UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Nejc Gašper

Kiosk z vizualno komunikacijo na daljavo s klasičnim telemarketingom

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Peter Peer

Ljubljana, 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00240/2012
Datum: 02.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **NEJC GAŠPER**

Naslov: **KIOSK Z VIZUALNO KOMUNIKACIJO NA DALJAVO S KLASIČNIM TELEMARKETINGOM**

KIOSK WITH VISUAL COMMUNICATION AT A DISTANCE WITH CLASSIC TELEMARKETING

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Najprej preučite strojno opremo, programska okolja, protokole, ki omogočajo razvoj kioska z vizualno komunikacijo na daljavo s klasičnim telemarketingom. Nato zasnujte in implementirajte rešitve ter jih ovrednotite predvsem glede na uporabniško izkušnjo, potrebno pasovno širino in obremenitev procesorja.

Mentor:
doc. dr. Peter Peer

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani

Nejc Gašper, z vpisno številko 63040035,

sem avtor diplomskega dela z naslovom:

Kiosk z vizualno komunikacijo na daljavo s klasičnim telemarketingom

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  doc. dr. Petra Peera
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 22. 10. 2012 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju, prijateljem in sorodnikom za razumevanje in potrpežljivost.

Kazalo

Seznam uporabljenih kratic in simbolov

720p (tudi HD, HD pomeni High Definition) – oznaka ločljivosti slike ali videa višine 720 pik in širine 1280 pik

1080p (tudi FHD oz. Full–HD, HD pomeni High Definition) – oznaka ločljivosti slike ali videa višine 1080 pik in širine 1920 pik

AAC (Advanced Audio Coding) – zvočni kodek, naslednik MP3; del standardov MPEG-2 in MPEG-4

AS (ActionScript) – programski jezik za programiranje v okolju Flash

CPE – glej CPU

CPU (Central Processing Unit) – centralna procesna enota ali tudi CPE, jedro vsakega računalnika; čip ki skrbi za večino izračunov in za izvajanje programov

DHTML (Dynamic HTML) – dinamični HTML, beseda ne označuje posamezne tehnologije, ampak katerokoli tehnologijo, ki omogoča izgradnjo dinamičnih spletnih strani v kombinaciji s HTML.

Flash (Adobe Flash) – platforma za multimedijske spletne aplikacije

Flex (Apache Flex) – orodja za razvoj aplikacij na Flash platformi

FLV (Flash VIdeo) – vsebnik za Flash medijske datoteke; vsebina je lahko kodirana z različnimi H.263 in H.264 kodeki

FLV1 – štiričrkovna koda za kodek Sorenson Spark

FPS (Frames Per Second) – slik (okvirov) na sekundo

FMS (Flash Media Server) – strežnik, ki ga ponuja Adobe kot komplement svoji Flash platformi; omogoča pretok podatkov in pa klice funkcij Flash aplikacijam

H.323 – protokol s podobno nalogo kot SIP, prav tako vzpostavlja ustrezne komplementarne protokole ter algoritme za delovanje sistema IP telefonije

H.264 (MPEG-4 AVC) – del standarda MPEG-4 za video kompresijo

HTML (HyperText Markup Language) – opisni jezik za prikaz spletnih strani

HTTP (HyperText Transfer Protocol) – aplikacijski protokol za prenos podatkov o spletnih straneh, osnova svetovnega spleta

IDE (Integrated Developement Environment) – računalniški program ali zbirka programov namenjenih za razvoj programske opreme; ponavadi vključuje urejevalnik kode, razhroščevalnik ter zbirko orodij za prevajanje programske kode v strojni jezik

IP (Internet Protocol) – protokol namenjen naslavljanju naprav v omrežjih

JNI (Java Native Interface) – vmesnik med Javo in gostiteljskim sistemom; omogoča direktne klice knjižnic iz Jave ali obratno

JSF (Java Server Faces) – tehnologija za delanje spletnih strani na Java platformi, nadomešča starejšo JSP

JSP (Java Server Pages) – starejša tehnologija za delanje spletnih strani na Java platformi

JVM (Java Virtual Machine) – javanski navidezni stroj; program, ki izvaja javanski bajt-kod, bodisi interpretirano ali pa direktno na strojni opremi

MP3 (MPEG-2 Audio Layer 3) – zvočni kodek; del standardov MPEG-1 in MPEG-2

QoS (Quality of Service) – nadzorni mehanizmi za izboljšanje kvalitete prenosa podatkov in dodeljevanje smiselnih prioritet prenosa, predvsem pri video in glasovni telefoniji, lahko pa tudi v zelo obremenjenih omrežjih in Internetu

RPC (Remote Procedure Call) – oddaljeni klic procedure

RTCP (RTP Control Protocol) – nadzorni protokol, ki deluje ob RTP in skrbi za kvaliteto storitve

RTMP (Real Time Messaging Protocol) – protokol za komunikacijo in prenos podatkov med aplikacijami na platformi Flash

RTP (Real-time Transport Protocol) – UDP protokol za prenos multimedijskih podatkov prek IP omrežja

SDP (Session Description Protocol) – protokol za opis podatkovnih tokov

SIP (Session Initiation Protocol) – protokol za vzpostavitev in nadzor podatkovnih tokov, predvsem v uporabi za (video) telefonijo

SVGA (Super Video Graphics Array) – oznaka ločljivosti slike ali videa višine 600 pik in širine 800 pik

TCP (Transmission Control Protocol) – temeljni internetni protokol, ki omogoča zanesljiv in urejen prenos podatkov

UDP (User Datagram Protocol) – hiter internetni protokol brez lastnega preverjanja konsistence in zaporedja prenosa podatkov

USB (Universal Serial Bus) – univerzalno serijsko vodilo; uporablja se za vse vrste vhodno-izhodnih naprav

XHTML (Extensible HyperText Markup Language) – gre za XML označevalni jezik, ki spominja na HTML

Povzetek

Video komunikacija je povsod okrog nas. Vendar pa ko se poglobimo in poskušamo ustvariti kaj več kot najbolj osnovne rešitve z video komunikacijo, kaj hitro ugotovimo, da ni vse tako preprosto. Za potrebe projekta ePomočnik smo raziskovali možne video komunikacijske rešitve. Možnosti je ogromno. Na kratko smo predstavili osnovne tehnologije – Java, Flash, protokole. Nato smo se lotili izgradnje in namestitve treh različnih rešitev, dve od njih sta osnovani na Flash, Java in Red5 platformah. Tretja rešitev pa je realizirana s pomočjo vsestranskega predvajalnika VLC. Pri vsaki rešitvi smo opazovali osnovna sistemska sredstva za nekaj scenarijev ter izvedli anketo pri uporabnikih. Izkazalo se je, da trenutno Flash rešitev še vedno ponuja več, kljub temu da je platforma v počasnem odhajanju. Red5 odprtokodni strežnik pa je brez težav kos svoji nalogi. VLC je sicer zelo dobra rešitev potrebuje pa, za komunikacijo v realnem času, precej ukvarjanja z detajli v nastavitvah in najverjetneje tudi kakšno manjšo spremembo v izvorni kodi. Nezahtevnim priporočamo, naj uporabijo Flash in Red5 kombinacijo, vsem ostalim pa da se poglobijo v x264 kodirnik in si prilagodijo VLC ali pa FFmpeg izvorno kodo za lastno uporabo.

Ključne besede:

video, komunikacija, Red5, Flash, VLC, VideoLAN, RTP, RTMP, Java, JavaCV, Xuggle, x264

Abstract

Video communication is rapidly developing and expanding field. The problem is, we found out early into the project, that things get quite complicated as you try to do more. We delved into the mysteries of video communication over web for a specific project ePomocnik (translates to eHelper). There is a lot of possibility but we singled out a few. First of all we present a few basic technologies used – Java, Flash, and some protocols used for video communication. What is more, we built three basic solutions for video communication. Two of them have a common base – Red5 streaming server. First is based on the standard Flash applets for video capture and playback, the second is using Java for video capture. The third solution is depending on VLC player as server and client. Going further from that point, we monitored CPU and network bandwidth usage for every solution and we did a short user survey regarding solution quality. What we concluded was quite interesting, even though nobody puts much faith in Flash anymore, Flash and Red5 combination is still a very strong option. By the choice of users and test data it is still the best solution followed by VLC. Problem with VLC is that it needs quite a bit of tinkering, while Flash is "ready to go". The recommendation for anyone not really demanding would be to just stick with Flash for now. If your resources permit it, you should use VLC or FFmpeg with x264 encoder by building your own distribution compiling it from source and customizing it.

Key words:

video, communication, Red5, Flash, VLC, VideoLAN, RTP, RTMP, Java, JavaCV, Xuggle, x264

Poglavje 1

Uvod

Imamo preprost problem. Želimo postaviti video komunikacijo med reklamnim panojem in operaterjem nekje v pisarni. Tako racionaliziramo poslovanje zaradi manj poslovalnic in nižjega stroška najema, hkrati pa obogatimo oglaševalski del z živimi informacijami. Kje začeti? Naredili bomo prototipno rešitev s primerno video komunikacijo.

ePomočnik je multidisciplinaren projekt, ki združuje tehnologijo za vizualno komunikacijo na daljavo s klasičnim telemarketingom. Kiosk zamišljen v projektu ePomočnik je dinamičen in interaktiven, ni samo električni plakat za informacije. Osnova projekta je preprosta: naredimo dvostransko video komunikacijo in z njo razširimo informacijsko točko. Predvidene so še druge možne razširitve delovanja poleg svetovanja živega agenta, na primer vgradnja čitalca kartic za direktne nakupe z dostavo na dom, vgradnja tiskalnika za nakup vstopic ali tiskanje fotografij, merilec krvnega tlaka in pulza za pomoč ostarelim itd.

To diplomsko delo je produkt raziskav, ki smo jih začeli v okviru projekta ePomočnik, in njihovo nadaljevanje. Izhodiščna situacija v diplomskem delu je osnovana na ideji ePomočnika, prav tako pa vse predpostavke in osnovni cilji - usmerili na obravnavo video komunikacijske rešitve primerne za ePomočnika, torej video komunikacijska rešitev za kiosk (slika 1.1).

1.1 Video komunikacija pri ePomočniku

Projekt ePomočnik ima specifične potrebe glede video komunikacije. Predvsem potrebujemo dovolj dobro kvaliteto zvoka in slike, da ima uporabnik občutek živega sogovornika. Torej osnovno prepoznavanje mimike sogovornika, razločen zvok, dobra sinhronizacija zvoka in slike ter čim krajši zamik za pogovor brez

Slika 1.1: Primer kioska s priključeno napravo za merjenje pritiska.

neprijetno dolgih premorov. Upoštevajmo, da bo kiosk velik, uporabnik bo videl sliko sogovornika v skoraj naravni velikosti.

Prav tako potrebuje agent možnost nadzora nad zvokom in vsebino za več kot enega odjemalca na agenta. Agent je v tem primeru zaposleni pri neki organizaciji, ki upravlja s sistemom. Imenovali ga bomo *upravljalec*. *Odjemalec* je kiosk, torej osebni računalnik postavljen na javnem ali delno javnem mestu, vendar z omejenim dostopom do sistema. *Uporabnik* je oseba, ki uporablja ta kiosk za komunikacijo z upravljalcem.

Prednost sistema je lociran kvalitetnejši kader na centralni lokaciji, potrebno pa je omogočiti osnovni nivo storitve, ki bo ustrezal prodajalcu na lokaciji ter ne bo odgnal uporabnikov.

1.2 Osnovne predpostavke

Odločili smo se za uporabo odprtih rešitev. Prav tako ne bomo uporabljali sestavljenega video signala (angl. video compositing). S tem mislimo na sestavljanje vse vsebine v en sam video signal, ki ga nato ciljna naprava predvaja kot bi ga televizor. Izvedli bi radi rešitev prek več preprostih video in podatkovnih kanalov, preko katerih direktno prenašamo sliko in podatke z ene in druge strani. Uporabili bomo dinamično spletno stran ter vgradne rešitve za video komunikacijo, torej bo ta dinamična spletna stran poskrbela za pravilno prezentacijo in razporeditev informacij na zaslonu. Uporabniki ne bodo imeli direktnega nadzora nad sistemom.

