Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan koulutusohjelma

Kandidaatintyö

Arttu Hanska

OPETUSKÄYTTÖÖN SUUNNATUN VIDEONEUVOTTELUPORTAALIN PROTOTYYPIN SUUNNITTELU JA TOTEUTTAMINEN AVOIMILLA TEKNIIKOILLA

Työn tarkastaja: Dosentti Jouni Ikonen

Työn ohjaaja: Dosentti Jouni Ikonen

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan koulutusohjelma

Arttu Hanska

Opetuskäyttöön suunnatun videoneuvotteluportaalin prototyypin suunnittelu ja toteuttaminen avoimilla tekniikoilla

Kandidaatintyö - aihe on hyväksytty 4.6.2012.

28 sivua, 9 kuvaa

Työn tarkastaja: Dosentti Jouni Ikonen

Hakusanat: videoneuvottelu, WebRTC, HTML5, WebSocket,
Keywords: video conferencing, WebRTC, HTML5, WebSocket

Työ tutkii, miten yksinkertaisen opetuskäyttöön suunnatun videoneuvotteluportaalin prototyypin rakentaminen onnistuu avoimilla tekniikoilla. WebRTC mahdollistaa selaimen päällä toimivan videoneuvottelusovelluksen rakentamisen avoimia ohjelmistorajapintoja käyttäen. Prototyypin toteuttamiseen käytetään WebRTC:n lisäksi WebSocket-standardia keskustelun tahojen yhdistämiseen. WebSocket toteutetaan Node.js:ää käyttävällä socket.io-kirjastolla. Sovelluksen avulla on tarkoitus pystyä muodostamaan ääni- ja videoyhteys kahden asiakasohjelman välille. Opetuskäyttöä tukevina ominaisuuksina mukana on myös tekstipohjainen keskustelu ja mahdollisuus opetuskalvojen selaamiseen. Lopputuloksena on laajennettavissa oleva avoin videoneuvottelusovelluksen prototyyppi.

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology Management
Degree Program in Information Technology

Arttu Hanska

Planning and implementation of teaching focused video conferencing portal prototype with open techniques

Bachelor's Thesis – subject accepted June 4th, 2012

28 pages, 9 figures

Examiner: Docent Jouni Ikonen

Keywords: video conferencing, WebRTC, HTML5, WebSocket

This thesis studies how it is possible to build a simple video conferencing portal prototype aimed for teaching with open techniques. WebRTC makes it possible to build a video conferencing application working on browser with free application programming interfaces. Prototype is created with WebRTC and WebSocket standard to connect conversation endpoints. WebSocket is implemented with Node.js socket.io-library. The goal is to be able to form audio and video connection between two clients. As features to support teaching there is a text based chat and possibility to browse study slides. Outcome is a prototype of open and extensible video conferencing application.

SISÄLLYSLUETTELO

SYMBOLI- JA LYHENNELUETTELO

BOSH	Bidirectional-streams Over Synchronous HTTP
DTLS	Datagram Transport Layer Security
GIPS	Global IP Solutions
HTML5	HyperText Markup Language 5
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICE	Interactive Connectivity Establishment
IETF	Internet Engineering Task Force
IP	Internet Protocol
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
NAT	Network address translation
RoR	Ruby on Rails
RTCWeb	Real-Time Communication Web
RTMP	Real Time Messaging Protocol
SCTP	Stream Control Transmission Protocol
SIP	Session Initiation Protocol
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TURN	Traversal Using Relays around NAT
UDP	User Datagram Protocol
WebRTC	Web Real-Time Communication
WHATWG	Web Hypertext Application Technology Working Group
W3C	World Wide Web Consortium
XMPP	Extensible Messaging and Presence Protocol

1 JOHDANTO

Vuorovaikutteinen media on yleistynyt huomattavasti internetin myötä. Sille on löydetty useita erilaisia käyttötarkoituksia esimerkiksi viihdekäytössä ja viestinnässä. Useat näistä sovelluksista ovat internetselaimella toimivia websovelluksia, mikä tekee niiden käyttöönotosta helppoa ja usein laitteistoriippumatonta. Selainten ominaisuuksien kasvaessa näitä mahdollisuuksia voidaan soveltaa myös laajemmin esimerkiksi opetuskäyttöön.

1.1 Tausta

Nykypäivänä monet palvelut ovat muuttuneet pilvipalveluita käyttäviksi websovelluksiksi, joita voidaan käyttää pelkällä internetselaimella. Sovelluksiin on helppo päästä käsiksi mistä tahansa. Uusin teknologiatulokas on HTML5 (HyperText Markup Language 5), joka toi mukanaan paljon lisäominaisuuksia, kuten sisäänrakennetun videoiden suoratoiston, 2D-renderöinnin ja geopaikannuksen. Videoneuvottelusovellukset ovat pitkään ollut kaupallisten toteutusten varassa. WebRTC-standardi (Web Real-Time Communication) on kuitenkin tuomassa helpon ja avoimen tavan toteuttaa videoneuvotteluyhteys selaimessa ilman erillisiä lisäohjelmia.

1.2 Tavoitteet ja rajaukset

Työn tavoitteena on suunnitella ja toteuttaa opetuskäyttöön suunnatun videoneuvotteluportaalin prototyyppi, joka mahdollistaa sujuvan etäopetuksen opettajan ja oppilaan välillä. Itse portaalin tarkoituksena on kasata yhteen opettajat sekä oppilaat ja tarjoaa opetusta edistäviä palveluita, kuten tilastoja, käyttäjille. Prototyyppi keskittyy kuitenkin opetusistunnon vaatimien ominaisuuksien toteuttamiseen. Tavoitteena on, että sovellusta voidaan käyttää normaalilla internet-selaimella ja ainoat käyttäjältä vaadittavat investoinnit ovat webkamera ja mikrofoni. Tärkeänä osana on myös sovelluksen avoimuus.

Kaikkien käytettyjen tekniikoiden tulee olla avoimia ja myös lopullinen sovellus tulee olemaan avoin jatkokehitykselle ja tutkimukselle.

Työssä käytetään olemassa olevia avoimia tekniikoita, ja tarkoituksena ei ole kehittää omia tekniikoita. Sovellusta kehittäessä tärkeitä mittareita ovat skaalautuvuus, helppo toteutus ja resurssienkäyttö. Työ käsittelee lähinnä avoimia tekniikoita, mutta sivuaa myös suljettuja ratkaisuja. Vaikka tavoitteena ei ole rakentaa täysimittaista toimivaa portaalia, niin prototyypin on tarkoitus olla mahdollisimman helposti laajennettavissa.

1.3 Työn rakenne

Luvussa 2 käsitellään videoneuvottelun historiaa ja nykyistä tilannetta. Käydään läpi yleisiä termejä ja tekniikoita sekä määritellään tutkimusongelma ja käytettävät ratkaisumenetelmät. Luvussa 3 analysoidaan tekniikoita ja ratkaisuvaihtoehtoja. Luvussa 4 kerrotaan työn vaiheet, toteutustavat sekä tehdään yhteenveto toteutuksesta. Lopuksi esitetään työn pohjalta tehdyt johtopäätökset ja arviot tulevaisuudesta. Luku 5 sisältää tiivistetyt johtopäätökset.

2 VIDEONEUVOTTELUTEKNIIKAT

Videoneuvottelu tarkoittaa kahden tai useamman henkilön välistä reaaliaikaista kuva- ja ääniyhteyttä. Termiä on aiemmin käytetty erityisesti tähän tarkoitukseen rakennetuista laitteistoista ja ohjelmistoista. Nopeiden internetyhteyksien ja uusien tekniikoiden myötä videopuhelut ovat kuitenkin levinneet kaikkien saataville ja termin raja on hämärtynyt. Nykypäivänä videoneuvottelulla voidaan tarkoittaa mitä tahansa kahden tai useamman henkilön välistä reaaliaikaista videoyhteyttä.

