrittelemistä protokollista, kuten HTTP:stä (Hyper Text Transfer Protocol) [19] sekä SMTP:stä (Simple Mail Tranport Protocol) [20]. HTTP:stä on lainattu asiakas-palvelin -arkkitehtuuri ja URL:n (Uniform Resource Locator) [21] käyttö. SMTP:stä on puolestaan lainattu tekstikoodaukseen perustuva ajatusmaailma sekä otsikkotyyli. Vaikka yhtäläisyyksiä onkin muihin protokolliin nähden, niin SIP noudattaa IETF:n filosofiaa "yksi ongelma, yksi protokolla". SIP käyttää siis muita IETF:n protokollia mm. median siirtämiseen ja kuvaamiseen. [18]

SIP käyttää kuljetusprotokollanaan joko UDP:ta (User Datagram Protocol) [22] tai TCP:tä (Transmission Control Protocol) [23]. SIP-protokolla on kuitenkin suunniteltu siten, ettei se ole riippuvainen käytettävästä kuljetusprotokollasta. Onkin olemassa yhdyskäytäviä, jotka muuntavat IP-verkosta tulevat SIP-viestit muiden fyysisten verkkojen tai protokollien kanssa yhteensopiviksi viesteiksi [24].

SIP-protokolla muodostuu kahdesta eri tarkoitukseen luodusta komponentista, SIP-käyttäjäagentista (engl. user agent) ja SIP-palvelimesta. Käyttäjäagentit ovat tyypillisesti osana päätelaitteita, kun palvelimet ovat verkkokomponentteja, jotka käsittelevät useiden puheluiden signalointia [25]. On kuitenkin mahdollista, että käyttäjäagentti on osana jotain palvelinta, kuten esimerkiksi tallennuspalvelinta, mediapalvelinta tai erityisesti SIP:n laajennuksia käytettäessä osana mitä tahansa sulautettua laitetta, joka tarjoaa palvelujaan käyttäjälle [26].

Käyttäjäagentti voidaan puolestaan jakaa kahteen toiminnalliseen osaan, käyttäjäagenttiasiakkaaseen (engl. SIP User Agent Client, UAC) sekä käyttäjä-agenttipalvelimeen (engl. SIP User Agent Server, UAS). Asiakasosan tehtävänä on aloittaa puhelu. Palvelinosan tehtävänä on puolestaan vastata siihen.[25]

SIP-palvelimet voidaan jakaa seuraaviin kolmeen tyyppiin [18]:

1. Tilaton tai tilallinen välipalvelin (engl. stateless/stateful proxy server)

Välipalvelimen tehtävänä on toimia SIP-käyttäjäagentin edustajana puhelun muodostamisessa. Välipalvelin vastaanottaa käyttäjäagentilta yhteydenmuodostus-pyynnön ja alkaa käsitellä sitä. Tavallisesti välipalvelimella on pääsy tietokantaan tai rekisteröintipalvelimelle, joiden avulla se selvittää, mihin kyseinen yhteydenottopyyntö tulee lähettää.

Välipalvelin poikkeaa käyttäjäagentista kahdella olennaisella tavalla. Välipalvelin ei aloita itsenäisesti puhelua, vaan se vain vastaa käyttäjäagentin palvelupyyntöihin. Sen lisäksi välipalvelimella ei ole kykyä käsitellä mediaformaatteja.

Tilaton välipalvelin käsittelee viestejä vain niiden otsikoiden perusteella, eikä tallenna aikaisempia viestejä. Näin tilaton välipalvelin ei pysty lähettämään uudelleen viestejä, eikä käytä siten SIP:n ajastimia hyväkseen. Tästä huolimatta tilaton välipalvelin pystyy *Via*-otsikoiden perusteella havaitsemaan ja estämään silmukoiden syntymisen palvelupyyntöketjuun.

Tilallinen välipalvelin tallentaa lähettämänsä ja vastaanottamansa viestit. Näin tilallinen välipalvelin voi lähettää uudelleen palvelunpyynnön puhelun vastaanottajalle, jos tämä ei vastaa. On myös mahdollista, että tilallinen välipalvelin monistaa palvelupyynnön useampaan eri osoitteeseen (engl. forking proxy server), joita puhelun vastaanottajalla mahdollisesti on. Tällöin välipalvelin välittää ensimmäisen myönteisen vastauksen palvelupyynnön esittäjälle ja peruu muihin haaroihin esitetyt pyynnöt.

2. Uudelleenohjauspalvelin (engl. redirect server)

Uudelleenohjauspalvelimen ero välityspalvelimeen on siinä, että uudelleenohjauspalvelin ei lähetä palvelupyyntöjä eteenpäin, vaan vastaa niihin mahdollisuuksien mukaan. Se käyttää hyväkseen tietokantoja ja rekisteröintipalvelimia selvittääkseen palvelupyynnön kohteen sijainnin. Sen sijaan, että se itse edelleen lähettäisi pyynnön vastaanottajalle se vastaa pyynnön esittäjälle kertoen samalla vastaanottajan sijainnin. Tätä tietoa soittajan käyttäjäagentti käyttää hyväkseen lähettäessään pyynnön itsenäisesti vastaanottajan käyttäjäagentille.

3. Rekisteröintipalvelin (engl. registration server)

Rekisteröintipalvelin vastaanottaa REGISTER-viestejä, joiden avulla käyttäjäagentit kertovat käyttäjän tai resurssin nykyisen sijainnin. On huomattava, että rajapintaa, jonka avulla rekisteröintipalvelusta voidaan tehdä kyselyitä, ei ole määritelty SIP-protokollassa. Useissa toteutuksissa rekisteröintipalvelin on integroitu osaksi väli- tai uudelleenohjauspalvelinta.

2.1.4 Resource Reservation Protocol

RSVP:n (Resource Reservation Protocol) [10] tehtävänä on neuvotella kahden IP-verkossa olevan laitteen välille yhteys, jonka laatu on taattu. Normaalisti IP-verkon laatua ei ole taattu mitenkään, vaan verkko tekee parhaansa pakettien perille saattami-

seksi. RSVP:n avulla neuvotellaan jokaiselle päätelaitteiden välissä olevalle solmulle yhteyden laatuvaatimukset, joita solmut noudattavat.

RSVP tukee sekä yhden että usean vastaanottajan yhteyksiä. RSVP-neuvottelu aloitetaan tiedon vastaanottajan toimesta, joka selvittää reitin tiedon lähettäjälle ja sopii matkalla olevien solmujen kanssa käytettävästä palvelun laadusta. On huomattava, että neuvoteltu yhteys on yksisuuntainen, vaikka tietoa siirtyisikin molempiin suuntiin. RSVP:ta tukevat solmut käyttävät kolmea mekanismia liikenteen kontrollointiin:

1. **Pakettien luokittelu.** Solmu selvittää selvittää jokaisen paketin laatuluokan ja mahdollisuuden reitittää se.

2. **Käyttöoikeuksien kontrollointi.** Selvitetään, onko resurssien varaaminen sallittua kyseiselle pyynnölle.

3. **Pakettien aikatauluttaminen.** Aikatauluttamalla pakettien lähtö saavutetaan tavoitellut laatuvaatimukset.

2.1.5 Session Description Protocol

SDP:ta (Session Description Protocol) [27] käytetään kuvaamaan mediayhteyden tyyppiä ja ominaisuuksia. SDP kuvaa mm. käytettävän median tyypin ja formaatin, käytettävän siirtoprotokollan sekä mahdollisesti tarkat IP-osoitteet ja portit varsinaisen yhteyden muodostamiseksi. SDP-protokolla itsessään ei ole siirtoprotokolla, vaan SDP:n kuvaama informaatio siirretään jonkun toisen protokollan avulla. Tyypillisesti siirtoprotokollina toimivat SAP (Session Announcement Protocol) [28], SIP (Session Initiation Protocol) tai RTSP (Real-Time Streaming Protocol).

2.1.6 H.323

H.323 on ITU-T:n suositus multimediakommunikaatiolle pakettipohjaisten verkkojen yli. Suositus sisältää useita eri standardeja [29]:

H.323	Koko systeemin spesifikaatio
H.225.0	Puhelun kontrollointi ja alustus; mediavirtojen paketointi ja synkronisointi
H.235	Turvallisuus protokolla
H.245	Moodin vaihto ja toimintakykytietojen vaihto

H.450.x Lisäpalvelut

H.246 Yhteistoiminta piirikytkentäisten verkkojen kanssa

H.332 Suurikokoisen konferenssien hallinta

H.26x Videokoodekit

G.7xx Audiokoodekit.

H.323 on aiemmin ollut varsin merkittävässä asemassa multimediajärjestelmien signaloinnissa. Protokollaperhettä on käytetty erillisissä konferenssijärjestelmissä sekä ohjelmistopohjaisissa järjestelmissä kuten esimerkiksi Microsoftin NetMeeting-sovelluksessa.

2.2 Olemassa olevat multimediajärjestelmät

Seuraavassa esitellään käytetyimmät multimediajärjestelmät, jotka sisältävät tuen jatkuva-aikaiselle multimedialle. Tarkastelun lähtökohtana on toteutusten standardinmukaisuus ja mahdollisuus käyttää niitä referenssitoteutuksina. Lisäksi pyrittiin selvittämään, voisiko mahdollisesti saatavilla olevia lähdekoodeja hyödyntää toteutettavassa järjestelmässä.

QuickTime

QuickTime[6] on Applen kehittämä teollisuusstandardi, jota tuetaan usealla laitteistoalustalla. QuickTime tukee monia eri koodekkejä ja kykenee siten toimimaan useissa eri käyttötarkoituksissa. Sen lisäksi, että QuickTimea käytetään multimedian levittämiseen, sitä tukevat myös useat multimedian tuottamiseen ja editointiin tarkoitetut sovellukset. QuickTimessa on perinteisen jatkuva-aikaisen median lisäksi tuki synkronisoidulle grafiikalle, äänelle ja tekstille, sekä mahdollisuus käyttää interaktiivisia virtuaalitodellisuusmalleja.

Sovelluskehittäjälle Apple tarjoaa kattavan tuen sekä Java- että Windows-ympäristöön. Asiakassovellusten lähdekoodeja Apple ei kuitenkaan ole julkaissut. Lisäksi Applelta on saatavissa Darwin Streaming Server [33], joka on RTP:tä ja RTSP:tä tukeva jatkuva-aikaista multimediaa jakeleva palvelinsovellus. Darwinin lähdekoodit ovat saatavissa Applen oman lisenssin [34] alaisena.

[6] http://www.apple.com/quicktime/

Darwin mediapalvelinta käytettiin tässä työssä RTP- ja RTSP-toteutusten testaamiseen. Darwin on lähes standardinmukainen, mutta käsittelee esimerkiksi mediavirtojen tunnisteita eri tavoin kuin standardi vaatii. Darwin-palvelimen käyttö osoittautui pakolliseksi lähinnä siitä syystä, että se on ainut saatavilla oleva ohjelmisto, joka noudattaa standardin vaatimaa riippumattomuutta kuljetusprotokollasta. Lisäksi testauksessa käytettiin Applen QuickTime-asiakasohjelmistoa toteutettujen palvelinohjelmistojen testaamiseen.

RealNetworks

RealNetworks[7]:in tuotteet kattavat varsin laajasti jatkuva-aikaisen multimedian luomisen, jakelun ja katselun.

RealNetworks käyttää Surestream-tekniikkaa, jossa tiedostoformaattiin on tallennettu samasta medialeikkeestä usealla eri bittitaajuudella oleva versio. Näin mediapalvelimen on helpompi mukautua erilaisiin asiakkaisiin, joilla on erilaiset vaatimukset medialle. Lisäksi tietyn asiakkaan palvelua voidaan muuttaa yhteyden aikana, jos verkko-olosuhteet muuttuvat. [3]

RealNetworksiltä oli saatavissa (alkuvuodesta 2001) GPL-lisenssin [35] alaiset referenssitoteutukset RTP:lle ja RTSP:lle. Projektissa tutustuttiin kyseiseen toteutukseen tarkemmin, sillä se toimi esimerkkinä minimalistisesta toteutuksesta. Toteutus oli kuitenkin arkkitehtuuriltaan niin sekava, ettei sitä voinut käyttää oman toteutuksen lähtökohtana. Toteutuksen tutkiminen antoi kuitenkin käsityksen multimediasovellusten toteuttamisesta ja toimi varoittavana esimerkkinä rakenteellisesta ongelmista, jotka omassa toteutuksessa onnistuttiin välttämään.

RealNetworksin RealPlayer -asiakasohjelmistoa käytettiin RTP- ja RTSP-protokollien palvelinosien testaamiseen.

Windows Media Technologies

Windows Media Technogies[8] on Microsoftin arkkitehtuuri jatkuva-aikaiselle medialle. Windows Media tarjoaa mahdollisuuden käyttää sekä todellista jatkuva-aikaista mediavirtaa tukevia palvelimia että HTTP:hen perustuvaa rajoitettua siirtotekniikkaa. Median siirtämiseen Windows Media käyttää omaa protokollaansa, joka tukee UDP:ta, TCP:ta

[7] http://www.realnetworks.com
[8] http://www.microsoft.com/windows/windowsmedia/

ja toimii myös HTTP:n yli. Heidän käyttämänsä tiedostoformaatti tukee RealNetworksin tapaan useita bittitaajuuksia samassa tiedostossa.

Yleisesti Windows Median formaatteja pidetään tehokkaina, mutta sen suurin ongelma on kuitenkin suljettu teknologia, eikä sen soveltaminen muussa kuin Windows-ympäristössä ole mahdollista. Koska Microsoft haluaa tukea omien protokolliensa käyttöä, ei Windows Mediaan ole toteutettu standardeja protokollia, kuten RTP tai RTSP. Näin siis Windows Median käyttö testaamiseen oli mahdotonta.

DivX

DivX on MPEG-4 pohjainen, mutta ei sen kanssa yhteensopiva arkkitehtuuri videokuvan pakkaamiselle. DivX-arkkitehtuurista on olemassa periaatteessa kaksi eri versiota. DivXNetworks[9] tarjoaa kaupallista tuotteistettua versiota, jonka pohjalta on kehitetty avoimen lähdekoodin alainen OpenDivX[10]-järjestelmä.

