OULUN SEUDUN
AMMATTIKORKEAKOULU

Juha Ahvenlampi

VOIP-TUKI ANDROID-KÄYTTÖJÄRJESTELMÄSSÄ

VOIP-TUKI ANDROID-KÄYTTÖJÄRJESTELMÄSSÄ

Juha Ahvenlampi
Opinnäytetyö
Kevät 2013
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Tekijä(t): Juha Ahvenlampi
Opinnäytetyön nimi: VoIP-tuki Android-käyttöjärjestelmässä
Työn ohjaaja(t): Eino Niemi
Työn valmistumislukukausi ja -vuosi: Kevät 2013 Sivumäärä: 35

Opinnäytetyön tilaajana toimi Neusoft Mobile Solutions Oy. Neusoft Mobile Solutions Oy:ltä löytyy GSMA:n akkreditoima RCS Neusoft Silta -sovellus, jota oli tarkoitus laajentaa IP-pohjaisella VoIP-puhelu mahdollisuudella.

Työn tarkoitus oli tutkia miten VoIP on tuettuna Android-käyttöjärjestelmässä ja miten Neusoft Silta -Android-sovellusta voidaan laajentaa tukemaan IP-pohjaisia VoIP-puheluita. Tutkimuksen ohessa tehtiin pelkistetty versio VoIP-sovelluksesta, jonka avulla käyttäjät voivat tehdä VoIP-puheluita Neusoft Silta -sovelluksesta ja jota voidaan myöhemmin jatkokehittää.

Projektissa saatiin luotua Neusoft Silta -sovellukseen integroitu VoIP-sovellus. VoIP-sovellus käyttää audiohallintaan Androidin tarjoamia ohjelmistokirjastoja ja sovelluksen käyttöliittymä on yhteneväinen muiden Neusoft Silta -sovelluksen käyttöliittymien kanssa. VoIP-puhelun luonnissa, sekä audiosiirrossa käytettiin RCS-standardin mukaista signalointi- sekä audiosiirtoprotokollaa. Työssä keskityttiin myös tuleviin jatkokehitysmahdollisuuksiin, joihin kuuluvat äänenlaatuun sekä käyttöliittymään liittyvät parannukset.

Asiasanat: Android, VoIP, RCS, SIP, RTP

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Juha Ahvenlampi
Title of thesis: VoIP Support in Android Operation System
Supervisor: Eino Niemi
Term and year of completion: Spring 2013 Number of pages: 35

This bachelor's thesis was commissioned by Neusoft Mobile Solutions Ltd. Neusoft Mobile Solutions Ltd. has a GSMA accredited RCS Neusoft Silta application which was meant to be expanded with an IP-based VoIP call support.

The aim of this thesis was to explore how VoIP is supported in Android based operation systems and in which way the Neusoft Silta Android application can be expanded to support IP-based VoIP calls. During this thesis a simplified version of a VoIP application was made, it allowed users to make VoIP calls from Neusoft Silta application and it later can be further developed.

The result of the project was an integrated VoIP client within the Neusoft Silta application. The VoIP client uses Android program libraries for audio handling and the client's user interface is consistent with the Neusoft Silta application's user interfaces. For initiation of a VoIP call and for audio data transfer, RCS-standard signalling and audio transfer protocols were used. During the project different development possibilities, which contain improvements to sound quality and VoIP user interface, were also assessed.

Keywords: Android, VoIP, RCS, SIP, RTP

SISÄLLYS

1 JOHDANTO

Opinnäytetyöni aihe syntyi Neusoft Mobile Solutions Oy:n tarpeesta laajentaa olemassa olevaa RCS-sovellusta kattamaan myös IP-pohjaiset äänipuhelut. Työn tilaajana toimivalla Neusoft Mobile Solutions Oy:llä on GSMA:n akkreditoima RCS Neusoft Silta -sovellus, jota on tarkoitus laajentaa tukemaan myös RCS:n versio 5.1 ominaisuuksia, joihin myös IP-pohjaiset äänipuhelut kuuluvat.

Neusoft Silta -sovellus toteuttaa RCS-dokumentaation mukaisia IP-pohjaisia kommunikaatiopalveluita. Näistä keskeisimpiä ovat chat, tiedostonsiirto sekä videonjako GSM-puhelun aikana. Sovelluksen avulla RCS-käyttäjät voivat laajentaa yhteydenpitoa IP-pohjaisilla kommunikaatiopalveluilla perinteisten tekstiviestin (SMS) sekä multimediaviestin (MMS) lähetyksen rinnalla. Neusoft Silta -sovellus integroituu eri mobiilialustojen paikallisiin sovelluksiin, jolloin IP-pohjaiset kommunikaatiopalvelut saadaan tuotua lähemmäksi käyttäjää.

Tässä opinnäytetyössä tutkittiin, miten VoIP-puheluiden tekeminen on tuettuna Android-käyttöjärjestelmässä ja miten Neusoft Silta -sovellusta voidaan laajentaa VoIP-puhelutuella. Tutkimuksen ohella oli myös tarkoitus toteuttaa pelkistetty versio VoIP-sovelluksesta, jolla on mahdollista tehdä VoIP-puheluita Neusoft Silta -sovelluksesta.

2 VAATIMUSMÄÄRITTELY

Työn aloitusvaiheessa toteutettiin vaatimusmäärittely, joka määritti VoIP-puhelussa käytettävät protokollat sekä vähimmäismäärittelyt audiolaadulle ja VoIP-sovelluksen käyttöliittymälle.

2.1 Protokollavaatimukset

VoIP-yhteyden luonnissa pitää käyttää SIP-protokollaa, koska SIP-protokollaa käytetään myös muissa Neusoft Silta -sovelluksen IP-palveluissa. SIP-protokollan tulee hallinnoida puhelun muodostusta ja sen lopettamista. SIP-signaloinnin yhteydessä käytettävä SDP-protokolla tulee hoitaa neuvottelu vastapäiden välillä, mitä audiokoodekkia tullaan puhelun aikana käyttämään. Audiotiedonsiirtoon puhelun aikana tulee käyttää RTP-protokollaa. Vastapään VoIP-kyvykkyys tulee määritellä SIP OPTIONS -viestien avulla. Sama menetelmä on käytössä selvitettäessä myös muiden IP-kommunikaatiopalveluiden kyvykkyyttä.

2.2 Käyttöliittymävaatimukset

2.2.1 Yleiset käyttöliittymävaatimukset

VoIP-puhelun käyttöliittymän ja puhelunäkymän tulee seurata Neusoft Silta -sovelluksessa käytettävää tapaa asemoida toimintapainikkeita sekä integroitua osaksi Neusoft Silta -sovellusta.

2.2.2 Puhelun aloitus

VoIP-puhelun aloituksen tulee olla mahdollista Neusoft Silta -sovelluksesta. Vastapään VoIP-kyvykkyys tulee selvittää, ennen kuin VoIP-puhelun voi aloittaa. Jos vastapää ei tue VoIP-puhelua, tulee tarjota GSM-puhelu vaihtoehto.

2.2.3 Sisääntuleva puhelu

Sisääntulevan VoIP-puhelun aikana pitää soittaa soittoääni, jota käytetään myös sisääntulevan GSM-puhelun aikana. VoIP-puhelunäkymä tulee näyttää käyttäjälle sisääntulevan puhelun ajan. Näkymästä käyttäjällä on mahdollisuus hyväksyä tai hylätä sisääntuleva VoIP-puhelu. Puhelimen näppäinlukon ollessa päällä, sisääntulevan VoIP-puhelun aikana, se tulee avata ja VoIP-puhelunäkymä tulee näyttää käyttäjälle.

