УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

Дејан Стефановић

Софтверско решење система за контролу дигиталних огласних панела и дистрибуцију мултимедијалних садржаја

– М А Г И С Т А Р С К И Р А Д –

Ментор:

проф. др. Никола Теслић

Нови Сад, 2012

Желео бих да се најискреније захвалим ментору проф. др. Николи Теслићу на исказаном стрпљењу, подршци и мотивацији пруженој у току израде ове тезе. Такође се захваљујем свим члановима комисије на поклоњеној пажњи и корисним сугестијама.

Спровођење у реалност комплексног система попут овог описаног у тези је могуће једино као резултат тимског рада, а ја сам имао среће да сарађујем са малим тимом изузетних сарадника, којима се овим путем захваљујем на исказаном залагању и раду у реализацији овог истраживања.

Породица је увек била уз мене спремна да ми упути речи охрабрења и подстрека као и да разуме неопходност данонoћног рада који нас је понекад одвајао. Стога желим да овај рад, који представља заједнички успех, посветим породици.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Магистарски рад
Аутор, **АУ**:		Дејан Стефановић, дипл. инж.
Ментор, **МН**:		проф. др Никола Теслић
Наслов рада, **НР**:		Софтверско решење система за контролу дигиталних огласних панела и дистрибуцију мултимедијалних садржаја
Језик публикације, **ЈП**:		Српски / ћирилица
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2012.
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		7 поглавља / 122 стране / 48 цитата / 22 табеле / 59 слика
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника
Предметна одредница/Кључне речи, **ПО**:		Дигитално оглашавање, дигитални огласни панели, дистрибуција мултимедијалних садржаја, контрола дигиталних огласних панела
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		Ова магистарска теза се бави истраживањем у области дистрибуције мултимедијалних садржаја и контроле дигиталних огласних панела. Циљ тезе је развој решења које је комерцијално применљиво и по функционалностима представља унапређење постојећих решења по питању платформске независности, могућности дистрибуције произвољних садржаја, могућности интеграције са различитим платформама за репродукцију, ефикасности коришћења пропусног опсега код комуникације серверске платформе са клијентским апликацијама и дијагностике уређаја за репродукцију рекламних садржаја. У тези се анализирају релевантни аспекти проблематике, предлаже се решење у складу са постављеним циљевима и користе се објективне мере за мерење перформанси појединих делова решења.
Датум прихватања теме, **ДП**:		22.11.2010.
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	др. Владимир Ковачевић, проф. емеритус
	Члан:	др. Миодраг Темеринац, ред. проф.
	Члан:	др. Мирослав Поповић, ред. проф.
	Члан:	др. Јован Ђорђевић, ред. проф.
	Члан, ментор:	др Никола Теслић, ред. проф.

Потпис ментора

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		MSc Thesis	
Author, **AU**:		Dejan Stefanović, BSc	
Mentor, **MN**:		Nikola Teslić, PhD	
Title, **TI**:		Software solution for digital signage control and multimedia content distribution	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2012.	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7 chapters / 122 pages/ 48 references / 22 tables / 59 pictures	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		Digital Signage, Digital signs, Multimedia content distribution, Digital sign control	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This MSc thesis addresses area of Digital Signage from aspects of control of digital signs and distribution of media contents. Goal of research presented in this thesis is to provide solution that is market-ready and represents step forward compared to existing solutions, offering platform independence, ability to distribute arbitrary media contents, ability to integrate with different digital signs (in broader sense), providing efficient bandwidth use in server-client communication and advanced digital sign diagnostic features. Thesis analyses important aspects of the problem, proposes solution based on given goals and applies objective measures to measure performance of certain parts of the system.	
Accepted by the Scientific Board on, **ASB**:		22.11.2010.	
Defended on, **DE**:			
Defended Board, **DB**:	President:	dr. Vladimir Kovacević, prof. emeritus	
	Member:	dr. Miodrag Temerinac, Professor	
	Member:	dr. Miroslav Popović, Professor	Menthor's sign
	Member:	dr. Jovan Đorđević, Professor	
	Member, Mentor:	dr Nikola Teslić, Professor	

Садржај

СПИСАК СЛИКА

СПИСАК ТАБЕЛА

СКРАЋЕНИЦЕ

SaaS	Software as a Service
OPS	On-Premisses Software
GWT	Google Web Toolkit
JavaEE	Jave Platform, Enterprise Edition
RPC	Remote Procedure Call
EJB	Enterprise Java Bean
WS	Web Service
JWS	Java Web Start
JMS	Java Messaging Service
SOAP	Simple Object Access Protocol
REST	Representational state transfer
AJAX	Asynchronous JavaScript and XML
DOM	Document Object Model
DHTML	Dynamic Hypertext Markup Language
SOA	Service Oriented Architecture
WSDL	Web Service Definition Language
UTC	Coordinated Universal Time
JSON	JavaScript Object Notation
VPN	Virtual Private Network
POPAI	Point of Purchase Advertising International
CRUD	Create, Read, Update and Delete
WSDL	Web Service Definition Language
XML	Extensible Markup Language
CRT	Cathode Ray Tube
BITS	Background Intelligent Transfer Service
RSS	Really Simple Syndication
LGPL	Lesser General Public License
TCP	Transmission Control Protocol
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
LCD	Liquid Crystal Display
CSV	Comma Separated Values
PDF	Portable Document Format
LAN	Local Area Network
ADSL	Asymmetric Digital Subscriber Line
LED	Light-emitting diode

ПОГЛАВЉЕ 1.

Увод

Маркетинг је један од изузетно битних чинилаца модерне економије, који је изузетан узлет доживео крајем 19. и почетком 20. века, пратећи експанзију масовне производње у том периоду. Читав низ медија се користио и још увек се користи у сврхе маркетинга, од штампаних медија, преко радио и ТВ станица, Интернета па до билборда и дигиталних екрана постављених на јавним местима. Крајња намена свих видова рекламирања, упркос различитим медијима коришћеним као носиоцима рекламне поруке, јесте повећање продаје одређених производа и креирање позитивне слике о произвођачу у свести потрошача.

Оглашавање у савременом свету се углавном ослања на дигиталне медијуме за пренос рекламних порука уз тежњу да се оне презентују јасно дефинисаним циљним групама. Један од начина да се успешно допре до циљних група купаца је оглашавање на јавним местима или на продајним местима (енгл. *in-store advertising*). Ова врста оглашавања у почетку се заснивала на штампаним рекламама истакнутим на уочљивим местима, да би се у новије време са ширењем *LCD* технологије рекламе почеле приказивати на великим *LCD* екранима.

Осим циљаног оглашавања, савремени трендови у оглашавању на јавним местима се крећу ка интеракцији са посматрачем, централизованој контроли и дистрибуцији рекламних садржаја и ефективној расподели рекламног времена.

Присуство дигиталних рекламних панела на најразличитијим локацијама, попут бензинских пумпи, супер-маркета, робних кућа и аеродрома поставља пред системе за дигитално оглашавање захтев за флексибилношћу у достави нових рекламних садржаја уз истовремени захтев за поузданом репродукцијом. Могућност удаљене дијагностике уређаја који учествују у приказивању рекламних садржаја је још један од захтева који се поставља пред све сложеније системе за дигитално оглашавање. Прикупљање статистичких података о интеракцији са потенцијалним купцима је начин да се прикупе вредне информације о навикама и потребама потрошача чиме се омогућава побољшање квалитета будућих рекламних кампања и њиховог ефекта на продају рекламираних производа и услуга.