2.3 Ohjelmointirajapinnat

Ohjelmistojen kehitysympäristöissä on varsin harvoin valmista tukea monipuoliselle multimedian käsittelylle. Usein ominaisuudet rajoittuvat paikallisella tallennusmedialla olevan äänen toistamiseen. Eri ympäristöihin on kuitenkin edellä esitettyjen multimediajärjestelmien lisäksi saatavilla kirjastoja, joissa on tuki jatkuva-aikaisen median siirtämiselle ja toistamiselle.

Java Media Framework

Sun Microsystemsin toteuttama Java Media Framework[11] (JMF) sisältää mm. videokuvan tai äänen kaappaamiseen, jatkuva-aikaisen median siirtämiseen ja toistamiseen omissa sovelluksissa. Arkkitehtuuriltaan JMF on varsin selkeä ja antaa hyvän kuvan multimedia-järjestelmien toiminnasta. JMF oli myös harvoja järjestelmiä, jotka olivat hyvin dokumentoituja.

JMF:n lähdekoodit ovat saatavilla Sunin SCSL-lisenssin [36] alaisena. Projektin käyttötarkoituksiin JMF ei kuitenkaan soveltunut, sillä Java-kielen käyttö projektissa oli mahdotonta.

[9] http://www.divxnetworks.com
[10] http://www.projectmayo.com
[11] http://java.sun.com/products/java-media/jmf/index.html

Vovida

Vovida Networks[12]:in ylläpitämä järjestelmä perustuu avoimeen lähdekoodiin. Järjestelmä sisältää kokonaisten sovellusten lisäksi useita multimediaan liittyviä protokollia erillisinä kokonaisuuksinaan, kuten mm. SIP, MGCP, RTP ja RTSP. Toteutuskielenä Vovida käyttää C++:aa.

Vovidan toteutukset olivat varsin huonosti dokumentoituja. Kokonaiskuvaus arkkitehtuurista ja ohjelmiston rakenteesta oli erittäin puutteellinen. Osasta luokista oli olemassa luokkakuvaukset, mutta kokonaiskuvaa eivät nekään pystyneet antamaan. Toteutus oli myöskin varsin massiivinen, joten sen soveltuvuudesta sulautettuun laitteeseen ei olisi myöskään ollut varmuutta.

SR-RTP

Edellä esiteltiin DivX-järjestelmä, jonka osaprojekti SR-RTP[13] on. SR-RTP keskittyy jatkuva-aikaisen median siirtämiseen ja on toteutettu pääasiassa erään MIT:n yksikön toimesta. SR-RTP-kirjasto sisältää toteutuksen RTP, RTSP ja RTCP protokollista. Tämän työn alkuvaiheessa kyseistä kirjastoa ei ollut saatavilla, mutta nykyisen dokumentaation ja avoimmuuden perusteella kyseinen toteutus olisi voinut olla varsin varteenotettava vaihtoehto. Toisaalta GNU-lisenssin [35] alaisen koodin käyttö kaupallisessa tuotteessa voi olla hankalaa.

2.4 Signalointiprotokollien valmiit toteutukset

Useita SIP-protokollaa tukevia toteutuksia on saatavissa, ja osasta on saatavissa myös lähdekoodit. Seuraavassa esitellään muutamia keskeisiä toteutuksia. Toteutusten käsittelyssä keskitytään lähinnä niiden käyttökelpoisuuteen referenssitoteutuksena.

Columbia University

Columbia University:n toteutus sisältää kaiken SIP-protokollaan olennaisesti liittyvän: käyttäjäagentin ja monipuolisen palvelinsovelluksen. Toteutuksen taustalla on Henning Schulzrinne, joka on SIP-protokollan pääsuunnittelijoita. Nykyisin alunperin yliopiston omistuksessa ollut toteutus on siirretty itsenäisen yhtiön (Sip Communcations, Inc.[14])

[12] http://www.vovida.org
[13] http://nms.lcs.mit.edu/software/videocm/
[14] http://www.sipcomm.com

hallintaan, josta toteutus on edelleenkin lisensoitavissa. Sovelluksista on saatavilla sekä lähdekoodit että valmiit ajettavat sovellukset.

Tässä projektissa käytettiin Columbia Universityn kehittämiä käyttäjäagentti- ja palvelintoteutuksia ensisijaisina referenssitoteutuksina.

Dynamicsoft

Dynamicsoft[15] tarjoaa laajan kirjon ammattimaiseen käyttöön tarkoitettuja SIP-sovelluksia, sekä käyttäjäagentteja että palvelinsovelluksia. Lisäksi on saatavana 60 päivän kokeiluversiot Java- ja C++-pohjaisista käyttäjäagenteista.

Dynamicsoftin toteutusta ei ehditty käyttämään toteutuksen testaamiseen.

Ubiquity

Ubiquity[16] on yritys, joka tarjoaa useita erilaisia SIP-protokollaan perustuvia sovelluksia. Vapaasti on saatavissa käyttäjäagentti, jolla testattiin SIP-toteutuksen toimivuutta.

Ubiquityn toteutusta käytettiin valmiin protokollatoteutuksen testaamisen. Ubiquityn sovellus ei ollut helppokäyttöinen. Erityisesti aiheutti hankaluuksia sovelluksen lisensointi, joka piti uusia varsin usein.

2.5 Pohdinta

Projektin alkuvaiheessa jouduttiin tekemään päätös, mitä protokollaa käytetään signalointiprotokollana. Vaihtoehtoja oli kaksi, H.323 ja SIP. Näillä protokollilla on lähes sama käyttötarkoitus ja toiminnallisuus, mutta muuten eroavuuksia löytyy kuitenkin melkoisesti. Protokollia on vertailtu kirjallisuudessa, josta on seuraavassa koostettu olennaisimmat erot ja yhtäläisyydet. [29, 37, 38]

1. Monimutkaisuus

H.323 on huomattavasti monimutkaisempi kuin SIP. H.323:n spesifikaatio on yhteensä 736 sivua, kun SIP:n 128 sivua. Monimutkaisuus näkyy myös viesteissä. H.323:ssa on määriteltynä satoja elementtejä, kun SIP sisältää vain 37 otsikkokenttää. Suurimpana kompleksisuuden lisääjänä H.323:ssa on viestien koodaus. H.323 koodaa viestit binaa-

[15] http://www.dynamicsoft.com
[16] http://www.ubiquity.net

rimuotoon käyttäen ASN.1- ja PER- (packet encoding rules) koodausta[17], kun SIP:in viestit ovat tekstimuotoista ABNF-koodausta (augmented Backus-Naur form) [39]. ASN.1:n jäsentäminen (engl. parse) on hankalaa ja etenkin PER-koodauksen purkuun olevat työkalut ovat harvinaisia ja erittäin kalliita. Geneerinen jäsentäjä SIP-viesteille voidaan puolestaan toteuttaa muutamalla sadalla rivillä C-koodia. [29, 37, 40]

SIP:n ABNF-koodaus ei kuitenkaan noudata RFC 2234:ssa [39] määriteltyä ABNF-koodausta, vaan määrittelee sen itse. Näin ei ole mahdollista käyttää geneerisiä jäsentäjiä SIP-viestien käsittelyssä. [38]

2. Laajennettavuus

Laajennettavuutta pidetään SIP:n suurena vahvuutena. Sen kehittämisessä on otettu oppia aiempien Internet-protokollien historiasta ja huomioitu se, että protokolla kehittyy voimakkaasti ajan mukana. Edellä mainittu tekstipohjainen koodaus helpottaa myös uusien ominaisuuksien toteuttamista. Tekstipohjaiset otsikkokentät ovat hyvin intuitiivisia, ja geneerista jäsentäjää käytettäessä uuden ominaisuuden toteuttaminen on triviaali tehtävä. H.323 on myös laajennettava, mutta siten rajoitettu, että uusia otsikkokenttiä voidaan lisätä vain ennalta määrättyihin kohtiin viestissä. Toisaalta ongelmia voi aiheutua siitä, että terminaalit eivät voi H.323:n avulla vaihtaa informaatiota siitä, mitä viestilaajennuksia he tukevat. [37]

Tekstimuotoisen esityksen ongelmana nähdään sen viemä suuri tilan määrä. Ongelmia seuraa erityisesti siitä, kun yhden viestin koko ylittää sallitun MTU:n (maximum transimission unit) koon, jolloin on olemassa riski pakettien pirstoutumiselle reitittimissä. [38]

3. Skaalautuvuus

H.323:lla on ongelmia skaalautuvuuden kanssa. Se on alun perin suunniteltu toimimaan yhden LAN-verkon alueella, eikä siinä olet tukea silmukoiden havaitsemiselle multidomain-ympäristössä. SIP-palvelimet voivat olla tilallisia, mutta ne voivat toimia myös tilattomassa moodissa, jolloin palvelin ei kuormitu niin paljon suurilla puhelumäärillä. H.323-palvelin voi toimia vain tilallisessa moodissa. Lisäksi H.323-palvelimen viestiliikenne toimii TCP:n päällä, joten sen on ylläpidettävä yhteys koko puhelun ajan. [29, 37]

Yhtenä SIP:n ongelmana mainitaan siltä puuttuva laskutusmalli. H.323:een on jo alunperin rakennuttu malli, jolla puheluista voidaan laskuttaa. Tämän puuttuminen SIP:stä

[17] ITU-T Recommendation X.691 (1997) Information Technology - ASN.1 encoding rules - Specification of Packed Encoding Rules.

voi vaikeuttaa sen hyväksikäyttöä. [38] Tämä on yleinen Internet-maailman ongelma: on vaikea nähdä mistä ja miten käyttäjiä voidaan laskuttaa.

Vaikka edellä onkin esitetty varsin vahva vastakkainasettelu SIP:n ja H.323:n välillä, on todennäköistä, että ne elävät rinnakkain jatkossakin. Molemmilla on oma käyttötarkoituksensa, ja on huomattava, että 3GPP tukee useita H.323-perheeseen kuuluvia protokollia IETF:n protokollien rinnalla.

Projektissä päädyttiin käyttämään SIP:tä signalointiprotokollana, koska se oli yksinkertaisempi, laajennettavampi ja skaalautuvampi. Lisäksi todettiin, että esimerkiksi 3GPP soveltaa SIP-teknologiaa IP-multimedia-alijärjestelmässään [4]. Näillä perusteilla asiakas päätyi käyttämään SIP-teknologiaa.

Muiden protokollien käytöstä ei jouduttu tekemään valintoja, sillä vaihtoehtoja ei juuri ollut. IETF:n määrittelemät protokollat ovat ainoita vaihtoehtoja, jos haluttiin spesifikaation olevan vapaasti saatavilla ja käytettävissä. Lisäksi IETF:n protokollat ovat Internet-maailmassa de facto -standardin asemassa ja lähitulevaisuudessa niitä käytetään myös seuraavan sukupolven matkapuhelin-järjestelmissä.

Tarjolla olleet valmiit toteutukset tai ohjelmointirajapinnat eivät olleet käyttökelpoisia. Suurin osa toteutuksista oli niin huonosti dokumentoituja, että niiden toiminnan selvittämiseen olisi mennyt aivan liian paljon aikaa saatavaan hyötyyn nähden. Usein itse toteutus oli niin huono, ettei dokumentointikaan olisi auttanut. Muutamia mahdollisia kirjastoja olisi voinut käyttää, mutta seuraavassa luvussa esiteltävät vaatimukset ohjelmistolle tiputtivat viimeisimmätkin pois.

3. Järjestelmän kuvaus

3.1 Vaatimukset

Asiakkaan tarpeiden takia toteutettavalle järjestelmälle oli asetettu seuraavia vaatimuksia:

1. Integroituvuus

Tärkein ehto toteutettavalle järjestelmälle oli, että sen piti integroitua sujuvasti asiakkaan aikaisempien tuotteiden kanssa. Siksi luotavan alustan arkkitehtuurista piti suunnitella sellainen, että siihen voidaan helposti integroida asiakkaan nykyinen MPEG-4-pohjainen videokuvan pakkaus- ja purkupiiri tai hänen vastaava ohjelmistopohjainen kirjastonsa. Toisaalta järjestelmän osien piti olla irrotettavissa toisistaan, jotta kohtuullisella integrointityöllä saadaan muodostettua haluttu kokonaisuus videokuvarajapinnan kanssa yhteistyössä toimivista protokollista.

2. Standardinmukaisuus

Suuri painoarvo oli myös standardien noudattamisella, sillä yrityksen kannalta on elintärkeää, että luotava alusta on yhteensopiva muiden valmistajien tuotteiden kanssa. Aluksi tehtiin markkinatutkimus, jossa pyrittiin selvittämään alalla olevat merkittävimmät teknologiat sekä nyt että tulevaisuudessa. Näiden trendien selvittämisessä IETF:n ja 3GPP:n näkemykset osoittautuivat määrääviksi.

3. Uudelleenkäyttö

Vaikka protokollien ja alustan toteutus ei ollutkaan suoraa tuotekehitystä, vaan enemmänkin toteuttamiskelpoisuuden tutkimista prototyyppien kautta, oli tärkeätä pitää mielessä myös tuotenäkökohta. Siksi uudelleenkäyttöön kiinnitettiin huomiota jo suunnittelun alkuvaiheessa ja etenkin toteutusvaiheessa. Uudelleenkäyttö voi tapahtua eri tuotteiden välillä tai tuotteen eri versioiden välillä [42]. Uudelleenkäytöksi voidaan myös lukea toteutuksien siirtäminen muihin käyttöjärjestelmäympäristöihin. Tämä pyrittiin tekemään mahdollisimman helpoksi ja joustavaksi. Käytännön toimina uudelleenkäytön mahdollistamiseksi käytettiin hyväksi komponentointia, joka luontaisesti tukee uudelleenkäyttöä. Lisäksi käytetty ohjelmointikieli valittiin siten, että sitä voitaisiin helposti käyttää myös muissa ympäristöissä.