2.2.4 Puhelunäkymä

VoIP-puhelunäkymässä tulee näyttää vastapään kontaktinimi tai numero, jos kontaktitietoa ei saada selvitettyä. VoIP-puhelunäkymästä tulee löytyä lopetuspainike VoIP-puhelun lopettamiseen.

2.3 Audiovaatimukset

2.3.1 Äänentoistovaatimukset

Äänentoisto tulee suorittaa Android-käyttöjärjestelmästä löytyviä ohjelmistokirjastoja hyödyntämällä. Ääni tulee toistaa äänipuheluille tarkoitettua kaiutinta käyttämällä. Käyttäjällä tulee olla mahdollisuus säätää puhelun aikaista äänenvoimakkuutta käyttämällä Android-puhelimesta löytyviä äänenvoimakkuuspainikkeita.

2.3.2 Äänenlaatuvaatimukset

VoIP-sovelluksen tulee tukea vähintään yhtä audiokoodekkia, joka pakkaa audiosignaalin ennen lähetystä ja purkaa sen vastapäässä ennen äänentoistoa. Audiokoodekilla pakattu audiosignaali tulee toimittaa vastapäälle RTP-protokollaa hyödyntämällä.

3 TOIMINTAYMPÄRISTÖ

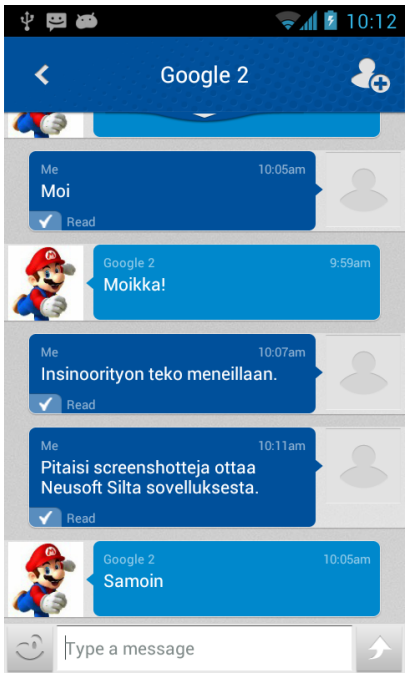
3.1 Neusoft Silta -sovellus

Neusoft Silta on GSMA:n akkreditoima RCS-sovellus. RCS (Rich Communication Services) on suurimpien mobiilioperaattoreiden yhteinen pyrkimys luoda yhteensopivaa IP-pohjaista kommunikointistandardia. RCS tuo käyttäjille IP-pohjaiset viestien lähetys- sekä tiedostonsiirtomahdollisuudet, jotka parantavat kommunikointia perinteisten tekstiviestin (SMS) sekä multimediaviestin (MMS) lähetyksen rinnalla. Muita keskeisiä RCS-palveluita ovat videon sekä kuvan jakaminen GSM-puhelun aikana. (Rich Communication Services 2013.)

Neusoft Silta -sovellus on saatavilla eri mobiilialustoille ja tarjoaa käyttäjälle RCS-standardin mukaisia IP-pohjaisia kommunikaatiopalveluita. Neusoft Silta -sovellus pyrkii hyödyntämään eri mobiilialustojen vahvuuksia sekä integroituu mobiilialustojen paikallisiin sovelluksiin, jolloin IP-pohjaiset kommunikaatiopalvelut saadaan tuotua lähemmäksi käyttäjää. (Neusoft Silta 2013.)

3.1.1 Chat

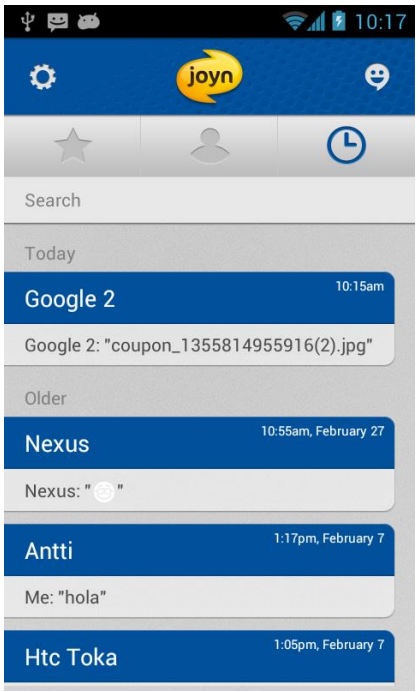
Neusoft Silta -sovellus tarjoaa RCS-dokumentaation mukaisen IP-yhteyttä käyttävän chat palvelun (kuva 1). Neusoft Silta -sovelluksesta voidaan valita helposti käyttäjä, jonka kanssa halutaan aloittaa chat-keskustelu. Sisääntulevat chat-viestit indikoidaan Androidin notifikaatio-ohjelmistorajapinnalla. Chat-viestin saapuessa Android puhelimen notifikaationäkymään lisätään chat-viestinotifikaatio, jota painamalla käyttäjä pääsee suoraan chat-keskustelunäkymään ja voi aloittaa chat-keskustelun vastapään kanssa.



KUVA 1. Neusoft Silta -sovelluksen chat-keskustelunäkymä.

3.1.2 Chat-historia

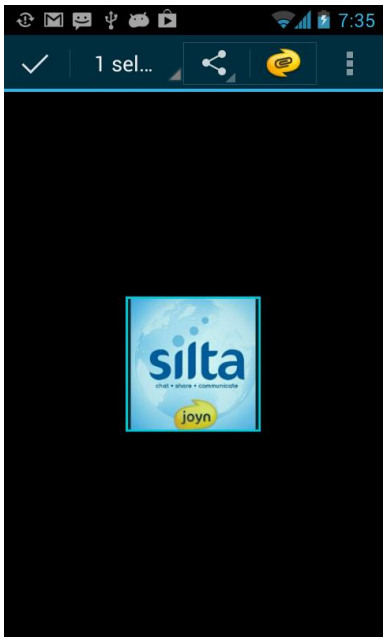
Neusoft Silta -sovellus tallentaa tietokantaan käydyt chat-keskustelut ja käyttäjällä on helppoa jatkaa vanhaa chat-keskustelua chat-historianäkymästä. Historianäkymässä olevat chat-keskustelut ryhmitellään päivämäärän mukaan ja viimeisimmät keskustelut näytetään ensimmäisenä. (Kuva 2.)



KUVA 2. Neusoft Silta -sovelluksen chat-historianäkymä.

3.1.3 Tiedostonsiirto

Neusoft Silta -sovellus tarjoaa RCS-dokumentaation mukaisen IP-yhteyttä käyttävän tiedostonsiirtopalvelun. RCS-käyttäjät voivat jakaa tiedostoja Neusoft Silta -sovelluksen avulla, eikä tuetuilla tiedostotyypeillä ole rajoitteita (kuva 4). Neusoft Silta -sovellus integroituu Android-puhelimen tiedostonkäsittelyohjelmistoihin ja tiedostonsiirto onnistuu suoraan esimerkiksi Androidin Gallery sovelluksesta (kuva 3).

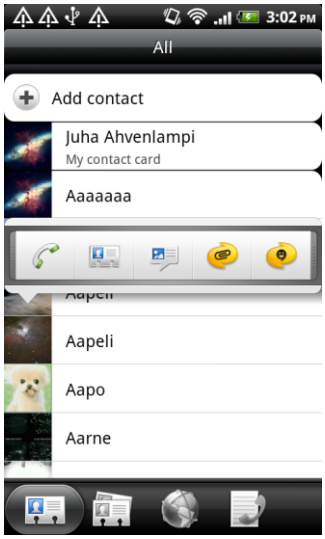


KUVA 3. Neusoft Silta -tiedostonsiirtopainike Androidin Gallery sovelluksessa.