2.1 Taustatiedot

Videoneuvotteluprosessi ja sen komponentit voidaan jakaa tasoihin. Päällimmäisenä on käyttöliittymä, jonka avulla voidaan hallita muita tasoja. Käyttöliittymän alla toimii yhteyksienhallinta, joka varaa tarvittavat resurssit, kuten portit, varmistaa oikeiden tahojen paikalla olemisen ja aloittaa videoneuvottelun. Yhteyksienhallinta voidaan laskea tämän työn tapauksessa osaksi käyttöliittymää. Signaalitaso yhdistää päätepisteet ja muodostaa yhteyden niiden välille. Mediataso huolehtii äänen ja videon kaappaamisesta, yhdistämisestä sekä toistosta. [1, s21-22]

Useiden tasojen takia videoneuvottelusovelluksen kehittäminen on monivaiheinen prosessi. Mediatasolla huomioitavia asioita ovat kuvalle ja äänelle käytettävät pakkausformaatit sekä niiden toteutus koodekkeja käyttäen. Signaalitasolla yhteyttä muodostaessa ongelmina ovat esimerkiksi osoitteenmuuntajien ja palomuurien läpäisy, tiedon suojaus ja kaistankäytön hallinta. Lisäksi on kehitettävä neuvottelun laatua parantavia ominaisuuksia, kuten kaiunpoisto ja kohinan hallinta.

Yritysten ja muiden isojen tahojen erikoistuneita videoneuvottelujärjestelmiä käytetään edelleen, mutta palvelut ovat levinneet myös kuluttajille. Yleisin selainympäristössä käytetty ratkaisu on lisäosana asennettava Adobe Flash. Toinen kuluttajien helposti

käytettävä ratkaisu on Microsoftin omistama Skype, joka tarjoaa erillisen asiakasohjelman videoneuvotteluille. Nämä ovat kuitenkin suljettuja kaupallisia ratkaisuja, joten palvelua tarvitsevien on turvauduttava niihin tai kehitettävä omia useita eri osaamisalueita vaativia sovelluksia.

Videoneuvottelupalvelun voidaan ajatella koostuvan kolmesta komponentista: käyttöliittymä, mediamoottori ja signaaliprotokolla. Signaalitasolla on olemassa kaupallisia ratkaisuja, kuten Skype ja Adoben Flashin käyttämä RTMP (Real Time Messaging Protocol), mutta tarjolla on myös avoimia protokollia, kuten H.323, SIP (Session Initiation Protocol) [2] ja XMPP (Extensible Messaging and Presence Protocol) [3]. Mediatasolla avoimia, laadukkaita ja helposti kehitettäviä ratkaisuja on kuitenkin huomattavasti vähemmän. Videoneuvottelupalvelun rakentaminen on yleensä vaatinut, että palvelua tarvitseva tahon on joko rakennettava oma mediamoottorinsa tai ostettava sellainen toiselta yritykseltä. Uuden mediamoottorin rakentaminen on raskas ja kallis prosessi, joten palvelun ostaminen on usein helpompi ratkaisu.

Google osti vuonna 2012 yrityksen nimeltä Global IP Solutions (GIPS), joka tarjosi muille yrityksille palveluna valmiita mediamoottoriratkaisuja. Google julkaisi GIPS:n sovelluksia avoimella lisenssillä ja tästä syntyi nykyisin WebRTC:nä tunnettu aloite rakentaa avoin ja helposti sovellettava mediamoottori, joka toimii ilman lisäosia tavallisessa internet-selaimessa. Kehityksessä ovat mukana myös W3C (World Wide Web Consortium), WHATWG (Web Hypertext Application Technology Working Group) ja IETF (Internet Engineering Task Force).

2.2 Käsitteiden ja termien määrittely

Mediamoottori tarkoittaa videoneuvottelusovelluksen osaa, joka huolehtii median kaappauksesta, pakkaamisesta ja purkamisesta. Joissakin tapauksissa mediamoottori käsittelee myös median lähettämistä. Signaaliprotokollalla tarkoitetaan signaalintason totuttamiseen käytettyä mallia. Yleisesti käytettyjä avoimia protokollia ovat IETF:n SIP ja XMPP sekä ITU-T:n (ITU Telecommunication Standardization Sector) H.323.

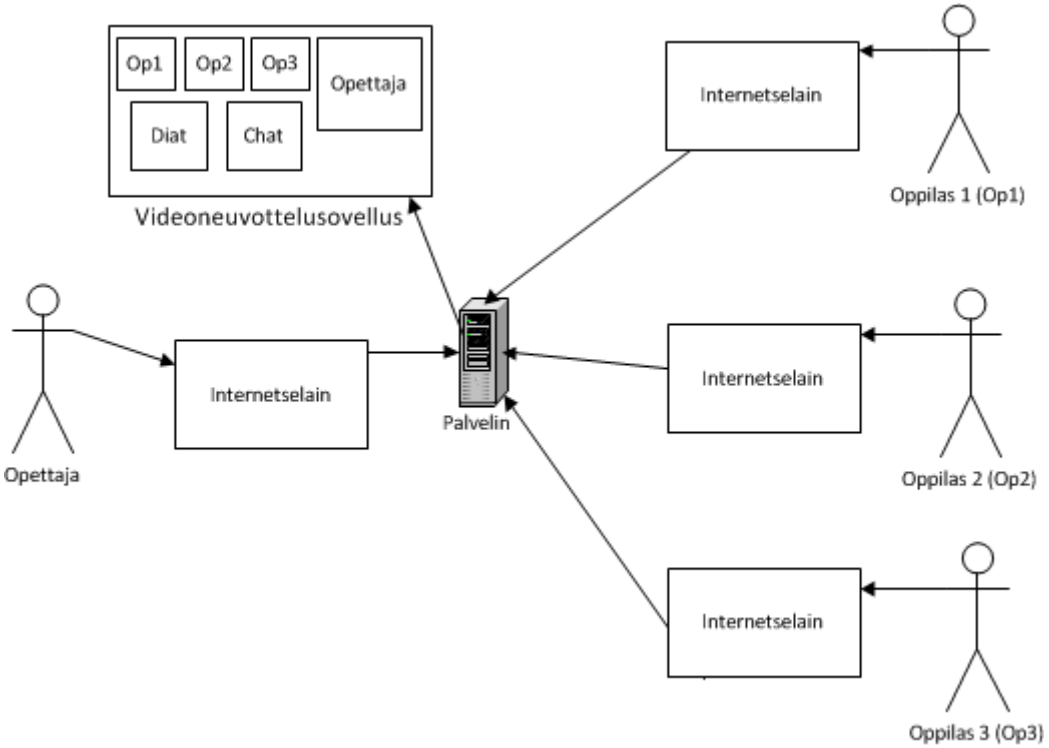
WebRTC on selaimessa toimiva avoin mediamoottori, joka mahdollistaa paikallisen kuvan ja äänen kaappaamisen sekä tietovirran lähettämisen ja vastaanottamisen. Kehitystyö on kesken, mutta useiden merkittävien selainten testiversiot tukevat jo WebRTC:n eri ohjelmointirajapintoja. Mukava ovat Google, Mozilla, Opera ja Microsoft. Ainostaan Apple ja ei ole tehnyt aloitetta tekniikan tukemiseksi. Tuettuja äänikoodekkeja ovat G.711, G.722, iLBC ja iSAC. Tuettu videokoodekki on VP8.[4] RTCWeb (Real-Time Communication Web) on IETF:n kehityksessä oleva avoin protokollamääritelmä WebRTC:lle. Yhdessä RTCWeb ja WebRTC mahdollistavat selainten välisen videoneuvotteluyhteyden muodostamisen sekä yksinkertaisen videoneuvottelusovellusten kehittämisen. [5]

Osoitteenmuuntaja sallii useiden isäntäkoneiden jakavan yhden tai useamman julkisen IP-osoitteen (Internet Protocol) sisäverkon IP-osoitteita käyttäen. Osoitteenmuunto aiheuttaa ongelmia erityisesti vertaisyhteyksissä. [6] Pakkausformaatti on määrittely digitaalisen videon ja äänen pakkaamiselle. Pakkaaminen ja purkaminen hoidetaan joko ohjelmallisesti tai rautatasolla sopivaa koodekkia käyttäen. Formaatit voivat olla joko häviöllisiä tai häviöttömiä. Videoneuvottelussa pyritään mahdollisimman hyvään pakkaustehoon, koska video on toistettava ilman huomattavaa puskurointia.