Теза се бави истраживањем у области развоја система за контролу, дистрибуцију и репродукцију дигиталних рекламних садржаја. Циљ тезе је да се развије платформски независтан систем за контролу репродукције рекламних садржаја на удаљеним дигиталним рекламним панелима, који обједињује функције дистрибуције рекламних садржаја, њихове контроле и дијагностике дигиталних рекламних панела уз минималне трошкове експлоатације. Систем треба да задовољи следеће услове:

- Платформска независност централног сервера као и клијентске апликације која контролише удаљене дигиталне рекламне панеле
- Систем мора бити проширив како би се лако могао повећавати број дигиталних рекламних панела које систем контролише
- Рекламни садржај који се дистрибуира може бити произвољан како би се могла постићи компатибилност са постојећим системима за репродукцију садржаја на дигиталним рекламним панелима

У поглављу 2 дат је преглед развоја области система за контролу дигиталних огласних панела и дистрибуцију дигиталних рекламних садржаја, са освртом на проблеме који се постављају пред систем и оквире истраживања.

Поглавље 3 се бави истраживањем досадашњих достигнућа у области. У циљу постављања оквира система анализирани су постојећи производи на тржишту. Базе патената са једне стране представљају веома значајан извор научних информација, док са друге пружају увид у заштићена решења која су

сродна циљном систему. Научна достигнућа из области од интереса такође су истражена у овом поглављу, као и постојећи стандарди.

У поглављу 4 описана су мерења које се користе за оцену квалитета и перформанси појединих делова система. Поједина мерења су коришћена за одређивање технологија најпримеренијих за поједине делове система, поједина су коришћена за одређивање оптималних подешавања одређених подсистема, док одређена мерења обезбеђују информације о границама оптерећења система.

У поглављу 5 дат је детаљан опис система, архитектуре апликације и спрежних подсистема, као и поступака дистрибуције рекламних садржаја и контроле дигиталних рекламних панела путем клијентских контролних апликација.

Поглавље 6 приказује резултате мерења описаних у поглављу 4. Осим приказа резултата, ово поглавље се бави и њиховом анализом везаном за примену у пројектованом систему.

У поглављу 7 дат је закључак истраживања са даљим правцима развоја.

ПОГЛАВЉЕ 2.

ПРЕГЛЕД РАЗВОЈА ОБЛАСТИ ИСТРАЖИВАЊА И ПОСТАВКА ЦИЉЕВА ИСТРАЖИВАЊА

Током историје, знаци у виду натписа, симбола, табли и других објеката су коришћени за упућивање, упозоравање, обавештавање, рекламирање и уопште за упућивање различитих врста порука особама које се налазе у њиховој близини. Још су древни Египћани, Грци и Римљани користили знаке, па је тако познат обичај Римљана да као знак за гостионицу користе жбун. Крст, као симбол хришћанства, коришћен је од давних времена да привуче хришћане, док су симболи попут сунца и месеца коришћени у сличне сврхе код пагана. Као и у другим областима, коришћење знака је током времена еволуирало и добијало нове примене. Једна од значајнијих примена знакова је у рекламирању, као начин привлачења пажње потенцијалних клијената, што је у овој области остао један од основних циљева знакова до данашњих дана.

Почетком коришћења електричне енергије у рекламним панелима може се сматрати 1910. година, када је Француски инжењер, хемичар и проналазач Џорџ Клод (*Georges Claude*) јавности представио прве неонске лампе на сајму у Паризу, да би пет година касније патент светлеће неонске цеви био званично прихваћен од стране патентне организације Сједињених Америчких Држава [PatClaude]. Осим неонских цеви у раним данима електричних реклама су у великој мери коришћене сијалице, да би 1960-тих година *LED* технологија

сазрела за комерцијалну примену и у значајној мери почела да се користи у електричним рекламним панелима. Значајан пробој на пољу *LED* технологије постигнут је 1980-тих година, када је нови материјал, *GaAlAs*, донео и до 10 пута већи осветљај уз значајно мању потрошњу.

Дигитални огласни панели су релативно нова форма рекламних знакова која користи електронске екране за приказивање информација, реклама и сличних садржаја. Претече дигиталних огласних панела датирају из 1970-тих година, када су се појављивали углавном у продајним објектима у виду ТВ пријемника са катодном цеви који приказују садржај са уређаја за репродукцију видео садржаја (енгл. *Video Player*). Пионири у овој фази дигиталног оглашавања су биле модне куће које су снимале своје модне ревије на видео траке које су потом дистрибуиране у продајне објекте и тамо приказиване.

Деведесете године прошлог века доносе значајан пробој на два поља значајна за дигитално оглашавање – развој Интернет рекламирања и значајан напредак у технологији израде тањих екрана веће дијагонале (плазма и *LCD*). До краја деведесетих година прошлог века значајно је повећан број инсталираних равних екрана базираних на плазма и *LCD* технологији, а искуства из Интернет рекламирања почињу да се користе и у области дигиталних огласних панела, пре свега у дистрибуцији и контроли рекламних садржаја. До данашњих дана развијају се различити системи за контролу дигиталних рекламних панела који теже да оптимизују тзв. КДИ циклус – креирање, дистрибуцију и инсталацију рекламних садржаја (енгл. *CDI – Creation, Distribution and Installation* [Schaeffler]). Дистрибуција рекламних садржаја се углавном врши путем Интернета, а приказ рекламних садржаја преко *LCD* екрана који су крајем 2007. године у продаји претекли класичне екране са катодном цеви.

2.1 Поставка циљева истраживања

Ова теза ће идентификовати битне функционалности које систем за контролу дигиталних огласних панела и дистрибуцију дигиталних рекламних садржаја мора поседовати. Теза се усредсређује на систем намењен раду у реалном окружењу са великим бројем просторно удаљених дигиталних огласних панела, уз потребу да се ефикасно и поуздано пренесу велике количине

дигиталних рекламних садржаја и да се непрекидно прати стање свих панела. Главне карактеристике окружења у коме овакав систем функционише су:

- Дигитални рекламни панели су распоређени на удаљеним географским локацијама тако да им није могуће физички приступити
- Комуникација са удаљеним дигиталним рекламним панелима се одвија преко Интернета
- У одређеним временским интервалима дигитални рекламни панели нису повезани на Интернет
- Постоји опасност од неовлашћеног приступа, како дистрибутивно-контролном серверу, тако и клијентским контролним апликацијама
- Толеранција на прекиде у раду система је минимална

Додатне отежавајуће околности у раду система могу бити:

- Грешке у преносу података између система за дистрибуцију и дигиталних рекламних панела (клијената)
- Могућност изненадног прекида везе у тренуцима преноса података
- Преоптерећење централног система за дистрибуцију у случајевима преноса велике количине података великом броју рекламних панела
- Постојање потребе за минимизацијом количине података пренетих између сервера и рекламних панела

Циљ тезе је да предложи решења поменутих проблема које је упоредиво са карактеристикама постојећих решења и као такво и само може постати комерцијално решење.

У оквиру тезе треба проучити постојеће изворе информација (постојећа решења, базе патената и изворе научних радова) како би се поставила одговарајућа ограничења пред систем у погледу:

- проширивости
- поузданости
- понуђених функционалности

За реализацију решења потребно је изабрати платформу која се уклапа у постављена ограничења.

Експериментална потврда тезе биће изведена реализацијом система на изабраној платформи. Мерење перформанси решења биће обављено у складу са постављеном методологијом.

ПОГЛАВЉЕ 3.

ПРЕГЛЕД РЕЛЕВАНТНИХ ИЗВОРА ИНФОРМАЦИЈА

Са циљем адекватног позиционирања теме истраживања, потребно је анализирати актуелна достигнућа у разним доменима. Са становишта ове тезе, сматра се да су следеће области од важности:

- постојећа програмска решења система за контролу дигиталних огласних панела и дистрибуцију дигиталних рекламних садржаја,
- базе патената,
- научна достигнућа и стандарди у области дигиталних огласних панела
- потенцијалне платформе за експерименталну потврду тезе

У даљем тексту дат је преглед релевантних области. Након анализе, утврђују се оквири истраживања са јасним циљем – реализације система за система за контролу дигиталних огласних панела и дистрибуцију дигиталних рекламних садржаја.