KUVA 4. Vastaanotettu kuvatiedosto Neusoft Silta -sovelluksen chat-keskustelunäkymässä.

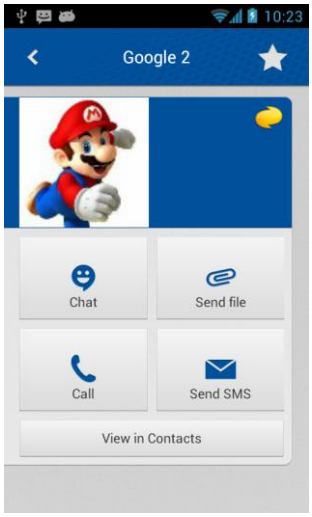
3.1.4 Puhelinluettelo

Neusoft Silta -sovellus integroituu Androidin puhelinluetteloon, jonka ansiosta chat-keskustelun aloitus sekä tiedostonsiirto onnistuvat helposti olemassa olevasta puhelinluettelosta. (Kuva 5.)



KUVA 5. Androidin puhelinluettelo, josta RCS-kyvykkään kontaktin kommunikointimetodeista voidaan valita myös chat- sekä tiedostonsiirtovaihtoehto.

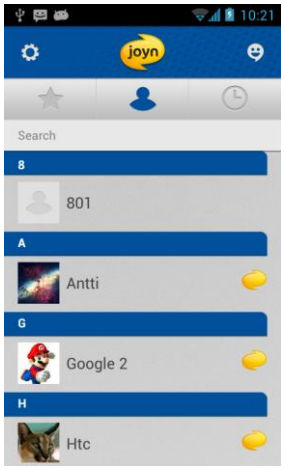
Myös Neusoft Silta -sovelluksesta löytyy puhelinluettelo, josta kontaktin valitsemalla avataan kontaktikortti. Kontaktikortin kautta chat-keskustelun aloitus sekä tiedostonsiirto onnistuvat helposti. Kontaktikortilta voidaan myös aloittaa puhelu, lähettää tekstiviesti sekä muokata kontaktitietoa. (Kuva 6.)



KUVA 6. Neusoft Silta -sovelluksen kontaktikorttinäkymä.

3.1.5 RCS-kyvykkyyden tiedostaminen

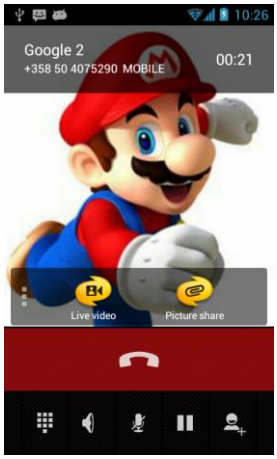
RCS-käyttäjien tulee rekisteröityä RCS-palveluun. Palveluun rekisteröityjen käyttäjien välinen RCS-kyvykkyyden tiedostaminen hoidetaan SIP OPTIONS -viestien avulla. Myös Neusoft Silta -sovellus käyttää SIP OPTIONS -viestejä ja RCS-kyvykkäät käyttäjät ovat helposti löydettävissä Neusoft Silta -sovelluksen puhelinluettelosta (kuva 7).



KUVA 7. Neusoft Silta -sovelluksen puhelinluettelonäkymä. Keltainen ikoni nimen vieressä ilmaisee käyttäjän RCS-kyvykkyyden.

3.1.6 Kuvan ja videon jakaminen GSM-puhelun aikana

Neusoft Silta -sovellus integroituu Android-käyttöjärjestelmän GSM-puhelunäkymään. GSM-puhelun aikana käyttäjällä on mahdollisuus aloittaa kuvan tai videon jakaminen, joka toteutetaan IP-yhteyttä käyttämällä. (Kuva 8.)



KUVA 8. Keltaiset RCS-toimintapainikkeet GSM-puhelunäkymässä kuvan ja videon jakamiseen.

3.2 Orange RCS -engine

Neusoft Silta –Android-sovellus käyttää Orange Labsin kehittämää RCS-engineä. Orangen RCS -engine tarjoaa tuen SIP-protokollaa käyttäville IP-pohjaisille kommunikaatiopalveluille. Tällä hetkellä Orange enginen tukemat IP-pohjaiset kommunikaatiopalvelut ovat chat, tiedostonsiirto, sekä kuvan ja videon jakaminen GSM-puhelun aikana. Orange Labsin RCS -engine on suunniteltu siten, että uusien SIP-protokollaa käyttävien IP-kommunikaatiopalveluiden lisääminen onnistuu siihen helposti. (Android-rcs-ims-stack 2013.)

3.3 Rich Communication Services

Rich Communication Services (RCS) on kansainvälinen, suurimpien mobiilioperaattoreiden yhteinen hanke kehittää IP-pohjaisia kommunikaatiopalveluita. RCS-standardista on olemassa useita versioita. Keskeisimpiä RCS-kommunikaatiopalveluita ovat chat, tiedostonsiirto ja videon jakaminen GSM-puhelun aikana. RCS:n versio 5.1 tuo mukanaan mm. IP-pohjaisen ääni- sekä videopuhelun. (Rich Communication Services 2013.)

3.4 VoIP-teknologia

VoIP (Voice over Internet Protocol) on termi tekniikalle, jonka avulla ääntä voidaan siirtää reaaliaikaisesti Internetin yli. VoIP sisältää yhteyden muodostuksessa käytettävän protokollan, sekä äänensiirtoon käytettävän protokollan. (Voice over IP 2013.)

3.4.1 SIP

SIP (Session Initiation Protocol) on IP-teknologian signaaliprotokolla, jota käytetään IP-yhteyksien luomiseen, muokkaamiseen ja päättämiseen (Session Initiation Protocol 2013). Neusoft Silta -sovellus käyttää SIP-protokollaa luodessaan chat-keskusteluyhteyden, tiedostonsiirrossa, sekä kuvan ja videon jakamisessa GSM-puhelun aikana. VoIP-puheluiden yhteydessä SIP-protokollaa käytetään puhelun luomiseen ja päättämiseen.

3.4.2 SDP

SDP (Session Description Protocol) on protokolla, jolla voidaan kuvata suoratoistettavan median alustusparametreja. VoIP-puheluissa protokollaa käytetään neuvoteltaessa, mitä audiokoodekkia tullaan puhelun aikana käyttämään. SDP-protokollaa käytetään Neusoft Silta -sovelluksessa myös neuvoteltaessa videon jakamisessa käytettävä videokoodekki. (Session Description Protocol 2013.)

3.4.3 RTP

RTP (Real Time Protocol) on IP-yhteydessä käytettävä video- ja audiodatan siirtoprotokolla (Real Time Protocol 2013). VoIP-puhelussa RTP määrittää standardin pakettimuodon äänensiirtoon Internetin välityksellä. Neusoft Silta -sovellus käyttää RTP-protokollaa myös IP-pohjaisessa videon jakamisessa.

3.4.4 Audiokoodekki

IP-puheluissa käytettävä audiokoodekki on algoritmi, joka pakkaa ja purkaa audiosignaalia. VoIP-puheluissa koodekki pakkaa audiosignaalin, jolloin audiosignaali käyttää vähemmän tiedonsiirtokapasiteettia.

3.5 Android-käyttöjärjestelmä

Android on puhelimille ja muille mobiililaitteille suunniteltu ohjelmistopino, joka sisältää käyttöjärjestelmän, väliohjelmistoja ja käyttäjän perusohjelmia. Siinä käytetään avoimen lähdekoodin GPLv2-lisensoitua Linux-käyttöjärjestelmäydintä. Androidia kehitti alun perin Android Inc., jonka Google myöhemmin osti. Nykyisin sen kehittämisestä vastaa Open Handset Alliance. Androidiin tarkoitettua koodia kirjoitetaan Java-kielellä ja se käyttää Googlen kehittämiä Java-kirjastoja. (Android 2013.)