2.3 Tutkimusongelma

Työ tutkii kuinka opetuskäyttöön suunniteltu webportaali voidaan toteuttaa WebRTC:tä käyttäen. Ratkaistavina ongelmina ovat sovelluksen käyttöliittymän kehittäminen, mediamoottorin käyttöönotto sekä signaalitason ratkaisun valinta ja käyttöönotto. Eräs keskeinen ongelma on sovelluksen toimiminen mahdollisimman monella selaimella. Tutkitaan myös, onko videoneuvotteluyhteys mahdollista toteuttaa kuvan 1 mukaisesti useamman tahon välille. WebRTC on vielä kehitysvaiheessa, joten työssä arvioidaan myös sen asemaa tulevaisuudessa

Ominaisuuksia toteuttaessa huomioidaan opetuskäyttöön soveltuvuus. Oletuksena keskustelussa ovat mukana opettaja sekä yksi tai useampi oppilas. Jos oppilaita on useampi, niin opettajan videoruudun tulee erottua selvästi muista. Opettaja voi näyttää opetusmateriaalia oman kameransa kautta, mutta käytännöllisempää on erillinen ominaisuus luentokalvojen selaamiseen. Kalvojen selaamisen tulee toimia siten, että ainoastaan opettajalla on oikeus niiden hallintaan. Mahdollisten yhteys- tai laiteongelmien takia mukana on myös tekstipohjainen keskustelu, jonka avulla oppilas voi osallistua opetukseen ilman webkameraa tai mikkiä. Tutkitaan myös, että onko ilman kuvaa pelkällä mikin välityksellä mahdollista. Kuvassa 1 on esitetty esimerkkiratkaisu opettajalle ja kolmelle oppilaalle.



Kuva 1: Videoneuvottelusovelluksen esimerkkitoteutus

2.4 Tutkimusongelman ratkaisumenetelmät

Tutkimusongelman ratkaisu etenee videoneuvottelusovelluksen osa-alue kerrallaan: mediamoottori, signaaliprotokolla ja käyttöliittymä. Yleisenä etenemismallina käytetään asiaa käsittelevien tutkimusten ja muun lähdemateriaalin lukemista, teoriaosuuden

koostamista ja tämän jälkeen käytännön toteutuksen laatimista. Lopullinen yhteenveto laaditaan näiden kaikkien vaiheiden pohjalta.

WebRTC kattaa portaalin mediatason toteutuksen. Tekniikan nuoren iän ja keskeneräisyyden takia kattavia tutkimuksia ei vielä ole. Kehittämis- ja tutkimusmateriaalina käytetään lähinnä W3C:n ja IETF:n standardivedoksia [4, 5] sekä muiden tahojen ohjelmointiesimerkkejä. Signaalitason ratkaisuja on useita, ja niistä löytyvä tutkimusmateriaali on laajempaa. Työn pääpaino on kuitenkin WebRTC:ssä ja portaalin prototyypin toteutuksessa, joten signaaliratkaisun tutkimiseen ei käytetä yhtä paljon aikaa. Signaaliratkaisu päätetään helpon ylläpidon, tehokkuuden ja skaalautuvuuden mukaan.

Kyseessä on websovellus, joten käyttöliittymä toimii internetselaimella. Portaalin prototyypin toteutuksessa käytetään Ruby on Railsia (RoR), koska työn tekijällä on eniten kokemusta sen käyttämisestä. WebRTC käyttää HTML5:n ohjelmointirajapintoja ja videontoisto-ominaisuuksia, joten sen käyttäminen on välttämätöntä.

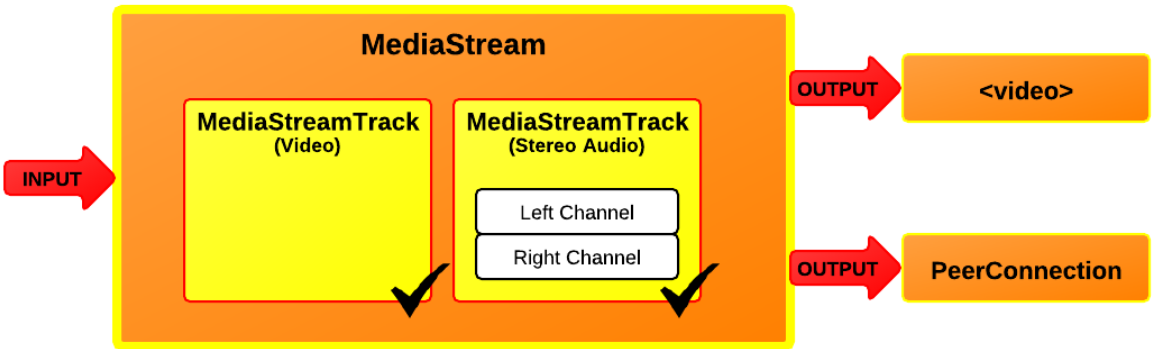
3 RATKAISUVAIHTOEHTOJEN ANALYSIONTI

Luku esittelee tekniikoita ja ratkaisuvaihtoehtoja prototyypin toteuttamiseksi. Tarvittavat osat ovat mediamoottori, signaaliprotokolla sekä tekniikka jolla media- ja signaalitaso kommunikoivat. Käyttöliittymä käsitellään neljännessä kappaleessa. Työ keskittyy WebRTC:n ympärille, joten huomattava osa ajasta on käytetty sen tutkimiseen.

3.1 Mediamoottori: WebRTC

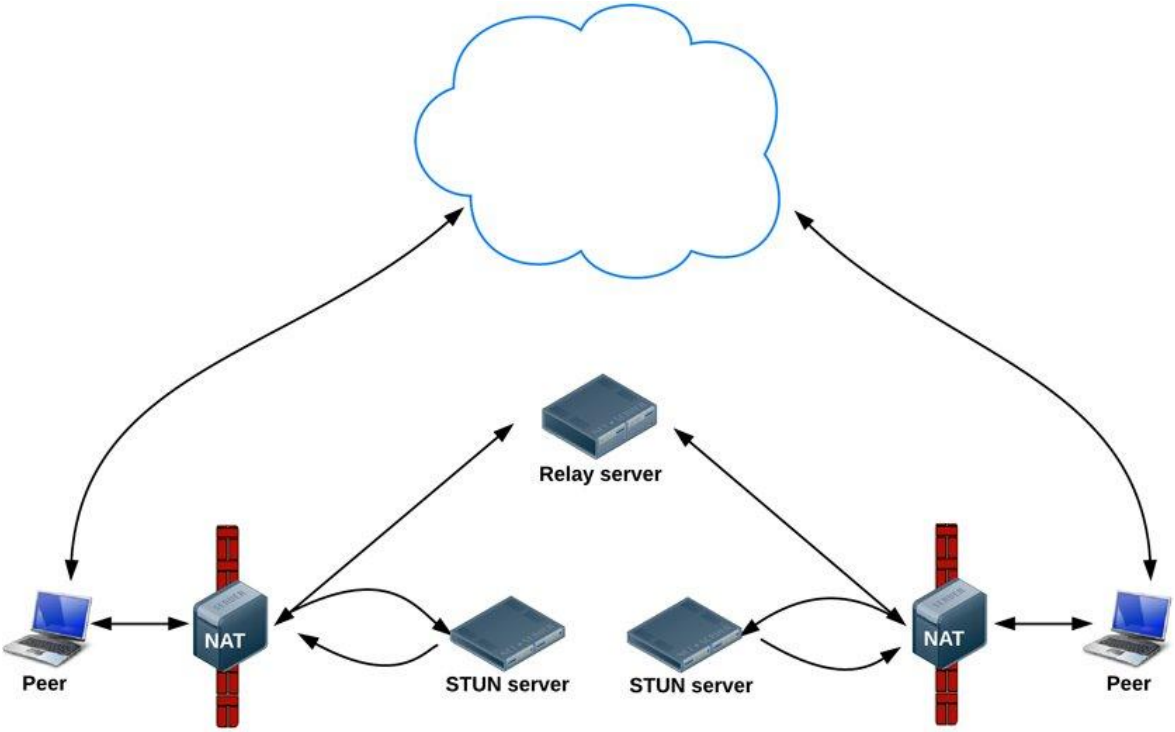
WebRTC koostuu tällä hetkellä pääasiassa neljästä ohjelmointirajapinnasta. MediaStream-rajapinnan avulla voidaan kaapata tietovirtoja, kuten ääntä ja kuvaa. Paikallisia tietovirtoja varten on erillinen GetUserMedia-rajapinta[7]. Tietovirtojen lähettämistä varten on vertaisyhteyksiä käyttävä RTCPeerConnection-rajapinta. Lisäksi on DataChannel-rajapinta, joka mahdollistaa kaksisuuntaisen reaaliaikaisen tiedonsiirron geneeriselle tiedolle.