Uudelleenkäyttö tähtää siihen, että luotuja ohjelmiston osia voitaisiin hyödyntää toisessa kokonaisuudessa. Toinen näkymä uudelleenkäyttöön on tutkia, voidaanko luotavaan

ohjelmistoon sisällyttää jotain ennalta toteutettua. Niin tehtiin tässäkin tapauksessa. Saatavissa olevien protokollatoteutusten lähdekoodeista ei kuitenkaan löytynyt sellaista, jota olisi voitu käyttää uudelleen luodussa ympäristössä – ellei niistä hankittua toimialatietämystä (engl. domain knowledge) lasketa sellaiseksi.

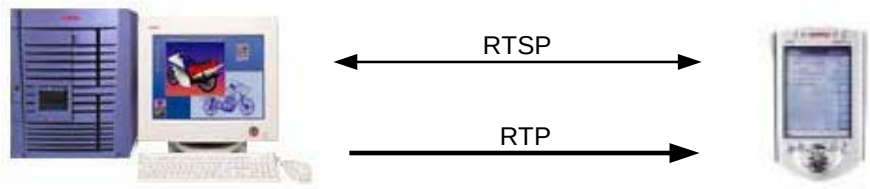
4. Muut laatuvaatimukset

Toteutuksen tehokkuus nähtiin tärkeimmäksi ominaisuudeksi, koska lopputulosta tultaisiin käyttämään sulautetussa laitteessa toimivassa sovelluksessa ja siten hyvä suorituskyky on erittäin tärkeää. Edelleen sulautetun laitteiston vaatimuksista aiheutui vaatimus toteutuksen pienelle koolle. Lisäksi pidettiin tärkeänä tehdä toteutuksesta mahdollisimman helposti laajennettava.

3.2 Käyttötapaukset

Projektin alkuvaiheessa asetettiin tavoitteeksi kahden inkrementaalisen käyttötapauksen toteuttaminen.

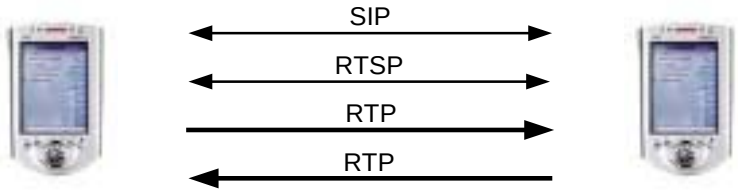
Ensimmäiseksi käyttötapaukseksi valittiin kaikkein yksinkertaisin tapaus, jossa jatkuva-aikaista multimediaa siirretään palvelimelta päätelaitteeseen. Kuvassa *3* on esitetty käyttötapauksessa käytettävät laitteet ja vaaditut protokollat.



Kuva 3. Multimedian siirto palvelimelta päätelaitteeseen.

Tapaus on mahdollista jakaa vielä useampaan iteratiiviseen vaiheeseen, joissa oman toteutuksen osuutta lisätään vähitellen. Ensimmäisessä vaiheessa käytettiin olemassa olevaa kaupallista media-palvelinta. Omaa toteutusta olivat asiakassovelluksen media-protokollat. Seuraavassa vaiheessa toteutettiin myös oma media-palvelinsovellus. Jo suunnitteluvaiheessa pyrittiin tietoisesti erottamaan erottamaan asiakas- ja palvelinosien toteutus itsenäisiin kokonaisuuksiin. Tällä pyrittiin parantamaan toteutuksen yhteensopivuutta jo olemassa olevien toteutuksien kanssa.

Toisena käyttötapauksena toteutettiin reaali-aikainen videopuhelu kahden sulautetun päätelaitteen välillä (kuva 4). Ratkaisussa täytyi integroida ensimmäisessä käyttötapauksessa toteutetut asiakas- ja palvelinosat samaan sovellukseen. Lisäksi täytyi toteuttaa signalointiprotokolla ja muuntaa sovelluslogiikka tapaukseen soveltuvaksi.



Kuva 4. Videopuhelu kahden päätelaitteen välillä.

3.3 Ohjelmiston arkkitehtuuri

Kuvassa *5* [42] on esitetty koko toteutetun järjestelmän arkkitehtuuri. Käytetty notaatio ei noudata mitään standardia, vaan on yrityksen sisäisesti käyttämä kuvaustapa[18]. Seuraavassa esitellään kuvassa käytetyt elementit:

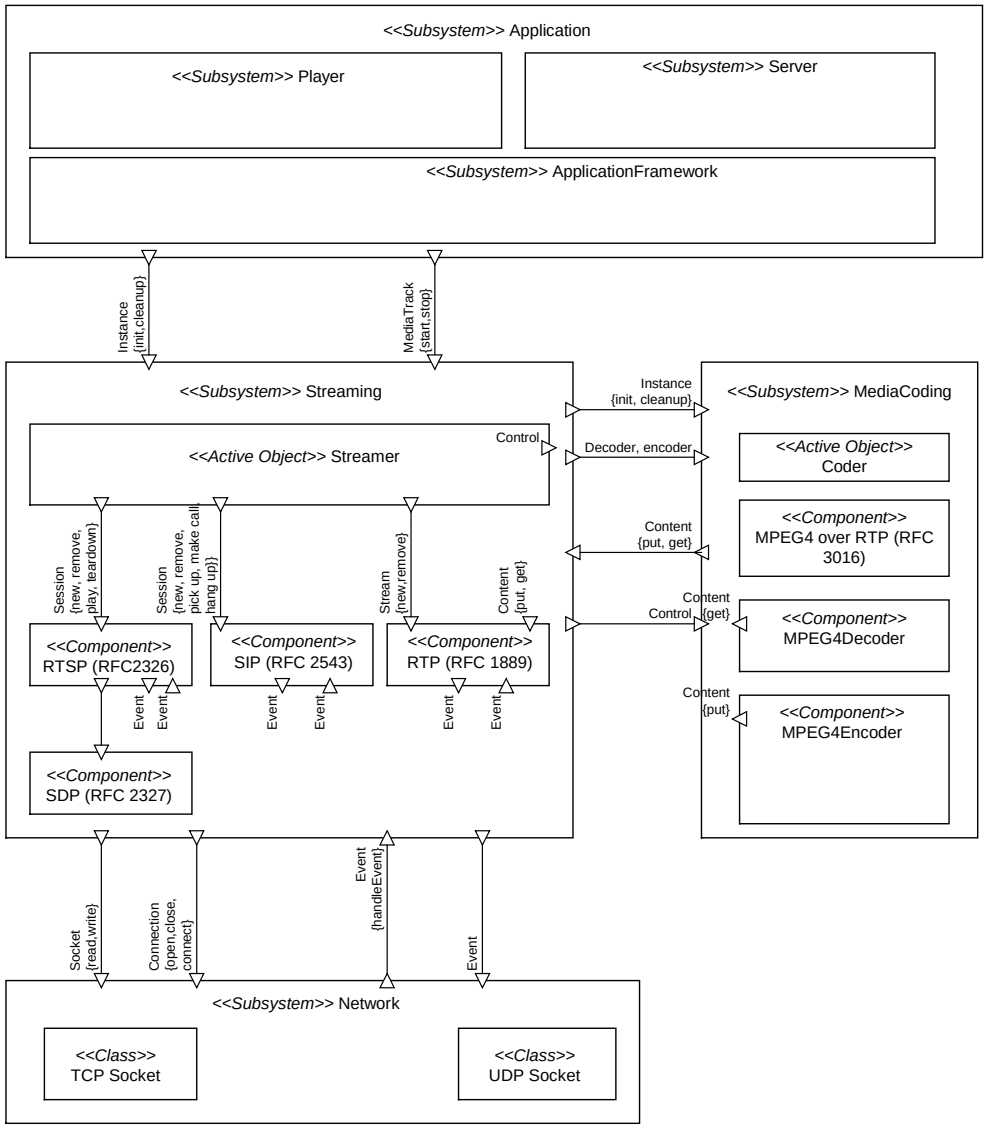
- **Alijärjestelmä (engl. subsystem).** Alijärjestelmällä tarkoitetaan tiettyä teknistä kokonaisuutta, joka koostuu samaan aihealueeseen (engl. domain) kuuluvista komponenteista.
- **Aktiivinen olio (engl. active object).** Aktiivinen olio voi olla säie, muttei kuitenkaan oma prosessi.
- **Komponentti (engl. component).** Komponentti on korvattavissa oleva kokonaisuus.
- **Luokka (engl. class).** Tyypillisesti staattisesta kirjastosta saatava olio-tyyppi.
- **Tuleva nuoli.** Esittää tarjottua rajapintaa.
- **Lähtevä nuoli.** Esittää vaadittua rajapintaa.

Toteutettu järjestelmä koostuu neljästä alijärjestelmästä: sovellus-, multimedia-, median koodaus- ja verkko-alijärjestelmästä. Alijärjestelmät esitellään myöhemmin tarkemmin. Koko järjestelmälle on asetettu seuraavat tehtävät:

- **Vastaanota komento käyttäjältä.** Sovelluksessa on käyttöliittymä, jonka avulla käyttäjä voi kontrolloida järjestelmän toimintaa.
- **Muodosta puhelu.** Signalointiprotokollan tehtävänä on paikallistaa puhelun vastaanottaja ja muodostaa siihen yhteys.

[18] Perustuu OBP Research Oy:n ReaGeniX-työkalun symbolipalettiin.

- **Muodosta mediayhteys.** RTSP:n tehtävänä on esittää vastaanottajalle pyyntö aloittaa tiedonsiirto.
- **Lue tieto videokameralta.** Järjestelmässä on tuki videokameralle, jonka kuvaa voidaan reaaliaikaisesti lähettää verkkoon.
- **Koodaa videokameralta luettu tieto MPEG-4 -formaattiin.** Järjestelmä on kykenevä pakkaamaan videokameran kuvan vähemmän tilaa vievään muotoon.
- **Lähetä tieto verkkoon.**
- **Lue tieto verkosta.**
- **Pura MPEG-4 koodaus.** Vastaanotettu pakattu videotieto puretaan ennen sen näyttämistä näytöllä.
- **Näytä videokuva ruudulla.**



Kuva 5. Toteutetun järjestelmän arkkitehtuuri.

Seuraavassa esitellään toteutetut alijärjestelmät tarkemmin.

Sovellus-alijärjestelmä (kuvassa 5 'application')

Sovellus suunniteltiin MVC-suunnittelumallin (Model-View-Controller) mukaisesti. Sovellus jaetaan siis kolmeen eri osaan siten, että osat ovat itsenäisiä ja toisistaan eristettyjä. MVC-mallin osat jaetaan seuraavasti [41]:

1. **Malli (engl. model)** sisältää varsinaisen sovelluskohtaisen tiedon.

2. **Näkymät (engl. view)** esittävät mallin sisältämän tiedon käyttäjälle.

3. **Kontrolliosa (engl. controller)** tarjoaa keinot mallin sisältämän tiedon manipulointiin. Yleensä tämä tapahtuu näkymien kautta eli graafinen käyttöliittymä sisältää tavallaan sekä näkymät että kontrolliosan.

Multimedia-alijärjestelmä (kuvassa 5 'streaming')

Koska kysymyksessä oli tutkimustyyppinen hanke, jonka alussa ei ollut tarkkaa tietoa eri protokollien suhtautumisesta toisiinsa tai sovellukseen, päädyttiin käyttämään liitutauluarkkitehtuuria (engl. blackboard). [42] Liitutauluarkkitehtuuri määrittelee kaksi erityyppistä komponenttia. Yleinen tietorakenne hallitsee järjestelmän tilaa ja yhteistä tietoa. Itsenäiset komponentit puolestaan käsittelevät ja muokkaavat yhteistä tietoa. Yleinen tietorakenne on usein aktiivinen komponentti, joka kutsuu tiedon muuttuessa itsenäisiä komponentteja. Itsenäisillä komponenteilla on vähän tai ei lainkaan interaktiota keskenään. [43, 44]

Erityisesti järjestelmän toteuttamisen jälkeentulivat liitutauluarkkitehtuurin edut selkeästi näkyviin. Vielä arkkitehtuurisuunnittelun ja toteutuksen alkuvaiheenkin aikana oli epäselvää, mitkä komponentit kytkeytyvät toisiinsa, millaiset ovat niiden väliset rajapinnat ja mikä toiminnallisuus kuuluu mihinkin komponenttiin. Siksi oli erityisen tärkeää, että sidosten muuttaminen ja toiminnallisuuden siirtäminen oli mahdollista ja helppoa ilman, että koko järjestelmän rakenne romahtaa ja joudutaan tekemään suuria muutoksia kaikkiin luotuihin komponentteihin.

Multimediaosan ytimenä on aktiivinen streamer-komponentti, johon mediaprotokollat yhdistettiin. Streamer-komponentti ylläpitää liitutaulu-arkkitehtuurin määrittämää yleistä tietorakennetta. Streamer-komponentti sisältää tiedon kaikkien siihen liitettyjen protokollakomponenttien tiloista. Kaikki alijärjestelmien ja komponenttien välillä siirtyvä kontrolli-informaatio ja tieto siirtyy streamer-komponentin kautta. Protokollat toteutettiin toisistaan mahdollisimman riippumattomina tapahtumapohjaisina kompo-

nentteina. Näin pystyttiin testaamaan ja kehittämään jokainen protokolla täysin itsenäisenä osanaan. Toinen kantava ajatus tämän arkkitehtuurin taustalla oli se, että näin tehtiin mahdolliseksi vaihtaa sovelluksessa oleva protokollatoteutus toiseen tekemällä vain pieniä muutoksia koostavaan komponenttiin. Näin suunniteltuna helpotetaan protokollakomponenttien integrointia muihin olemassa oleviin sovelluksiin. [42]

Tarkempi tutkimus paljastaa streamer-komponentin toiminnan. Se on aktiivinen komponentti, joka huolehtii yhteyksistä muihin alijärjestelmiin ja yhdistää multimedia-alijärjestelmän komponentit toisiinsa. Komponentin ytimenä on silmukka, joka huolehtii kaikesta sovelluksen sisältämästä toiminnallisuudesta.