3.1 Постојећа решења система за контролу дигиталних огласних панела и дистрибуцију мултимедијалних садржаја

У овом поглављу су анализирана постојећа решења у области система за оглашавање путем дигиталних огласних панела како би се могле уочити њихове основне функционалности и карактеристике и упоредити са онима које систем обрађен у овој тези пружа.

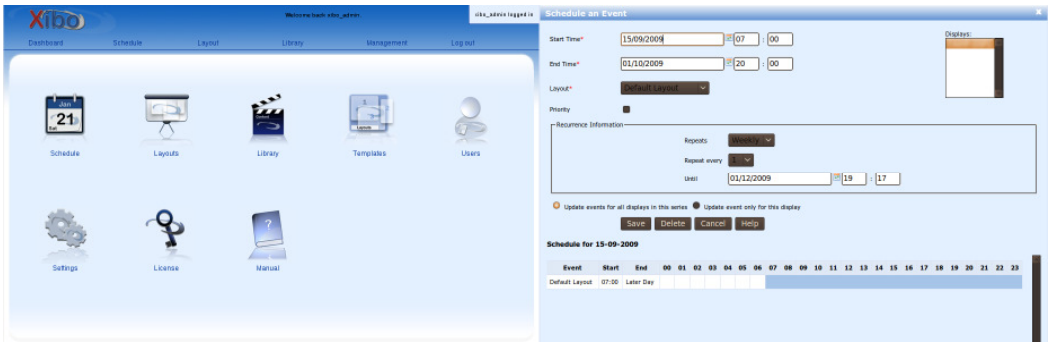
На тржишту постоји низ решења у области дигиталних огласних панела, како бесплатних, тако и комерцијалних. Решења за дигиталне огласне панеле се типично састоје од два дела:

- Део за репродукцију садржаја, који може бити специјализовани уређај физичке архитектуре или програмско решење које се извршава на персоналном рачунару
- Део за манипулацију рекламним садржајима и управљање деловима за репродукцију, који у одређеним случајевима укључује и решења за креирање дигиталних рекламних садржаја. Овај део је најчешће реализован у виду серверске апликације.

Тежиште овог рада је на делу за манипулацију садржајима и управљање дигиталним огласним панелима, тако да ће у овом поглављу акценат бити стављен на решења која припадају датој области.

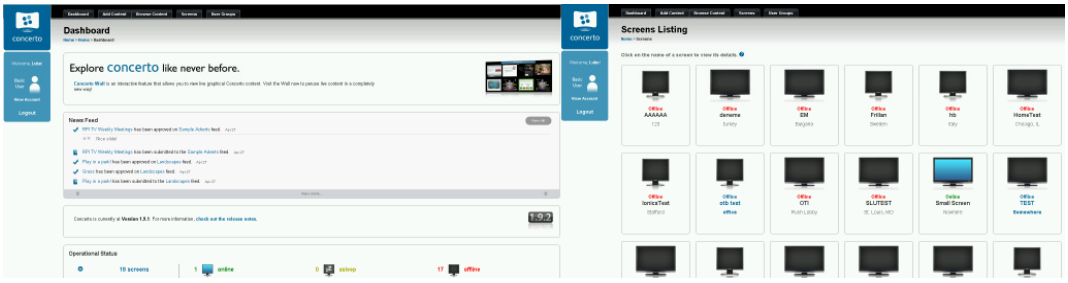
Xibo [ProdXibo] је решење за дигитално оглашавање које се дистрибуира у виду отвореног кода (енгл. *open source*). Овај програм нуди контролу великог броја дигиталних рекламних панела путем централне веб апликације, која обједињује и систем за креирање рекламних садржаја. Централна апликација за контролу система нуди могућност креирања временских распореда приказа презентација (енгл. *scheduling*), уз њихову дистрибуцију на расположиве дигиталне рекламне панеле. Формат презентација које *Xibo* дигитални рекламни панели приказују је специфичан, и може се састојати од слика (*PNG*, *JPG* и *GIF*), текстуалних садржаја, видео садржаја (формати који се могу репродуковати *Windows Media Player*-ом), *Flash* садржаја, веб страница, *Power Point* презентација и *RSS* токова. Серверски део апликације се заснива на *PHP* скрипт језику и могуће га је инсталирати на различитим веб серверима и оперативним

системима. Клијентска апликација је базирана на *.NET* платформи и као таква везана за *Windows* оперативни систем, мада је у току и рад на *Python* верзији која би требала да омогући платформску независност.



Слика 1 - приказ *Xibo* апликације

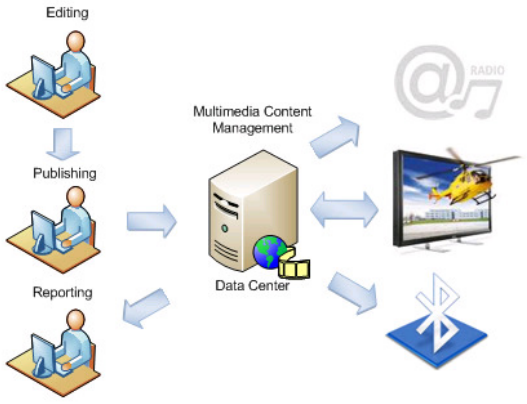
Concerto [ProdConcerto] је бесплатно решење отвореног кода за дигитално оглашавање. Систем подржава искључиво приказ слика (*JPEG*, *PNG* и *GIF* формати), *PDF* докумената и *Power Point* презентација, без могућности рада са видео садржајима. Графички и текстуални садржаји се дистрибуирају путем *RSS* токова, при чему се на сваком дигиталном рекламном панелу може приказивати више токова уз примену одређених тежинских фактора којима се расподељује време приказа. Временско распоређивање приказа садржаја је крајње поједностављено, и састоји се у задавању почетног и крајњег датума приказа сваког садржаја и трајања приказа. Серверски део система је имплементиран у *PHP* скрипт језику и као такав је независтан од платформе, док се клијентски део система извршава на персоналном рачунару, и покреће се са *Linux CD*-а без претходне инсталације на клијентски рачунар.



Слика 2 - приказ *Concerto* апликације

Web Signage [ProdWebSignage] је комерцијално решење које се састоји од апликације за репродукцију дигиталних садржаја, веб сервера за контролу апликација за репродукцију и дистрибуцију рекламних садржаја, алата за

креирање дигиталних рекламних садржаја, као и апликације за контролу уређаја за репродукцију намењене *iPhone* мобилној платформи. Дигитални садржаји који се репродукују на уређају за репродукцију су организовани у виду шаблона (енгл. *template*), у оквиру којих се могу приказивати видео садржаји, текстуални садржаји који се преузимају преко *RSS* токова или сличних динамичких извора, као и мали програми (енгл. *widgets*) који служе за приказ специфичних садржаја попут временске прогнозе, садржаја програма и слично. *Web signage player* је *Windows* апликација која служи за преузимања дигиталних садржаја са *Web signage server*-а, контролу временског распореда репродукције и саму репродукцију. Осим репродукције дигиталних садржаја, *Web signage player* укључује и праћење понашања купаца путем видео камере, као и праћење параметара клијентске платформе и њихово слање на сервер (дијагностика). *Web signage server platform* је централни део система из кога се управља појединачним уређајима за репродукцију или групама уређаја за репродукцију, при чему је систем организован по *SaaS* (енгл. *System as a Service* - систем у виду сервиса) моделу. Сервер представља централни део система, и укључује манипулацију корисницима, садржајима, временским распоредима, контролу стања уређаја за репродукцију. Дистрибуција садржаја са сервера на клијенте реализована је као комбинација *Windows Azure* мреже за доставу садржаја (енгл. *Content Delivery Network - CDN*) на серверској страни и *Windows BITS* сервиса за синхронизацију садржаја [PolizziFontana]. Контрола репродукције и прикупљање информација са уређаја за репродукцију реализована је коришћењем *SOAP XML* спрежног подсистема.