V diplomskem delu se bomo osredotočili na video komunikacijski del. Naša motivacija je postavitev poceni in fleksibilnega video sistema, tako ob prodaji sistema ni večina vrednosti v ceni namenskih strojnih rešitev, ampak v naših servisih.

Poglavje 2

Strojna oprema in programska okolja

2.1 Strojna oprema

Sistem kot je ePomočnik lahko glede na strojno opremo izvedemo na več načinov. V predpostavkah smo se omejili na programske načine brez specializirane strojne opreme. Namenska strojna oprema je sicer na voljo [1], vendar bi bilo vseeno nekaj dela pri integraciji takšne rešitve, želimo namreč več kot samo zaslon za video komunikacijo. Predvsem pa bi uporaba namenske strojne opreme povečala osnovno ceno sistema in nas postavila v podrejen položaj pri trženju. Fleksibilnost namenskih rešitev je seveda proporcionalna s samo ceno.

Ker nas zanima nizka cena nismo pretiravali pri strojni opremi. Glavni testni sistem je osebni računalnik z USB kamero. V času pisanja je možno primerljiv sistem kupiti za ≈500€.

2.1.1 Osebni računalniki

Na glavnem testnem osebnem računalniku teče operacijski sistem Windows 7. CPE je štirijedrnik AMD Phenom II 955BE. Teče pri 3,2 GHz. Drugi testni sistem je podoben, le da gre za prenosnik z i7-2670QM CPE. Na koncu smo se odločili, da uporabimo za meritve prvega, zaklenjenega na fiksno frekvenco, predvsem zaradi lažjega odčitavanja rezultatov. Dinamično spreminjanje frekvence bi samo dodalo več spremenljivk v problematično odčitavanje porabe časa CPE.

2.1.2 Kamere

Pri kamerah je najbolj zanimiva Logitechova C510, trenutno sicer že obstaja novejši model, ki pa je po lastnostih in ceni precej podoben. Gre za cenovni razred 50€, torej za široko dostopno napravo. Zanima nas ali naprava proizvede zadovoljivo kvaliteto.

- Kamera Logitech HD C510, podpira ločljivosti SVGA in 720p
- Vgrajena kamera HP, podpira ločljivosti VGA in 720p
- Kamera Logitech QuickCam, podpira ločljivosti QVGA in VGA

2.2 Uporabljena razvojna orodja

Izhodiščna točka razvoja je bilo zavedanje, da lahko uporabimo Flash za video komunikacijo. Po kratkem raziskovalnem delu smo ugotovili, da obstaja tudi odprtokodni prosti strežnik za Flash: Red5. Podpira pa večino funkcij lastniškega FMS, predvsem pa nam pomemben prenos videa. Red5 je napisan v Javi, pripravljena pa ima osnovna orodja za razvoj strežniških aplikacij kot modul za Eclipse. Predvsem to ter znanje Jave in poznavanje okolja Eclipse sta glavna dejavnika za izbiro tega okolja.

2.2.1 Eclipse

Eclipse [2] je razvojno okolje napisano predvsem v Javi namenjeno pa primarno razvoju v Javi. Vendar pa to ni le razvojno okolje (IDE), ampak ponuja precej več prek vtičnikov. Na voljo so različice tega okolja za precej različnih jezikov, že za samo Javo je več različnih prilagoditev: klasičen Java razvoj, Java EE in Android razvoj. Prav tako so na voljo okolja za C++, Cobol, PHP, Flash in še kaj bi se našlo. Gre torej za prosto razširljiv izdelek pri katerem si lahko vsak uporabnik sestavi svoj izbor različnih modulov.

Eclipse je zanimiva izbira, ker je prost in zastonjski program. Torej je možna uporaba brez ovir in stroškov, hkrati pa je seveda dovolj zmogljiv, da ga marsikdo navaja kot svoje primarno okolje za razvoj. Obstajajo tudi komercialne različice Eclipse razvojnega okolja, na primer RedHatov JBoss Developement Studio, IBM-ov RAD in pa recimo Adobe Flash Builder.

Prek Eclipse orodja poteka razvoj za Red5 strežnik in razvoj za Apache Tomcat (JSP/JSF aplikacije). V tem okolju smo razvili preprosto aplikacijo za zajem slike.

2.2.2 Adobe Flash Builder

Adobe Flash Builder [3] je komercialni IDE za razvoj Adobe Flash aplikacij. Med drugim omogoča tudi, z nekaj spremembami, nastavitev projekta razvoj Adobe Air programov, česar pa se nismo lotevali.

Razvij Flash programov poteka deklarativno za vmesnike, prav tako pa imamo možnost sprotne vizualizacije. Elemente lahko s palete nosimo na zaslon in nastavljamo atribute; kot alternativo, lahko pa direktno urejamo .mxml datoteke, ki so v standardnem formatu XHTML.

Koda programov Flash pa je pisana v AS. Gre za preprosto zadevo, ki precej spominja na javansko kodo, potrebujemo pa osnovno API poznavanje.

Na voljo je zastonjska časovno omejena preizkusna različica, ki je zadostovala za obseg tega projekta. Zgradili smo dve preprosti aplikaciji.

2.2.3 Spletni brskalnik

Potrebujemo spletni brskalnik, ki omogoča način kiosk. Kiosk način delovanja omeji uporabnikov nadzor nad napravo. Predvsem želimo, da je brskalniško okno čez celoten zaslon in da ga ni možno zapreti. Poleg tega mora imeti izbrani brskalnik podoro za vtičnike Flash, Java in VLC.

Preizkusili smo brskalnika Mozilla Firefox [4] in Opera [5] oba izpolnjujeta vse pogoje za delovanje naših rešitev.

2.3 Uporabljena tehnologija

Članek [6] razloži nekaj osnov pretočnega oddajanja slike in protokolov, med drugim tudi omeni nekaj protokolov, ki smo jih uporabili.

2.3.1 Protokol RTMP

Gre za protokol v lasti podjetja Adobe, namenjen pa je prenosu zvoka, slike in pa tudi podatkov. Delna specifikacija protokola je na voljo za javno rabo [7]. Klasična verzija uporablja TCP vrata 1935. Obstaja nekaj različic RTMP protokola za posebne situacije: kriptirana povezava (RTMPE, deluje samo na Adobe izdelkih), varna povezava SSL (RTMPS) ter tunelska različica prek protokola HTTP (RTMPT).

Protokol RTMP je v rabi pri Flash aplikacijah, vendar ne samo za prenos podatkovnih tokov kot sta zvok in slika. Poleg tega omogoča še signalizacijo

ter sinhronizacijo objektov na daljavo. Prav vsebuje možnosti za oddaljeni klic procedur (RPC).

2.3.2 Protokol RTP

RTP protokol [8] je za razliko od RTMP precej bolj preprost. Gre za standardiziran protokol pri prenosu video podatkov, bodisi pri video telefoniji ali telekonferencah. Gre za UDP protokol, ima pa vgrajene mehanizme za zaznavanje nepravilnega zaporedja paketov in odpravljanje problemov nekonsistentne hitrosti prenosa (angl. packet jitter). Prav tako je RTP namenjen samo prenosu podatkov. Po potrebi pa lahko uporabimo druge protokole vzporedno z RTP za vzpostavitev dodatne funkcionalnosti. RTCP se uporablja za sinhronizacijo večih RTP tokov in QoS, SIP ali H.323 za signalizacijo (npr. video telefonija) ter SDP za prenos metapodatkov o podatkovnem toku.

Ena izmed prednosti tega protokola je večkratno sočasno pošiljanje podatkov (angl. multicast). Prav tako je pomembeno zavedanje, da RTP pošilja podatke na določene naslove, lahko tudi multicast naslove kot je omenjeno. Gre torej za potisni protokol (angl. push-based). Predvsem zaradi tega potrebujemo spremljevalne storitve, ki poskrbijo, da oddajamo podatke le kadar je potrebno.

Potrebujemo odprta vhodna IP vrata pri prejemniku slike.

2.3.3 Protokol HTTP

Protokol HTTP [9] se sicer uporablja za prenos spletnih strani in spremljevalnega gradiva. Lahko pa se uporabi tudi za prenos pretočnega videa. Prednost je predvsem v prostem pretoku prek protokola HTTP, le-ta je namreč protokol, ki bo deloval v kar največ situacijah, saj ga blokiramo le, kadar želimo popolnoma onemogočiti uporabo svetovnega spleta.

V našem primeru VLC rešitev omogoča oddajanje slike prek protokola HTTP, je pa ta način delovanja nasproten RTP načinu, in imamo klasičen vlečni protokol (angl. pull-based).

Težava HTTP protokola je predvsem v nekaj manjši hitrosti zaradi dodatnih spremljevalnih podatkov, uporablja se namreč TCP prenosni protokol.

Potrebujemo odprta vhodna IP vrata pri izvoru slike.

2.3.4 Flash

Flash je zelo razširjena platforma za izdelavo zahtevnih dinamičnih spletnih strani, ki pa se uporablja vedno manj. Predvsem se je prijel tehnologije Flash slab glas med uporabniki in razvijalci zaradi počasnosti, požrešnosti in velike porabe energije. Klasična raba vstavkov na tej tehnologiji se vedno manj uporablja, je pa pri dani nalogi precej koristen za snemanje in predvajanje posnetkov. Počasi ga izpodriva HTML5, vendar pa je zaenkrat lažje ostati pri preverjenih tehnologijah.

Tipično se namesti Flash okolje kot vtičnik v vseh spletnih brskalnikih, na spletno stran pa se vključi prek html <*embed*> oznake. Tako vključen Flash objekt nato deluje kot ločen program znotraj brskalnika. Tudi povezuje se ločeno od brskalnika prek RTMP povezave ali katere od različic te povezave. Zgodovinsko gledano je bil to dolgo eden najbolj priljubljenih načinov za vključevanje dinamične vsebine ali animacije v spletne strani konec prejšnjega in na začetku tega tisočletja. V zadnjih letih ga izpodrivajo DHTML spletne strani na različnih tehnologijah. Trenutno ni več nobenega razloga za izgradnjo spletne strani v Flash tehnologiji Pri ostalih funkcijah, kot je spletni video ter igre pa se Flash še vedno uporablja.

2.3.5 Java

Java je programski jezik, hkrati pa se tako imenuje tudi platforma, platfoma je sestavljena iz prevajalnika (angl. compiler), knjižnic (angl. class libraries) in JVM izvršnega okolja. Razvili so jo sredi devetdesetih let v podjetju Sun, trenutno pa je v lasti Oracla. Sintaktično je zelo podobna jeziku C++, po katerem ima tudi objektni model z nekaj manjšimi spremembami. Zasnovali so jo kot programski jezik s čimmanj odvisnostmi od programskih platform in strojne opreme. Programska koda se vedno najprej prevede v Java bajt-kod (angl. bytecode). Le-to pa nato interpretira javanski navidezni stroj ali pa se prevede v strojni jezik ciljne platforme tik pred izvajanjem (angl. just-in-time compilation). Na ta način je možno izvajati javanske programe na vseh platformah, za katere obstaja JVM.

Primer platforme, ki uporablja Java programski jezik, ni pa zmožna poganjati programov prevedenih za standardno Java platformo je Android platforma. Jezik je sicer skupen, knjižnice in JVM pa niso.