- Luetaan käyttöjärjestelmältä tulevat viestit ja toimitaan niiden mukaisesti. Tyypillisiä käyttöjärjestelmäviestejä ovat käyttöliittymältä ja koodekilta tulevat ohjausviestit.

- Luetaan verkosta tuleva tieto ja toimitetaan tiedo siitä vastuussa olevalle komponentille.

- Luetaan komponenteilta mahdollisesti tuleva tieto ja lähetetään se verkkoon.

Streamer-komponentti sitoo siis kaikki muut alijärjestelmän komponentit yhteen ja huolehtii kaikesta kommunikaatiosta alijärjestelmän ulkopuolelle. Multimedia-alijärjestelmän kaikki laitteistoriippuvat osat on toteutettu streamer-komponentissa. Siksi muita alijärjestelmän komponentteja ei tarvitse muuttaa, jos halutaan siirtää toteutus toiseen käyttöjärjestelmäympäristöön.

Streamer-komponentti oli koko järjestelmän ainoa aktiivinen komponentti. Tästä seurasi muutamia positiivisia seikkoja toteutuksen kannalta. Nyt ei ollut tarvetta pohtia eri säikeiden käynnistysjärjestystä tai sitä, kuinka säikeet saadaan hallitusti ajettua alas. Monisäikeisessä järjestelmässä virhetilanteista toipuminen on usein vaikeata.

Verkko-alijärjestelmä (kuvassa 5 'network')

Järjestelmään suunniteltiin asynkroninen verkko-alijärjestelmä, jossa on tuki TCP- ja UDP-yhteyksille. Tutkimuksissa on huomattu, että monisäikeinen ympäristö yhdistettynä asynkroniseen tietoliikenteeseen on tehokkain tapa toteuttaa jatkuva-aikaista multimediaa jakeleva palvelin [45]. Alijärjestelmän rajapinta määriteltiin siten, että se ei itsessään ottaisi kantaa käytettävään verkkoteknologiaan. Tällä pyritään helpottamaan multimedia-alijärjestelmän siirtämistä ympäristöihin, joiden verkko-rajapinta ei ole identtinen tässä toteutuksessa käytetyn BSD-rajapinnan kanssa.

Median koodaus -alijärjestelmä (kuvassa 5 'media coding')

Median koodaus -alijärjestelmästä on kuvassa 5 esitetty sen keskeiset komponentit ja rajapinnat. Alijärjestelmän sisäistä toimintaa tai rakennetta ei kuitenkaan tässä yhteydessä voida käsitellä, vaan median koodaus-alijärjestelmä nähtiin black-box tyyppisenä komponenttina, johon rakennettiin sovelluksen käyttöön sopiva sovitin.

3.4 Pohdinta

Tieto-keskeinen (engl. data-centered) arkkitehtuuri osoittautui soveliaaksi prototyypin toteuttamiseen. Tyyli tukee luontaisesti muokattavuutta, joka on ensisijaisen tärkeätä tutkimushankkeessa. Muokattaessa järjestelmää tuotteenomaisemmaksi voi jokin muu arkkitehtuurityyli olla parempi vaihtoehto. Esimerkiksi tietovirta-arkkitehtuurilla (engl. data flow) on mahdollista saavuttaa parempi tehokkuus ja skaalautuvuus, jotka ovat tuotteelle tärkeitä laatuattribuutteja.

Käyttötapaukset antoivat kattavan kuvan vaadittavista protokollista. Luoduilla protokollilla olisi mahdollista toteuttaa myös muita mielenkiintoisia käyttötapauksia, kuten multimediaviestien välittäminen. Näin erityisesti SIP- ja RTSP-protokollille aiheutuvat laajemmat vaatimukset tulisivat esiin.

4. Alustan toteutus

4.1 Toteutusympäristö

Järjestelmä toteutettiin käyttäen Windows NT -työasemia ja Microsoft:in Visual C++ 6.0 ja Embedded Visual C++ 3.0 -kehitysympäristöjä. Protokollien toteutuskielenä käytettiin ANSI C:tä ja sovellusten laitteistoriippuvaiset osat, kuten käyttöliittymät ja verkkoliittymät, toteutettiin ANSI C:llä tai C++:lla.

Sovelluksista toteutettiin versiot sekä Windows NT -ympäristöön että PocketPC[20] -ympäristöön. PocketPC-ympäristössä laitteistona käytettiin Compaq:in iPaq[21]-laitteita, joihin oli liitetty MPEG-4-pakkauspiiri ja videokamera. Tiedonsiirtoyhteytenä käytettiin sekä langallista että langatonta LAN-linkkiä.

4.2 Real-Time Protocol

Toteutettu RTP-protokolla on minimitoteutus RFC 1889:sta [7]. Komponenttiin ei toteutettu samaisessa spesifikaatiossa määriteltyä RTCP-protokollaa. RTCP nähtiin parhaaksi toteuttaa erillisenä komponenttina, mutta sitä ei aikataulun vuoksi ehditty tehdä. RTCP:n käyttö testiympäristössä ei osoittautunut pakolliseksi, vaan sovellukset toimivat mainiosti myös ilman sitä.

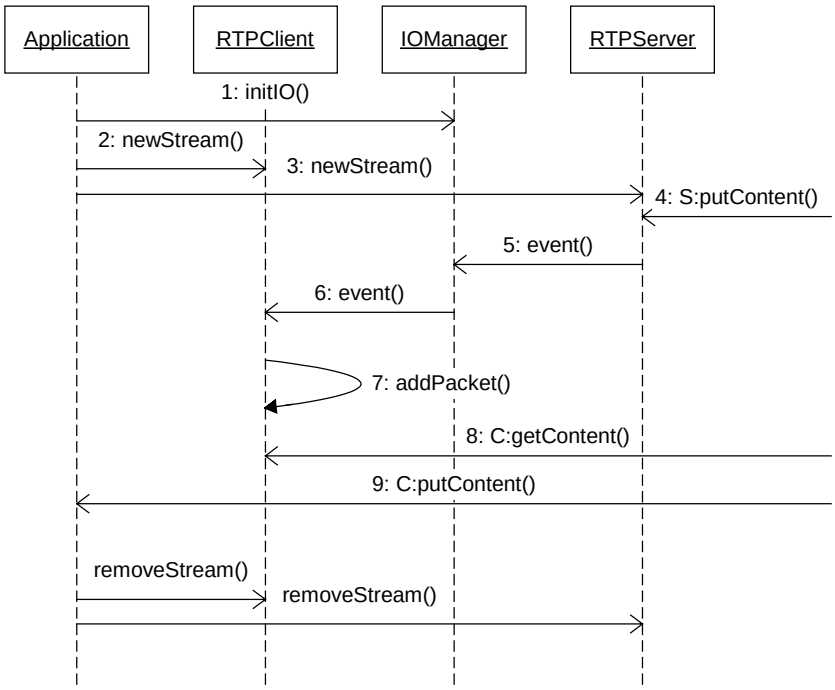
Kuvassa *6* [42] olevassa viestikaaviossa on esitetty RTP-komponentin tyypillinen käyttöskenaario. Käyttöskenaario ei kuvaa toteutettua sovellusta, vaan toimii esimerkkinä siitä, kuinka komponenttia tulisi käyttää. Lisäksi on huomattava, että kuvassa on esitetty erillisinä osina RTP:n asiakas- ja palvelinosat, vaikka todellisuudessa ne ovatkin samassa komponentissa.

Kuvassa oleva skenario kuvaa kaksisuuntaisen multimediapuhelun tiedon siirtoa. Sovellus alustaa IOManagerin (1) sekä RTP:n asiakas- ja palvelinosat (2 ja 3). Tämän jälkeen enkooderi aloittaa videokameralta saamansa ja pakkaamansa tiedon syöttämisen palvelinosalle (4). Palvelinosa huolehtii tämän tiedon lähettämisestä edelleen vastaanottajalle IOManagerin avustuksella (5). Samanaikaisesti aloitetaan vastapään lähettämän tiedon vastaanottaminen. IOManager syöttää vastaanottamansa tiedon asiakasosalle (6), joka käsittelee sen myöhemmin esitettävällä tavalla (7). Dekooderi käy noutamassa

[19] http://www.microsoft.com/mobile/pocketpc/
[20] http://www.microsoft.com/mobile/pocketpc/
[21] http://www.compaq.com/products/iPAQ/

asiakasosalta saapunutta tietovirtaa (8) ja purettuaan koodauksen syöttää se tiedon edelleen sovellukselle (9), joka näyttää sen näytöllä.

RTP:n spesifikaatiossa ei ole tarkasti määrätty toteutustekniikkaa protokollalle. Siksi jouduttiin pohtimaan, mitkä tehtävät kuuluvat varsinaiselle protokollakomponentille ja mitkä koodekeille tai järjestelmän muille osille. Kysymykset eivät olleet helppoja, sillä monissa seikoissa vastuu olisi ollut varsin helppo antaa kummalle tahansa. Seuraavassa käsitellään tarkemmin muutamia RTP:n tehtäviä, jotka liittyvät multimediatiedon käsittelyyn.



Kuva 6. RTP-komponentin käyttöskenaario.

Synkronointi

Spesifikaation mukaan RTP-paketit sisältävät aikaleiman, joka ilmaisee paketissa olevan mediatiedon näytteistämishetken. Yleensä mediapalvelimet sijoittavat kuitenkin aikaleimaksi kuluneen media-ajan kyseisen medialeikkeen alusta alkaen. Näin siis RTP-protokolla pystyy synkronisoimaan media-ajan ja todellisen ajan toisiinsa. Riippuu koko järjestelmän toteutusarkkitehtuurista, onko RTP:n tarpeen huolehtia synkronisoinnista vai ei. Molemmat vaihtoehdot ovat mahdollisia. Luodussa toteutuksessa synkronisointia ei annettu RTP:n tehtäväksi. Sillä sen lisäksi, että haluttiin RTP-komponentin

olevan passiivinen, haluttiin median kontrollointi ja synkronisointi toteuttaa erillisessä media-komponentissa.

Tärkein syy synkronisoinnin ulkoistamiselle on joustavuuden saavuttaminen. Jos ajatellaan esimerkiksi mediapalvelinta, niin se voi vaatia RTP:ltä kahdenlaista toiminallisuutta. Kun palvelin lähettää reaaliaikaista mediatietoa, on tyypillistä, että RTP lähettää kaiken saatavilla olevan tiedon välittömästi. Jos taas kyseessä on mediapalvelin, joka lähettää esimerkiksi ennalta tallennettuja uutissähkeitä, täytyy olla jokin mekanismi, jolla säädellään sitä, kuinka nopeasti RTP lähettää tietoa. Koska nämä vaatimukset vaihtelevat sovelluskohtaisesti, ei ole järkevää toteuttaa synkronisointia kiinteästi RTP-komponenttiin.

Toinen syy synkronoinnin ulkoistamiseen on se, että siten kyseistä RTP-toteutusta on mahdollista käyttää ulkoisten synkronisointitulkkien kanssa, jotka ohjeistuksen mukaan synkronisoivat ääntä, kuvaa ja tekstiä esimerkiksi www-selaimeen. Tällaisia synkronisointitulkkeja ovat mm. SMIL-toteutukset [46].

Puskurointi ja vanhentuneiden pakettien tuhoaminen

Toteutetussa järjestelmässä RTP-komponentti huolehtii pakettien puskuroinnista ja vanhentuneiden pakettien tuhoamisesta. Pakettien tuhoamisessa ei kuitenkaan mennä synkronisoinnista vastaavan komponentin vastuualueelle, vaan synkronisoinnista vastaava komponentti kertoo, mitä ajan hetkeä vanhemmat paketit on luvallista tuhota. Periaatteessa RTP voisi itsenäisestikin hoitaa tämän tehtävän, mutta tällä tavoin voidaan ulkoistaa RTP:stä monia järjestelmäriippuvia osia, kuten ajastimet ja kellot.

Näin menetellen saadaan luotua RTP:stä monikäyttöisempi, sillä sovelluksen käyttäytymistä virhetilanteissa voidaan hallita paremmin. Käyttäytyminen riippuu sovelluksen käyttötilanteesta. Jos katsotaan staattista tietoa mediapalvelimelta, on järkevää näyttää käyttäjälle kaikkien pakettien sisältö riippumatta siitä, kuinka pitkään ne ovat puskuroituneet. Toisaalta tilanteessa, jossa media on reaaliaikaista, ei ole tarvetta ylläpitää suurta puskuria, sillä silloin suuri osa tiedosta on jo vanhentunutta. Tällaisen toimintalogiikan rakentaminen RTP-komponentin sisään ei ole perusteltua, vaan on järkevämpää jättää asian päätäntä sen ulkopuolelle.

Eri järjestyksessä tulleiden pakettien järjestäminen

Koska RTP käyttää siirtoprotokollanaan UDP:tä, on mahdollista, että lähetetyt paketit saapuvat verkosta eri järjestyksessä kuin ne on lähetetty. RTP-pakettien otsikossa on järjestysnumerokenttä, jonka perusteella protokolla tietää pakettien oikean järjestyksen. RTP huolehtii siis siitä, että koodekille annettavat paketit ovat oikeassa järjestyksessä.

Tietokehysten eheyden hallinta

RTP-otsakkeen M-bittiä käytetään yleensä ilmaisemaan, onko kyseisessä RTP-paketissa kokonainen mediakehys. Koodekin toiminnallisuuden mukaan voidaan koostaa useasta paketista kokonainen mediakehys, informoida koodekkia keskeneräisestä mediakehyksestä tai jättää tämä huomioimatta ja jättää vastuu koodekille. Tässä tapauksessa bitin merkitystä ei huomioitu, sillä kaikkien mediakehysten oletettiin olevan niin pieniä, että ne mahtuvat yhteen UDP-kehykseen.