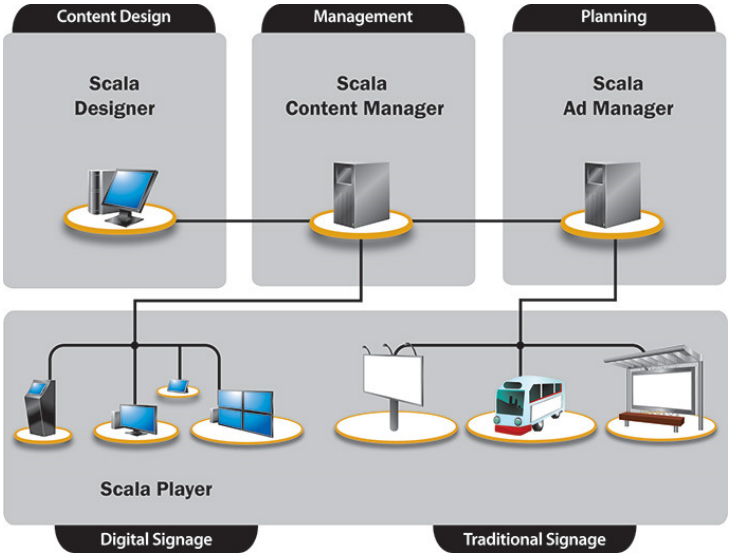
Слика 3 - структура *Web Signage* производа

Scala Connected Signage Network [ProdScala] је комерцијално решење које обухвата комплетан систем за креирање, дистрибуцију и репродукцију дигиталних рекламних садржаја. Ово решење се састоји из низа компоненти приказаних на слици 4 од којих су најзначајније:

- *Designer*, компонента намењена креирању дигиталних рекламних садржаја у *ScalaScript* формату
- *Content Manager,* централна компонента намењена дистрибуцији дигиталних садржаја, креирању временских распореда, контроли репродукције и праћењу стања компоненти за репродукцију. Ова компонента је доступна и као *SaaS* и као *OPS* (енгл. *on premises software* – програм који се инсталира код корисника).
- *Player,* компонента намењена приказу дигиталних садржаја на различитим хардверским платформама, од *LCD* фото-оквира до персоналних рачунара.

Поред подршке за свој заштићени формат дигиталних презентација, ова платформа пружа подршку и за *JavaScript*, *Python* и *VBScript* језике као начин интеграције између програма за репродукцију и других система. Када су дигитални садржаји у питању, *Scala* пружа подршку за широк опсег популарних мултимедијалних формата:

- Слике: *BMP, JPG, GIF, PNG, TIFF*
- Видео: *AVI, SWF, MPG, WMV, X.264*
- Звук: *WAV, MP3*



Слика 4 - елементи *Scala* производа

Преглед постојећих решења указао је на типичну структуру система за креирање, дистрибуцију, контролу и репродукцију дигиталних рекламних садржаја, који се састоји од:

1. Дела за креирање дигиталних рекламних садржаја
2. Централизованог система за дистрибуцију садржаја и контролу репродукције
3. Дела за репродукцију садржаја

Поједини системи обједињују делове 1 и 2, при чему типично уводе специфичне формате за запис рекламних садржаја, што уноси потребу да и део 3 мора подржавати исти формат, чиме се опет смањује модуларност система и могућност комбиновања са производима других произвођача.

Различити производи нуде и различите начине дистрибуције дигиталних садржаја, од коришћења *RSS* токова, преко сопствених решења па до коришћења специјализованих подсистема намењених дистрибуцији садржаја.

Сви разматрани системи располажу неким од видова временског планирања репродукције, било да се ради о изузетно једноставним, што је случај код *Concerto* апликације, било да је реч о напредним планирању [ScalaWP] које подржава *Scala*.

Разматрањем постојећих решења уочено је и да је ***доказ репродукције*** (енгл. *Proof of Play*) битна информација коју системи за дигитално оглашавање морају да обезбеде. Доказ репродукције даје информацију о томе када и у ком трајању

су појединi рекламни садржаји репродуковани, да ли је било грешака у току репродукције и генерално се користе код наплаћивања рекламних услуга.

3.2 Преглед базе патената

[*PatStrand*] описује систем за дистрибуцију и презентацију дигиталних рекламних садржаја који се састоји од централног сервера који управља већим бројем дигиталних огласних панела (клијената). Централни сервер прикупља информације из више извора, које потом шаље клијентима, који од добијених информација селектује оне које су релевантне за локални систем и приказује их. Тежиште овог патента је проналажењу начина да се аутоматизује избор рекламних садржаја који се приказују на страни клијента у складу са локалним параметрима и да се на тај начин повећа ефекат рекламе.

Компанија *Pinpoint Incorporated* патентирала је систем за приказ рекламних садржаја на дигиталним рекламним панелима [PatPinpoint] који садржаје приказује зависно од присутности посматрача. Систем описан у патенту откривање присутности посматрача врши путем бежичних уређаја које посматрачи носе са собом, при чему се исти уређај користи и за добијање информација о профилу посматрача. На тај начин је могуће приказивање садржаја прилагођених профилу конкретних корисника.

Тенденција прилагођавања рекламног материјала профилу купца исказана је и у патенту [PatSonyUK], у коме се користи технологија препознавања лица (енгл. *Face detection*). Овај систем предочава и могућност коришћење више камера у комбинацији са блоком за препознавање лица постављених у различитим деловима продајног објекта. Из скупа расположивих камера, један део се користи за прилагођавање рекламних садржаја купцима, док се други део користи за откривање присутности тих истих купаца у другим деловима продајног објекта (нпр. код касе, тако да се може пратити утицај приказаних реклама на продају). У овом патенту се уочава још један захтев који се поставља испред система за дистрибуцију и приказ дигиталних реклама, а то је потреба за постојањем повратних информација о утицају приказаних садржаја на купце.

Будући да типичан систем за дистрибуцију и презентацију дигиталних рекламних садржаја користи Интернет као везу између сервера и клијената,

постоји више могућих видова комуникације између сервера и клијената. У патентној пријави [PatCoreptronic] представљен је метод комуникације заснован на периодичном повлачењу података (енгл. *pooling*), у коме клијенти у регуларним интервалима контактирају централни сервер, захтевајући команде. Сервер доставља команде клијентима, које они извршавају и потом шаљу резултате извршења серверу. Наведена метода се уједно користи и за одређивање присутности појединих клијената на мрежи.

Још један од битних аспеката система за дистрибуцију и презентацију дигиталних рекламних садржаја је сама организација садржаја који се дистрибуира. У патенту [PatFactorx] описан је систем код кога се дигитални рекламни садржаји организују помоћу шаблона, који се на различитим дигиталним панелима могу приказати на адекватне начине. Систем доноси и организацију већег броја дигиталних панела у канале, дистрибуцији шаблона по каналима и контролу приступа појединим фазама креирања и дистрибуције садржаја.

У патентној пријави [PatChief] описана је специфична топологија мреже за дигитално оглашавање, која осим централног сервера названог медија сервер (енгл. *Media Server*) уводи и локалне сервере (*NMP FS*), који су врста посредника између централног сервера и дигиталних огласних панела. У овој топологији, локални сервер има задатак да комуницира са централним сервером, врши аутентификацију, преузима команде, временске распореде и садржаје и дистрибуира их дигиталним огласним панелима у локалној мрежи. На овај начин се постиже већа ефикасност код дистрибуције садржаја уз мање оптерећење сервера.

Дистрибуција мултимедијалних садржаја на велики број клијената представља велике захтеве пред сервер, а у неким случајевима и клијентске апликације могу имати префериране периоде у којима ваља дистрибуирати садржаје. Патент [PatNCTS] се бави начинима планирања саобраћаја у оквиру мреже за дистрибуцију дигиталних садржаја према унапред планираним временским распоредима, како би се дистрибуција садржаја распоредила на термине када је мрежа мање заузета.