MediaStreamin avulla tietovirrat voidaan kaapata joko paikallisesta laitteesta GetUserMedia-rajapinnan avulla tai etäkohteelta RTCPeerConnect-rajapinnan avulla. Vastaavasti kaapattu tietovirta voidaan joko toistaa paikallisesti tai lähettää RTCPeerConnection-rajapintaa käyttäen etäkohteelle, kuten kuvassa 2 esitetään. Yksittäinen tietovirta voi sisältää useita ääni- ja kuvalähteitä.



Kuva 2. MediaStream-rajapinta tietovirtojen kaappaamiseen [8]

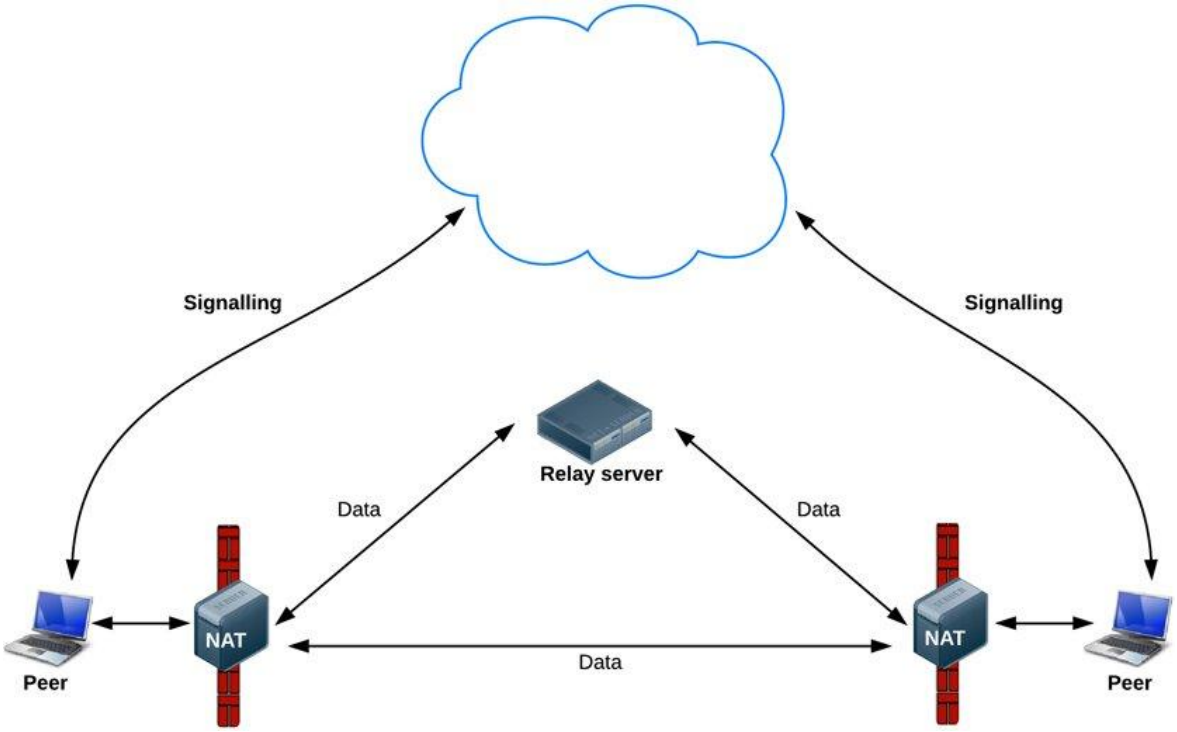
RTCPeerConnection-rajapinta mahdollistaa kahden käyttäjän suoran kommunikaation selaimelta selaimelle. WebRTC tukee tällä hetkellä ainoastaan kahden käyttäjän vertaisyhteyksiä. Yhteyden muodostamiseen käytetään määrittelemätöntä signaalinratkaisua, mutta rajapinta tarjoaa kuitenkin ratkaisun osoitteenmuuntajien läpäisyyn. Yhteyttä muodostaessa luodaan ICE (Interactive Connectivity Establishment) agentti, joka pyrkii etsimään parhaan reitin päätepisteeseen [9]. Jos asiakasohjelma on osoitteenmuuntajan takana, apuna käytetään kuvan 3 esittämällä tavalla STUN-palvelinta (Session Traversal Utilities for NAT), joka pyrkii etsimään yhteydelle julkisen osoitteen sekä portin.



Kuva 3. Nopeimman avoimen yhteysreitin etsintä [10]

ICE agentti pyrkii aluksi etsimään suoraa yhteyttä UDP:n (User Datagram Protocol) avulla. Jos UDP epäonnistuu, yritetään TCP:tä (Transmission Control Protocol); ensin suojaamaton HTTP (Hypertext Transfer Protocol) ja sitten suojattu HTTPS (Hypertext Transfer Protocol Secure). Jos yhteyden suora muodostaminen ei onnistu, vaihdetaan TURN-välityspalvelimeen (Traversal Using Relays around NAT), joka uudelleen ohjaa kaiken liikenteen. Tämä lisää viivettä ja vaatii huomattavasti enemmän kaistaa. Kuvassa 4