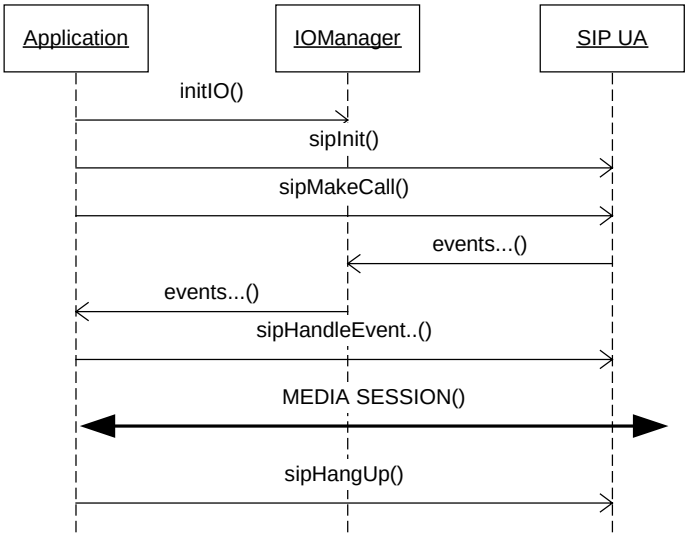
4.3 Session Initiation Protocol

SIP-komponentti toteuttaa SIP-käyttäjäagentin asiakas- ja palvelinosat (engl. SIP user agent client and server). Koska RFC:n määrittelemä toiminnallisuus on erittäin laaja ja moninaisiin käyttötarkoituksiin soveltuva, on tämän komponentin toiminnallisuutta rajattu. Rajaus on tehty siten, että se noudattaa varsin läheisesti RFC 2543:n [8] määrittämiä minimitoteutuksen toiminnallisia vaatimuksia, ja viesteihin sisällytetyt otsikkokentät ovat 3GPP:n minimimääritysten mukaiset [47]. Näiden vaatimusten lisäksi toteutettiin lisätoiminnallisuuksia omien tarpeiden mukaisesti.

Toteutuksessa tuettuja viestejä ovat INVITE, ACK, BYE, CANCEL ja REGISTER. Viestit mahdollistavat puhelun perustoiminnot, kuten puhelun perustamisen ja purkamisen sekä käyttäjän rekisteröinnin hakemistopalveluun. Komponentti on kykenevä muodostamaan puhelun ottamalla suoran yhteyden vastaanottajaan tai muodostamalla puhelun välipalvelimen välityksellä.

Protokollakomponentti toteutettiin itsenäisenä tapahtumapohjaisena komponenttina. Vaikka komponentti toteutettiin ANSI C:llä, se on rakenteeltaan varsin oliomainen. Kun puhelua alustetaan, komponentilta pyydetään uusi SIP-istunto (engl. session). Komponentti palauttaa sovellukselle osoittimen struktuuriin, joka sisältää kaiken puheluun ja sen tilaan liittyvän tiedon. Tässä struktuurissa oleva tieto ei ole relevanttia sovellukselle, joten sovellus ei tiedä struktuurin muotoa ja sisältöä, vaan joutuu kysymään kaiken tarvitsemansa komponentilta. Tällä tavoin mahdollistetaan se, että sovellus pystyy helposti käsittelemään useita yhtäaikaisia puheluita.

Kuvassa *7* esitetään tyypillinen komponentin käyttöskenario, jossa puhelu perustetaan ja puretaan. On huomattava, että kuvassa ei ole esitetty varsinaisia lähetettäviä SIP-viestejä, vaan kuvauksessa keskitytään komponentin käyttöön. Puhelun muodostuksessa ja purkamisessa siirtyvät SIP-viestit esitellään myöhemmin.



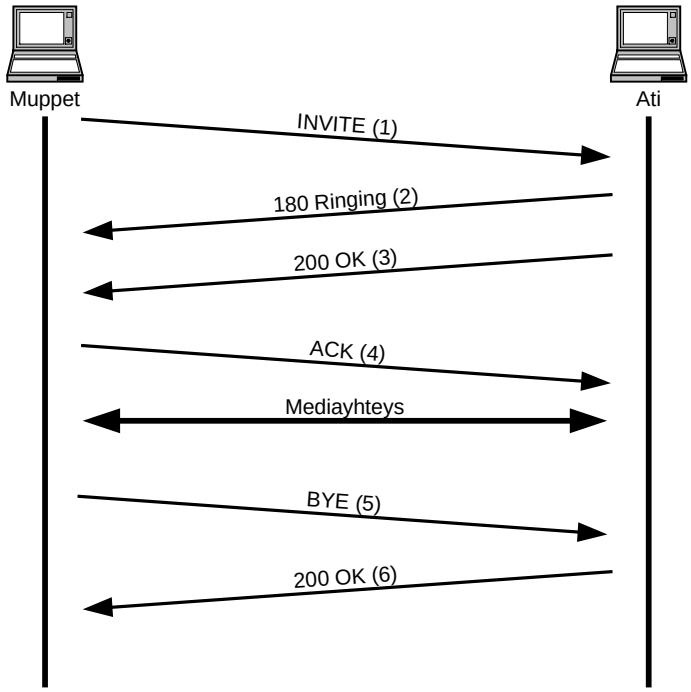
Kuva 7. SIP-komponentin käyttöskenaario.

Aluksi sovellus alustaa IOManagerin ja pyytää SIP-komponentilta uutta istuntoa. Istunnon tunniste toimi avaimena, jonka perusteella sovellus voi erotella sillä samanaikaisesti käynnissä olevat SIP-puhelut. Kun käyttäjä on valinnut puhelun kohteen, sovellus pyytää SIP-komponenttia muodostamaan puhelun. Tämän jälkeen SIP-komponentti kommunikoi välipalvelimen ja puhelun vastaanottajan kanssa vaihtaen useita viestejä. Koska kaikki saapuva liikenne kulkee sovelluksen kautta, tietää sovellus puhelun tilan kaiken aikaa. Kun puhelu on saatu pystytettyä, sovellus voi muodostaa mediayhteyden saatujen parametrien perusteella. Lopuksi puhelu lopetetaan.

Seuraavassa esitetään kaksi tyypillistä SIP-protokollan käyttötapausta. Tapauksien avulla saa kuvan SIP:n toiminnasta ja viestien sisäisestä rakenteesta. Esimerkkiviesteinä käytetään toteutetun protokollan lokitiedostoista poimittuja paketteja.

4.3.1 Suora puhelu vastaanottajalle

Kuvassa *8* on esitetty SIP-puhelun muodostaminen ja purkaminen suoraan kahden osapuolen välillä. Esitetyssä tapauksessa Muppet aloittaa puhelun muodostamisen ja Ati vastaa siihen.



Kuva 8. Suora puhelu vastaanottajalle.

Puhelun syntyminen alkaa siitä, kun soittaja (Muppet) lähettää puhelun vastaanottajalle (Ati) kuvassa *9* esitetyn INVITE-viestin (1). Jos muodostettava puhelu sisältää jonkin mediayhteyden, lähetetään tiedot lähettäjän mediaparametrit kuvattuna SDP-protokollalla. Seuraavassa esitettävät viestit on poimittu versiosta, jota ei oltu integroitu sovellukseen. Tästä syystä viestit eivät sisällä SDP-muotoisia kuvauksia, vaan pelkästään tekstin, jolla testattiin hyötykuorman toimivuus.

```
INVITE sip:ati@130.188.91.200 SIP/2.0
Contact: sip:muppet@130.188.92.125:5060
To: sip:ati@130.188.91.200
From: sip:muppet@ele.vtt.fi
Call-ID: 1764057112@130.188.92.125
CSeq: 0 INVITE
Expires: 3600
Subject:
Via: SIP/2.0/UDP 130.188.92.125:5060
Content-Type: NONE
Content-Length: 22

MediaOsoiteeni on 1234
```

Kuva 9. INVITE-viesti (1).

INVITE-viesti on esimerkki tyypillisestä kyselyviestistä (engl. request). Viestin ensimmäinen rivi ilmaisee viestin tyypin, vastaanottajan ja käytettävän protokollan. Sen jälkeen on joukko otsikoita, jotka tarkentavat kyseisen viestin sisältöä. Viestin lopussa on

mahdollisesti hyötykuorma (engl. content), jossa voidaan siirtää sovelluskohtaista informaatiota.

Esitetyssä viestissä on seuraavat otsikot. Contact-otsikko ilmaisee lähettäjän fyysisen sijainnin, johon viestin vastaanottajan pyydetään lähettämään vastauksensa. To- ja From-otsikot ilmaisevat puhelun alkuperäisen vastaanottajan ja soittajan. On huomattava, että reititystä ei missään tilanteessa tehdä To-otsikon perusteella vaan aina käyttäen ensimmäisellä rivillä olevaa Request-URI-kenttää. Call-ID-otsikko ilmaisee puhelun tunnisteen. Tunniste voi olla mikä tahansa uniikki tunniste, mutta yleensä se muodostetaan satunnaisesta osasta ja soittajan IP-osoitteesta. Via-otsikko kertoo reitin, jota pitkin viesti on kulkenut. Tätä tietoa voidaan käyttää vastauspaketin reititykseen. Kaikilla viesteillä on järjestysnumero, joka lähetetään CSeq-otsikkossa. Content-Type-otsikko ilmaisee viestissä mahdollisesti olevan hyötykuorman tyypin. Content-Length-otsikko puolestaan ilmaisee hyötykuorman pituuden.

Vastaanottaja vastaa INVITE-viestiin kertomalla tilansa. Kuvassa 10 esitetty '180 Ringing' -viesti (2) on esimerkki vastausviestistä. Esitetty viesti kertoo, että käyttäjäagentti on saanut viestin vastaan ja koettaa nyt herättää käyttäjän huomion.

```
SIP/2.0 180 Ringing
Contact: sip:ati@130.188.91.200:5061
To: sip:ati@130.188.91.200
From: sip:muppet@ele.vtt.fi
Call-ID: 1764057112@130.188.92.125
CSeq: 0 INVITE
Record-Route:
Via: SIP/2.0/UDP 130.188.92.125:5060; received=127.0.0.1
Content-Type:
Content-Length: 0
```

Kuva 10. 180 Ringing -viesti(2).

Vastausviesti on muodostettu kopioimalla useita INVITE viestin otsikoita: To, From, Call-ID, CSeq ja Via. On huomattava, että To- ja From-otsikot eivät vaihda paikkaa, vaan säilyvät samana koko puhelun ajan. Ne ilmaisevat siis puhelun suuntaa, ei kulkevan viestin suuntaa.

Kun käyttäjä on huomannut tulevan puhelun ja päättää vastata siihen, lähettää käyttäjäagentti soittajalle kuvassa 11 esitetyn '200 OK'-viestin (3). Viesti ilmaisee, että soittajan pyytämä puhelu on hyväksyttävä, mukaanlukien mahdollisessa SDP-kuvauksessa kerrotut mediaparametrit.

```
SIP/2.0 200 OK
Contact: sip:ati@130.188.91.200:5061
To: sip:ati@130.188.91.200
From: sip:muppet@ele.vtt.fi
Call-ID: 1764057112@130.188.92.125
CSeq: 0 INVITE
Record-Route:
Via: SIP/2.0/UDP 130.188.92.125:5060; received=127.0.0.1
Content-Type: NONE
Content-Length: 22

MediaOsoiteeni on 4321
```

Kuva 11. 200 OK -viesti (3).

'200 OK'-viesti on muodostettu samoin kuin '180 Ringing'-viestikin, kopioimalla kenttiä INVITE-viestistä. Lisäksi viestiin sisällytetään puhelun vastaanottajan mediainformaatio.

Kun soittaja saa vastaanottajan hyväksynnän puheluun, lähettää soittaja vielä kuittauksen vastaanottajalle. Tämän jälkeen osapuolet muodostavat mediayhteyden, jonka avulla he kommunikoivat. Kuvassa 12 esitetty ACK-viesti (4) on siis hyväksyntä sille, että soittaja on kykenevä ja halukas käyttämän vastaanottajan mediaparametreja.

```
ACK sip:ati@130.188.91.200:5061 SIP/2.0
To: sip:ati@130.188.91.200:5061
From: sip:muppet@ele.vtt.fi
Call-ID: 1764057112@130.188.92.125
CSeq: 0 ACK
Via: SIP/2.0/UDP 130.188.92.125:5060
Content-Type:
Content-Length: 0
```

Kuva 12. ACK-viesti (4).

ACK-viestin lähetettyään soittaja muodostaa '200 OK'-viestissä saamiensa tietojen perusteella mediayhteyden vastaanottajaan. Mediayhteydet ovat tyypillisesti yksisuuntaisia, joten vastaanottaja muodostaa ACK-viestin jälkeen yhteyden soittajaan INVITE-viestissä saamiensa tietojen perusteella.

Kun jompikumpi osapuolista haluaa lopettaa puhelun, lähettää se kuvassa *13* esitetyn BYE-viestin (5). Tässä tapauksessa puhelun soittaja päättää puhelun.

```
BYE sip:ati@130.188.91.200 SIP/2.0
To: sip:ati@130.188.91.200
From: sip:muppet@ele.vtt.fi
Call-ID: 1764057112@130.188.92.125
CSeq: 1 BYE
Route:
Via: SIP/2.0/UDP 130.188.92.125:5060
Content-Type:
Content-Length: 0
```

Kuva 13. BYE-viesti (5).