Ensimmäinen Android-käyttöjärjestelmän versio julkaistiin 5. marraskuuta 2007. Tämän jälkeen Google on julkaissut Android-käyttöjärjestelmästä useita uusia versiota. Uudet versiot yleensä korjaavat virheitä ja lisäävät uusia ominaisuuksia.

4 TOTEUTUS

Tämän insinöörityön aikana tutkittiin, minkälaista tukea Android-käyttöjärjestelmästä löytyy VoIP-sovelluksen tekemiseen sekä miten Neusoft Silta -sovellus on laajennettavissa IP-pohjaisella puhelutuella. Työn aikana toteutettiin myös pelkistetty versio VoIP-sovelluksesta, joka mahdollistaa VoIP-puhelun tekemisen Neusoft Silta -sovelluksesta RCS-käyttäjien kesken.

Projektin lähtökohtana oli, että kaikkien protokollatason toteutusten tulee käyttää samoja menetelmiä, joita käytetään myös muissa Neusoft Silta -sovelluksen IP-kommunikaatiopalveluissa. Lähtökohtana oli myös, että audiohallinnassa sekä VoIP-puhelun käyttöliittymässä tulisi käyttää mahdollisimman paljon Androidin tarjoamia ohjelmistopalveluita.

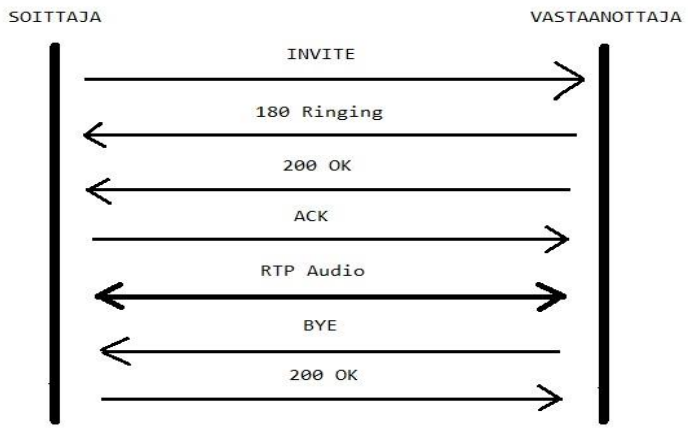
4.1 Android-käyttöjärjestelmän versio

Androidin käyttöjärjestelmän versioksi päädyttiin valitsemaan Android JellyBean 4.1. Android JellyBean tarjoaa suoran tuen joihinkin yleisiin audiokoodekkeihin, joten audiokoodekeista perittävä lisenssimaksu on suoritettu, eikä koodekkikirjastoja käyttävien sovellusten tarvitse huolehtia niistä enää.

4.2 VoIP-puhelun hallinnointi

VoIP-puhelun luonnissa käytettiin SIP-standardin mukaisia signalointimetodeja. SIP-signalointi on käytössä myös muissa Neusoft Silta -sovelluksen IP-palveluissa.

VoIP-puhelun aikaisessa SIP-signaloinnissa soittaja lähettää SIP INVITEn, johon vastapää vastaa SIP 180 RINGING, jolloin vastapäässä näytetään sisääntuleva VoIP-puhelu ja soitetaan hälytysääni. Vastapää voi halutessaan hylätä puhelun, jolloin soittajalle vastataan SIP 183 BUSY HERE -viestillä ja soittajaa indikoidaan puhelun hylkäyksestä. Jos vastapää hyväksyy puhelun, se lähettää SIP 200 OK -viestin, jonka soittaja kuittaa SIP ACK -viestillä. Tällöin SIP-yhteys on muodostettu ja voidaan aloittaa audiotiedonsiirto vastapäiden välillä käyttäen RTP-protokollaa. Kun puhelu halutaan lopettaa, lopettava pää lähettää SIP BYE -viestin, johon toinen pää vastaa SIP 200 OK -viestillä. Tällöin VoIP-puhelu on lopetettu ja kaikki puhelua varten varatut resurssit voidaan vapauttaa. (Kuva 9.)



KUVA 9. VoIP-puhelun aikainen SIP-signalointi.

4.3 Audiokoodekki

Audiokoodekiksi valittiin AMR NB (Adaptive Multi-Rate Narrowband) ja näytteenottotaajuudeksi 8000 Hz. Audiosignaalin enkoodaus ja dekoodaus hoidettiin Androidin tukemilla ohjelmistokirjastoilla. Android JellyBean -käyttöjärjestelmä versiosta löytyy MediaCodec-rajapinta, jonka avulla käsittelemätön audiosignaali voidaan pakata haluttua audiokoodekkia käyttämällä pienempään muotoon (koodi 1; koodi 2). Samaa MediaCodec-rajapintaa käyttämällä pakattu audiosignaali saadaan purettua vastapäässä audiosignaaliksi, jota voidaan toistaa Androidin audiotoistoon käytettäviä rajapintoja hyödyntämällä (MediaCodec 2013).

```
/**
 * Constructor
 * @param Codec type
 * @param sampleRate
 * @param bit rate
 * @param channel count
 */
public AudioCodecEncoder(String type, int sampleRate,
        int bitRate, int channelCount) {
    if (logger.isActivated()) {
        logger.info("AudioCodecEncoder: " + type + ", SR: " + sampleRate +
            ", BR: " + bitRate);
    }
    String audioMime = AUDIOMIME + type.toLowerCase();
    MediaFormat mediaFormat =MediaFormat.createAudioFormat(audioMime,
          sampleRate, channelCount);
    mediaFormat.setInteger(MediaFormat.KEY_BIT_RATE, bitRate);
    mediaFormat.setInteger(MediaFormat.KEY_SAMPLE_RATE, sampleRate);
    mediaCodec = MediaCodec.createEncoderByType(audioMime);

    mediaCodec.configure(mediaFormat, null, null,
          MediaCodec.CONFIGURE_FLAG_ENCODE);
    mediaCodec.start();
    }
```

KOODI 1. Androidin MediaCodec-rajapinnan alustaminen (MediaCodec 2013).

```
    /**
     * Encodes bytes
     * @param data to be encoded
     * @return encoded data
     */
    public byte[] encodeData(byte[] data) {

        ByteBuffer[] codecInputBuffers;
        ByteBuffer[] codecOutputBuffers;

        codecInputBuffers = mediaCodec.getInputBuffers();

        codecOutputBuffers = mediaCodec.getOutputBuffers();

        int inputBufIndex = mediaCodec.dequeueInputBuffer(0);
        if (inputBufIndex >= 0) {
            ByteBuffer dstBuf = codecInputBuffers[inputBufIndex];
            dstBuf.clear();
            dstBuf.put(data);
            mediaCodec.queueInputBuffer(inputBufIndex,
                                        0,
                                        data.length,
                                        0,
                                        0);

            MediaCodec.BufferInfo bufferInfo = new MediaCodec.BufferInfo();

            while (true) {
                final int res = mediaCodec.dequeueOutputBuffer(bufferInfo, 0);
                if (res >= 0) {
                    int outputBufIndex = res;
                    ByteBuffer buf = codecOutputBuffers[outputBufIndex];
                    final byte[] chunk = new byte[bufferInfo.size];
                    buf.get(chunk);
                    buf.clear();

                    mediaCodec.releaseOutputBuffer(outputBufIndex, false);
                    return chunk;

                } else if (res == MediaCodec.INFO_OUTPUT_BUFFERS_CHANGED) {
                    codecOutputBuffers = mediaCodec.getOutputBuffers();

                } else if (res == MediaCodec.INFO_OUTPUT_FORMAT_CHANGED) {
                } else {
                    break;
                }
            }
        }

        return null;
    }
}
```

KOODI 2. Audiosignaalin enkoodaus, jossa pakkaamaton audiosignaali enkoodataan pienempään muotoon MediaCodec-rajapintaa käyttämällä (MediaCodec 2013).