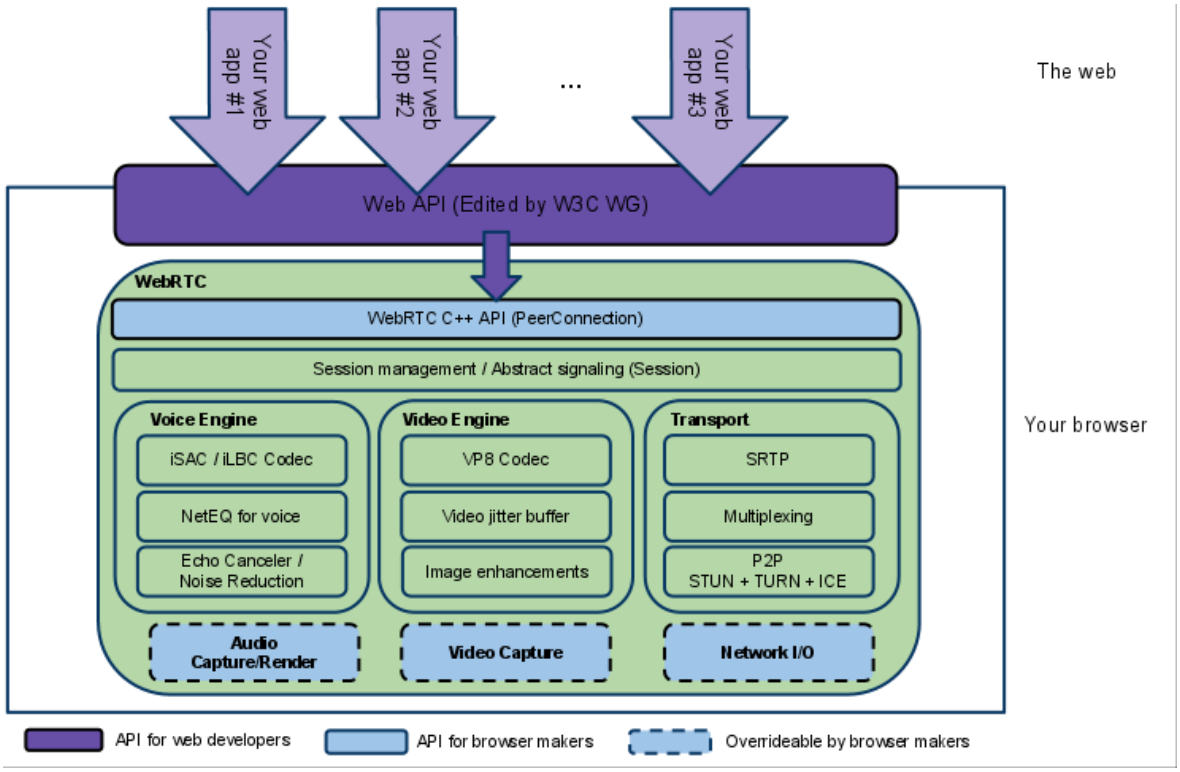
nähdään esimerkki WebRTC:n yhteysreiteistä; vertaisyhteys käyttäjien välillä tai uudelleen ohjattua yhteys TURN-palvelimen välityksellä.



Kuva 4. WebRTC:n yhteysmallit.[11]

Sovelluksen yhteydessä halutaan usein lähettää myös muuta kuin ääntä ja videokuvaa. Tätä varten on DataChannel-rajapinta, joka mahdollistaa geneerisen tiedon lähettämisen. Toimintatapa muistuttaa hyvin paljon MediaStream-rajapintaa, ja rajapintojen on tarkoitus toimia yhdessä. DataChannel mahdollistaa esimerkiksi reaaliaikaisten keskustelusovellusten tai pelien kehittämisen selaimeen WebRTC:tä käyttäen. Rajapinta käyttää SCTP-yhteyttä (Stream Control Transmission Protocol), joka suojataan DTLS:ää (Datagram Transport Layer Security) käyttäen. [12]

WebRTC:n ohjelmointirajapinnat toteutetaan kuvan 5 mukaisesti selainvalmistajien toimesta, joten niiden käyttöönotto ei vaadi erillistä asiakasohjelmaa tai lisäosaa. Tällöin poistuu tarve ylläpitää ja julkaista versiot eri käyttöjärjestelmille ja selaimille. Tämä helpottaa myös loppukäyttäjää, koska erillisten sovellusten asentaminen lisää virheiden määrää. Sovellusta voidaan myös käyttää mobiililaitteella suoraan laitteen selaimella.



Kuva 5. WebRTC:n arkkitehtuurikaavio. [13]

3.2 Reaaliaikaisen websovelluksen toteuttaminen

HTTP-yhteys on tilaton, mikä tarkoittaa, että yhteyksiä hallitaan toisistaan riippumattomina. Useat nopeasti päivittyvät websovellukset, kuten keskusteluohjelmat, vaativat kuitenkin mahdollisimman reaaliaikaista tietoa. Yleinen tapa asiakasohjelman ja palvelimen väliseen tiedon päivittämiseen on kyselyt. Asiakasohjelma kysyy palvelimelta tietyin aikavälein päivitettyä tietoa ja palvelin joko ilmoittaa, että tietoa ei ole päivitetty tai lähettää uuden tiedon. Nopeasti päivittyvissä sovelluksissa ongelmaksi tulee, että uutta tietoa ei lähetetä heti vaan vasta asiakasohjelman sitä pyytäessä. Tämä lähestymistapa toimii sovelluksissa, jotka eivät vaadi jatkuvaa päivitystä, mutta reaaliaikaisissa vuorovaikutteisissa sovelluksissa se voi saada käytön tuntumaan tahmealta.

Toinen versio kyselystä on pitkä kysely. Asiakasohjelma suorittaa yhden kyselyn, palvelin jättää yhteyden avoimeksi ja ilmoittaa, kun tieto päivittyy. Tällöin palvelin voi lähettää päivitetyn tiedon heti sen saadessaan eli yhteys on kaksisuuntaisesti reaaliaikainen. Tämä

kuitenkin syö enemmän palvelimen resursseja, koska jokaista asiakasohjelmaa kohden tarvitaan useita avoimia yhteyksiä; yksi tiedon lähettämiseen ja uusi jokaiselle vastaanotettavalle yhteydelle. Jokainen yhteys sisältää HTTP:n protokollatiedot, mikä aiheuttaa turhaa tiedonsiirtoa. Lisäksi HTTP-yhteys ei sovellu hyvin sovelluksiin, joissa vaaditaan pieniä vasteaikoja. Pitkä kysely ei ole varsinainen standardi vaan sovellettu ratkaisu, joka emuloi kaksisuuntaista liikennettä. Olemassa olevia ratkaisuja ovat esimerkiksi Comet ja BOSH (Bidirectional-streams Over Synchronous HTTP).

Uudempi tapa toteuttaa kaksisuuntainen reaaliaikainen tiedonsiirtoyhteys on WebSocket-protokolla, joka muodostaa yhteyden yksittäistä TCP-yhteyttä käyttäen. Erona pitkään kyselyyn tarvitaan ainoastaan yksi yhteys ja palvelin ei käsittele yhteyttä nettisivuna vaan socket-sovelluksena, jolloin sekä palvelin että asiakasohjelma voivat lähettää toisilleen tietoa halutessaan. Tietoa voidaan myös jakaa palvelinprosessin sisällä ilman, että sitä tarvitsee erikseen tallentaa tietokantaan. Tieto voidaan tallentaa keskusmuistiin tai välittää suoraan toiseen socket-yhteyteen. Kaistankäyttö ja vasteajat ovat myös pienempiä. WebSocket käyttää HTTP:n tavoin porttia 80 normaalille yhteydelle ja suojatulle TSL-yhteydelle (Transport Layer Security) porttia 443. [14]

3.3 Signaaliprotokolla

WebRTC tarjoaa yleisen signaalitasosta riippumattoman mediaratkaisun, joka sisältää ratkaisun osoitteenmuuntajien läpäisyyn. Eräs keskeinen ongelma on signaaliprotokollan valitseminen. Yleisesti käytettyjä avoimia protokollia ovat H.323, SIP ja XMPP. Koska kehitettävä sovellus on websovellus, niin tärkeä kriteeri on webyhteensopivuus. Käytännössä tämä tarkoittaa webympäristöön soveltuvaa JavaScript-toteutusta tai vastaavaa.

H.323 on vanha standardi, jota käytetään isoissa yritysratkaisuissa. Sille ei ole olemassa yhtään websovellukselle sopivaa toteutusta, joten sen käyttäminen ei ole mahdollista. Standardia ei luultavasti tulla näkemään selainympäristössä.

SIP tarvitsee TCP-tuen, mikä on mahdollista WebSocket-protokollan avulla. SIP-toteutuksen avulla sovellus voidaan yhdistää vanhoihin palveluihin olemassa olevia palvelimia käyttäen. SIP:lle on olemassa joitakin JavaScript-toteutuksia, mutta niiden kehitysaste on vielä varhainen. Standardi alkaa olla jo vanha, eikä sitä ole ensisijaisesti suunniteltu selainympäristöön.