Прегледом базе патената из области дигиталних огласних панела (енгл. *Digital Signage*) уочено је да постоји релативно мали број патената који се баве овом тематиком. Проучени патенти покривају различите делове система за дигитално оглашавање, од опште структуре система, преко начина комуникације између централног сервера и дигиталних огласних панела, преко начина организације рекламних садржаја па до начина за распоређивање оптерећења централног сервера.

Информације дате у патентима су углавном генералне и недовољне за комплетну реконструкцију решења, будући да теже томе да генерализацијом решења покрију што ширу област од интереса. Стога је немогуће ослонити се на патенте као на једини извор информација и потребно је анализирати и резултате научних истраживања у областима од интереса за овај рад.

3.3 Преглед стандарда и научних достигнућа

У овом поглављу су изнесени резултати проучавања релевантних научних радова који се баве било комплетним системом за дигитално оглашавање било појединим његовим деловима. Осим научних радова, поглавље доноси и преглед основних елемената система за дигитално оглашавање које дефинише *POPAI* стандард, једини шире прихваћени стандард у области од интереса.

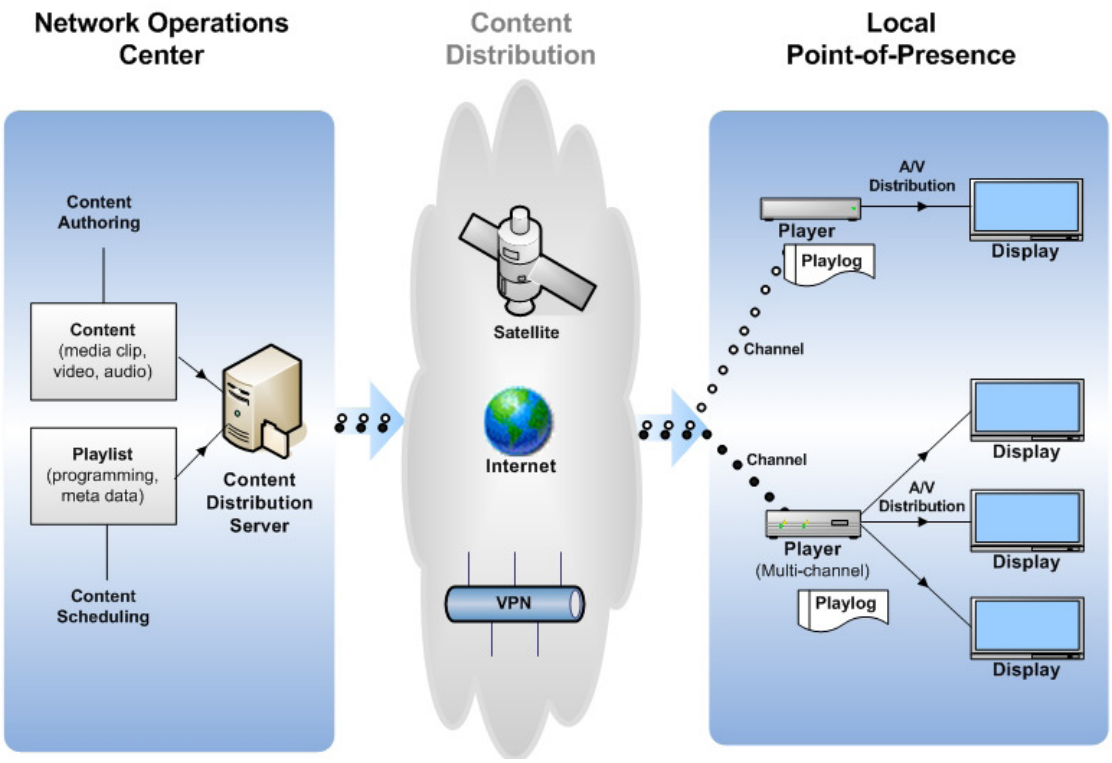
3.3.1 Референтни систем *POPAI* стандарда

У области система за дистрибуцију рекламних садржаја и контролу дигиталних огласних панела тренутно не постоји опште прихваћен стандард који прописује терминологију, протоколе и правила која се примењују на систем. Један од покушаја уређивања ове области представља и низ предлога стандарда *POPAI* групе [POPAI].

Слика 5 приказује структуру референтног система за дистрибуцију рекламних садржаја и контролу дигиталних огласних панела дефинисану у предлогу стандардне терминологије [TermsPOPAI]. Са слике се види да се систем састоји од три целине:

- Дистрибутивно-контролног центра рекламне мреже
- Физичке мреже за дистрибуцију рекламних садржаја

- Одредишне рекламне тачке



Слика 5 - референтни систем *POPAI* групе

У елементе везане за дистрибутивно-контролни центар мреже спадају:

- Алати за креирање мултимедијалних садржаја (енгл. *content authoring*)
- Мултимедијални рекламни садржаји (аудио, видео и било која друга врста мултимедијалних садржаја – енгл. *content*)
- Алати за планирање временског распореда (енгл. *content scheduling*)
- Временски распореди репродукције (енгл. *playlist*)
- Сервер за дистрибуцију садржаја (енгл. *content distribution server*)

Физичка мрежа за дистрибуцију рекламних садржаја је комуникациони канал преко кога се врши достава рекламних садржаја од сервера за дистрибуцију до уређаја за репродукцију на одредишној рекламној позицији, и може се радити о сателитској вези, Интернету, виртуелној приватној мрежи (енгл. *VPN – Virtual Private Network*) или било ком другом комуникационом каналу погодном за пренос дигиталних садржаја.

Елементи одредишне рекламне позиције дефинисани *POPAI* стандардом су:

- Уређај или програм за репродукцију рекламних садржаја (енгл. *player*).
- Репродукциони канал (енгл. *channel*) који представља низ уређаја који репродукују исти рекламни садржај који се на њима синхронизовано мења
- Запис репродукције (енгл. *playlog*) који пружа детаљне информације о репродукованим садржајима и променама параметара система
- Уређај за приказ (енгл. *display*) на коме се приказују рекламни садржаји. Најчешће се ради о некој врсти екрана, али генерално може бити било која врста уређаја која је способна да прикаже рекламни садржај.
- Вишеканални уређај или програм за репродукцију (енгл. *multi-channel player*) је уређај или програм који може истовремено репродуковати више различитих садржаја на више уређаја за приказ

Свака од наведених целина поставља посебан низ захтева пред систем, а самим тим захтева и посебно разматрање адекватних технологија за њену имплементацију.

Осим градивних делова система, слика 5 показује и њихове међусобне спреге, које ће такође бити дефинисане у раду.

3.3.2 Преглед релевантних научних достигнућа

[Harrison] даје доста информација о историји развоја дигиталног оглашавања и пореди га са класичним оглашавањем уз навођење битних предности. Рад се концентрише на систем за размену дигиталних реклама (енгл. *DSE – Digital Signage Exchange*), у коме власници рекламних панела продају рекламни простор купцима заинтересованим да своје рекламне садржаје прикажу на тим рекламним панелима. Представљени *DSE* систем дефинише модел података којим се са једне стране описују захтеви купца око термина приказа, понављања приказа, циљаних локација приказа, док се са друге стране кроз исти модел нуде алтернативне опције приказа уколико оригинални захтеви не могу бити испуњени (нпр. неки од жељених термина за приказ су заузети). На

основу модела приказаном у овом раду развијена је варијација која се у тези користи за временско планирање распореда репродукције рекламних садржаја.