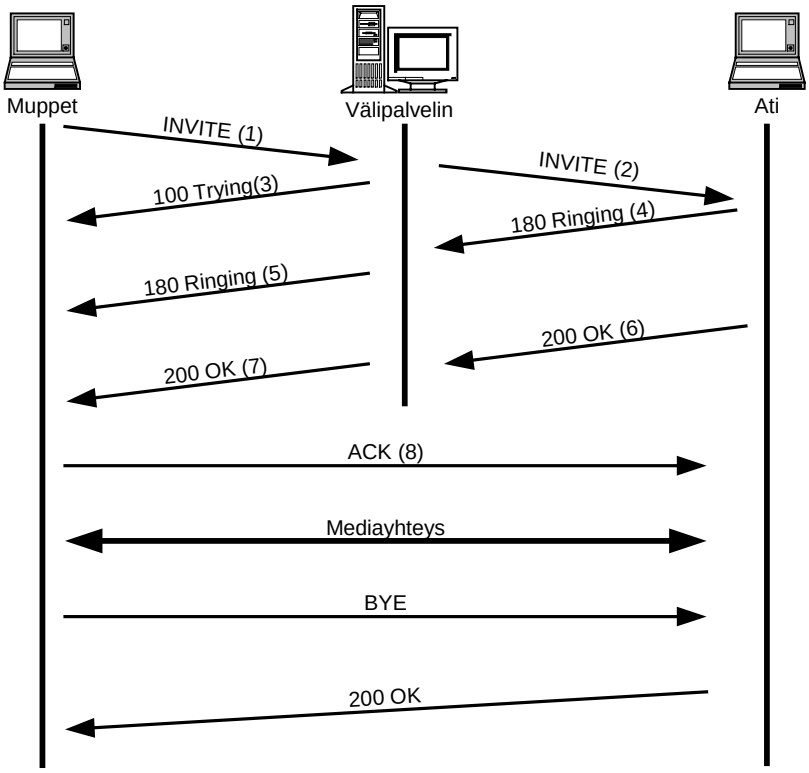
Vastaanotettuaan BYE-viestin puhelun vastaanottaja hyväksyy puhelun lopettamisen lähettämällä kuvassa 14 esitetyn '200 OK'-viestin (6). Tämän jälkeen puhelun osapuolet purkavat media yhteyden.

```
SIP/2.0 200 OK
Contact: sip:ati@130.188.91.200:5061
To: sip:ati@130.188.91.200
From: sip:muppet@ele.vtt.fi
Call-ID: 1764057112@130.188.92.125
CSeq: 1 BYE
Record-Route:
Via: SIP/2.0/UDP 130.188.92.125:5060
Content-Type:
Content-Length: 0
```

Kuva 14. 200 OK -viesti (6).

4.3.2 Puhelu välipalvelimen kautta

Kuvassa *15* on esitetty välipalvelimen kautta tapahtuva puhelun muodostus. Siinä soittaja (Muppet) haluaa muodostaa puhelun vastaanottajan kanssa (Ati), mutta ei kuitenkaan tiedä, missä fyysisessä osoitteessa vastaanottajan kyseisellä hetkellä on. Tällöin soittaja lähettää pyynnön välipalvelimelle, jonka avulla puhelu muodostetaan.



Kuva 15. Välipalvelimen avulla muodostettava puhelu.

Välipalvelinta hyödyntävä puhelu alkaa kuvassa 16 esitetyllä INVITE-viestillä (1). Erona suoran puhelun tapaukseen on, että puhelun vastaanottajan fyysistä sijaintia ei tiedetä. Tästä syystä viestin Request URI-kenttä on sähköposti-osoitteen kaltainen SIP-URI, joka ei välttämättä osoita mihinkään fyysiseen osoitteeseen. Lisäksi viesti lähetetään välipalvelimelle, joka pyrkii etsimään puhelun vastaanottajan fyysisen sijainnin tai palvelimen, joka tietää sen.

```
INVITE sip:ati@ele.vtt.fi SIP/2.0
Contact: sip:muppet@130.188.92.125:5060
To: sip:ati@ele.vtt.fi
From: sip:muppet@ele.vtt.fi
Call-ID: 1818308920@130.188.92.125
CSeq: 1 INVITE
Expires: 3600
Subject:
Via: SIP/2.0/UDP 130.188.92.125:5060
Content-Type: NONE
Content-Length: 22

MediaOsoiteeni on 1234
```

Kuva 16. INVITE-viesti (1).

Välipalvelin muokkaa viestin (1) ja lähettää sen kuvassa 17 esitetyssä muodossa (2) eteenpäin vastaanottajalle.

```
INVITE sip:ati@130.188.94.100:5061 SIP/2.0
Via: SIP/2.0/UDP lin1064.ele.vtt.fi:5060; branch=1068710129-1
Via: SIP/2.0/UDP 130.188.92.125:5060
Contact: sip:muppet@130.188.92.125:5060
To: sip:ati@ele.vtt.fi
From: sip:muppet@ele.vtt.fi
Call-ID: 1818308920@130.188.92.125
CSeq: 1 INVITE
Expires: 3600
Subject:
Content-Type: NONE
Content-Length: 22

MediaOsoiteeni on 1234
```

Kuva 17. INVITE-viesti (2).

Välipalvelin lisää viestiin toisen Via-otsikon, joka ilmaisee, minkä välipalvelimen kautta viesti on kulkenut. Jos välipalvelimia on reitillä useampia, lisäävät kaikki oman Via-otsikkonsa. Näin pystytään välttämään silmukoiden syntymistä viestiketjuissa.

Kun välipalvelin on lähettänyt INVITE-viestin puhelun vastaanottajalle, lähettää välipalvelin puhelun soittajalle ilmoituksen siitä, että välipalvelin on saanut viestin vastaan ja on tekemässä puhelulle jotain. Tässä tapauksessa välipalvelin lähettää soittajalle kuvassa 18 esitetyn '100 Trying'-viestin(3).

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 130.188.92.125:5060
From: sip:muppet@ele.vtt.fi
To: sip:ati@ele.vtt.fi
Call-ID: 1818308920@130.188.92.125
Cseq: 1 INVITE
Date: Tue, 27 Nov 2001 08:12:14 GMT
Server: Columbia-SIP-Server/1.17
Content-Length: 0
```

Kuva 18. 100 Trying -viesti (3).

Kun vastaanottajan käyttäjäagentti saa INVITE-viestin vastaan, lähettää se kuvassa 19 esitetyn '180 Ringing'-viestin (4). Koska viesti on kulkenut välipalvelimen kautta vastaanottajalle, lähetetään myös vastaus kyseiseen viestiin välipalvelimen kautta. Yleisesti vastausviesti lähetään aina osoitteeseen, joka on ensimmäisessä Via-otsikossa.

```
SIP/2.0 180 Ringing
Contact: sip:ati@130.188.94.100:5061
To: sip:ati@ele.vtt.fi
From: sip:muppet@ele.vtt.fi
Call-ID: 1818308920@130.188.92.125
CSeq: 1 INVITE
Record-Route:
Via: SIP/2.0/UDP lin1064.ele.vtt.fi:5060; branch=1068710129-1
Via: SIP/2.0/UDP 130.188.92.125:5060; received=127.0.0.1
Content-Type:
Content-Length: 0
```

Kuva 19. 180 Ringing -viesti (4).

Vastaanottamastaan vastausviestistä välipalvelin poistaa oman Via-otsikkonsa ja lähettää kuvassa 20 esitetyn muokatun '180 Ringing' -viestin(5) edelleen puhelun soittajalle.

```
SIP/2.0 180 Ringing
Contact: sip:ati@130.188.94.100:5061
To: sip:ati@ele.vtt.fi
From: sip:muppet@ele.vtt.fi
Call-ID: 1818308920@130.188.92.125
CSeq: 1 INVITE
Record-Route:
Via: SIP/2.0/UDP 130.188.92.125:5060; received=127.0.0.1
Content-Length: 0
```

Kuva 20. 180 Ringing -viesti(5).

Kun vastaanottaja on päättänyt vastata puheluun, lähettää vastaanottajan käyttäjäagentti kuvassa 21 esitetyn '200 OK'-viestin (6) välipalvelimelle.

```
SIP/2.0 200 OK
Contact: sip:ati@130.188.94.100:5061
To: sip:ati@ele.vtt.fi
From: sip:muppet@ele.vtt.fi
Call-ID: 1818308920@130.188.92.125
CSeq: 1 INVITE
Record-Route:
Via: SIP/2.0/UDP lin1064.ele.vtt.fi:5060; branch=1068710129-1
Via: SIP/2.0/UDP 130.188.92.125:5060; received=127.0.0.1
Content-Type: NONE
Content-Length: 22

MediaOsoiteeni on 4321
```

Kuva 21. 200 OK -viesti(6).

Välipalvelin puolestaan muokkaa viestiä poistamalla toisen Via-otsikon ja lähettää viestin (7) edelleen puhelun soittajalle. Jos puhelun soittaja on tyytyväinen media parametreihin, se vastaa suoraan puhelun vastaanottajalle kuvassa 22 esitetyllä ACK-viestillä (8).