XMPP:lle on olemassa useita JavaScript-toteutuksia ja standardi on suunniteltu selainyhteensopivaksi. Sen avulla voidaan luoda myös muita reaaliaikaisia sovelluksia ja se voidaan liittää jo olemassa oleviin sovelluksiin, kuten pikaviestimiin. Reaaliaikaisen yhteyden luomiseen käytetään yleensä pitkää kyselyä, mutta myös WebSocket-kirjastot ovat yleistymässä.

Erikoistuneiden ratkaisujen lisäksi signaalitaso voidaan toteuttaa WebSocketin ansiosta myös yksinkertaisesti HTTP:tä ja JavaScriptia käyttäen. Toteutus ja käyttöönotto on yksinkertaista, mutta tarjolla ei ole valmiiden ratkaisujen monipuolisia ominaisuuksia. Etuna kuitenkin on, että signaaliratkaisulle riittää pelkkä HTTP-palvelin.

4 TYÖN TOTEUTUS

Kappaleen aluksi esitellään valitut tekniikat ja kerrotaan työn toteuttamisesta. Tulosten yhteenvedossa kootaan kaikki tulokset yhteen, kuvataan prototyypin lopullista toimintaa ja pohditaan prototyypin laajentamismahdollisuuksia. Tulosten tulkinta paneutuu WebRTC:n ja webvideoneuvottelun asemaan ja tulevaisuuteen.

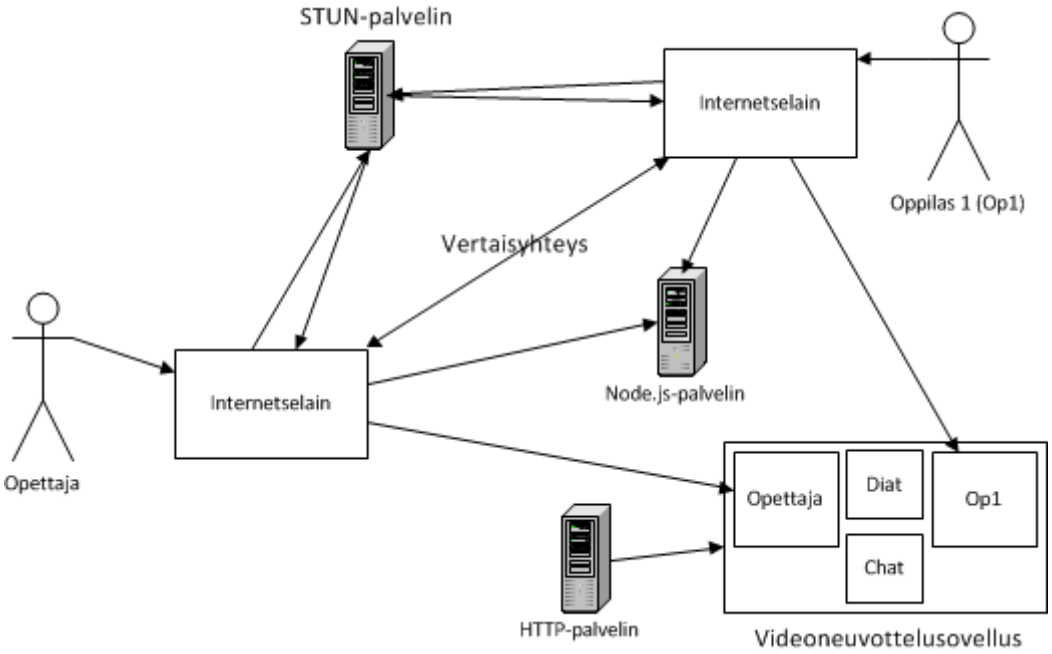
4.1 Työn kuvaus ja toteuttaminen

Lopulliseksi signaaliratkaisuksi päädyttiin käyttämään pelkkää HTTP:tä ja WebSockettia. WebSocket on edullinen tapa muodostaa reaaliaikainen kaksisuuntainen yhteys palvelimen ja asiakasohjelman välille. Myös XMPP:lle on olemassa WebSocket-kirjastoja, mutta helpon toteutuksen ansiosta prototyypin signaaliratkaisun toteutukseen valitaan HTTP ja WebSocket.

Työssä käytetään WebRTC:n MediaStream-rajapintaa, joka käyttää GetUserMedia-rajapintaa äänen ja videokuvan kaappaamiseen paikallisilta laitteilta. Tämän jälkeen keskustelun osapuolet yhdistetään WebSocket-palvelinta käyttäen. WebSocket-palvelimen toteutuksessa käytetään socket.io-kirjastoa, joka tarjoaa Node.js:llä toteutetun palvelin- ja asiakasohjelman. [15] Socket.io tarjoaa myös taaksepäin yhteensopivuuden muihin tekniikoihin, kuten Adobe Flashiin, mutta WebRTC:tä tukevat selaimet tukevat myös pääasiassa WebSockettia.

Kun osapuolet ovat löytäneet toisensa käytetään RTCPeerConnection-rajapintaa suoran vertaisyhteyden luomiseen, jota pitkin ääni ja video lähetetään suoraan käyttäjien välillä. MediaStream vastaanottaa etäkohteelta tulevan tietovirran, joka voidaan toistaa HTML5 videona. Osoitteenmuuntajien läpäisyyn käytössä on Googlen tarjoama STUN-palvelin ulkoisen IP:n ja portin löytämiseksi. Erillistä TURN-palvelinta koko yhteyden

välittämiseksi ei käytetä. Kuvassa 6 esitellään tutkimustyön pohjalta laadittu kaavio sovelluksesta.



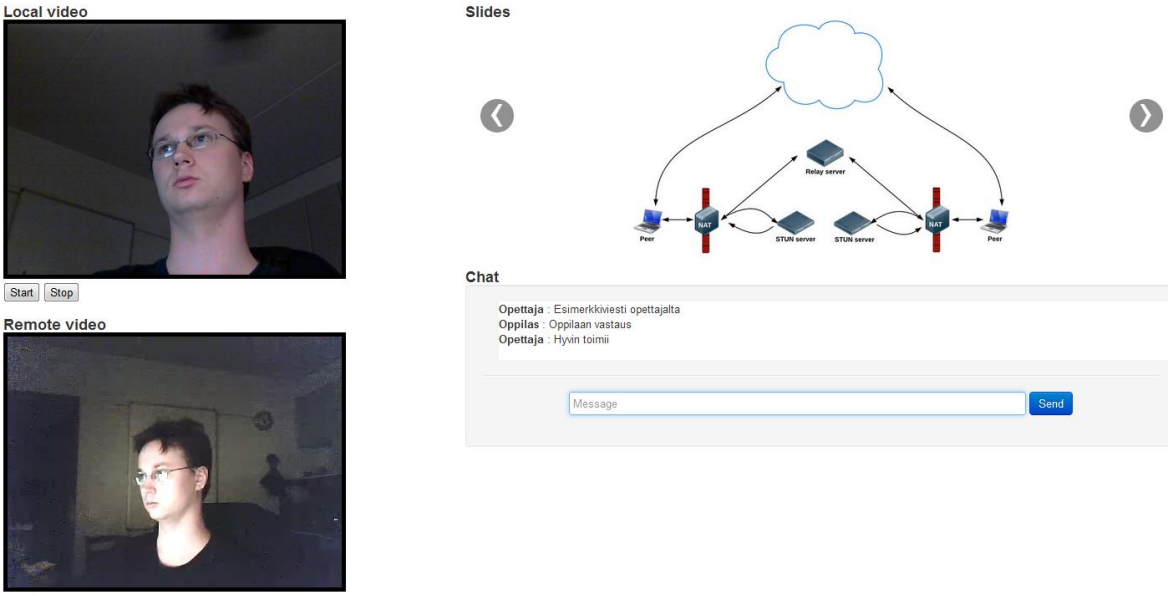
Kuva 6. Sovelluksen toteutus suunnittelun pohjalta

Prototyypin käyttöliittymä keskittyy videoneuvottelun ja opetuskäyttöön suunnattujen ominaisuuksien kehittämiseen. Käytössä ovat Ruby on Rails 3.2.1 ja HTML5 sekä muotoiluun bootstrap-sass 2.0.0. Sovelluksen ajamiseen käytetään yksinkertaista WEBrick HTTP-palvelinta. Sovellus toimii Chrome-selaimen versiolla 21 tai uudemmalla. Chromen lisäksi Opera tarjoaa tuen tarvittaville ohjelmointirajapinnoille, mutta syntaksit eroavat toisistaan.