[BozdagEtAl] Даје преглед различитих техника којима се у веб апликацијама постиже брза комуникација са сервером код које корисник има утисак двосмерне комуникације. Рад објашњава ограничења *HTTP* протокола која спречавају иницирање слања података са сервера у веб апликацију и описује методе којима се ова ограничења превазилазе. На крају, рад даје резултате експеримената у којима се пореде перформансе *CometD* библиотеке ([CometD]) са класичном техником периодичног повлачења. *CometD* библиотека имплементира *BAYEUX* комуникациони протокол ([BAYEUX]) којим се постиже симулација трајне везе између веб апликације и сервера која омогућава иницирање слања података од стране сервера (енгл. *server push*). Овај рад је послужио као један од извора информација на основу којих су креиране тест апликације за мерење референтних вредности *Comet* техника за пренос података.

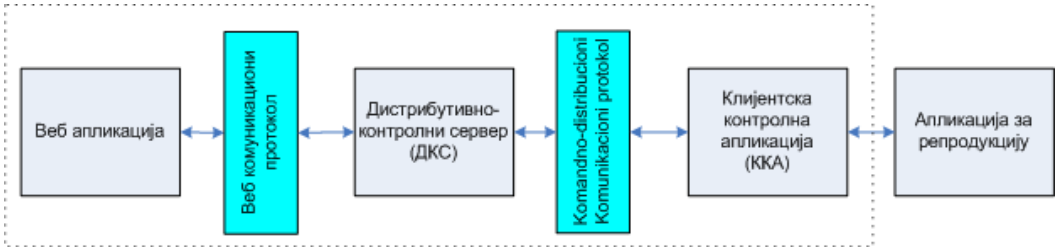
Рад [GibsonRossiter] је разматран као део истраживања могућности да се *SOAP* веб сервис користи за доставу датотека до клијентских контролних апликација. Рад описује могућност преноса датотека у реалном времену (енгл. *streaming*) преко *SOAP* протокола додавањем новог обрасца размене порука (енгл. *MEP – Message Exchange Pattern*). Осим тога, рад описује и коришћење *MTOM* (енгл. *Message Transmission Optimization Mechanism*) за ефикасно преношење датотека посредством веб сервиса.

[Rizzo] описује коришћење *dummynet* [Dummynet] мрежног симулатора за симулирање различитих врста мрежних сметњи. Рад описује начин на који *Dummynet* функционише, могуће примене и ограничења, као и неколико примера његовог коришћења. *Dummynet* је коришћен у мерењима описаним у секцији 4.2.1. за симулирање услова постојања сметњи на комуникационом каналу.

3.4 Преглед могућих платформи за реализацију решења

Захтеви постављени као циљ ове тезе послужиће као критеријуми за избор одговарајућих програмских платформи за имплементацију верификације тезе. Будући да се теза бави сложеним системом, који се састоји од три лабаво

спрегнуте целине (Слика 6), за сваку од ових целина ће бити размотрене технологије које су најбољи кандидати за имплементацију. Осим технологија за имплементацију појединих делова система, у овој секцији биће размотрени и начини спрезања појединих компоненти система. Као додатни фактор код избора најбољих технологија користиће се и њихова интероперабилност, то јест једноставност међусобне интеграције. Поређењем блок-дијаграма приказаног на слици 6 са референтним системом приказаним на слици 5, уочава се постојање клијентске контролне апликације као посредника између серверске платформе и апликације за репродукцију, што представља једну од специфичности решења изнетог у овој тези, које ће бити детаљније образложено у поглављу 5.



Слика 6 - основни блокови пројектованог система

3.4.1 Платформа за развој дистрибутивно-контролне апликације

Серверска апликација представља језгро система које прима команде од корисника преко веб апликације и сходно њима управља удаљеним клијентским контролним апликацијама, који добијене команде на адекватан начин достављају апликацијама за репродукцију. У сваком тренутку, ДКС мора бити у стању да опслужује велики број веб апликација и да комуницира са великим бројем ККА. Осим тога, ДКС мора бити у стању да прихвати, ускладишти и достави рекламне пакете који се састоје од великог броја датотека које саме по себи могу бити изузетно велике (нпр. промотивни видео материјали високе резолуције). Сви ови захтеви представљају велико оптерећење за серверску платформу, која мора бити у стању да се проширује сходно расту оптерећења услед повећања броја клијената (како веб апликација, тако и ККА).

Серверска платформа такође мора да омогући и лако спрезање са клијентским апликацијама, при чему је пожељно да подржава што већи број стандардних комуникационих протокола како би се могао направити избор

комуникационог протокола који на најбољи начин спреже две стране. Уобичајен начин складиштења података у комплексном систему оваквог типа је коришћењем базе података, тако да серверска платформа мора подржавати рад са базама података. Аспекти сигурности, манипулације правима корисника и подршке трансакционог извршења задатака такође су неизбежни захтеви који се постављају пред овакав систем.

Узевши у обзир све претходно наведене захтеве, *Java EE* (енгл. *Java Enterprise Edition*) се поставља као логична платформа за имплементацију серверске апликације. *Java EE* се састоји од окружења (*Java EE* контејнер) и низа *API*-ја дизајнираних за развој комплексних, вишеслојних, проширивих, поузданих и сигурних *Java* апликација [JavaEE]. *Java EE* контејнер изабран за израду експерименталне потврде ове тезе је *JBoss* апликативни сервер [Jbossas], верзија *6.1.0 Final*. Ова платформа изабрана је због повољне дозволе за коришћење (*LGPL*), једноставности инсталације на различитим оперативним системима (током рада на тези равноправно су коришћене инсталацији на *Windows* и *Linux* оперативним системима), доброг сета развојних алата (*JBoss tools Eclipse* проширење) као и богата документација и подршка заједнице која учествује у развоју апликативног сервера. Скалабилност *JBoss* апликативног сервера обезбеђена је подршком за једноставно повезивање више инстанци сервера у кластере (енгл. *clustering*) [Jbossclust], што је још један од разлога за избор ове платформе. Обзиром да *Java EE* платформа дефинише *API*-је које све *Java EE* компатибилне платформе морају имплементирати, прелазак на неки други *Java EE* контејнер био би једноставан, уз прилагођавање конфигурационих параметара новој платформи.

3.4.2 Платформа за развој веб апликације

Веб апликација је основни алат преко кога корисници приступају систему и управљају његовим радом. Савремене веб апликације заснивају се на *AJAX* технологији [AJAX] која представља унапређење *DHTML* технологије увођењем асинхроне комуникације са сервером [DOMScripting]. Код *AJAX* технологије, апликације се састоје од серверског и клијентског дела, при чему се клијентски део извршава у веб претраживачу као *JavaScript* апликација који манипулише

деловима странице, ослањајући се на *DOM* (енгл. *DOM - Document Object Model*) репрезентацију странице. Комуникација са серверском апликацијом која се може базирати на произвољној технологији, одвија се путем асинхроних упита који се одвијају у позадинским процесима и не захтевају поновно учитавање странице сваки пут када је потребно разменити податке са сервером, чиме се постиже бржи одговор апликације на акције корисника и смањена употреба пропусног опсега сервера. Данас постоји велики број технологија које су у основи *AJAX*, али нуде програмеру велике библиотеке готових компоненти које значајно убрзавају време развоја апликација. Избор технологије која ће се користити за реализацију прототипа решења тезе заснива се на примени три битна критеријума на неке од најкоришћенијих *AJAX* платформи (Табела 1).

Технологија/ Критеријум	Компатибилност са веб претраживачима	Развојна окружења	Спрезање са серверском апликацијом	Укупна оцена
Ручно писани *AJAX*	0	2	0	2
GWT	2	2	2	6
Dojo	2	2	1	5
jQuery	2	2	1	5

Табела 1 - поређење карактеристика *AJAX* платформи

Први коришћени критеријум за квалитет *AJAX* платформе је компатибилност са веб претраживачима. Осим ручно писаног *AJAX* програма, остале платформе кроз свој *API* нуде приступ ресурсима претраживача уз решење проблема компатибилности садржано у самој имплементацији *API*-ја.