Osnovne knjižnice so precej bogate, prav tako je preprosto nalagati zunanje knjižnice. Zadeva se pa pogosto zatakne, ko želimo dostopati do naprav bolj direktno, ali ko ne obstajajo javanski API/knjižnice, potrebne za določeno opravilo. Ker znotraj javanske kode nimamo dostopa do sistemskih klicev

gostiteljskega sistema je to seveda težava. Obstaja rešitev, in sicer je to vmesnik JNI, ki omogoča direktno povezavo javanskega programa z nejavanskimi knjižnicami. Tega se sicer izogibamo, kjer je možno, včasih pa ne gre drugače. Predvsem pa s tem izgubimo prenosljivost Java programa.

V našem primeru smo uporabili vmesnike za knjižnice, ki omogočajo zajem in obdelavo videa. Ti vmesniki pa komunicirajo preko standarda JNI s knjižnicami direktno na gostiteljskem OS. Problem tega pristopa je, da moramo imeti na voljo enake knjižnice za vse OS, kjer želimo da naš program deluje.

2.3.6 Red5

Red5 [10] je odprtokodni projekt, katerega rezultat je strežnik, ki lahko nadomesti FMS. Koda za strežnik je spisana v Javi in teče znotraj Apache Tomcat strežnika.

2.4 Namestitev in razvoj

2.4.1 Eclipse in Red5 razvijalsko okolje

Namestitev samega okolja Eclipse je zelo preprosta. Najprej potrebujemo javansko okolje JDK [11], za delovanje ga potrebujeta Eclipse IDE in Red5 strežnik. Nato na spletno stran fundacije Eclipse poiščemo in prenesemo *Eclipse Indigo for Java EE (32-bit)*. Vsi deli okolja bodo ostali 32-bitni, da smo prepričani glede združljivosti med seboj in za nazaj.

1. Java JDK namestimo prek namestitvenega programa.

2. Eclipse preprosto razširimo v poljubno mapo.

3. Namestimo Red5 IDE vtičnik za Eclipse, navodila so na spletni strani Red5 [12].

Ti koraki so potrebni pri postavitvi razvojnega okolja za aplikacije Red5. Eclipse smo uporabili še za razvoj spletnih aplikacij, kamor je vgrajena video komunikacija prek vtičnikov.

2.4.2 Flash Builder razvijalsko okolje

Gre za okolje zgrajeno na Eclipse platformi. Za namestitev imamo dve možnosti. Prva je prenos celotnega razvijalskega paketa, ki vključuje Eclipse in vse že prednastavljene module. Alternativno pa lahko prenesemo samo vtičnik za Eclipse, ki ga lahko namestimo na že obstoječi Eclipse namestitvi. V vsakem primeru pa moramo za uporabo tega orodja kupiti licenco po preteku 90 dni preizkusnega obdobja. Poudariti je vredno še to, da moramo za razhroščevanje Flash programov (angl. debug) namestiti posebno različico Flash okolja, ki se poveže z vgrajenim razhroščevalnikom in omogča ustavitev izvajanja programa v poljubni točki kode (angl. breakpoint).

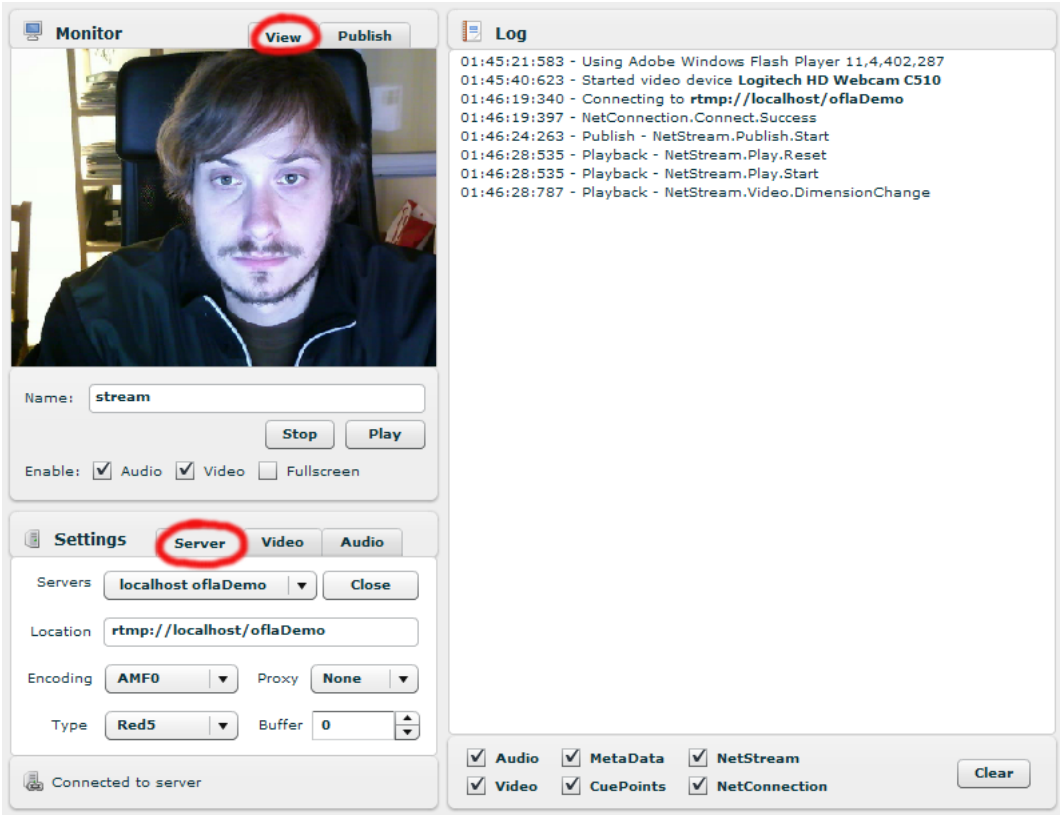
2.4.3 Red5 pretočni strežnik

Red5 strežnik namestimo prek standardnega namestitvenega programa [13]. Servis konfiguriramo, da se zažene samodejno. Med naloženimi demonstracijskimi aplikacijami je *oflaDemo*, ki zadostuje našim potrebam pri osnovnem testiranju: aplikacija v realnem času posreduje posnetke, ki jih na strežnik pošilja snemalni program, odjemalcem. Prav tako je zanimiv program za testiranje *"publisher"*. Slednji opravlja vlogo snemalnega in predvajalnega programa. Primeren je za testiranje in demonstracijo delovanja, za vgradnjo v končno rešitev pa ni primeren, saj ima že izdelan uporabniški vmesnik (sliki 2.1 2.2) in ne podpira ločljivosti 720p.

2.4.4 Flash rešitev

Flash program je nameščen na spletnem strežniku, pri odjemalcih pa namestitev ni potrebna, saj se prenese kot del spletne strani. Na spletni strani je namreč vključen kot html *embed* objekt. Pri strankah na kiosku moramo poskrbeti za nastavitve: Flash si mora zapomniti varnostne nastavitve kamere. Poziv stranki glede omogočanja kamere namreč ni zaželjen.

Izdelava Flash odjemalca [14] ni zahtevna. Zahteva znanje jezika AS in poznavanje Flex API. Ustvarili smo dva različna Flash odjemalca z uporabo Flash Builder razvijalskega okolja. Prvi program je prikazovalnik slike, zmore se povezati z Red5 strežnikom in prikazovati zvok ter sliko. Drugi program pa pošilja zvok in sliko z lokalne kamere ter mikrofona na strežnik. Oba je možno krmiliti z nekaj osnovnimi parametri. Večji del obeh programov je AS koda, manjši del pa predstavlja .mxml XHTML datoteka s postavitvijo, saj imamo le en element.

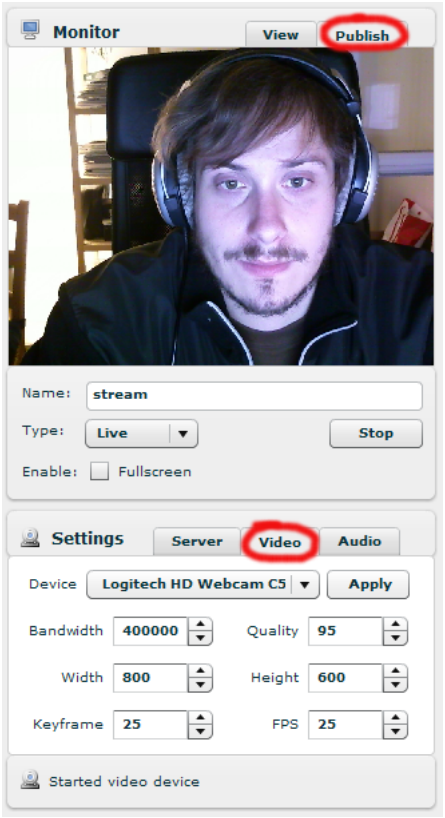


Slika 2.1: Demonstracijski program "Publisher": predvajanje video kanala z gumboma play in stop zgoraj levo, povezava na strežnik spodaj levo, okno s sistemskimi sporočili (angl. log) desno.

```
<?xml version="1.0" encoding="utf-8"?>
<capvid:CaptureVideo xmlns:mx="http://www.adobe.com/2006/mxml"
                     xmlns:capvid="org.simple.captureVideo.*"
                     paddingLeft="0" paddingRight="0" paddingTop="0" paddingBottom="0">

    <mx:HBox id="box" width="100%" paddingLeft="0" paddingRight="0" paddingTop="0"
             paddingBottom="0">
        <mx:UIComponent id="videoHolder"/>
    </mx:HBox>
</capvid:CaptureVideo>
```

Koda 2.1: Primer .mxml datoteke.



Slika 2.2: Demonstracijski program "Publisher": oddajanje video kanala zgoraj in nastavitve kamere spodaj

```
//uvozimo potrebne razrede
import flash.media.Video;
import flash.media.NetConnection;
//...//

//definiran razred, tipa Application
public class CaptureVideo extends Application {
//definicije spremenljivk
    private var connection:NetConnection = null;
    private var camera:Camera = Camera.getCamera();
\\...\\

//izsek iz funkcije, ki doda video na xhtml element videoHolder
    video = new Video(width, height);
    videoHolder.addChild(video);
}
```

Koda 2.2: Prikaz nekaj izsekov ActionScript kode

Razvoj Flash programov je možen tudi brez komercialne programske opreme, saj je razvijalski paket Flex prosto dostopen [15]. Zanimivi predlogi glede tega se nahajajo v [13].

2.4.5 Java rešitev

Ideja za java program za zajem slike prihaja iz manjšega programa z imenom *red5-screenshare* [16]. Omenjeni program zajema sliko namizja v določenem intervalu prek paketa *java.awt.Robot*. To nato pošlje strežniku Red5 po standardni poti prek protokola RTMP, na enak način kot pošiljajo ostali programi zajeto sliko s kamere. Odjemalec potem lahko s standardnim predvajalnim programom predvaja video zajetega namizja v realnem času. En posnetek na sekundo ponavadi zadostuje.

Ideja je bila napraviti podobnega, vendar s kamero kot virom. Rešitev pa se je izkazala za bolj kompleksno. Z uporabo knjižnic Xuggle [17] in JavaCV [18] za kompresijo in zajem slike nam je navsezadnje to tudi uspelo. Prednost namestitve je, da potrebujemo le JRE javansko okolje, javanski paket JAR nato deluje brez težav bodisi kot vtičnik v brskalniku, bodisi preko JNDI zagona, bodisi direktno lokalno kot klasičen program.

Težava pri tem pa je uporaba zunanjih knjižnic, če jih uspemo vse zapakirati v JAR datoteko, bodo zadeve delovale, sicer pa moramo te knjižnice naložiti posebej. Zunanje knjižnice so problematične v heterogenih okoljih (npr. Linux, Windows), kjer so knjižnice seveda različne.