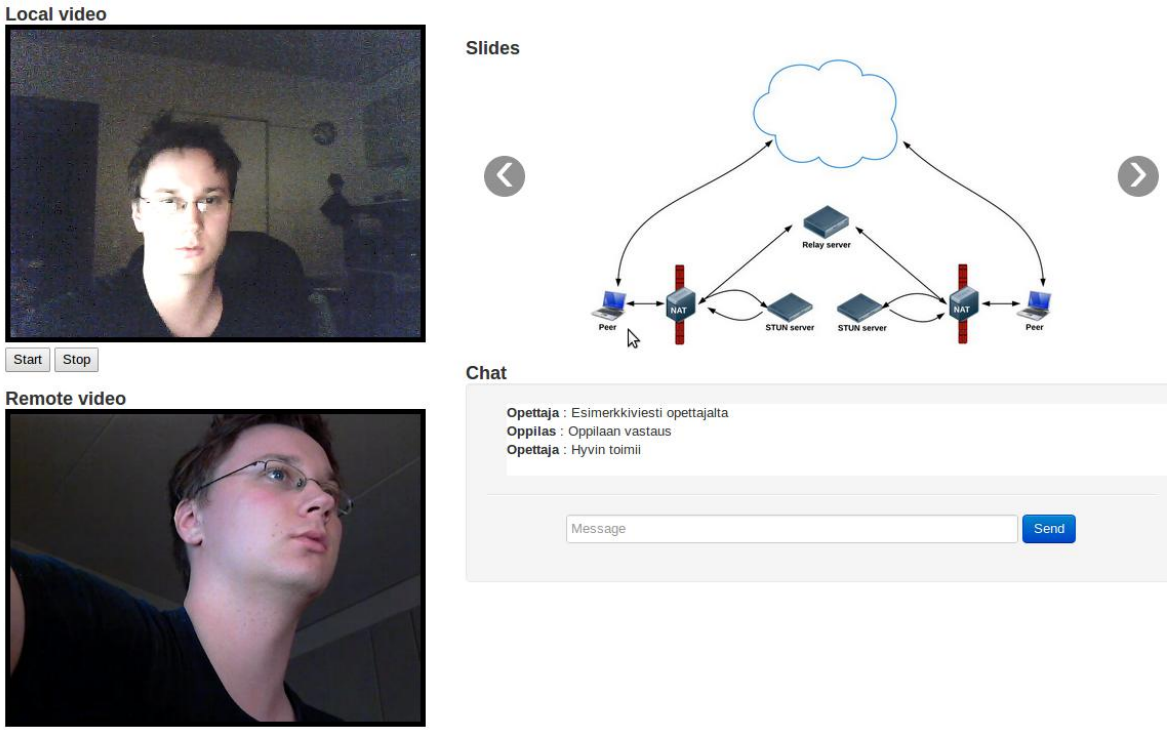
4.2 Työn tulokset ja yhteenveto

Videoneuvottelun käyttöliittymässä on kuvien 7 ja 8 mukaisesti elementit paikalliselle sekä vastaanotettavalle videolle, luentokalvoille ja tekstipohjaiselle keskustelulle. Paikallisen videoelementin alla on painikkeet neuvottelun aloittamiseen ja lopettamiseen.

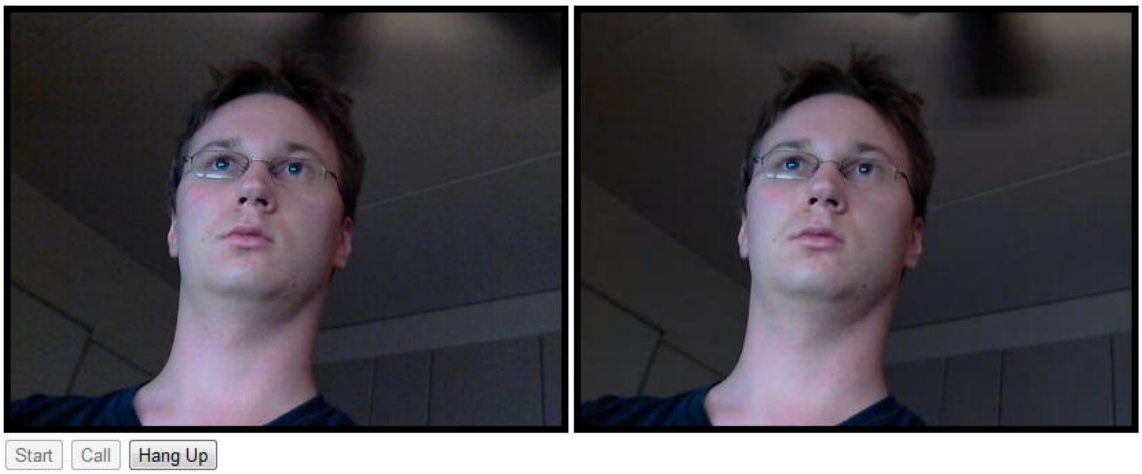
Sovelluksessa on myös kuvan 9 mukainen esimerkki ilman signaalitasoa toteutetusta RTCPeerConnection-yhteydestä, joka kaappaa videon ja toistaa sen suoraan samalle sivulle.



Kuva 7. Sovelluksen käyttöliittymä



Kuva 8. Esimerkki-istunnon toinen osapuoli



Kuva 9. Paikallisesti toistettava videoyhteys ilman signaaliratkaisua

Opetuskäyttöön suunnattuja ominaisuuksia ovat reaaliaikainen tekstipohjainen keskustelu ja vaihdettavat luentokalvot. Niiden toteutuksessa käytetään WebSocket-yhteyttä RTCPeerConnectionin ja DataChannel-rajapinnan sijaan, koska DataChannel on työtä kirjoittaessa vielä merkittävästi keskeneräinen. WebSocket soveltuu myös hyvin tällaiseen käyttöön nopeutensa ansiosta.

Puuttuvia ominaisuuksia valmiiseen portaaliin verrattuna ovat videoneuvottelun keston tallentaminen, tilastojen laskeminen, käyttäjienhallinta, kirjautuminen ja ajanvarausjärjestelmä. Signaaliratkaisun korvaaminen XMPP:llä WebSockettia käyttäen on myös yksi tutkimisen arvoinen lähestymistapa. Prototyypissä ei ole myöskään tietokantaa tietojen tallentamiseen. TURN-välityspalvelimen käyttöönotto voi myös olla tarpeellista, jos käyttäjät ovat vaikeasti läpäistävien osoitteenmuuntimien tai palomuurien takana. Tämä kuitenkin vaatii huomattavasti enemmän resursseja palvelimelta.

Eräs jatkotutkittava ongelma on WebRTC-sovelluksen toteuttaminen useammalle samanaikaisesti samaan keskusteluun osallistuvalle taholle. Opetuskäytössä tämä voi tarkoittaa opettajaa ja useampaa oppilasta. Koska WebRTC tukee tällä hetkellä ainoastaan vertaisyhteyksiä, niin ratkaisu vaatisi todennäköisesti erillistä palvelinta, johon kaikki keskustelun tahot ovat yhteydessä. Avoin kysymys on, kuinka sovellus toteutetaan siedettävillä kaistavaatimuksilla?