Развојна окружења су у различитим облицима доступна за све платформе, тако да у овој области ни једна платформа није фаворизована.

Спрезање са серверском апликацијом је тачка у којој *GWT* (енгл. *Google Web Toolkit*) платформа добија предност у односу на *Dojo* и *jQuery* платформе. Иако *Dojo* и *jQuery* имају развијен *API* за асинхрону комуникацију са серверском страном апликације, обзиром на избор *Java EE* серверске платформе образложен у секцији 3.4.1, *GWT* платформа, која је такође базирана на *Java* програмском језику, нуди бољу интеграцију са серверском страном путем *RPC* позива. Осим тога, коришћење *Java* програмског језика на обе стране омогућава дељење програмског кода између серверске и клијентске апликације, чиме се развој комплетног система оптимизује у смислу тестирања и времена имплементације.

3.4.3 Платформа за развој клијентске контролне апликације

Клијентска контролна апликација прима команде и садржаје од серверске апликације и на основу њих контролише апликацију за репродукцију рекламних садржаја. Ова апликација се најчешће мора налазити на истом уређају као и апликација за репродукцију рекламних садржаја и сходно томе мора поседовати висок степен платформске независности. Платформска независност је неопходна како би решење предложено у тези могло да се примени у што ширем спектру уређаја. Као опције програмских језика за развој клијентске апликације у овој тези су разматрани:

- *C#*
- *C/C++*
- *Java*

Захтеви који се постављају пред клијентску апликацију нису критични ни у погледу перформанси, комплексности, нити постоји потреба за блиском интеграцијом са циљном платформом. Због тога коришћење језика *C* или *C++*, који би у случају претходно наведених захтева представљали најбољи избор али и довели до делимичне платформске зависности, није дошло у обзир. Иако за *C#* постоје програмске платформе које омогућавају висок степен преносивости програма на разне оперативне системе ([Mono] спада у најбоље прихваћене платформе), ни једна од њих није опште-прихваћена, уз додатно постојање претњи по легалност оваквих платформи, имајући у виду бројне патенте којима је *C#* платформа заштићена. *Java* платформа данас важи за *de-facto* стандард подржан од стране већине значајних оперативних система, а будући да се и остатак система заснива на овој платформи, *Java* представља логичан избор за имплементацију клијентске контролне апликације.

3.4.4 Веб комуникациони протокол

Комуникација између веб апликације и серверске апликације са собом доноси низ проблема које треба на адекватан начин решити, а који проистичу из ограничења која доноси традиционални модел комуникације између веб претраживача и сервера. Класичан вид комуникације подразумева да веб претраживач увек иницира комуникацију као и да сервер не чува стање између

два сукцесивна позива (тзв. *RESTful* начин комуникације). Додатно ограничење потиче из *HTTP 1.1* [HTTP1.1] спецификације и налаже да један веб претраживач ни у једном тренутку не би требао да има више од две истовремено успостављене везе са истим сервером. Са друге стране, постоји реална потреба да се подаци достављају од стране сервера према веб клијенту, што поменута ограничења додатно отежавају. Овај рад као додатни захтев доноси потребу да се минимизује количине пренетих података између централног сервера и клијентских апликација у које спада и веб апликација. Стога разматрање комуникационог протокола у овом случају обухвата две целине:

- Комуникацију иницирану од стране веб апликације
- Комуникацију иницирану од стране серверске апликације

Избор *GWT* платформе за развој веб апликације намеће као најбоље решење за комуникацију иницирану од стране веб апликације *GWT RPC* механизам. Овај механизам је специфичан за *GWT* платформу и користи начин преноса података који не одговара ни једном постојећем стандарду, али доноси неколико предности које га чине најбољим избором за комуникацију са сервером:

- Протокол омогућава позивање функција на страни сервера из веб апликације (енгл. *RPC – Remote Procedure Call*)
- Једноставно програмирање – програмер и на страни веб и на страни серверске апликације ради са објектима, без потребе да их додатно серијализује у *XML* или *JSON* формат ради преноса
- И веб и серверска апликација се развијају у *Java* програмском језику
- Рад са грешкама и сигурносни аспекти комуникације су транспарентно подржани од стране платформе
- Протокол компресује податке на доста ефикасан начин што минимизује коришћење пропусног опсега

За комуникацију иницирану од стране серверске апликације и усмерену ка веб апликацији се типично користи термин *Comet* [COMET], мада је у употреби и неколико других термина, као што су „*Ajax Push*“, „*Reverse Ajax*“, „*Two-way-web*“, „*HTTP Streaming*“ и „*HTTP server push*“. Карактеристично за све *Comet* имплементације је да се ослањају на привидно трајне конекције иницијално

успостављене са стране веб апликације. Постоје три основне технике остваривања везе са сервером.

Прва је техника скривеног *iFrame* елемента, који се додаје на страни веб апликације и као извор му се поставља серверска страница која доставља поруке намењене веб апликацији. Страна која се учитава у *iFrame* је декларисана као *multi-part* одговор и садржи страницу са низом *script* тагова која се никада не завршава. Техника се ослања на чињеницу да се *HTML* страница приказује инкрементално и да се сваки *script* таг извршава чим стигне са сервера, тако да сервер поруке доставља клијентској *JavaScript* апликацији тако што доставља *script* таг који садржи позив повратне функције главне странице са садржајем поруке као једним од параметара. Уколико нема корисних порука, сервер периодично доставља *script* тагове који садрже позив повратне функције за одржање везе, како би клијентска страна имала информацију о томе да је комуникациони канал и даље валидан. Највећи недостатак ове технике немогућност поузданог руковања грешкама као и немогућност праћења стања упита.

Друге две технике познате као *HTTPXMLRequest long pooling* и *Script tag long pooling* функционишу на сличном принципу. Код ових техника, *JavaScript* на начин који одговара датој техници позива серверску страницу, која уколико има података за клијентску страну одмах враћа одговор (формат и у овом случају зависи од технике), а уколико одговора нема, страница блокира позив до пристизања поруке или истека предвиђеног времена (енгл. *timeout*). Техника дугог повлачења путем *HTTPXMLRequest* позива користи *HTTPXMLRequest* технику за асинхроне позиве сервера, а одговоре добија у произвољном облику. Техника дугог повлачења помоћу *Script* тага асинхрони позив постиже креирањем *Script* тага коме се као страница са изворним кодом поставља серверска страница за руковање порукама. Ова страница корисне податке враћа у виду *JavaScript* изворног кода који садржи позив повратне функције на главној страници са подацима (поруком) као параметром. Предност ове технике је у томе што омогућава комуникацију између различитих домена (енгл. *Cross Domain Communication*).

Поруке које се размењују између серверске и клијентске стране најчешће су у *JSON* [JSON] формату. *JSON* је отворени стандард за текстуалну репрезентацију једноставних структура података (објеката) и асоцијативних низова. Иако је независтан од језика у коме се користи, типично се везује за *JavaScript* код кога је *JSON* део језичке синтаксе. Предност коришћења *JSON* нотације за пренос података је поменута подршка од стране *JavaScript* језика и компактан запис.

За потребе ове тезе, разматране су две библиотеке које омогућавају комуникацију између веб апликације и серверске апликације разменом порука заснованом на моделу пошиљалац/претплатник (енгл. *publish/subscribe*). Прва разматрана библиотека је развијена од стране *Dojo* фондације [Dojo] под називом *CometD* и нуди подршку за *Dojo* и *jQuery JavaScript* библиотеке. Осим саме имплементације *Comet* комуникације, аутори *CometD* библиотеке стоје иза покушаја дефинисања стандарда за ову врсту комуникације [BAYEUX], познатог као *Bayeux* протокол. Друга разматрана библиотека је *JBoss Errai*, развијена од стране *JBoss* заједнице [Errai]. Ова библиотека представља проширење *GWT* платформе које омогућава транспарентно и једноставно додавање подршке за комуникацију засновану на размени порука али и *RPC* комуникацију у *GWT* апликацију коришћењем анотација. Обзиром на претходно образложен избор платформи за имплементацију веб апликације као и избор серверске платформе, *Errai* платформа је изабрана за експерименталну потврду резултата ове тезе.