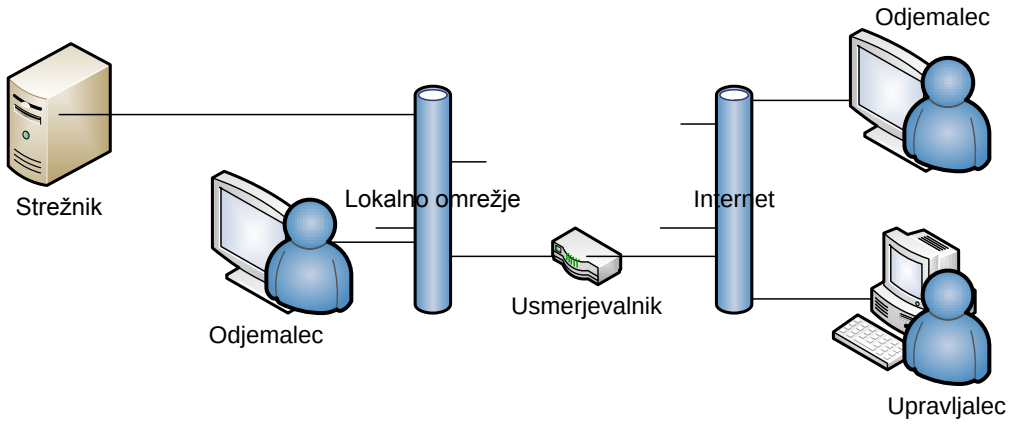
2.4.6 VLC predvajalnik in strežnik

VLC rešitev je strežnik, odjemalec, posrednik in ima na voljo cel paket kodirnikov. Po želji se namesti vtičnik v brskalnik, kar omogoči delovanje znotraj spletne strani. Namestitev poteka preko standardnega namestitvenega programa [19]. Pazimo, da se namestijo vtičniki za brskalnik, tako da lahko vgradimo sliko v spletno stran.

Poglavje 3

Predstavitev izdelanih rešitev

Najbolj preprosta in prva rešitev je uporaba Flash programov in tehnologije dinamičnih spletnih strani za vse odjemalce. Pri vseh rešitvah smo se držali podobne modularne zgradbe sistema, spletni strežnik in dinamične spletne strani torej ostajajo, video rešitev pa se spreminja. Pri VLC je potreben dodaten servis za nadzor in upravljanje. VLC lahko deluje kot grafični predvajalnik medijskih vsebin, lahko pa ga poženemo iz ukazne vrstice, kar omogoča nadzor prek zunanje aplikacije ali servisa. VLC zaradi svoje narave omogoča tudi direktne povezave med odjemalci, vendar pa to pomeni dodatne težave, ker moramo zagotoviti pravilno povezljivost vseh odjemalcev in smo se temu izognili.



Slika 3.1: Diagram sistema

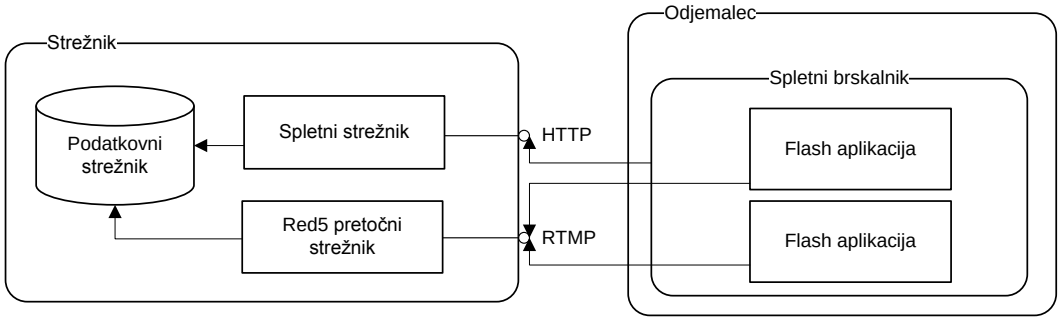
Strežnik je praviloma vedno za usmerjevalnikom in požarnim zidom. Za potrebe komunikacije se sprostijo omrežna vrata po potrebi. Odjemalci in

upravljalci se lahko nahajajo kjerkoli v omrežju (slika 3.1). Z vidika varnosti je v končni izvedbi predviden VPN dostop za vse uporabnike na strani interneta, tako imamo situacijo bolj pod nadzorom in dvignemo stopnjo varnosti z najmanjšim vloženim trudom.

3.1 Celoten sistem z uporabo tehnologije Flash

Jedro sistema so trije strežniki (glej sliko 3.1), lahko tudi fizično ločeni. V našem primeru tečejo vsi trije strežniki na istem stroju. Sami strežniki ne potrebujejo veliko sistemskih virov, količina uporabnikov pa je predvideno nizka. Izjema je omrežni prenos. Predvidenih je 400kB pasovne širine v obe smeri (angl. full duplex) na enega odjemalca. Trije strežniki so podatkovni strežnik MySQL, pretočni strežnik Red5 in Java Tomcat spletni strežnik.

Odjemalci se delijo na upravljalce in stranke. Upravljalci imajo popoln nadzor nad kanali in aplikacijo pri stranki. Stranki se prepreči nadzor nad aplikacijo z zakrivanjem principov delovanja in odvzemom dostopa do naprednih funkcij brskalnika. Stranka ima dostop do spletnega brskalnika le v načinu kiosk ter nima možnosti dostopa do sistemskih bližnjic.



Slika 3.2: Diagram sistema na tehnologiji Flash

3.1.1 Strežniki

Glavna funkcija podatkovnega strežnika je usklajevanje podatkov o uporabnikih ter dostopu do funkcij. Uporablja se za avtorizacijo/avtentikacijo pri video prenosih in na spletni strani. Prav tako ga koristimo za shranjevanje dinamičnih sprememb na spletni strani. Obremenitev je nizka.

Red5 je kombinacija javanskega spletnega strežnika in RTMP pretočnega strežnika. Teče znotraj Tomcat strežnika. Na Red5 strežniku teče aktivna

aplikacija, ki sprejema in posreduje sliko ter zvok na imenskih kanalih. Prav tako omogoča avtorizacijo in avtentikacijo s pomočjo podatkovne baze. Na vsak kanal se lahko prijavi večje število odjemalcev. Možno je vklopiti snemanje posameznega kanala kot .flv datoteko lokalno na strežniku. Nadzor nad kanalom ima snemalna Flash aplikacija pri odjemalcu. Aplikacije na Red5 strežniku so dostopne preko vrat RTMP.

Javanski spletni strežnik Tomcat servira dinamične spletne strani. Na tem strežniku tečeta dve dinamični spletni aplikaciji. Kontrolni podatki se pridobijo iz podatkovnega strežnika. Dostop do aplikacij je prek protokola HTTP. Nadzornik mora za dostop izpolniti prijavni obrazec, ki poskrbi za avtentikacijo in avtorizacijo. Stranke ne potrebujejo avtentikacije, za avtorizacijo pa pošljejo svojo identiteto. Tako ima vsaka stranka dostop do svojega kanala.

Na testnem sistemu smo hkrati poganjali več odjemalcev in strežnik. Strežnik Red5 porablja 300MB sistemskega pomnilnika, ta številka pa se ni spremenila tudi pri štirih odprtih odjemalcih. Prav tako poraba procesorskega časa s strani strežnika ni presegla 5%. Pasovna širina pa seveda narašča linearno za vsak kanal. Obnašanje pomnilnika je tipično za JVM, in sicer ima virtualni stroj rezerviran določen del pomnilnika. Groba primerjava pokaže, da bo na strežniku veliko prej problem pomanjkanja pasovne širine kot pa ostalih sistemskih virov. Pri meritvah se zato posvečamo večjim porabnikom sistemskih sredstev – odjemalcem.

3.1.2 Odjemalci

Odjemalec je lahko vsaka naprava na kateri teče spletni brskalnik s podporo Flash aplikacijam. Testirali smo osebne računalnike z operacijskimim sistemi Windows, MacOS X ter Ubuntu Linux in pri vseh deluje brez posebnih težav. Strojne podpore kodiranju v Flash platforma ne ponuja in je zato Flash rešitev nima. Možen odjemalec je lahko tudi mobilna naprava, v kolikor izpolnjuje zgoraj omejene pogoje (delujoč spletni brskalnik s podporo Flash vtičniku). Prav tako velja to za druge naprave. Slabša zmogljivost sistema pomeni tudi nižjo kvaliteto slike pri video komunikaciji. Predvsem se občuti manjša zmogljivost pri kodiranju slike in manj pri dekodiranju. Posledično vedno sogovornik občuti deficit naprave prej kot tisti, ki jo ima pred seboj. V primeru telemarketinga moramo nujno poskrbeti, da imajo agenti dovolj dobre osebne računalnike.

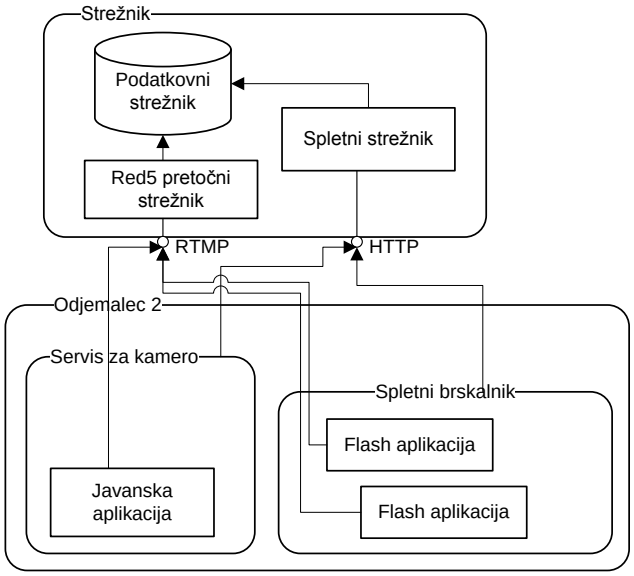
Pri strankah moramo omejiti dostop do sistemskih funkcij. Predviden je brskalnik v načinu kiosk in omejen dostop do tipkovnice s fizičnim kiosk sistemom. Takšen sistem lahko popolnoma omeji dostop do tipk, ponudi zmanjšan obseg tipk ali pa vsebuje zaslon na dotik (slika 3.3).

Slika 3.3: Kiosk brez okrasnega pokrova.

3.2 Sistem z uporabo tehnologije Flash in javanske rešitve za zajem videa

Bistvena razlika med to rešitvijo in prejšnjo je nadomestitev Flash aplikacije za zajem slike s kamere z aplikacijo napisano v Javi.

Java aplikacija omogoča več nadzora nad kodiranjem slike in seveda večji nabor kodirnih algoritmov. Podporne knjižnice za to aplikacijo so JavaCV za zajem slike in Xuggle za kodiranje v RTMP kanal. Aplikacija lahko v vseh funkcijah nadomesti enakonamensko Flash aplikacijo in prav tako teče znotraj spletnega brskalnika. Možna je tudi različica, kjer pustimo obe Flash aplikaciji za prikaz odhodne in dohodne slike – izklopimo zajemanje slike v aplikaciji Flash. Javanska aplikacija v tem scenariju zajema sliko v ozadju. Če uporabimo slednjo izvedbo, potrebujemo še lokalni sistemski servis, ki nadzira to aplikacijo za zajem. Javanska aplikacija teče v okviru servisa in se povezuje direktno preko RTMP protokola. Servis upravljamo preko spletnega servisa v upravljateljevi aplikaciji na spletnem strežniku. Trenutno uporabljamo drugo različico delovanja, kjer je Java aplikacija ločena od brskalnika, saj so določene



Slika 3.4: Diagram sistema na tehnologiji Flash z javansko rešitvijo za zajem slike

težave in knjižnic Xuggle ni možno uporabiti, če Java program deluje znotraj brskalnika.