4.4 Audiohallinta

4.4.1 Äänen nauhoitus

Lähetettävän audiosignaalin nauhoituksessa päädyttiin käyttämään Androidin AudioRecord -rajapintaluokkaa. AudioRecord-objektin avulla voidaan lukea puhelimen mikrofonista saatavaa audiosignaalia, joka voidaan toimittaa audiokoodekin pakattavaksi (AudioRecord 2013). Tämän jälkeen pakattu audiosignaali saadaan toimitettua vastapäälle sisällytettynä RTP-pakettiin. (Koodi 3.)

```
/**
 * Recorder thread class
 */
 private class RecorderThread extends Thread {
     // Variable to stop recording
     private boolean stopped = false;
     // Constructor
     public RecorderThread() {}

     @Override
     public void run() {
         android.os.Process
             .setThreadPriority(android.os.Process.THREAD_PRIORITY_URGENT_AUDIO);
         AudioRecord recorder = null;
         try {
             int rate = getValidSampleRatesForRecorder();
             int N = AudioRecord.getMinBufferSize(rate,
                       AudioFormat.CHANNEL_IN_MONO,
                       AudioFormat.ENCODING_PCM_16BIT);
             byte[][] buffers = new byte[256][320];
             int ix = 0;
             recorder = new AudioRecord(AudioSource.MIC, rate,
                               AudioFormat.CHANNEL_IN_MONO,
                               AudioFormat.ENCODING_PCM_16BIT, N*2);
             recorder.startRecording();

             while (!stopped) {
                 byte[] data = buffers[ix++ % buffers.length];

                 int count = recorder.read(data, 0, 320);
                 if (count > 0 && mAudioPlayer != null) {
                     mAudioPlayer.writeData(data);
                 }
             }
         } catch (Exception e) {
             Log.d(DTAG, "Recorder Exception", e);
             reportError();
         } finally {
             if (recorder != null) {
                 try {
                     recorder.stop();
                 } catch (Exception e) {}
                 try {
                     recorder.release();
                 } catch (Exception e) {}
                 recorder = null;
             }
         }
     }
```

```
    /**
     * Gets valid sample rate for recorder
     * @return sample rate
     */
     private int getValidSampleRatesForRecorder() {
         for (int rate : new int[] {16000, 8000, 22050, 44100, 11025}) {
             int bufferSize = AudioRecord.getMinBufferSize(rate,
                                 AudioFormat.CHANNEL_IN_MONO,
                                 AudioFormat.ENCODING_PCM_16BIT);
             if (bufferSize > 0) {
                 // buffer size is valid, Sample rate supported
                 return rate;
             }
         }

         return 0;
     }

     public void close() {
         Log.d(DTAG, "close Recorder");
         stopped = true;
     }
}
```

KOODI 3. Äänen nauhoituksessa käytettävä RecorderThread-luokka, jossa luodaan AudioRecord-objekti sekä luetaan audiosignaalia luodusta AudioRecord-objektista (AudioRecord 2013).

4.4.2 Äänentoisto

Sisääntulevan audiosignaalin toistossa päädyttiin käyttämään Androidin AudioTrack-rajapintaluokkaa (AudioTrack 2013). AudioTrack-objektin avulla voidaan audiokoodekilla purettu audiosignaali ohjata äänipuheluille tarkoitettuun kaiuttimeen. (Koodi 4.)

```
/**
 * Speaker thread class
 */
 private class SpeakerThread extends Thread {
     private boolean mRunning = true;
     private Data mData = null

     @Override
     public void run() {
         try {
             mFifoBuffer = new VoipFifoBuffer();
             android.os.Process.setThreadPriority(
                 android.os.Process.THREAD_PRIORITY_URGENT_AUDIO);
             int sampleRate = getValidSampleRates();
             int mMinimumBufferSize = AudioTrack.getMinBufferSize(
                 sampleRate, AudioFormat.CHANNEL_OUT_MONO,
                 AudioFormat.ENCODING_PCM_16BIT);
             Log.d(DTAG,"Speaker mMinimumBufferSize: " + mMinimumBufferSize);

             mSpeaker = new AudioTrack(AudioManager.STREAM_VOICE_CALL, sampleRate,
                        AudioFormat.CHANNEL_OUT_MONO,
                        AudioFormat.ENCODING_PCM_16BIT,
                        mMinimumBufferSize,
                        AudioTrack.MODE_STREAM);

             mSpeaker.play();
             mSpeaker.setPlaybackRate(sampleRate);

             while(mRunning) {
                 try {
                     mData = (Data)mFifoBuffer.getObject();

                     if (mData != null && mData.mData != null) {
                         mSpeaker.write(mData.mData, 0, mData.mData.length);
                     }
                 } catch(Exception e) {
                     Log.e(DTAG,"Speaker write Exception");
                     e.printStackTrace();
                 }
             }
         } catch (Exception e) {
             Log.e(DTAG, "Speaker init Exception");
             e.printStackTrace();
             reportError();
         } finally {
             if (mSpeaker != null) {
                 try {
                     mSpeaker.stop();
                 } catch (Exception e) {}
                 try {
                     mSpeaker.release();
                 } catch (Exception e) {}
                     mSpeaker = null;
                 }
             }
         }
```

KOODI 4. Äänentoistossa käytettävä SpeakerThread-luokka, jossa luodaan AudioTrack-objekti sekä kirjoitetaan sisääntulevaa audiosignaalia luodulle AudioTrack-objektille (AudioTrack 2013).

4.5 VoIP-kyvykkyys

4.5.1 Verkkovaatimukset

Tutkittaessa reaaliaikaisen äänensiirron vaatimaa tiedonsiirtokapasiteettia päädyttiin siihen, että VoIP-puhelua tehtäessä pitää olla saatavilla WLAN-yhteys tai mobiilidatayhteyttä käytettäessä vähintään 3G-yhteys. Saatavilla oleva verkkoyhteys saadaan selvitettyä Androidin Connectivity-Manager-rajapintaa käyttämällä (koodi 5).

```
/**
 * Checks if we are currently in supported network for VoIP calls.
 * @param context Application context
 * @return true if supported network available.
 */
public static boolean isSupportedNetworkForVoip(Context context) {
    ConnectivityManager connectivityMgr = (ConnectivityManager)context.
        getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = connectivityMgr.getActiveNetworkInfo();

    if (networkInfo != null &&
            networkInfo.getType() == ConnectivityManager.TYPE_WIFI ||
        (networkInfo.getType() == ConnectivityManager.TYPE_MOBILE &&
            isSupportedSubTypeForVoip(networkInfo.getSubtype()))) {
        return true;
    }

    return false;
}

/**
 * Checks if mobile network sub type is supported for VoIP calls.
 * @param subtype Mobile network sub type
 * @return true, if supported sub type
 */
public static boolean isSupportedSubTypeForVoip(int subtype) {
    switch (subtype) {
        case TelephonyManager.NETWORK_TYPE_GPRS:
        case TelephonyManager.NETWORK_TYPE_EDGE:
            return false;
        case TelephonyManager.NETWORK_TYPE_UMTS:
        case TelephonyManager.NETWORK_TYPE_HSPA:
        case TelephonyManager.NETWORK_TYPE_HSDPA:
        case TelephonyManager.NETWORK_TYPE_HSUPA:
        case 15: //TelephonyManager.NETWORK_TYPE_HSPAP
        case 13: //TelephonyManager.NETWORK_TYPE_LTE
            return true;
    }

    return false;
}
```

KOODI 5. Funktioissa tarkistetaan, voidaanko VoIP-puheluja tukea saatavilla olevaa verkkoyhteyttä käyttämällä.