```
ACK sip:ati@130.188.94.100:5061 SIP/2.0
To: sip:ati@130.188.94.100:5061
From: sip:muppet@ele.vtt.fi
Call-ID: 1818308920@130.188.92.125
CSeq: 1 ACK
Via: SIP/2.0/UDP 130.188.92.125:5060
Content-Type:
Content-Length: 0
```

Kuva 22. ACK-viesti (8).

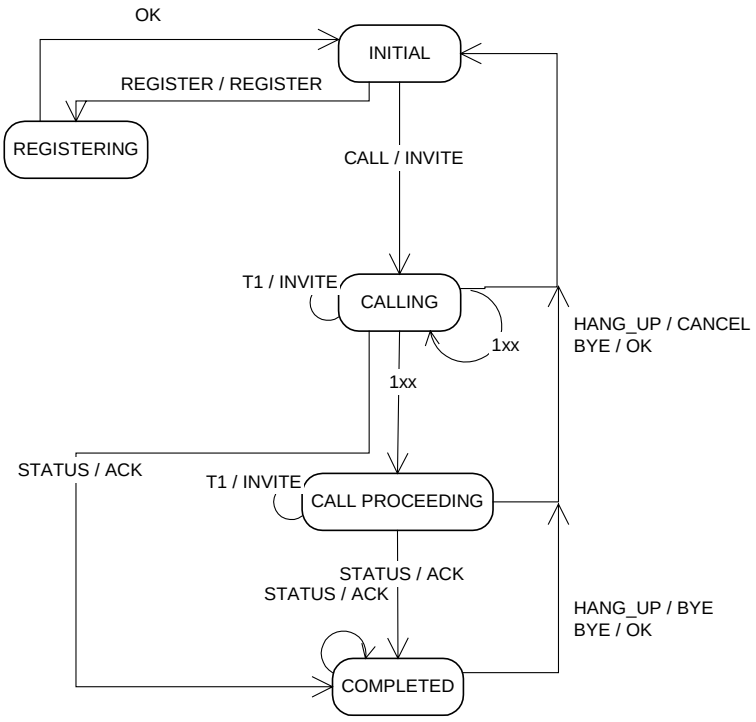
Tämän jälkeen muodostetaan media yhteys kuten edellisessäkin tapauksessa. Myös puhelun purkaminen tapahtuu kuten edellä.

4.3.3 Sisäinen toteutus ja testaus

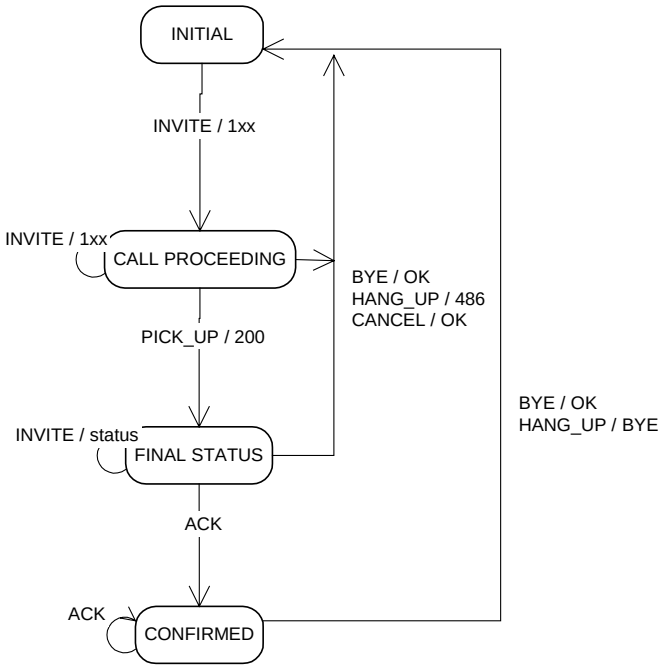
Sisäisesti SIP-komponentin toiminta toteutettiin käyttäen kahta tilakonetta, joilla kuvattiin puhelun käyttäytymistä asiakas- ja palvelinmoodeissa. Kuvissa 23 ja 24 on toiminnallisuus kuvattu tilakaavion muodossa. On huomattava, että osa tilasiirtymistä on tarkoituksella jätetty pois, jotta kuvasta olisi saatu selkeämpi. Kaikki olennaiset tilasiirtymät on kuitenkin kuvattu. Molemmissa tilakoneissa on esitetty *initial*-tila, joka on molemmille tilakoneille yhteinen. Riippuen ensimmäisen herätteen mukaan käyttäjäagentti siirtyy joko asiakas- tai palvelinmoodiin. Tilakoneiden suunnittelun pohjana on käytetty Schulzrinne:n artikkelia 'Signaling for Internet Telephony' [40], joka on kirjoitettu ennen RFC 2543:ta. Tilakoneet ovat kuitenkin yhteensopivia RFC:ssä määritellyn toiminnallisuuden kanssa.

Kuvassa 23 esitetty SIP-käyttäjäagentin asiakasmoodin tilakoneen toiminta perustuu viiteen tilaan. *Initial*-tila on molemmille tilakoneille yhteinen tila, jossa käyttäjäagentti on alustuksen jälkeen. Asiakasmoodissa käyttäjäagentti voi rekisteröidä käyttäjän rekisteröintipalvelimelle (*registering*-tila) tai aloittaa uuden puhelun muodostus (*calling*-tila). Kun käyttäjäagentti on päätynyt *calling*-tilaan, on se lähettänyt ensimmäisen INVITE-viestin vastaanottajalle tai välityspalvelimelle. Saamiensa vastausten mukaan käyttäjäagentti siirtyy *call proceeding* tai *completed* -tilaan. *Call proceeding* -tilassa käyttäjäagentti tietää vastaanottajan reagoineen jollain tavalla kutsuun. *Completed*-tilassa käyttäjäagentti on saattanut puhelun muodostuksen loppuun ja sovellus voi aloittaa mediayhteyden muodostamisen.

Kuvassa 24 esitetty SIP-käyttäjäagentin palvelinmoodin tilakoneen toiminta perustuu puolestaan neljään tilaan. Asiakasmoodin kanssa yhteisestä *initial*-tilasta käyttäjäagentti voi kutsun tultua siirtyä vain *call proceeding* -tilaan, jossa käyttäjäagentti on, kunnes käyttäjä on reagoinut kutsuun jollain tavalla. Jos käyttäjä haluaa vastaanottaa puhelun, siirtyy käyttäjäagentti *final status* -tilaan, johon se jää odottomaan puhelun lopullista vahvistamista. Vahvistuksen saavuttua käyttäjäagentti siirtyy *confirmed*-tilaa, jonka jälkeen sovellus voi muodostaa mediayhteyden.



Kuva 23. Asiakasmoodin tilakone.



Kuva 24. Palvelinmoodin tilakone.

Tilakoneet toteutettiin mahdollisimman varmatoimisiksi. Tilakone tarkistaa tulevien viestien toiminnan oikeellisuuden useilla eri tavoilla ja hylkää kaikki sopimattomat tapahtumat. Viestin täytyy liittyä parhaillaan käynnissä olevaan puheluun. Jos tilasiirtymä puolestaan vaikuttaa mahdottomalta, ei siihen reagoida lainkaan. Tarkistukset ovat tarpeellisia, sillä välipalvelin lähettää varsin usein vanhentuneita tai puhelusekvenssin ulkopuolisia viestejä. Näin tapahtuu erityisesti silloin, kun puhelun perustaminen keskeytetään ja hyvin pian sen jälkeen aloitetaan uuden puhelun perustaminen. Sekvenssiin kuulumattomia viestejä saapuu usein myös silloin, kun puhelun vastaanottaja on tavoittamattomissa ja puhelun perustaminen keskeytetään.

Komponentin toteutuksessa pyrittiin tekemään uusien viestien ja otsikoiden lisääminen mahdollisimman helpoksi. Sitä varten toteutettiin geneeriset toiminnot otsikkokenttien ja viestien käsittelyyn. Geneerisellä jäsentimellä (engl. parser) saavutettiin myös se etu, että näin toteutus ei ole niin haavoittuva tilanteessa, jossa sille syötetään väärin muodostettuja SIP-viestejä.

SIP-komponentti kehitettiin testaamalla sen toimintaa Columbia Universityn toteuttamaa SIP-palvelinta ja käyttäjäagenttia vasten. Kehittäminen tapahtui luomalla omaan toteutukseen viesti, joka lähetettiin joko palvelimelle tai käyttäjäagentille. Jos vastintoteutus reagoi pyydetysti, jatkettiin viestisekvenssiä uudella viestillä. Samalla tavalle kehitettiin oman käyttäjäagenttitoteutuksen käyttäytyminen sen vastatessa omalta tai Columbia Universityn toteutukselta lähettettyihin palvelupyyntöihin.

Columbian SIP-palvelimen tarjoamista palveluista toteutettu SIP-komponentti oli kykeneväinen käyttämään puhelun välittämistä ja käyttäjän rekisteröintiä. Käyttäjäagentin kanssa pystyttiin muodostamaan puhelu, mutta varsinaisia mediayhteyksiä ei muodostettu. Käyttäjäagentin kanssa muodostettua yhteyttä ei ollut mahdollista ajaa alas kaikissa tilanteissa hallitusti, sillä Columbian käyttäjäagentti ei hallinnut tilaa, jossa mediayhteyttä ei oltu muodostettu sen haluamalla tavalla. Lisäksi komponentin toimintaa testattiin Ubiquityn käyttäjäagentin kanssa.

4.4 Pohdinta

IETF:n määrittelemät protokollat osoittautuivat miellyttäviksi toteuttaa. Protokollien spesifikaatiot olivat erittäin selkeästi ja hyvin suunniteltuja. Erityinen huomio kiinnittyi protokollien sisäiseen skaalautuvuuteen. Spesifikaatio määrittelee protokollalle paljon toiminnallisuuksia, mutta toiminnallisuus on määritelty niin modulaariseksi, että omiin tarpeisiin soveltuva toiminnallinen kokonaisuus on helppo koostaa.

Koska IETF:n määrittelyt, nimenomaan vaadittujen toiminnallisuuksien määrittelyt, ovat varsin väljät, on testaaminen erityisen tärkeätä. Toteutettuja protokollia testattiin valmiiden kaupallisten sovellusten kanssa. Varsin yleistä oli, että olemassa olevat toteutukset eivät täysin noudattaneet spesifikaatioiden mukaista toiminallisuutta. Olisi kuitenkin hyvä, että uudet toteutukset olisivat mahdollisimman standardinmukaisia, mutta samalla myös yhteensopivia olemassa olevien toteutusten kanssa. Siksi jouduttiin tekemään varsin paljon yhteensopivuustestausta ja muutoksia siten, että toteutus tulisi toimeen myös epästandardien toteutusten kanssa.

Toteutuksista onnistuttiin tekemään sisäisesti skaalautuvia. Erityisesti SIP-protokollaan oli helppo lisätä myöhemmin uusia viestejä ja uutta toiminnallisuutta. Osa RTP:n tehtävistä jaettiin muille komponenteille, joten RTP:n vastuualue jäi varsin kapeaksi. Tällä saavutettiin joustavuutta, mutta toisaalta se vaatii komponentin käyttäjältä enemmän tietämystä. Olisikin ehkä mahdollista toteuttaa geneerisempi RTP-komponentti, jossa otettaisiin enemmän vastuuta kappaleessa 4.2. mainituista toiminnallisuuksista. Näin vähennettäisiin komponentin käyttäjän vastuuta.

Tällä hetkellä toteutetun järjestelmän suurin puute on RTCP:n puuttuminen. Testiympäristössä sen käyttö ei osoittautunut välttämättömäksi, joten se jätettiin kokonaisuudessaan toteuttamatta, vaikka sen vaikutukset arkkitehtuuriin huomioitiin. Jos toteutusta käytetään todellisessä ympäristössä, jossa verkko-olosuhteet vaihtelevat ajan myötä, on tarpeen toteuttaa multimedia-alijärjestelmään uusi komponentti, joka vastaanottaa RTCP-paketteja ja tietoja paikallisesta järjestelmästä ja muokkaa järjestelmän toimintaa niiden perusteella.

Jotta toteutettuja protokollia voitaisiin käyttää paremmin todellisessä ympäristössä, tulee yhteensopivuustestausta muiden toteutusten kanssa tehdä paljon enemmän. SiPit[22] tarjoaa erinomaiset mahdollisuudet kehittää SIP-protokollan vikasietoisuutta ja yhteensopivuutta. Protokolla tarjoaa valmiin kuormitustestiympäristön sekä joukon viallisesti muodostettuja paketteja, joita toteutuksen pitäisi pystyä käsittelemään. Lisäksi järjestetään tapahtumia, joissa omaa toteutusta voi testata muiden toteutuksia vasten. Myös muita toteutettuja protokollia tulisi testata enemmän olemassa olevia toteutusten kanssa.

[22] http://www.cs.columbia.edu/sip/sipit/

5. Yhteenveto

Työn tavoitteena oli kehittää jatkuva-aikaista multimediaa hyödyntävien sovellusten kehitysalusta. Tavoitteen saavuttamiseksi tutustuttiin olemassa oleviin multimediajärjestelmiin, joiden avulla pyrittiin hankkimaan tietoa käytetyistä protokollista ja samalla saavuttamaan mahdollisimman hyvä yhteensopivuus olemassa olevien toteutusten kanssa. Lisäksi selvitettiin mahdollisuus käyttää uudelleen olemassa olevia ohjelmistokomponentteja.

Alalle on muodostunut selkeä joukko de facto -protokollia, joiden käyttäminen on elintärkeätä yhteensopivuuden kannalta. Näin tarvittavien protokollien valinta ei ollut vaikea tehtävä. Ainoastaan signalointiprotokollan valinnassa oli mahdollisuus käyttää useampaa kuin yhtä protokollaa. Alustaan valittiin toteutettavaksi SIP-protokolla, koska se oli yksinkertaisempi, laajennettavampi ja skaalautuvampi.

Toteutun alustan ominaisuudet määriteltiin kahden käyttötapauksen avulla. Ensimmäisessä käyttötapauksessa siirrettiin jatkuva-aikaista videokuvaa palvelimelta mobiiliin päätelaitteeseen. Toisessa käyttötapauksessa siirrettiin jatkuva-aikaista kaksisuuntaista videokuvaa kahden mobiilin päätelaitteen välillä. Käyttötapaukset määrittelivät tarvittavat protokollat sekä antoivat yhdessä asiakkaan vaatimusten kanssa pohjan arkkitehtuurisuunnittelulle.

Alustan arkkitehtuuri jakaantui neljään alijärjestelmään, joista tässä työssä keskityttiin multimediaprotokollat sisältävään alijärjestelmään. Media-alijärjestelmä muodostui itsenäisistä protokollakomponenteista, jotka koostettiin toimivaksi kokonaisuudeksi liitutauluarkkitehtuurin mukaisesti yhdellä hallitsevalla komponentilla. Liitutaulunarkkitehtuurin käyttö osoittatui onnistuneeksi valinnaksi, sillä sen avulla pystyttiin muokkaamaan helposti komponttien välisiä liitoksia ja siirtämään tehtävävastuita komponenttien välillä.

Lähdeluettelo

[1] Packer, R. (1999) Just What Is Multimedia, Anyway? IEEE Multimedia Volume:6 Issue:1, Jan-March 1999, s. 11–13.

[2] The UMTS Third Generation Market(2001) Phase II: Structuring the Service Revenue Oppurtunities. Report 13 from the UMTS Forum, UMTS Forum, 2001, 107 s.

[3] Lammi, J. (2001) Internet Streaming Media Platform. Diplomityö. Tampereen teknillinen korkeakoulu, Tampere, 55 s.

[4] 3GPP TS 23.228 (v5.1.0 2001-06) IP Multimedia (IM) Subsystem - Stage 2. 3rd Generation Partnership Project, Sophia Antipolis, France, 135 s.

[5] 3GPP TS 26.234 (v4.0.0 2001-03) Packet-switched Streaming Service (PSS); Procols and Codecs. 3rd Generation Partnership Project, Sophia Antipolis, France, 32 s.

[6] Schulzrinne, H & Rosenberg, J.(1999) Internet Telephony: Architecture and Protocols an IETF Perspective. Computer Networks and ISDN Systems, vol. 31, nro. 3, s. 237–255.

[7] Schulzrinne,H., Casner, S. & Frederick, R. (1996) RTP: A transport protocol for real-time applications. RFC 1889, Internet Engineering Task Force, 75 s.

[8] Handley, M., Schulzrinne, H., Schooler, E.& Rosenberg ,J. (199??? mikä vuosi?) SIP: Session Initiation Protocol. RFC 2543, Internet Engineering Task Force, 153 s.

[9] Schulzrinne, H., Rao, A. & Lanphier, R. (1998) Real Time Streaming Protocol (RTSP). RFC 2326, Internet Engineering Task Force, 92s.

[10] Branden, R., Zhang, L., Berson ,S., Herzog,S. & Jamin, S. (1997) Resource ReSerVation Protocol (RSVP). RFC 2205, Internet Engineering Task Force, 53 s.

[11] Delgrossi, L. & Berger, L. (1995) Internet Stream Protocol Version 2 (ST2). RFC 1819, Internet Engineering Task Force, 109 s.

[12] Naugle, M. G. (1999) Illustrated TCP/IP. John Wiley & Sons, 499 s.

[13] Karjalainen, J. (2001) IP-puhe palvelunlaatuominaisuuksilla varustetussa Internetissä. Diplomityö. Tampereen Teknillinen korkeakoulu, Tampere, 56 s.

[14] Blake, S., Black, D., M Carlson, M., Davies, E., Wang, Z. & Weiss. W. (1998) An architecture for differentiated services. RFC 2475, Internet Engineering Task Force, 36 s.

[15] Clark, D. D. & Tannenhouse, D. L. (1990) Architectural Considerations for a New Generation of Protocols. In: ACM SIGCOMM´90, September 1990, Philadelphia, Pennsylvania, s. 200–208.

[16] Schulzrinne, H.(1996) RTP Profile for Audio and Video Conferences with Minimal Control. RFC 1890, Internet Engineering Task Force, 18 s.

[17] Braun, T. (1997) Internet protocols for Multimedia Communications Part 2. IEEE Multimedia, Volume: 4, Issue: 4, October-December 1997, s. 74–82.

[18] Johnston, A. (2001) SIP – Understanding the Session Initiation Protocol. Artech House Publishers, 201 s.

[19] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach P. & Berners-Lee, T. (1999) Hypertext Transfer Protocol -- HTTP/1.1. RFC 2616, Internet Engineering Task Force, 176 s.

[20] Postel, J. (1982) Simple Mail Transfer Protocol. RFC 821, Internet Engineering Task Force, 68 s.