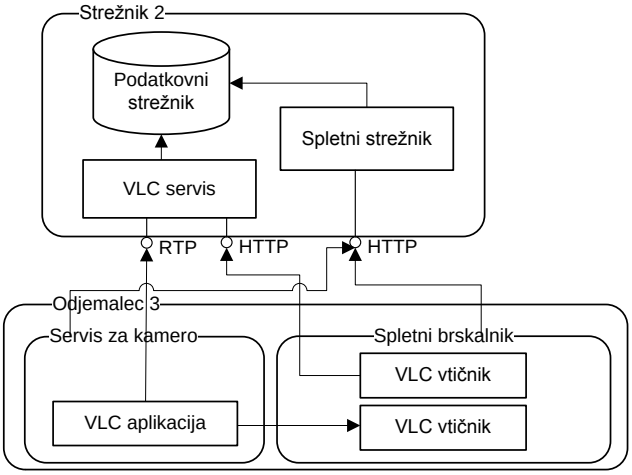
Strežniki v tej rešitvi so identični strežnikom v prejšnji.

3.3 Rešitev s pomočjo predvajalnika VLC

Rešitev s tehnologijo VLC (slika 3.5) je v principu podobna, vendar ima nekaj očitnih razlik v primerjavi s prejšnjimi. Spletni in podatkovni strežnik lahko obdržimo, prav tako večino tehnologije spletnih strani. Red5 strežnika pa v tem primeru ne moremo uporabiti. VLC namreč privzeto ne podpira lastniškega protokola RTMP. Možnosti podpore obstajajo, zahtevajo pa ročno prevajanje VLC kode in dodajanje posebnih modulov. Je pa zato rešitev z VLC elegantnejša. VLC je lahko tudi strežnik, ne samo odjemalec, prav tako pa podpira vrsto različnih načinov kodiranja in nekaj protokolov.

Program VLC je sedaj hkrati program za zajem videa, za prikaz videa in za posredovanje podatkov. Za vsak podatkovni tok imamo na strežniku nov VLC proces, ki skrbi, da pridejo podatki na pravo mesto. Seveda potrebujemo servis, ki se napaja iz podatkovne baze in skrbi, da tečejo pravilni procesi s preprostimi klici iz ukazne vrstice. V tem primeru uporabimo RTP protokol za

podatkovno povezavo, odjemalec mora vedeti naslov strežnika in prosta vrata za povezavo. Na strežniku so potem objavljeni ustrezni kanali prek HTTP protokola.



Slika 3.5: Diagram sistema na tehnologiji VLC

Lokalni zajem s spletne kamere, kodiranje v x264 standardu in pošiljanje podatkov strežniku lahko sprožimo s preprostim ukazom (glej koda 3.1).

```
vlc dshow://
    --dshow-vdev="Logitech␣HD␣Webcam␣C510"
    --dshow-size=800x600
    --dshow-aspect-ratio=4:3
    --dshow-fps=25
    --live-caching=100
    --sout="#transcode{vcodec=h264,vb1024,fps=25,scale=1,venc=x264{tune=zerolatency,
            vbvmaxrate=1024,vbv-bufsize=40,intra-refresh,sliced-threads},acodec=mp3,ab
            =128,channels=2,samplerate=44100}:rtp{dst=ip_naslov_streznika,port=
            vrata_za_sprejem,mux=ts,ttl=1}"
```

Koda 3.1: Primer zagona zajema slike in prekodiranja z VLC. Uporabimo ločljivost SVGA in kodirnik x264.

Za ogled videa, ki ga strežnik posreduje je dovolj, da imamo omogočen brskalniški vtičnik za VLC ter preprosto HTML kodo (koda 3.2). Brskalnik prepozna podatkovni tip *video/mp4* in zažene VLC vtičnik v označenem območju. VLC nato samodejno zazna spletni naslov kot vir pretočne vsebine in ga poskuša prikazati.

```
<embed src="http://ip_naslov_streznika:8080/ime_kanala.ts" type="video/mp4" width="800
        " height="600"/>
```

Koda 3.2: Primer vključevanja videa v spletno stran.

Poglavje 4

Metodologija in orodja primerjave

Določene rešitve so že na prvi pogled slabe in jih lahko zgodaj izločimo. Nekaj očitnih primerov: zelo zrnata slika (ni možno prepoznati osnovnih obraznih potez, tudi kadar je obraz na 1/4 slike), preskakovanje slike ali neenakomerno predvajanje (angl. jitter), očitne napake kodiranja; kot so napačne barve, slabo sinhroniziran zvok in slika ter veliki zamiki zaradi kodiranja. Predvsem zadnji problem je manj očiten na prvi pogled, vendar zelo pomemben. Če preteče veliko časa od trenutka oddajanja do trenutka, ko prejemnik vidi in sliši sporočilo, komunikacija v realnem času ni možna.

Obravnavane rešitve primerjamo na več dejavnikih. Nekatere dejavnike je možno primerjati s pomočjo preproste kvantitativne analize, predvsem gre za sistemska sredstva. Video prenos je bil vedno zahteven problem za strojno in programsko opremo, ki si jo lahko privošči manjše podjetje ali posameznik, zato smo opazovali obnašanje rešitev glede na obremenitev sistemskih parametrov.

Seveda so osebni računalniki že pred desetletjem omogočali video komunikacijo. Vendar pa zaradi omejitev strojne opreme postaja komunikacija s sliko poleg zvoka za uporabnike zanimiva predvsem v zadnjem času. Povečanje ločljivosti iz standarne QVGA pred 10 leti na VGA ter 720p potencira računsko zahtevnost, slika se namreč veča v dveh dimenzijah.

Če uporabnik opazuje sliko od blizu je pomembno, da ne more razločiti posameznih pik, sicer bi slika izgledala zrnato že v najboljši možni situaciji. Še nekaj let nazaj je bila standardna ločljivosti za uporabniške video rešitve QVGA, VGA le redko. Kljub veliko višjim ločljivostim pri predvajanju videa, pri snemanju teh možnosti ni bilo. Zadnjih nekaj let pa so dostopne tudi višje ločljivosti. In predvsem te višje ločljivosti nas zanimajo. Vključili smo tudi

kratko ime	dimenzije		št. slikovnih elementov
QVGA	320	240	76.800
VGA	640	480	307.200
SVGA	800	600	480.000
720p	1280	720	921.600
1080p	1920	1080	2.073.600

Tabela 4.1: Tabela nekaterih standardnih ločljivosti.

tabelo (glej tabelo 4.1) za primerjavo standardnih ločljivosti.

Za testiranje in vrednotenje rešitev smo izbrali ločljivosti SVGA in 720p. Nižje ločljivosti v našem primeru niso primerne, saj imamo velik kiosk in želimo prikazati sliko osebe v naravni velikosti. Za to torej moramo uporabiti čim višjo ločljivost slike, vendar pa smo omejeni tudi s karakteristikami strojne opreme. Čeprav bi bilo zanimivo testirati tudi višje ločljivosti pa so nas lastnosti testne kamere omejile na 720p kot zgornjo mejo.

4.1 Orodja

4.1.1 Pasovna širina: Netlimiter v3, Windows Resource Monitor

Netlimiter sicer ni potreben za samo merjenje prenosa podatkov, za to zadostuje v operacijski sistem vgrajen merilec sredstev (Windows Resource Monitor). Potrebujemo pa Netlimiter za testiranje obnašanja algoritmov pri pomanjkanju sredstev na podatkovni liniji.

4.1.2 Obremenitev CPE: Windows Resource Monitor

Podobno kot pri merjenju pasovne širine lahko filtriramo podatke po procesih, na voljo pa imamo tudi povprečno vrednost, ki jo je možno ročno odčitati.

4.2 Primerjalna anketa

Manjšo skupino uporabnikov (7) smo anketirali glede pripravljenih komunikacijskih rešitev. Pokazali smo jim vse tri rešitve pri vsaki po dve ločljivosti videa. Za vsako od teh kombinacij rešitve in ločljivosti so uporabniki podali po štiri številčne ocene (tabela 4.2).

ocena za	razpon
splošni vtis	1–10
kvaliteta slike v mirovanju	1–5
kvaliteta slike v gibanju	1–5
časovni zamik pri komunikaciji	1–5

Tabela 4.2: Ocene rešitev

4.3 Ostali dejavniki

Pomembni dejavniki, poleg čisto kvantitativno določljivih dejavnikov, so seveda čas, kompleksnost postavitve in izdelave posamezne rešitve. Posledično seveda tudi cena, četudi so rešitve neplačljive ali odprte, čas potreben za vzpostavitev sistema ni nujno zastonj.

Pomembno je premisliti tudi odzivnost rešitev, nalagalne čase in upoštevati celostno izkušnjo tako subjektivno kot objektivno. Zanima nas, katera rešitev bi bila uporabnikom najbolj blizu. Delno so ti faktorji razvidni iz primerjalne ankete, sicer pa poskušamo razložiti razlike tudi opisno.

Poglavje 5

Vrednotenje rešitev

5.1 Nastavitve kodirnikov in optimizacije

S poskusi in opazovanjem smo poiskali nekaj uporabnih konfiguracij za nadaljno bolj podrobno testiranje. Uporabili smo kamero Logitech HD C510. Ostale kamere so se izkazale za slabše rešitve. Tudi slednja je po naši oceni minimalni nivo kvalitete za dosego naših ciljev. Na ločljivosti 720p je že opazna sled gibanja na sliki, več pa kamera ne podpira.

Vse rešitve ponujajo sinhroniziran zvok in sliko brez vidnega zamika. Opazen pa je zamik zaradi kodiranja. Pri kodiranjih H.264 je ta problematičen. Sogovornik dobi sliko in zvok z zamikom ene do pet sekund odvisno od specifične metode in parametrov.

5.1.1 Flash

Parametre za nadzor nad Flash programom (tabela 5.1) smo pri izgradnji tega programa pripravili vnaprej. Kasneje smo s pomočjo nadzorovanih sprememb v teh parametrih opazovali in merili obnašanje Flash rešitve. Podrobneje predstavljeni parametri:

- *Kvaliteta slike* – razen v Flash testu odvisnosti pasovne širine od kvalitete (glej sliko 5.2) smo uporabili povsod 95%. Zanimive so nastavitve od 90 do 99 zaradi precej dobre kvalitete slike in pa 0, saj potem Flash uporabi maksimalno možno kvaliteto še znotraj dovoljene pasovne širine.
- *Pasovna širina* – gre za trdo omejitev pasovne širine, možno je podobno kot za prejšno opcijo, uporabiti 0 za dinamično obnašanje. To smo uporabili pri testu odvisnosti pasovne širine od kvalitete (glej sliko 5.2). Pri

parameter	ime v programu	razpon vrednosti
kvaliteta slike (angl. quality)	quality	0-100 [%]
pasovna širina (angl. bandwidth)	bandwidth	od vkjučno 0 [B/s]
ločljivost (angl. resolution)	width, height	širina, višina; mora podpirati kamera
slik na sekundo (angl. FPS)	fps	več od 0 [slik/s]
ključna slika (angl. keyframe)	keyframe	več od 0 [slik]
medpomnilnik (angl. buffer)	-	nastavljen na 0 v kodi
RTMP naslov (angl. address)	server	rtmp://naslov/kanal

Tabela 5.1: Parametri Flash rešitve.

ostalih testih smo uporabili 400kB/s. Ko smo postopno zmanjševali pasovno širino, se je opazno slabšala kvaliteta slike. Vse kar je pod 200kB/s povroči že zelo vidne kockaste vzorce med gibanjem. Če imamo slabši kanal, lahko zmanjšamo ta parameter do razumne mere in preprečimo popolno zasičenje kanala s strani Flash programa.

- *Ločljivosti* sta SVGA in 720p.

- *Število slik na sekundo* smo nastavili na 25, kar je pogosta standardna vrednost za video posnetke pri nas.

- *Število slik med ključnimi slikami (angl. keyframe) pri kodiranju* uporabili smo 25, kar prinese eno ključno sliko na sekundo. Če zmanjšujemo to vrednost, zmanjšamo potrebno pasovno širino za ceno slabše kakovosti, sploh pri bolj dinamičnih posnetkih.