4.5.2 VoIP-kyvykkyyden tiedostaminen

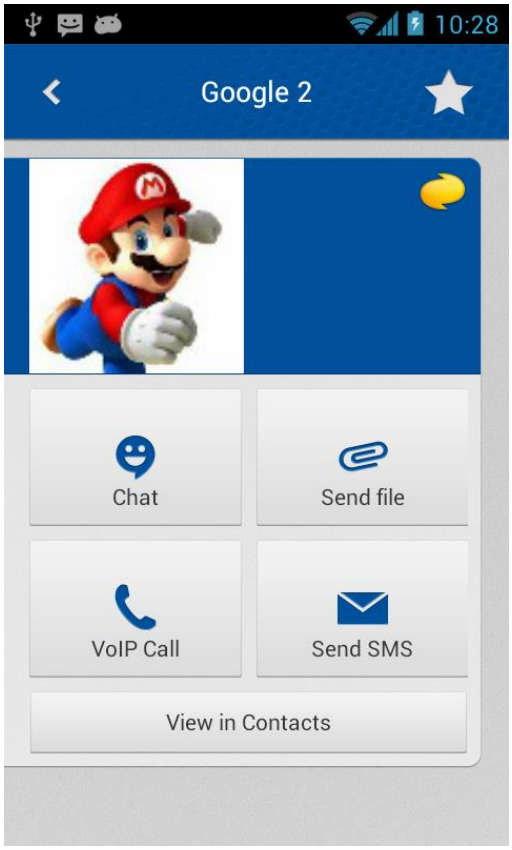
Ennen kuin VoIP-puhelu on mahdollista, pitää vastapäiden olla tietoisia toistensa mahdollisuudesta tehdä VoIP-puhelu. Vastapään VoIP-puhelukyvykkyyden selvittämisessä käytetään SIP OPTIONS -viesti menetelmää. Sama menetelmä on käytössä myös muissa Neusoft Silta -sovelluksen IP-palveluissa.

VoIP-kyvykkyys ilmastaan lisäämällä SIP OPTIONS -viestiin GSMA:n julkaisema VoIP-kyvykkyyttä tarkoittava tagi (”urn%3Aurn-7%3A3gppservice.ims.icsi.mmtel”). SIP OPTIONS -viestin vastaanottaja vastaa viestiin SIP 200 OK -viestillä ja lisää viestiin VoIP-tagin, jos on kyvykäs vastaanottamaan VoIP-puheluita.

4.6 VoIP-puhelun käyttöliittymä

4.6.1 Puhelun aloitus

VoIP-puhelun aloittaminen toteutettiin Neusoft Silta -sovelluksen kontaktikortilta. Kontaktikortilla oleva GSM-puhelun aloituspainike vaihdetaan VoIP-puhelun aloituspainikkeeksi, kun vastapäät pystyvät tekemään VoIP-puhelun. (Kuva 10.)



KUVA 10. Neusoft Silta -sovelluksen kontaktikorttinäkymä VoIP-puhelutuen kanssa.

4.6.2 Puhelunhallintapainikkeet

Vaatimusmäärittelyn mukaisesti käyttöliittymästä täytyi löytyä mahdollisuus hyväksyä tai hylätä sisääntuleva VoIP-puhelu, sekä mahdollisuus lopettaa aktiivinen VoIP-puhelu.

Puhelunhallinta suoritettiin lisäämällä puhelunäkymään toimintapainikkeita käyttämällä Androidin tarjoamaa XML-rajapintaa (koodi 6).

```
<FrameLayout
    android:layout_width="136dp"
    android:layout_height="88dp"
    android:layout_marginLeft="4dp"
    android:background="@drawable/bg_generic_button"
    android:clickable="true"
    android:onClick="onEndCallButtonClick" >

    <ImageView
        android:id="@+id/endCallItem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal|center_vertical"
        android:src="@drawable/ic_contact_details_button_call" />

    <TextView
        android:id="@+id/endCallTextItem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|center_horizontal"
        android:text="End call"
        android:textColor="@color/general_foreground" />
</FrameLayout>
```

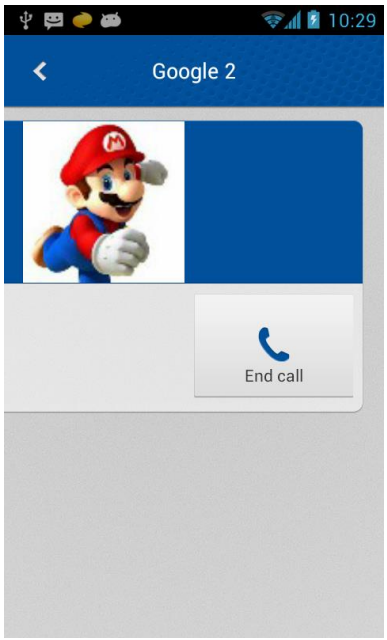
KOODI 6. Androidin XML:llä toteutettu puhelunlopetuspainike. Painiketta painamalla puhelun lopetus suoritetaan Java-kielellä kirjoitetussa funktiossa onEndCallButtonClick.

4.6.3 Soittajan esittäminen

Soittajan esittäminen päädyttiin toteuttamaan näyttämällä VoIP-puhelunäkymässä Androidin puhelinluettelosta löytyvät soittajan kontaktinimi sekä kontaktikuva (kuva 11). Kontaktinimen esittämisessä VoIP-puhelunäkymässä käytettiin Androidin XML TextView -objektia ja Kontaktikuvan esittämisessä Androidin XML ImageView –objektia (koodi 7).

```
<ImageView
    android:id="@+id/avatar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:scaleType="centerCrop"
    android:src="@drawable/ic_contact_details_empty_contact_image"
/>
```

KOODI 7. Androidin XML:llä toteutettu kontaktikuvan näyttävä ImageView-objekti.



KUVA 11. VoIP-puhelunäkymä, jossa ovat soittajan kontaktinimi, kontaktikuva sekä toiminta-painike puhelun lopettamiseen.