4.3 Tulosten tulkinta

Yksinkertaisen videoneuvottelusovelluksen toteuttaminen avoimilla tekniikoilla on mahdollista ja verrattain helppoa WebRTC:n, HTML5:n ja WebSocketin avulla. WebRTC tarjoaa ohjelmointirajapinnat videokuvan kaappaamiseen ja vertaisyhteyttä pitkin lähettämiseen. Keskustelun tahot löytävät toisensa WebSocket-palvelimen avulla, minkä jälkeen tieto liikkuu vertaisyhteyttä käyttäen ilman palvelimen suurta rasitusta. Osoitteenmuuntajan läpäisyä varten käytössä on Googlen tarjoama STUN-palvelin julkisen IP-osoitteen ja portin selvittämiseksi.

WebRTC tuo webkehitykseen kauan kaivatun lisäkomponentin, mutta sen selvä heikkous on keskeneräisyys. Tällä hetkellä tekniikka vaikuttaa lupaavalta, mutta kestää kauan ennen kuin se on kaikkien käyttäjien saatavilla. Standardointiprosessi on myös vielä kesken ja täytyy vain toivoa, että tässä vaiheessa ei synny hajaannuttavia kiistoja sen lopullisen version toteutuksesta. Isoista selainvalmistajista Apple ei ole myöskään osoittanut kiinnostustaan WebRTC:tä kohtaan, mikä johtuu todennäköisesti yrityksen FaceTime-sovelluksesta. Jos tekniikka kuitenkin onnistuu lyömään itsensä läpi, niin luultavasti myös Apple ottaa sen käyttöön Safari-selaimessaan.

WebRTC:n tietoturva on myös keskeinen asia, koska kaikilla tuetuilla selaimilla on pääsy käyttäjän webkameraan ja mikrofoniin. Käytännössä tätä voisi ajatella selaimen mukana tulevana vakoiluohjelmana, jonka avulla haitallisen sivun täytyy vain ohittaa lupa käyttää laitetta. Ongelmalle ei ole vielä yksimielistä ratkaisua, mutta asiaan suhtaudutaan vakavasti.

Vanhat suljetut videoneuvottelusovellukset eivät luultavasti tule katoamaan kovin nopeasti vaikka WebRTC yleistyisi. Skype on jo rakentamassa omaa WebRTC-tukea, jolla voi yhdistää uudet käyttäjät olemassa olevaan palvelinverkkoon. WebRTC:n pakkausformaatteihin ei myöskään kuulu tällä hetkellä H.264, mille useat laitteet tarjoavat rautatason purkua. Käytännössä tämä tarkoittaa, että esimerkiksi matkapuhelimissa WebRTC:n käyttämän VP8-videon purkaminen käyttää pelkkää puhelimen prosessoria, mikä voi näkyä hitaampana purkunopeutena sekä akun kulutuksena. Uuden pakkausformaatin lisääminen avoimeen tekniikkaan on mahdollista, mutta jos lopullinen

standardi ei tue sitä, sen käyttöönotto jää sovelluksen kehittäjälle ja WebRTC-selaimet eivät oletusarvoisesti tue sitä.

5 JOHTOPÄÄTÖKSET

Tutkimuksessa rakennettiin selaimen päällä toimiva videoneuvottelusovellus WebRTC:tä, HTML5:ttä sekä WebSockettia käyttäen. Avoin WebRTC antaa webohjelmoijille helpon työkalun kehittää selaimessa toimivia videoneuvottelusovelluksia. Videot voidaan toistaa HTML5:ttä käyttäen ja WebSocket toimii signaaliratkaisuna, jonka avulla keskustelun osapuolet yhdistetään.

WebRTC on vielä kehitysvaiheessa, ja kuluu aikaa ennen kuin selaimet tukevat sitä ja sen kaikkia ominaisuuksia. Myös standardivedos tulee luultavasti muuttumaan kehitystyön jatkuessa. WebSocket on yleistyvä tekniikka, joka luultavasti syrjäyttää aiempia pitkään kyselyyn luottaneita ratkaisuja suoritustehon ja paremman tiedon käsittelyn ansiosta.

Videoneuvottelusovelluksen opetuskäyttöä tukevina ominaisuuksina toteutettiin tekstipohjainen keskustelu ja mahdollisuus opetuskalvojen selaamiseen. Nämä ovat esimerkkejä reaaliaikaisista sovelluksista, joita voidaan rakentaa videoneuvottelun yhteyteen oppimisen tueksi. Erityisesti opetuskäytössä tarpeellinen useamman henkilön välinen videoneuvottelu ei ole vielä mahdollista, koska WebRTC tukee tällä hetkellä ainoastaan vertaisyhteyksiä. Tähän voi tulla muutos WebRTC:n standardin mukana tai erillisenä signaaliratkaisuna, joka yhdistää kaikki keskustelun osapuolet.

LÄHTEET

1. Scott Firestone, Thiya Ramalingam, Steve Fry, Voice and Video Conferencing Fundamentals, Cisco Press, USA, 2007

2. Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, SIP: Session Initiation Protocol, Network Working Group, Proposed Standard, 2002, URL: http://tools.ietf.org/html/rfc3261 (13.8.2012)

3. Peter Saint-Andre, Cisco, Extensible Messaging and Presence Protocol (XMPP): Core, IEFT, Proposed Standard, 2011, URL: http://ebook.tools.ietf.org/html/rfc6120 (13.8.2012)

4. Adam Bergkvist, Daniel C. Burnett, Cullen Jenning, Anant, Anant Narayanan, WebRTC 1.0: Real-time Communication Between Browsers, W3C Editor's Draft 21 July 2012, URL: http://dev.w3.org/2011/webrtc/editor/webrtc.html (13.8.2012)

5. Harald T. Alvestrand, Overview: Real Time Protocols for Brower-based Applications draft-ietf-rtcweb-overview-04, Google, Network Working Group Internet-Draft, 2012, URL: http://tools.ietf.org/html/draft-ietf-rtcweb-overview-04 (13.8.2012)

6. Veera Andersson, Network Address Translator Traversal forthe Peer-to-Peer Session Initiation Protocol on Mobile Phones, Aalto University, Master's Thesis, 2010, URL: https://aaltodoc.aalto.fi/bitstream/handle/123456789/3223/urn100205.pdf (13.8.2012)

7. Daniel C. Burnett, Anant Narayanan, Media Capture and Streams, W3C Editor's Draft 13 August 2012, URL: http://dev.w3.org/2011/webrtc/editor/getusermedia.html (13.8.2012)

8. Daniel C. Burnett, Anant Narayanan, 2012, URL: http://dev.w3.org/2011/webrtc/editor/images/media-stream.png (13.8.2012)

9. Jonathan Rosenberg, Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols, IETF Proposed Standard, 2010, URL: http://tools.ietf.org/html/rfc5245 (13.8.2012)

10. Sam Dutton, HTML5 Rocks Tutorials, 2012, URL: http://www.html5rocks.com/en/tutorials/webrtc/basics/stun.png (13.8.2012)

11. Sam Dutton, HTML5 Rocks Tutorials, 2012, URL: http://www.html5rocks.com/en/tutorials/webrtc/basics/dataPathways.png (13.8.2012)

12. Randell Jesup, Salvatore Loreto, Michael Tuexen, WebRTC Data Channel Protocol draft-jesup-rtcweb-data-protocol-00, RTCWeb Working Group Internet-Draft, 2012, URL: http://tools.ietf.org/html/draft-jesup-rtcweb-data-protocol-00 (13.8.2012)

13. Google, WebRTC, Architecture, 2012, URL: http://www.webrtc.org/_/rsrc/1317202919504/reference/WebRTCpublicdiagramforwebsite%20%282%29.png (13.8.2012)

14. Ian Fette, Alexey Melnikov, The WebSocket Protocol, IETF Proposed Protocol, 2011, URL: http://tools.ietf.org/html/rfc6455 (13.8.2012)

15. Guillermo Rauch, LearnBoost, Socket.IO, 2012, URL: https://github.com/LearnBoost/socket.io (22.8.2012)

LIITE 1. Prototyypin GitHub-sivu

https://github.com/Archinowsk/Realtimestudy