- *Medpomnilnik* (angl. buffer) nastavimo na 0. Meri se v sekundah medpomnjenih podatkov, večja številka pa pomeni večji časovni zamik med oddajo in prejemom signala. Ena sekunda bi bila sicer še tolerantna, vendar pa ni drugih opaznih učinkov medpomnilnika.

- *Naslov strežnika RTMP* Gre za klasičen spletni naslov, le da je protokol rtmp namesto http. Prav tako velja izpostaviti, da je del za zadnjim / namenjen imenu podatkovnega toka. To velja za pošiljanje in prejemanje, ta del se dinamično spreminja, če imamo več komunikacijskih kanalov.

5.1.2 Java

Kodirnik v Javi je knjižnica FFmpeg prek knjižnice Xuggler. Paket Xuggler namreč vsebuje različico knjižnice FFmpeg, ki podpira RTMP. Knjižnico JavaCV smo izkoristili samo za zajem slike. Obe knjižnici imata sicer precej funkcionalnosti za manipulacijo slike, ki pa jih nismo izkoristili. Pripravili smo zanimive parametre za nadzor nad Java rešitvijo (tabela 5.2).

parameter	ime v programu	razpon vrednosti
kodek (angl. codec)	findEncodingCodec	FLV1 ali H.264
kvaliteta slike (angl. quality)	setGlobalQuality	0-100 [%]
pas. širina (angl. bandwidth)	setBitRate	od vključno 0 [B/s]
ločljivost (angl. resolution)	setWidth, setHeight	širina, višina; mora podpirati kamera
slik na sekundo (angl. FPS)	setFramerate	več od 0 [slik/s]
RTMP naslov (angl. address)	container.open	rtmp://naslov/kanal

Tabela 5.2: Parametri Java rešitve.

- *Kodek* – privzeto izbere okolje kodiranje po enem od kompatibilnih kodekov za format *.flv*, gre za kodirnik po standardu H.263 (v programu je to parameter FLV1). Lahko pa opcijsko zahtevamo kodek družine H.264, v tem primeru se uporabi x264 kodirnik.
- *Ločljivost* – podobno kot za Flash smo uporabili ločljivosti SVGA in 720p.
- *Število slik na sekundo* – uporabili smo 25 in pa 12,5. Prva vrednost je standardna, slednjo smo pa uporabili, da bi zmanjšali zahtevnost kodiranja. Cena za manjše število slik v sekundi pri kodiranju z x264 ni tolikšna, saj kodirnik že tako ne zmore procesirati vseh 25 slik in jih izpušča. Seveda pa se rezultat pozna na slabši kvaliteti gibajoče slike tako ali drugače.
- *Pasovna širina* – nadzor nad pasovno širino pri uporabi teh knjžnic ne deluje vedno po pričakovanjih. In sicer se parameter za pasovno širino obnaša različno pri različnih kodirnikih, prav tako pa številka nima očitne zveze s tem, kar pokažejo instrumenti o realnem stanju. Ne glede na nastavitve nam ni uspelo vplivati na FLV1 kodiranje, vedno smo dobili podobno rabo pasovne širine. Pri x264 kodiranju pa je vhodni podatek v bitih, torej če vnesemo 1 miljon dobimo približno 128kB/s.

5.1.3 VLC

VLC predvajalnik ima še največ parametrov (tabela 5.3), ki jih lahko spreminjamo. Kljub velikemu številu nastavitev pa sploh niso dosegljive vse možne nastavitve kodirnika x264. Za to bi bilo potrebno ustvariti lastno distribucijo iz izvorne kode. Podobno bi bilo možno seveda tudi za javansko rešitev.

parameter	ime v programu	razpon vrednosti
kodirnik (angl. encoder)	venc	x264, ffmpeg, theora
kodek (angl. codec)	vcodec	h264, mp2v, flv1 ...
predpripravljena nastavitev	preset	ultrafast, veryfast, fast, medium, slow ...
nastavitve glede na tip	tune	zerolatency, film, ...
razpršena ključna slika	intra-refresh	-
pasovna širina (angl. bandwidth)	vbX	X je v [kb/s]
ločljivost (angl. resolution)	–dshow-size	AxB, npr. 800x600 mora podpirati kamera
slik na sekundo (angl. FPS)	–dshow-fps in fps	več od 0 [slik/s]
naslov (angl. address)	:http{} ali :rtp{}	v oklepajih so parametri

Tabela 5.3: Parametri VLC rešitve

Parametri kodirnika x264, ki smo jih spreminjali:

- *Kodirnik in kodek* – uporabljali smo kodirnik x264 ter kodek H.264 (v parametru navedemo h264).

- *Predpripravljena nastavitev* – splošna vnaprej pripravljena nastavitev parametrov za kodiranje (angl. preset). Nastavitev smo za teste pustili prazno, vendar pa je možno s tem precej natančno vplivati na porabo procesorskega časa. Pri nastavitvi na *slow* smo imeli tri jedra skoraj polno zasedena, pri *very fast* pa ni bilo čisto zasedeno niti eno jedro testnega štirijedrnika. Vizualna razlika v kvaliteti je manjša in po potrebi si lahko privoščimo hitrejše nastavitve.

- *Posebne predpripravljene nastavitve glede na situacijo* (angl. tune) – uporabili smo *zerolatency* za kodiranje brez zamika.

- *Razpršene ključna slika* – tako nimamo vsakih nekaj slik ene velikanske ampak je ta porazdeljena (angl. intra-refresh). Medpomnilnik lahko zmanjšamo na velikost ene slike (angl. frame).

- *Pasovna širina* – se upravlja prek dveh parametrov, eden je vbX, kjer je X v kilobitih, prav tako pa postavimo velikost medpomnilnika prek parametra *vbv-maxrate* na enako vrednost. Uporabljali smo vrednosti 1024 kb/s za približno 100kB/s in 2048 kb/s za približno pasovno širino 200kB/s.

Kljub vsem nastavitvam za zmanjšanje zamika je le-ta še vedno okrog dve sekundi na strani prejemnika. Pri iskanju nastavitev za zmanjšanje latence smo si pomagali z blogom enega od razvijalcev x264 kodirnika [21], vseh predlaganih parametrov pa nismo našli med možnimi nastavitvami v VLC.

Kodiranje zvoka je lahko AAC, MP3 ali Vorbis (vsebnik ogg), med njimi ni opaznih razlik za končnega uporabnika prav tako pa ni opaznega vpliva na video, tako da smo uporabili kar najbolj znani MP3.

5.2 Anketa med uporabniki

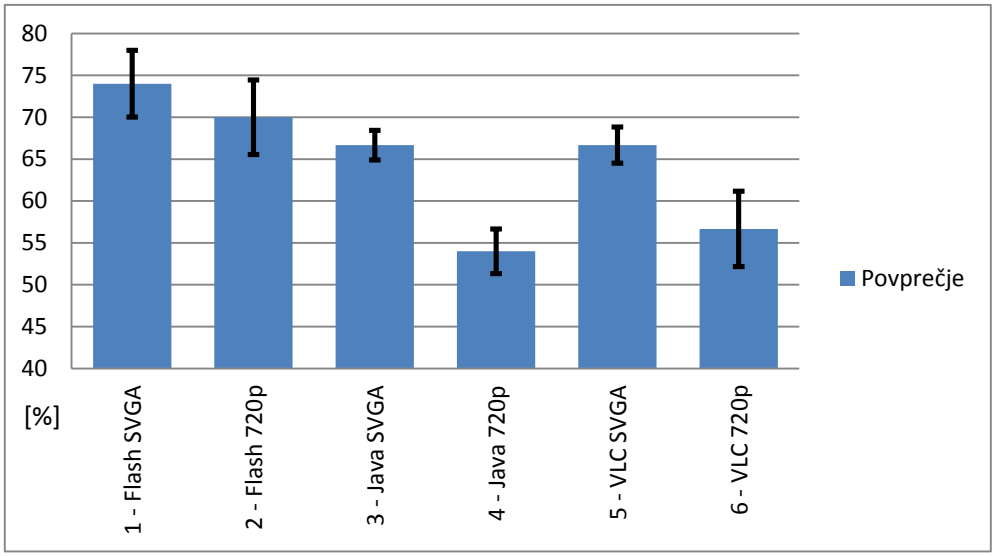
V tabeli 5.4 so pregledno predstavljeni parametri in rešitve uporabljeni pri anketi. Pri javanski rešitvi je pasovna širina izmerjeni približek pri teh nastavitvah, saj nastavitve v programu ni možno direktno pretvoriti v podatek.

	rešitev	ločljivost	kodek	kvaliteta	slike na sek.	pasovna širina
1	Flash	SVGA	FLV1	95%	25	400kB/s
2	Flash	720p	FLV1	95%	25	400kB/s
3	Java	SVGA	FLV1	95%	25	≈450kB/s
4	Java	720p	FLV1	95%	12,5	≈450kB/s
5	VLC	SVGA	x264	veryfast	25	100kB/s
6	VLC	720p	x264	veryfast	25	100kB/s

Tabela 5.4: Nastavitve za anketo.

Vsak sodelujoči je ocenil kvaliteto v štirih točkah:

- splošni vtis 1-10
- časovni zamik 1-5
- kvaliteto mirne slike 1-5
- kvaliteta slike v gibanju 1-5.



Slika 5.1: Rezultati ankete pri uporabnikih v % najvišje možne ocene.

Črne črte na sliki 5.1 označujejo standardno deviacijo. Izračunali smo povprečje vsote ocen vsakega posameznega ocenjevalca, razpon ocen je postavljen tako, da ima splošni vtis večji vpliv na skupno oceno kot ostale ocene. Osebni vtis vsakega uporabnika je zelo pomemben. Če rešitev ne naredi dobrega vtisa na uporabnika je tudi velika verjetnost, da je ne bo želel uporabljati.

Najbolje se je izkazala najbolj preprosta rešitev št. 1, ki je tudi najmanj požrešna z vseh vidikov. Prednost Flash rešitve pred ostalimi je predvsem to, da je že v osnovi rešitev zelo optimizirana. Sicer smo se posvetili optimizaciji ostalih rešitev, vendar pa očitno nismo dosegli nivoja Flash rešitve. Zamiki pri Flash rešitvi so skoraj neopazni, slika je gladka, deluje z najmanj težavami in se zelo dobro dinamično prilagaja situaciji.

Splošno slabšo oceno 720p rešitev in tudi večjo deviacijo ocen pripisujemo predvsem dvem faktorjem. In sicer se je izkazalo, da senzor kamere ni ravno vrhunski, tudi anketiranci so pripomnili v komentarjih, da vidijo zameglitev slike pri gibanju (angl. motion blur). Podobno so pokazale številčne ocene, saj je ocena za gibanje povsod najnižja. Ponekod je bila ocena statične slike boljša v 720p kot SVGA, kar tudi ni presenetljivo. Prav tako so VLC rešitve dobile kar dobro oceno za mirujočo sliko.

Kritika Java rešitve, ki je tudi najslabše ocenjena, je predvsem v požrešnosti in počasnosti. Očitno preveč nivojev vmesnikov in knjižnic pripomore k slabši kvaliteti, prav tako pa nižje število sličic na sekundo pri 720p dodatno poslabša

sliko. Zmanjšali smo ga za rešitev št. 4 zaradi tehničnih težav pri standardni številki.

Pogoste pripombe uporabnikov so bile proti dolgemu nalaganju H.264 kodiranih rešitev ter zamiku slike. Flash in javanska H.263 rešitev tega problema nimata, je pa zelo opazna hitra izguba kvalitete pri teh rešitvah v primeru, da nimamo dovolj širokega podatkovnega kanala.

Prav tako je vredno pripomniti, da se je kamera izkazala pod pričakovanji. Opazno se namreč megli slika ob hitrejšem gibanju pri ločljivosti 720p. Priporočali bi novejši model kamere ali pa nekaj dražje, boljše modele.