4.6.4 Sisääntulevan VoIP-puhelun hälytysääni

Sisääntulevan VoIP-puhelun hälytysäänen soittaminen toteutettiin Androidin MediaPlayer-ohjelmistorajapintaa käyttämällä (MediaPlayer 2013). Käyttäjän valitsema soittoääni saatiin tietoon Androidin RingtoneManager-rajapinnan kautta (RingtoneManager 2013). Sama soittoääni on käytössä myös sisääntulevan GSM-puhelun aikana. (Koodi 8.)

```
/**
 * Starts ringing tone player.
 */
public void playRingingTone() {
    if (mAudioManager.getRingerMode() != AudioManager.RINGER_MODE_SILENT) {
        mAudioManager.setMode(AudioManager.MODE_RINGTONE);
        mRingingTonePlayer = new MediaPlayer();
        Uri ringtoneUri=RingtoneManager.getDefaultUri(RingtoneManager.TYPE_RINGTONE);
        try {
            mRingingTonePlayer.setDataSource(mContext.getApplicationContext(),
                ringtoneUri);
            mRingingTonePlayer.setAudioStreamType(AudioManager.STREAM_RING);
            mRingingTonePlayer.prepare();
            mRingingTonePlayer.start();
        } catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

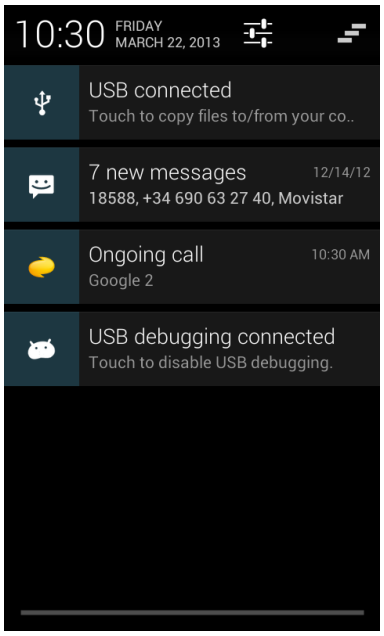
KOODI 8. Sisääntulevan VoIP-puhelun hälytysäänen toistaminen Android MediaPlayer -rajapintaa käyttämällä (MediaPlayer 2013).

4.6.5 VoIP-puhelunotifikaatio

Koska käyttäjä saattaa poistua VoIP-puhelunäkymästä puhelun aikana, piti käyttäjälle tarjota mahdollisuus palata VoIP-puhelunäkymään. Tämä toteutettiin näyttämällä VoIP-puhelun aikana Android NotificationManager -rajapintaa käyttävää VoIP-puhelunotifikaatiota (kuva 12). Notifikaatio näytetään Android puhelimen notifikaationäkymässä, ja notifikaatiota painamalla käyttäjä pääsee takaisin VoIP-puhelunäkymään. (Koodi 9.)

```
/**
 * Shows VoIP call notification.
 * @param contaxt Application context
 * @param contact Contact phone number
 * @param contactDisplayname Contact display name
 */
public void showNotification(Context context, String contact,
    String contactDisplayname) {

    Intent intent = new Intent();
    intent.setClass(context, VoipActivity.class);
    intent.putExtra("contact", contact);
    intent.putExtra("contactDisplayname", contactDisplayname);
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

    String notifTitle = context.getResources().getString(
            R.string.voip_notification_call_ongoing);
    String stripped = removeInvalidChars(contact);
    PendingIntent contentIntent = PendingIntent.getActivity(context,
        0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
    String message = ChatUtils.getContactNameFromNumber(stripped,
        contactDisplayname, context);

    Notification notif = new Notification(R.drawable.ic_stat_notification,
        notifTitle, System.currentTimeMillis());
    notif.defaults = 0;
    notif.tickerText = "";
    notif.setLatestEventInfo(context, notifTitle, message, contentIntent);

    // Send notification
    NotificationManager notificationManager = (NotificationManager) context
        .getSystemService(Context.NOTIFICATION_SERVICE);

    // Notification is updated by the given contact
    notificationManager.notify("VOIP_CALL",
        (int) Long.parseLong(stripped), notif);
}
```

KOODI 9. VoIP-puhelunotifikaation tekeminen Android NotificationManager -rajapintaa käyttämällä. Notifikaatiolle annettava Android Intent -luokka avaa VoIP-puhelunäkymän notifikaatiota painettaessa.



KUVA 12. VoIP-puhelunotifikaatio Android-puhelimen notifikaationäkymässä.

4.6.6 Sensori

VoIP-puhelun ajaksi päädyttiin käynnistämään läheisyyssensorin kuuntelu. Sensorin tarkoitus on estää vahingossa tehtävät toimintapainikkeen painallukset, kun puhelinta pidetään lähellä korvaa (koodi 11). Sensorin kuunteluun käytetään Androidin SensorEventListener-luokkaa (SensorEventListener 2013). Kun luokalta saadaan sensoritapahtuma, Android-puhelimen ollessa korvan lähettyvillä, voidaan toimintapainike toiminnot estää ohjelmallisesti. (Koodi 10.)

```
/**
 * Sensor event listener.
 */
private SensorEventListener mSensorListener = new SensorEventListener() {

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        if(event.sensor.getType()==Sensor.TYPE_PROXIMITY){
            float distance = event.values[0];
            Log.d(DTAG, "onSensorChanged distance: " + distance);
            mSensorActive = (distance >= 0.0 && distance < PROXIMITY_THRESHOLD
                && distance < event.sensor.getMaximumRange());
        }
    }
};
```

KOODI 10. Androidin läheisyyssensorin kuuntelija.