[21] Berners-Lee, T., Masinter, L. & McCahill, M. (1999) Uniform Resource Locators (URL). RFC 1738, Internet Engineering Task Force, 25 s.

[22] Postel, J. (1980) User Datagram Protocol. RFC 768, Internet Engineering Task Force, 3 s.

[23] Postel J. (1981) Transmission Control Protocol. RFC 793, Internet Engineering Task Force, 85 s.

[24] SIP: Session Initiation Protocol -- Implementations (20.12.2001) URL:http://www.cs.columbia.edu/%7Ehgs/sip/implementations.html

[25] What is SIP? (20.12.2001) URL: http://www.sipcenter.com/files/whatissip.pdf

[26] Networked Appliances Research (20.12.2001) URL: http://www.argreenhouse.com/iapp/

[27] Handley M., Jacobson V. (1998) SDP: Session Desription Protocol. RFC 2327, Internet Engineering Task Force, 42 s.

[28] Handley, M., Perkins, C., & Whelan, E. (2000) SAP - Session Announcement Protocol. RFC 2974, Internet Engineering Task Force, 18 s.

[29] Dalgic, D. & Fang.H. (1999) Comparision of H.323 and SIP for IP Telephony Signalling. In: Proc. of Photonics East, SPIE, September 1999, Boston, Massachusetts, 17 s.

[30] Verkkomedian perusteet -luentokalvot. (20.12.2001)URL:http://www.tml.hut.fi/Opinnot/Tik-110.250/2001/Luennot/luento9.pdf,Helsingin yliopisto.

[31] CodecCentral (6.11.2001) URL: http://www.icanstream.tv/CodecCentral/

[32] 3GPP TS 22.140 (v4.1.0 2001-3) Multimedia Messaging Service. 3rd Generation Partnership Project, Sophia Antipolis, France, 12 s.

[33] About Darwin Streaming Server (4.12.2001) URL: http://www.publicsource.apple.com/projects/streaming/AboutDarwinStreamingServer.pdf

[34] Apple Public Source License (20.12.2001) URL:http://www.darwin.org/apsl/

[35] GNU General Public License (04.12.2001) URL: http://www.gnu.org/licenses/gpl.txt

[36] Sun Community Source License Principles (04.12.2001) URL: http://www.sun.com/981208/scsl/principles.html

[37] Schulzrinne, H., & Rosenberg , J. (1998) A Comparison of SIP and H.323 for Internet Telephony. In Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), July 1998, Cambridge, England, s 83-86.

[38] H.323 versus SIP: A Comparison (4.12.2001) URL: http://www.packetizer.com/iptel/h323_vs_sip/

[39] Crocker, D. & Overell, P (1997) Augmented BNF for Syntax Specifications: ABNF. RFC 2234, Internet Engineering Task Force, 14 s.

[40] Schulzrinne, H., & Rosenberg J. (1998) Signalling for Internet Telephony. In: IEEE Sixth International Conference on Network Protocols, Austin, Texas, s. 298–307.

[41] Savikko, V-P. (2000) EPOC-sovelluksen rakentaminen", VTT, Espoo, 92 s.

[42] Parkkila A., & Tikkala A. (2001) Muppet Architecture. Sisäinen projekti-dokumentti, 21 s.

[43] Bass, L., Clements, P. & Kazman, R. (1998) Software Architecture in Practice. Addison-Wesley, Reading, Massachusetts. 452 s.

[44] Grahn, H. & Bosch, J. (1998) Some Initial Performance Characteristics of Three Architecturel Styles. In: Proceedings of the First International Workshop on Software and Performance (WOSP '98), Santa Fe, New Mexico, s. 197–98.

[45] Basso, A., Cash, G.L. & Civanlar, M. R., "Real-time MPEG-2 delivery based on RTP: Implementation issues", Signal Processing: Image Communication 15 (1–2), Elsevier Science, s. 165–78.

[46] Synchronized Multimedia Integration Language (SMIL) 1.0 Specification (04.12.2001) URL: http://www.w3.org/TR/REC-smil/

[47] 3GPP TS 24.229 (v0.0.7 2000-11) IP Multimedia Call Control Protocol based on SIP and SDP. 3rd Generation Partnership Project, Sophia Antipolis, France, 22 s.

Julkaisija

Vuorimiehentie 5, PL 2000, 02044 VTT
Puh. (09) 4561
Faksi (09) 456 4374

Julkaisun sarja, numero ja raporttikoodi

VTT Tiedotteita 2130
VTT–TIED–2130

Tekijä(t)

Tikkala, Aki

Nimeke

Jatkuva-aikaisten multimediasovellusten kehitysalusta

Tiivistelmä

Työn tavoitteena oli kehittää jatkuva-aikaista multimediaa hyödyntävien sovellusten kehitysalusta.

Työssä selvitettiin käytetyimpien multimediajärjestelmien ominaisuudet. Analyysin perusteella valittiin protokollat, jotka välttämättä tarvittiin sovelluksiin. Valintatehtävä osoittautui helpoksi, sillä alalle on muodostunut selkeästi joukko de facto -standardeja, joiden noudattaminen oli elintärkeää yhteensopivuuden kannalta. Lisäksi selvitettiin olemassa olevien toteutusten uudelleenkäyttömahdollisuuksia huomioiden alustalle asetetut vaatimukset.

Alustaan toteutettavat protokollat määriteltiin kahden käyttötapauksen avulla. Ensimmäisessä käyttötapauksessa siirrettiin jatkuva-aikaista videokuvaa palvelimelta mobiiliin päätelaitteeseen. Toisessa käyttötapauksessa siirrettiin jatkuva-aikaista kaksisuuntaista videokuvaa kahden mobiilin päätelaitteen välillä.

Luotu alusta jaettiin neljään alijärjestelmään, joista tässä työssä keskityttiin multimediaprotokollat sisältävään media-alijärjestelmään. Media-alijärjestelmä koostui itsenäisistä mediaprotokollakomponenteista, jotka koostettiin toimivaksi kokonaisuudeksi liitutauluarkkitehtuurityylin mukaisesti yhdellä hallitsevalla komponentilla. Näin saavutettiin modulaarinen järjestelmä, johon oli mahdollista liittää pienillä muutoksilla uusia protokollatoteutuksia.

Avainsanat

multimedia platform, RTP, Real-Time-Protocol, SIP, Session Initiation Protocol

Toimintayksikkö

VTT Elektroniikka, Sulautetut ohjelmistot, Kaitoväylä 1, PL 1100, 90571 OULU

ISBN			Projektinumero
951–38–5943–6 (URL: http://www.inf.vtt.fi/pdf/)			E2SU00035
Julkaisuaika	Kieli	Sivuja	Hinta
Maaliskuu 2002	Suomi, engl. tiiv.	55 s.	
Projektin nimi		Toimeksiantaja(t)	
MUPPET		Hantro Products Oy	
Avainnimeke ja ISSN		Myynti:	
VTT Tiedotteita – Research Notes 1455–0865 (URL: http://www.inf.vtt.fi/pdf/)		VTT Tietopalvelu PL 2000, 02044 VTT Puh. (09) 456 4404 Faksi (09) 456 4374	

Published by

Vuorimiehentie 5, P.O.Box 2000, FIN–02044 VTT, Finland
Phone internat. +358 9 4561
Fax +358 9 456 4374

Series title, number and report code of publication

VTT Research Notes 2130
VTT–TIED–2130

Author(s)

Tikkala, Aki

Title

Development platform for streaming multimedia applications

Abstract

The goal of this work was to develop a platform for streaming multimedia applications.

The work included studying the properties of the most used multimedia platforms. They were analysed to decide which ones must be included in the platform in order to achieve proper application functionality. The decision was easy to make because there are a number of de facto standards in this domain and those protocols must be supported in the name of compatibility. The possibility of using existing implementations was also examined.

The protocols implemented into the platform were defined with two use cases. In the first case, a live video was streamed from the server to the mobile terminal. In the second case, a dual-way live video was streamed between two mobile terminals.

The created platform was divided into four subsystems; of these four, this study was focused on the multimedia subsystem. The multimedia subsystem consists of independent protocol components which were assembled into the functional unit by using the repository defined by the blackboard architecture. This resulted in a modular system with a capability to easily add or change protocols.

Keywords

multimedia platform, RTP, Real-Time-Protocol, SIP, Session Initiation Protocol

Activity unit

VTT Electronics, Embedded Software, Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland

ISBN			Project number
951–38–5943–6 (URL: http://www.inf.vtt.fi/pdf/)			E2SU00035
Date	Language	Pages	Price
March 2002	Finnish, engl. abstr.	55 p.	
Name of project		Commissioned by	
MUPPET		Hantro Products Oy	
Series title and ISSN		Sold by	
VTT Tiedotteita – Research Notes 1455–0865 (URL: http://www.inf.vtt.fi/pdf/)		VTT Information Service P.O.Box 2000, FIN–02044 VTT, Finland Phone internat. +358 9 456 4404 Fax +358 9 456 4374	

VTT TIEDOTTEITA – MEDDELANDEN – RESEARCH NOTES

VTT ELEKTRONIIKKA – VTT ELEKTRONIK – VTT ELECTRONICS

1565 Jussila, Salme, Salmela, Olli & Stubb, Henrik. Impedanssispektroskopia elektroniikan -pakkaustekniikassa. Johdepolymeerien ja niiden kontaktirajapintojen karakterisointi. 1994. 30 s.

1593 Rytilä, Hannu. Reverse engineering technology as a tool of embedded software. A solution from PL/M code to structure charts. 1994. 84 p. + app. 16 p.

1614 Isomursu, Pekka, Juuso, Esko, Rauma, Tapio, Haataja, Kari, Kemppainen, Seppo & Myllyneva, Jaakko. Sumea säätö suomalaisessa prosessiteollisuudessa. 1994. 70 s.

1632 Maijanen, Satu. Segmentointi ja asiakastarpeiden analysointi tuotespesifikaation määrittelyssä. 1995. 57 s.

1642 Niemelä, Eila. Uudelleenkäytettävyys koneenohjausohjelmiston suunnittelussa. 1995. 114 s.

1655 Haapanen, Pentti, Korhonen, Jukka & Pulkkinen, Urho. Ydinvoimalaitosten ohjelmoitavien automaatiojärjestelmien tutkimushanke (OHA) 1995–1998. 1995. 23 s.

1746 Berg, Pauli. Toteutuskokemuksia Oppivien ja älykkäiden järjestelmien sovellukset -ohjelman hankkeista. 1996. 28 s. + liitt. 12 s.

1759 Korpipää, Panu. CAD-suunnittelutiedon hyödyntäminen mekatronisen laitteen kunnonvalvonnassa. 1996. 65 s. + liitt. 3 s.

1777 Röning, Juha, Kalaoja, Jarmo, Okkonen, Ari & Kauniskangas, Hannu. Reaaliaikaisten -konenäkösovellusten kehittäminen. 1996. 72 s. + liitt. 40 s.

1816 Pyhäluoto, Timo. Ohjelmistokomponenttien rajapintojen kuvaaminen. 1997. 55 s. + liitt. 22 s.

1825 Heimala, Päivi, Hokkanen, Ari, Keinänen, Kari, Keränen, Kimmo, Tenhunen, Jussi & Lehto, Ari. Mikroanturisysteemien tutkimusohjelma 1994–1996. 1997. 47 s.

1908 Tuominen, Arno. Joustavat ohjelmistoratkaisut tehtäväkriittisessä hajautetussa järjestelmässä. 1998. 74 s.

1911 Holappa, Mikko S. CORBAn soveltaminen joustavan valmistusjärjestelmän perusohjelmistoon. 1998. 95 s.

1913 Salmela, Mika. Testausympäristön konfigurointityökalun käytettävyyden parantaminen. 1998. 56 s.

1914 Korpipää, Tomi. Hajautusalustan suunnittelu reaaliaikasovelluksessa. 1998. 56 s. + liitt. 4 s.

1927 Lumpus, Jarmo. Kenttäväyläverkon automaattinen konfigurointi 1998. 68 s. + liitt. 3 s.

1933 Ihme, Tuomas, Kumara, Pekka, Suihkonen, Keijo, Holsti, Niklas & Paakko, Matti. Developing application frameworks for mission-critical software. Using space applications as an example. 1998. 92 p. + app. 20 p.

1965 Niemelä, Eila. Elektroniikkatuotannon joustavan ohjauksen tietotekninen infrastruktuuri. 1999. 42 s.

1985 Rauhala, Tapani. Javan luokkakirjasto testitapauseditorin toteutuksessa. 1999. 68 s.

2042 Kääriäinen, Jukka, Savolainen, Pekka, Taramaa, Jorma & Leppälä, Kari. Product Data Management (PDM). Design, exchange and integration viewpoints. 2000. 104 p.

2046 Savikko, Vesa-Pekka. EPOC-sovellusten rakentaminen. 2000. 56 s. + liitt. 36 s.

2065 Sihvonen, Markus. A user side framework for Composite Capability / Preference Profile negotiation. 2000. 54 p. + app. 4 p.

2088 Korva, Jari. Adaptiivisten verkkopalvelujen käyttöliittymät. 2001. 71 s. + liitt. 4 s.

2092 Kärki, Matti. Testing of object-oriented software. Utilisation of the UML in testing. 2001. 69 p. + app. 6 p.

2095 Seppänen, Veikko, Helander, Nina, Niemelä, Eila & Komi-Sirviö, Seija. Towards original software component manufacturing. 2001. 105 p.

2114 Sachinopoulou, Anna. Multidimensional Visualization. 2001. 37 p.

2129 Aihkisalo, Tommi. Remote maintenance and development of home automation applications. 2002. 85 p.

2130 Tikkanen, Aki. Jatkuva-aikaisten multimediasovellusten kehitysalusta. 2002. 55 s.

Tiedotteessa käsitellään jatkuva-aikaista multimediaa hyödyntävien sovellusten kehitysalustan kehittämistä. Aluksi selvitettiin käytetyimpien multimediajärjestelmien ominaisuudet. Analyysin perusteella valittiin protokollat, jotka välttämättä tarvittiin sovelluksiin. Valintatehtävä osoittautui helpoksi, sillä alalle on muodostunut selkeästi joukko de facto -standardeja, joiden noudattaminen oli elintärkeää yhteensopivuuden kannalta. Alustaan toteutettavat protokollat määriteltiin kahden käyttötapauksen avulla. Ensimmäisessä käyttötapauksessa siirrettiin jatkuva-aikaista videokuvaa palvelimelta mobiiliin päätelaitteeseen. Toisessa käyttötapauksessa siirrettiin jatkuva-aikaista kaksisuuntaista videokuvaa kahden mobiilin päätelaitteen välillä.

Tätä julkaisua myy
VTT TIETOPALVELU
PL 2000
02044 VTT
Puh. (09) 456 4404
Faksi (09) 456 4374

Denna publikation säljs av
VTT INFORMATIONSTJÄNST
PB 2000
02044 VTT
Tel. (09) 456 4404
Fax (09) 456 4374

This publication is available from
VTT INFORMATION SERVICE
P.O.Box 2000
FIN–02044 VTT, Finland
Phone internat. + 358 9 456 4404
Fax + 358 9 456 4374

ISBN 951–38–5943–6 (URL: http://www.inf.vtt.fi/pdf/)
ISSN 1455–0865 (URL: http://www.inf.vtt.fi/pdf/)