5.3 Pasovna širina in procesorska zahtevnost

Pasovno širino smo merili z orodjem Windows Resource Monitor. Orodje je del operacijskega sistema Windows 7. Prav tako smo v manjšem obsegu uporabljali NetLimiter3. Windows Resource Monitor povpreči meritev za zadnjih nekaj sekund. Možno je opazovati pasovno širino po posameznih procesih.

Pri vseh rešitvah so možnosti nadzora nad pasovno širino. Zato smo se pri večini odločili za določene nastavitve in se jih držali med testom z anketo. Kasneje pa smo še preverili rešitve pri različnih pasovnih širinah.

Za vsako meritev smo pustili video rešitev eno minuto preden smo odčitali podatke meritve. Tako se je povprečna pasovna širina ustalila. Rahle anomalije niso izključene, zanimali so nas predvsem trendi. Aplikacije so procesirale tipično predvideno nalogo, in sicer je to človek, ki generira manjše gibe pred statičnim ozadjem.

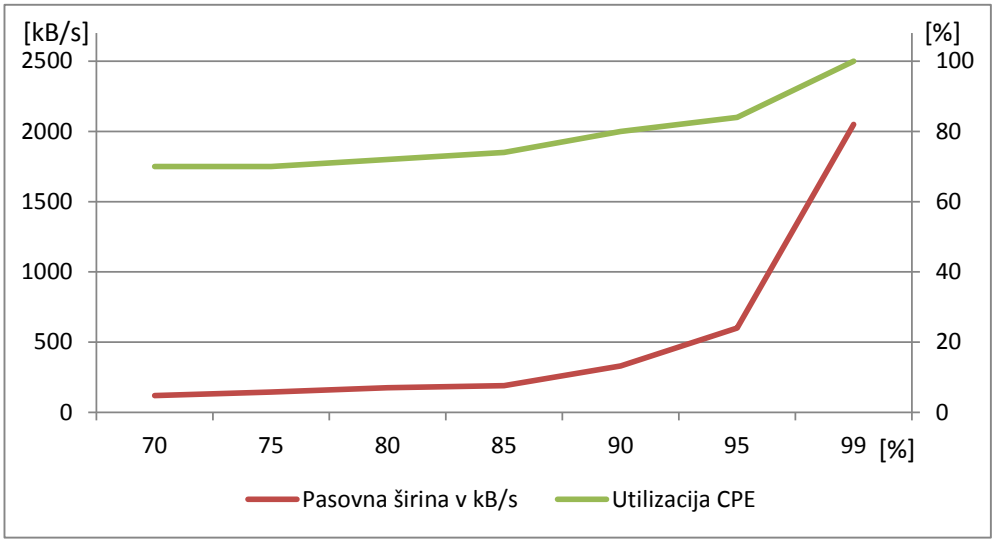
Vsi kodirni algoritmi družine H.263 so manj procesorsko zahtevni. Ti so v rabi pri osnovnem Flash kodiranju. Na ta način nismo dosegli 100% obremenitve na posameznem jedru testnega štirijedrnika.

Algoritmi družine H.264 imajo precej večjo potrebo po procesorskem času, vendar pa v zameno porabijo precej manj pasovne širine, groba ocena bi bila dvakrat več CPE časa in trikrat manj pasovne širine za podobno kvaliteto kot H.263. Z dobrim nadzorom nad pasovno širino lahko prilagodimo kvaliteto podatkovnemu kanalu ter zmogljivosti strojne opreme.

5.3.1 Flash kodirnik

Flash program ustvari .flv video s kodirnikom družine H.263. Značilna je manjša procesorska zahtevnost in velika poraba pasovne širine.

Iz slike 5.2 se vidi neugodna požrešnost kodirnika v našem Flash programu. Potrebno je še dodatno omeniti, da je pri nastavitvi kvalitete pod 90 že takoj



Slika 5.2: Diagram porabe pasovne širine v kB/s in utilizacije enega procesorskega jedra pri kvalitetah kodiranja 70% do 99%.

opazno drastično slabšanje kvalitete slike. Ta graf potrjuje smiselnost naše izbire osnovnih nastavitev za kodiranje videa.

5.3.2 Java kodirnik

Java program ustvari .flv video s kodekom družine H.263 (kodirnik FFmpeg, kodek FLV1) ali H.264 (kodirnik x264).

kodirnik	utilizacija CPE	časovni zamik	pasovna širina
FLV1 SVGA	36%	1 sekunda	500kB/s
FLV1 720p	45%	1 sekunda	750kB/s
x264 SVGA	58%	5 sekund	120kB/s
x264 SVGA	76%	5 sekund	120kB/s

Tabela 5.5: Pregledni prikaz testov Java rešitve.

Na žalost pri tej rešitvi nismo dobili uporabnih rezultatov glede pasovne širine in porabe procesorja, pri bolj uporabnem kodiranju FLV1, namreč upravljanje prek parametrov ni delovalo. Približek porabe pasovne širine in utilizacije CPE sta prikazana v tabeli 5.5.

Pri kodiranju x264 pa je težava najvišja poraba CPE časa in ne preveč tekoče delovanje. Prav tako imamo 5 sekund zamika, kar je enostavno preveč. Dokumentacija za vmesnih Xuggle pa nam na žalost ni zadostovala, da bi rešili te težave.

Nadaljnih testov pri Java rešitvi zato nismo izvajali. Rešitev je še vedno zanimiva, vendar bi bilo potrebno najprej rešiti nekaj izmed mnogih težav.

5.3.3 VLC kodirnik

VLC kodirnik se je izkazal za najbolj fleksibilnega od vseh. Sicer še vedno niso dostopne vse funkcije knjižnice x264, jih je pa dovolj, da precej izboljšamo izkušnjo. Pomembni parametri so vidni v ukazu za zagon VLC zajema (glej kodo 3.1).

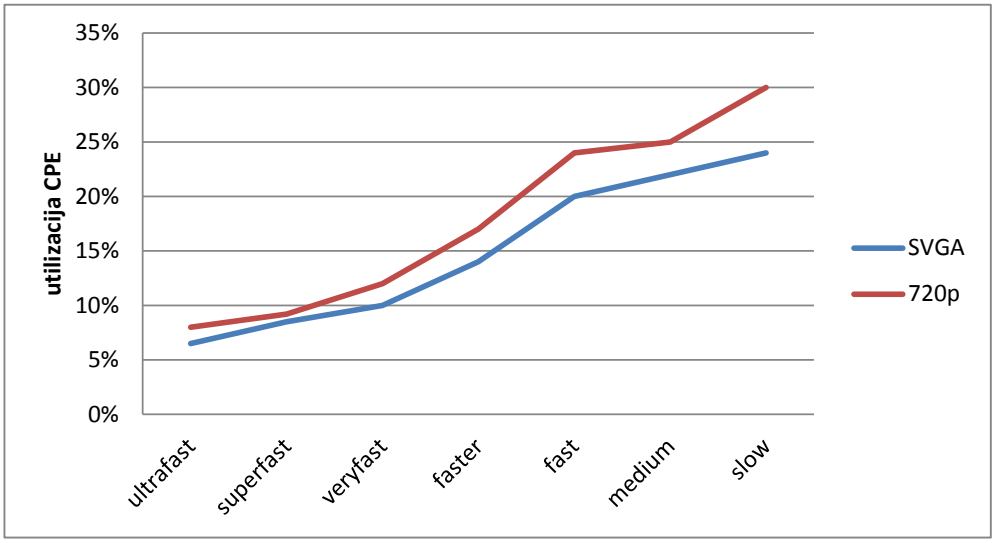
Najbolj vpliva na hitrejši začetek prenosa parameter *-tune zerolatency*, vklop tega parametra povzroči začetek prenosa v roku 2 do 10 sekund namesto 5 do 60 sekund. Prav tako se po vklopu tega zmanjša čas potovanja slike na skupno 2 sekundi iz predhodnih 5.

Prav tako zanimiv parameter je *-intra-refresh*, ki bolj enakomerno razporedi ključne slike v videu. Načeloma bi to naj pomagalo pri zamiku, vendar efekta nismo opazili, razen vidnega potovanja črte svežih ključnih slik prek zaslona kadar je slika rahlo pokvarjena.

Najbolj zanimiv parameter in zelo podoben parametru kvalitete pri Flash kodirniku je *-preset*, ki pove koliko procesorskega časa naj nameni kodirnik sliki. Če imamo težave s preobremenjenostjo CPE in nas ne moti slabša slika ter večji prenos podakov, lahko to zmanjšamo in seveda obratno, če želimo lepšo sliko ali pa boljše stiskanje podatkov.

Če opazujemo diagrama utilizacije CPE (sliki 5.3 in 5.4) vidimo, da se na določenih točkah graf prelomi v bolj strmo krivuljo. Opazno krivulja ni ravno gladka. Na vsakih 25% utilizacije CPE namreč kodirnik začne uporabljati dodatno jedro, posledica česa je nekaj dodatnih izgubljenih CPE ciklov za usklajevanje (angl. overhead).

Pri spreminjanju *-preset* parametra se opazno spreminja kvaliteta slike pri odjemalcih. Seveda pa se spreminja tudi utilizacija CPE. Če poznamo strojne specifikacije odjemalca lahko torej optimiziramo kodiranje za najboljšo kvaliteto. Verjetno lahko določimo lestvico možnih nastavitev, ki bodo delovale optimalno na posameznem stroju. Seveda bi za to potrebovali korelacijo podatkov sintetičnih testov CPE in pa testov kodiranja.



Slika 5.3: Rešitev VLC: CPE utilizacija štirijedrnika pri 128kB/s pasovne širine, na x osi vrednosti parametra *-preset*

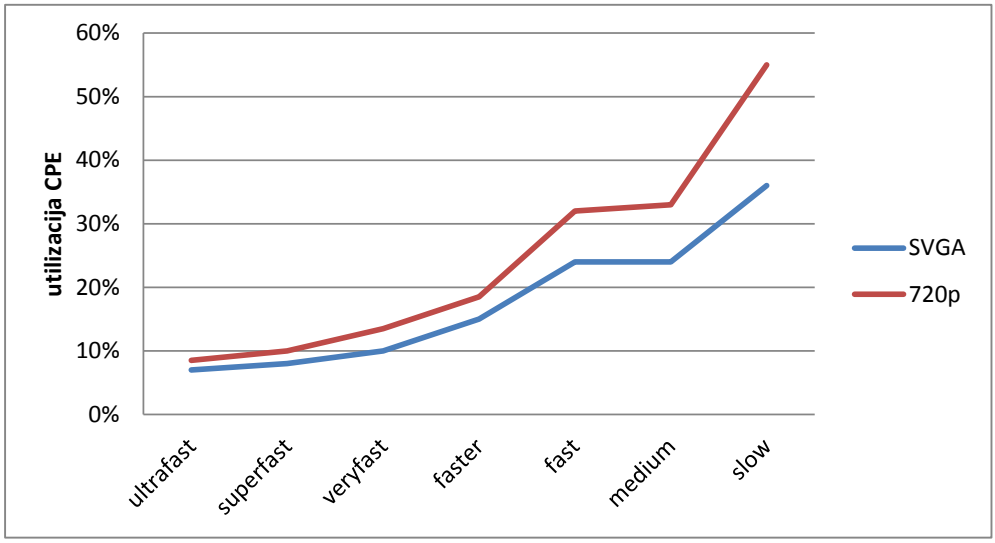
5.4 Časovni zamik pri kodiranju

Video kodiran s H.263 kodirniki ima minimalen zamik (Flash video), pri MPEG2 je zamik večji, največji pa pri H.264 kodirnikih (tabela 5.6).

rešitev	parametri	zamik
Flash	buffer 0, ostalo ne vpliva	ni zamika
Java	kodirnik FLV1	do 1 sekunde
Java	kodirnik H.264	$\approx$ 5 sekund
VLC	kodirnik x264, standardni profil	$\approx$ 5 sekund + nalaganje
VLC	kodirnik x264, optimirani profil	$\approx$ 2 sekundi + nalaganje
VLC	kodirnik mp2v, standardni profil	$\approx$ 2 sekundi

Tabela 5.6: Pregledni prikaz zamikov pri različnih vrstah kodiranja.