```java
/**
 * Called when end call button is clicked
 */
public void onEndCallButtonClick(View button) {
    Log.d(DTAG, "onEndCallButtonClick");
    if (mSensorActive) {
        Log.d(DTAG, "onEndCallButtonClick - sensor active");
        return;
    }
    button.setEnabled(false);
    mHandler.postDelayed(new Runnable() {
        @Override
        public void run() {
            endCallAndFinish();
        }
    }, 500);
}
```

KOODI 11. Java-puolen toteutus puhelunlopetuspainikkeelle.

5 JATKOKEHITYSMAHDOLLISUUDET

Tässä työssä saatiin aikaiseksi perustoteutus Neusoft Silta -sovellukseen integroidusta VoIP-puhelutuesta. VoIP-käytettävyyttä voidaan kuitenkin lisätä käyttöliittymään sekä audiolaatuun tehtävillä parannuksilla. Alla on listattuna joitakin Neusoft Silta VoIP-sovelluksen jatkokehitysmahdollisuuksia.

5.1 VoIP-käyttöliittymä

5.1.1 Androidin puhelinluettelo

VoIP-puhelun aloitusta voidaan parantaa integroimalla Androidin puhelinluetteloon VoIP-puhelumahdollisuus siten, että kontaktilistalla olevaa RCS-kontaktia painamalla voidaan aloittaa VoIP-puhelu suoraan Androidin puhelunluettelosta, kun molemmat päät tukevat VoIP-puheluja.

Androidin ohjelmistorajapinta tarjoaa tuen lisätä kontakteille kuvakkeellisia toimintapainikkeita. Mitä voidaan hyödyntää VoIP-toimintapainikkeiden lisäämisessä Androidin puhelinluettelon RCS-kontakteille.

5.1.2 VoIP-puheluhistoria

Neusoft Silta -sovelluksesta löytyvällä VoIP-puheluhistorianäkymällä voidaan parantaa käyttäjäkokemusta VoIP-palvelusta. Historianäkymässä käyttäjä voi selata VoIP-puheluhistoriaa. Listalla olevista VoIP-puheluista käyttäjä näkee puhelun suunnan, vastapään nimen sekä VoIP-puhelun keston.

Orange enginestä löytyvää RCS-tietokantaa voidaan hyödyntää VoIP-puheluhistorian tallennuksessa.

5.1.3 VoIP-puhelun äänentoisto kovaäänisestä

VoIP-puhelunäkymään lisättävä toimintapainike, jonka avulla puhelunaikainen äänentoisto saadaan ulos puhelimen kovaäänisen kautta, parantaa käyttäjäkokemusta, eikä käyttäjän tarvitse pitää puhelinta korvan vieressä VoIP-puhelun aikana.

Androidin AudioManager-ohjelmistorajapintaa hyödyntämällä äänentoisto voidaan ohjata kovaääniseen (AudioManager 2013).

5.2 Äänenlaatu

5.2.1 IP-pakettien puskurointi

IP-verkossa toimitettavien IP-pakettien viiveiden välillä voi olla eroavaisuuksia tai osa IP-paketeista voi kadota kokonaan. Tämä voi aiheuttaa ongelmia VoIP-puhelun äänenlaadulle. VoIP-sovelluksen tulisi puskuroida (jitter buffer) vastaanotetut IP-paketit ja toistaa ne tietynlaista synkronointialgoritmiä käyttämällä äänenlaadun säilyttämiseksi (Jitter Buffer 2013).

Android ei tarjoa IP-pakettien puskurointiin ohjelmistorajapintaa, mutta ulkoisia avoimen lähdekoodin puskurointiratkaisuja voidaan hyödyntää Neusoft Silta VoIP -toteutuksessa.

5.2.1 QoS

QoS (Quality of Service) on termi, jolla tarkoitetaan tietoliikenteen luokittelua ja priorisointia (Quality of Service 2013). Lisäämällä QoS tieto VoIP audiosignaalia kuljettaviin IP-paketteihin saadaan IP-paketit priorisoitua siten, ettei IP-verkot viivästytä turhaan audiosignaalia.

QoS tiedon lisäämisessä voidaan hyödyntää Androidin DatagramSocket-ohjelmistorajapintaa.

5.2.2 RTCP

RTCP on RTP-protokollan yhteydessä käytettävä kontrollointiprotokolla (RTP Control Protocol). RTCP välittää tietoa mm. RTP-yhteyden suorituskyvystä sekä verkon palvelutasosta (Real Time Protocol 2013). RTCP on käytössä Orangen RCS -enginen videon jaossa käytettävässä RTP-yhteydessä, ja samaa toimintaa voidaan hyödyntää myös VoIP-puhelun aikaisessa RTP-yhteydessä.

5.2.2 Kaiunpoisto

VoIP-puhelun äänenlaatua saadaan parannettua poistamalla kaiku äänentoistosta. VoIP-puhelussa kaikua saattavat aiheuttaa audiosignaalin pakkaamisessa käytettävät tekniikat sekä viiveet IP-pakettien lähetyksessä.

Androidin käyttöjärjestelmä versiosta 4.1 (JellyBean) löytyy AcousticEchoCanceler-ohjelmistorajapinta, jota hyödyntämällä voidaan välttää kaiun ilmestymistä VoIP-puhelun audiosignaaliin (AcousticEchoCanceler 2013).

5.3 Audiokoodekit

VoIP-käytettävyyttä voidaan parantaa myös lisäämällä tuettuja audiokoodekkeja. Android-käyttöjärjestelmän versio 4.1 (JellyBean) tarjoaa ohjelmistorajapinnan muutamalle yleiselle audiokoodekille, esimerkiksi AMR WB (Adaptive Multi-Rate Wideband) -audiokoodekille. Käyttämällä Android-käyttöjärjestelmä versiota 4.1 sekä Androidin tukemia audiokoodekkeja saadaan tuettujen audiokoodekkien listaa kasvatettua suhteellisen helposti (MediaCodec 2013).

Myös lisenssivapaita audiokoodekkeja voidaan tukea, esimerkiksi C-kielellä kirjoitettua Opus-audiokoodekkia. Androidin NDK -ohjelmistotyökaluja apuna käyttämällä voidaan C-kielellä kirjoitettuja ohjelmistokirjastoja integroida Androidin Java-kielen ymmärtämään muotoon (Android NDK 2013).

6 POHDINTA

Opinnäyte työssä tutkittiin, minkälaisen tuen Android-käyttöjärjestelmä tarjoaa VoIP-sovellusten tekemiseen ja kuinka Neusoft Silta -Android-sovellukseen saadaan integroitua VoIP-puhelumahdollisuus. Samalla toteutettiin pelkistetty versio Neusoft Silta VoIP -sovelluksesta, joka mahdollistaa VoIP-puhelujen tekemisen Neusoft Silta -sovelluksesta RCS-käyttäjien kesken.

Työ antoi hyvän kuvan VoIP-protokollasta ja siitä, kuinka hyvin Android-käyttöjärjestelmä tarjoaa tukea VoIP-puheluissa tarvittavaan audiohallintaan sekä käyttöliittymän luontiin. VoIP-puhelun audiohallinta sekä audiosignaalin pakkaaminen ja purkaminen saatiin toimimaan pelkästään Androidin ohjelmistorajapintoja hyödyntämällä. VoIP-puhelun muodostaminen ja lopettaminen SIP-protokollan avulla sekä audiosignaalin toimittaminen RTP-protokollaa käyttämällä saatiin toimimaan hyödyntämällä Orange enginen tarjoamia ohjelmistorajapintoja. Neusoft Silta -sovelluksessa olemassa olevat käyttöliittymät sekä Androidin käyttöliittymäsuunnitteluun tarkoitettu XML-rajapinta tarjosivat hyvää tukea VoIP-käyttöliittymän luomiseen.

Työtä tehtäessä kävi selväksi, että työn ohessa tehty pelkistetty versio VoIP-sovelluksesta ei ole riittävän käyttäjäystävällinen ja että myös äänenlaatuun tarvitaan parannuksia, koska reaaliaikaisessa äänensiirrossa IP-verkoissa voi ilmetä häiriöitä. Tämän johdosta työn ohessa pyrittiin keskittymään myös Neusoft Silta VoIP-sovelluksen mahdollisiin jatkokehitysmahdollisuuksiin.

LÄHDELUETTELO

AcousticEchoCanceler. 2013. Saatavissa: http://developer.android.com/reference/android/media/audiofx/AcousticEchoCanceler.html. Hakupäivä 30.3.2013.

Android. 2013. Saatavissa: http://fi.wikipedia.org/wiki/Android. Hakupäivä 20.2.2013.

Android NDK. 2013. Saatavissa: http://developer.android.com/tools/sdk/ndk/index.html. Hakupäivä 20.2.2013.

Android-rcs-ims-stack. 2013. Saatavissa: http://code.google.com/p/android-rcs-ims-stack. Hakupäivä 20.2.2013.

AudioManager. 2013. Saatavissa: http://developer.android.com/reference/android/media/AudioManager.html. Hakupäivä 20.3.2013.

AudioRecord. 2013. Saatavissa: http://developer.android.com/reference/android/media/AudioRecord.html. Hakupäivä 12.3.2013.

AudioTrack. 2013. Saatavissa: http://developer.android.com/reference/android/media/AudioTrack.html. Hakupäivä 12.3.2013.

Jitter Buffer. 2013. Saatavissa: http://www.webopedia.com/TERM/J/jitter_buffer.html. Hakupäivä 30.3.2013.

MediaCodec. 2013. Saatavissa: http://developer.android.com/reference/android/media/MediaCodec.html. Hakupäivä 8.3.2013.

MediaPlayer. 2013. Saatavissa: http://developer.android.com/reference/android/media/MediaPlayer.html. Hakupäivä 9.3.2013.

Neusoft Silta. 2013. Saatavissa: http://silta.neusoft.com. Hakupäivä 20.2.2013.

Quality of Service (QoS). 2013. Saatavissa: http://fi.wikipedia.org/wiki/QoS. Hakupäivä 30.3.2013.

Real Time Protocol (RTP). 2013. Saatavissa: http://fi.wikipedia.org/wiki/RTP. Hakupäivä 6.3.2013.

Rich Communication Services (RCS). 2013. Saatavissa: http://www.gsma.com/rcs. Hakupäivä 20.2.2013.

RingtoneManager. 2013. Saatavissa: http://developer.android.com/reference/android/media/RingtoneManager.html. Hakupäivä 21.3.2013.

SensorEventListener. 2013. Saatavissa: http://developer.android.com/reference/android/hardware/SensorEventListener.html. Hakupäivä 23.3.2013.

Session Description Protocol (SDP). 2013. Saatavissa: http://fi.wikipedia.org/wiki/Session_Description_Protocol. Hakupäivä 6.3.2013.

Session Initiation Protocol (SIP). 2013. Saatavissa: http://fi.wikipedia.org/wiki/SIP. Hakupäivä 5.3.2013.

Voice over IP (VoIP). 2013. Saatavissa: http://fi.wikipedia.org/wiki/Voip. Hakupäivä 3.3.2013.