Zaenkrat ostaja problem H.264 kodiranja zamik, zaradi medpomnilnikov ter specifike algoritmov kodiranja (iskanje vzorcev v zaporedju slik). Je pa možno zamik s pravilnim krmiljenjem kodirnika v večji meri odpraviti. Naš dosežek na VLC je 2 sekundi zamika pri predvajanju in do deset sekund nalaganja pri odpiranju. Originalno pa smo imeli na VLC dobrih 5 sekund zamika v prenosu in dobre pol minute od zagona predvajanja do prikaza slike (zvok



Slika 5.4: Rešitev VLC: CPE utilizacija štirijedrnika pri 256kB/s pasovne širine, na x osi vrednosti parametra *-preset*

se ponavadi pojavi prej).

5.5 Pregledni prikaz vseh rešitev

Za konec še kratek pregledni prikaz podatkov v tabeli 5.7. Java x264 rešitev anketiranci niso ocenjevali zaradi precej slabega delovanja.

rešitev	util. CPE [%]	čas. zamik [s]	pas. širina [kB/s]	ocena [%]
Flash SVGA	20	0	400	74
Flash 720p	22	0	400	70
Java FLV1 SVGA	36	1	500	66
Java FLV1 720p	45	1	750	54
Java x264 SVGA	58	5	120	-
Java x264 720p	76	5	120	-
VLC x264 SVGA	14	2	120	66
VLC x264 720p	18	3	120	56

Tabela 5.7: Pregledni prikaz rezultatov.

Poglavje 6

Zaključek

Izkaže se, da je Flash še vedno najboljša rešitev v večini primerov. Ne pa vedno. Šibka točka Flash rešitve je pasovna širina. Potrebujemo vsaj 200kB/s kanal za spodobno uporabniško izkušnjo. Če je le možno pa 400kB/s kanal za zelo dobro uporabniško izkušnjo. Procesorsko ni preveč zahteven ne glede na ločljivost slike. Možnosti vključevanja v spletne strani so odlične, prav tako združljivost. Trenutno deluje Flash predvajalnik na največ različnih platformah brez posebnih posegov, potrebno je samo namestiti Flash okolje. Seveda je na mobilni platformi zgodba drugačna, Flash okolje ni na voljo za vse naprave, kjer pa je imamo podobno situacijo kot na namizju in rešitev deluje brez dodatnih težav. Uporabna rešitev s Flash je tudi, če bi imeli manj zaprt sistem za splošno publiko. Flash programe je možno prilagajati in vgraditi vse varnostne mehanizme. Tu je velika možnost za olepšavo in nadgradnje. Problem je predvsem v tem, da so vsi trendi obrnjeni v počasno opuščanje Flash platforme.

Alternativa HTML5 za časa pisanja še nima vseh funkcij Flash okolja, predvsem je problem pošiljanje toka podatkov. Vsi trendi kažejo, da bo v roku nekaj let verjetno HTML5 imel večino Flash funkcionalnosti in takrat bo potrebno to tehnologijo zamenjati. Prav tako je kodirni algoritem vključen v Flash še iz generacije pred MPEG4 in je zato zelo požrešen pri pasovni širini, celo po standardih današnje dobe, ko so linije s pasovno širino 1MB/s in več standard, lahko precej hitro pretiravamo in zapolnimo celotno pasovno širino linije.

Tudi pri vrednotenju bi lahko izmerili še nekaj zanimivih podatkov. Uporaben podatek je količina izgubljenih podatkov, ponavadi gledamo količino izgubljenih slik (ang. frames). Prav tako bi bilo zanimivo izmeriti, kje točno se porajajo neenakomernosti v predvajanju videa (ang. jitter), kodiranje ali kanal.

Rešitev, kjer uporabimo javanski kodirnik za RTMP podatkovni tok je sicer zanimiva ideja. V praksi pa se izkaže za manj uporabno, kot smo predvideli. Zadeva je na žalost prevelika, saj potrebuje ogromno JNI knjižnic, ki so omejene na platformo. Prav tako je to najbolj požrešna rešitev z najslabšim izkupičkom. Če bi vključili v Xuggle svojo verzijo x264 in FFmpeg kodirnikov s posebnimi optimizacijami za živ pretok slike, bi bila situacija verjetno boljša, za ceno dodatnega dela. Želja po JNDI aplikaciji ali javanskemu vtičniku pa je zaradi velikosti in odvisnosti od knjižnic težje izvedljiva ter nepraktična. Ta rešitev je dober nadomestek samo v primeru, če smo res omejeni s pasovno širino, imamo dober procesor in želimo podtakniti javansko rešitev zelo neopazno, saj deluje znotraj osnovnega Red5 sistema in spreminjamo le program za zajem slike.

VLC alternativa je precej bolj zanimiva. Glavna težava je privzeta nepodpora VLC RTMP protokolu. Obstajajo sicer rešitve preko knjižnice Libav [20], vendar je potrebno vložiti precej truda, da ustvarimo svojo distribucijo VLC predvajalnika. Ker nam to ni uspelo dovolj hitro, smo se raje lotili bolj oprijemljive rešitve in ustvarili lasten preprost pretočni strežnik za to platformo. Red5 strežnik lahko zamenjamo z VLC strežniškimi procesi, katere pa krmili preprost servis. VLC se da namreč lepo kontrolirati iz ukazne vrstice. Prav tako lahko bodisi zajema sliko, jo prikazuje ali pa le preusmerja podatkovne tokove od izvora do ponora. Tako dobljena rešitev uporablja x264 kodirnik z nekaterimi izboljšavami za kodiranje pretočne slike [21]. VLC je tako dobra alternativa osnovnemu sistemu že v privzeti distribuciji. Potrebujemo pa seveda za kodiranje po H.264 standardu precej boljšo CPE ali pa se odpovemo kvaliteti slike.

6.1 Možne izboljšave

Če izhajamo iz rezultatov ankete opazimo, da uporabnike moti kvaliteta v gibanju. Po ogledu rezultatov se strinjamo, opazna je sled gibanja in zamegljenje ob hitrejših premikih. Potrebno bi bilo raziskati ali je to predvsem vpliv poceni kamere ali je mogoče razlog v načinu kodiranja. Sicer rezultati niso enaki pri različnih rešitvah, so pa pri vseh karakteristično slabši od splošnega vtisa in mirnosti slike.

Prav tako bi bilo potrebno še skrajšati čas potovanja podatka od zajema slike do prikaza na zaslonu pri rešitvah s kodirnikom x264. Viri [21] nakazujejo možnost izboljšav in optimizacij. Predvsem bi verjetno imeli več možnosti, če bi nam uspelo prevesti lastno različico programa VLC iz izvorne kode. Po-

magalo bi tudi dodatno raziskovanje parametrov in njihovega pomena. Ter seveda testiranje različnih vrednosti.

Flash različica za razliko od ostalih po našem mnenju že deluje zelo dobro in verjetno veliko bolje ne more. Razen seveda, če bi proizvajalec rešitve okolja Flash le-to nadgradil za primer zajema slike, omejitev je namreč v kodirnem algoritmu pri zajemu slike. Predvajalni del deluje brez težav tudi s H.264 kodiranimi tokovi podatkov.

Še ena zanimiva izboljšava bi bila uporaba prilagojenih kodirnikov, ki znajo izkoristiti strojno opremo modernih grafičnih kartic za zelo hitro kodiranje videa. Trenutno ni možno tega koristiti pri Flash rešitvi. Pri x264 kodirniku pa je opazna aktivnost na tem področju, ni pa še rešitev, ki bi jih lahko mi uporabili.

6.2 Priporočena rešitev

V luči zgornjih ugotovitev je naš zaključek, da bo najbolje deloval sistem osnovan na Flash in Red5 tehnologiji v kombinaciji z VLC rešitvijo. Osnova je sicer Flash sistem kateremu lahko dodamo še VLC. Na strežniku lahko VLC in Red5 sobivata, prav tako kot pri odjemalcih. S preprosto nastavitvijo v podatkovni bazi lahko določimo za vsako posamezno stranko, katero tehnologijo bo uporabljala glede na posebne zahteve lokacije. Seveda pa potrebujemo ali dobro linijo, kar je tudi priporočeno ali pa precej zmogljiv sistem.

Tudi iz stroškovnega vidika je ta rešitev najbolj upravičena. Sama vzpostavitev sistema Flash/Red5 je preprosta in ne zahteva veliko prilagajanja. Torej potrebujemo dosegljiv strežnik in Flash snemalnik in predvajalnik. Oboje predstavlja le nekaj vrstic AS kode oz. je možno na Internetu poiskati neplačljive različice.

Slike

Tabele

Literatura

[1] (2012) Polycom HDX družina produktov za video konference. Dostopno na: http://www.polycom.com/content/dam/polycom/www/documents/brochures/hdx-series-family-br-enus.pdf.

[2] (2012) Spletna stran fundacije Eclipse. Dostopno na: http://www.eclipse.org.

[3] (2012) Adobe Flash Builder skupek modulov za okolje Eclipse, ki omogočajo razvoj za Adobe Flash in Flex. Dostopno na: http://www.adobe.com/products/flash-builder.html.

[4] (2012) Spletni brskalnik Mozilla Firefox. Dostopno na: http://www.mozilla.org/firefox/.

[5] (2012) Spletni brskalnik Opera. Dostopno na: http://www.opera.com/.

[6] A. Begen: Watching video over the web: Part 1: Streaming protocols, IEEE Internet Computing, 15 (2), pp. 54-63, 2011.

[7] (2012) Specifikacija protokola RTMP. Dostopno na: http://www.adobe.com/devnet/rtmp.html.

[8] Colin Perkins, RTP, Addison-Wesley, 2003.

[9] (2012) Protokol HTTP, Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Http

[10] (2012) Spletna stran strežnika Red5. Dostopno na: http://www.red5.org.

[11] (2012) Java JDK okolje. Dostopno na: http://www.oracle.com/technetwork/java/javase/downloads/index.html.

[12] (2012) Vtičnik Red5 za Eclipse navodila. Dostopno na: http://www.red5.org/red5-ide-plugin/.

[13] C. Allen, J. Grden: The Essential Guide to Open Source Flash Development, Poglavje 12: Introducing Red5, Friends of Ed, 2008.

[14] A. Thomas, T. Green: Foundation Flash CS3 Video, Poglavje 10: The Camera Object and Flash Video, Friends of Ed, 2008.

[15] (2012) Apache Flex prosto dostopno okolje za razvoj Flash aplikacij. Dostopno na: http://incubator.apache.org/flex/.

[16] (2012) Red5-screenshare je preprosta aplikacija za zajem namizja in prenos slike prek RTMP. Dostopno na: http://code.google.com/p/red5-screenshare/.

[17] (2012) Xuggle je knjižnica, ki omogoča manipulacijo z videom v realnem času in kompresijo le-tega. Dostopno na: http://xuggle.com/.

[18] (2012) JavaCV je vmesnik za knjižnice OpenCV v Javi. Dostopno na: http://code.google.com/p/javacv/.

[19] (2012) Spletna stran predvajalnika VLC. Dostopno na: http://www.videolan.org/vlc.

[20] (2012) Libav knjižnica za pretvorbo multimedijskih formatov, podpira RTMP. Dostopno na: http://www.libav.org/.

[21] (2012) Blog razvijalca x264 kodirnika; Optimizacija kodiranja za pretočni video. Dostopno na: http://x264dev.multimedia.cx/archives/249.