3.4.5 Контролно-дистрибутивни комуникациони протокол

Приликом избора комуникационог протокола између серверске апликације и клијентске контролне апликације, у обзир су узета следећа ограничења:

- Комуникациони протокол мора да буде такав да се лако може користити и за друге апликације
- Клијентска контролна апликација може бити иза заштитног програма (енгл. *firewall*) или *proxy* сервера
- Имплементација мора бити што једноставнија
- Комуникација је асинхрона и заснива се на командама и одговорима на команде који не морају бити достављени одмах по пријему команде.

Као потенцијалне технологије за комуникацију разматрани су:

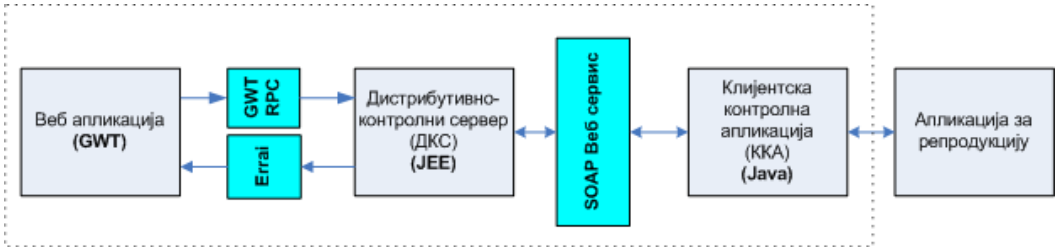
- *JMS*
- *SOAP* веб сервиси
- *RESTful* веб сервис

Иако *JMS* задовољава све функционалне захтеве, познато је да постоје проблеми у раду *JMS* клијената у мрежном окружењу иза заштитних програма или *proxy* сервера. Релативно компликовано конфигурисање мрежних параметара за несметан рад *JMS* клијента и реална могућност да сигурносна политика компаније у чијој се мрежи клијентска апликација инсталира не дозвољава потребне промене у конфигурацији мреже, разлози су због којих *JMS* није изабран као комуникациони протокол.

Са друге стране, и *SOAP* и *REST* веб сервиси задовољавају функционалне захтеве, уз додатну предност коришћења стандардног *HTTP* комуникационог канала који не подлеже посебним мрежним рестрикцијама. Потреба да се комуникациони канал што брже и лакше развије довела је до избора *SOAP* веб сервиса, који се захваљујући мноштву алата веома лако развија и користи. Постојећи алати омогућавају да се методе било које *Java* класе једноставно изложе као веб сервиси, за које се при томе аутоматски дефинишу *WSDL* документи. Тако генерисани *WSDL* документи омогућавају аутоматско генерисање клијентских класа, код којих се комуникација са веб сервисом своди на позивање метода генерисаних класа. Иако и *REST* базирани веб сервиси функционишу на истом принципу, број расположивих алата и подржаних развојних платформи је мањи, што је последица чињенице да је овај протокол релативно скоро почео да стиче већу популарност.

3.4.6 Преглед изабраних технологија

Анализом постављених захтева за имплементацију типичног система за контролу дигиталних огласних панела приказаног на слици 6 и расположивих технологија, изабране су технологије које ће се користити за имплементацију градивних блокова система, приказане на слици 7.



Слика 7 - преглед технологија коришћених за реализацију тезе

Сви блокови система се ослањају на *Java* платформу, што омогућава коришћење јединственог сета развојних алата за развој и тестирање циљног система. Још једна од предности коришћења јединствене платформе је и могућност коришћења заједничке базе програмског кода у различитим модулима система, чиме се скраћује време развоја система и смањује комплексност система.

ПОГЛАВЉЕ 4.

ПРЕГЛЕД МЕРА ЗА ОЦЕНУ ПЕРФОРМАНСИ РЕШЕЊА

Мерења описана у овом поглављу намењена су одређивању битних параметара комуникационих канала описаних у секцијама 3.4.4 и 3.4.5. У мерене параметре спадају ефикасност коришћења пропусног опсега, брзина комуникације као и проузроковано оптерећење система.

4.1 Поступак мерења параметара комуникације између веб апликације и дистрибутивно-контролне апликације

Циљ мерења описаних у овом поглављу је поређење захтевности појединих библиотека за комуникацију између веб апликације и серверске платформе по питању потрошње пропусног опсега, као и поређење перформанси сваке од техника. Део обављених мерења има за циљ постављање генералних референтних вредности за технологије на којима се коришћене библиотеке заснивају.

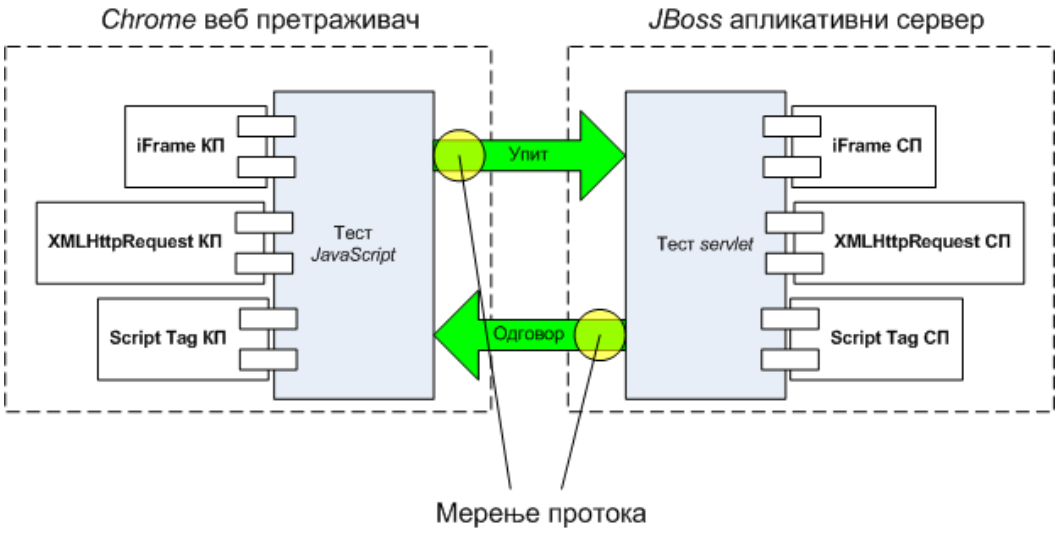
4.1.1 Референтне вредности базних *COMET* техника за пренос података

Први циљ ове групе мерења је да одреди у којој мери начин имплементације канала за пренос података утиче на укупну количину података пренету између сервера и веб клијента у случајевима када нема преноса корисних података – тј. поређење количина података које се пренесу у циљу одржавања отвореног комуникационог канала између веб клијента и сервера. Осим утицаја комуникационог канала, ово мерење ће истражити и начин на који коришћење компресије података на серверској страни утиче на количину пренетих података. Други циљ обављених мерења је утврђивање разлике у брзини преноса података од серверске апликације до крајњег корисника у веб апликацији. Као и у претходном случају, и ово мерење ће узети у обзир утицај компресије података на серверској страни на мерене вредности. У мерењу ће бити упоређене три најчешће коришћене технике које се користе у имплементацијама *COMET* модела комуникације. Резултати се могу применити и на *RPC* комуникацију која је делимично коришћена у решењу описаном у овом раду. Разматране технике комуникације су:

- Пренос података кроз скривени *iFrame* елемент
- Пренос података путем *HTTPXMLRequest*-а са дугим повлачењем (енгл. *long pooling*)
- Пренос података убацивањем *script* тага са дугим повлачењем

4.1.1.1 Мерно окружење и алати

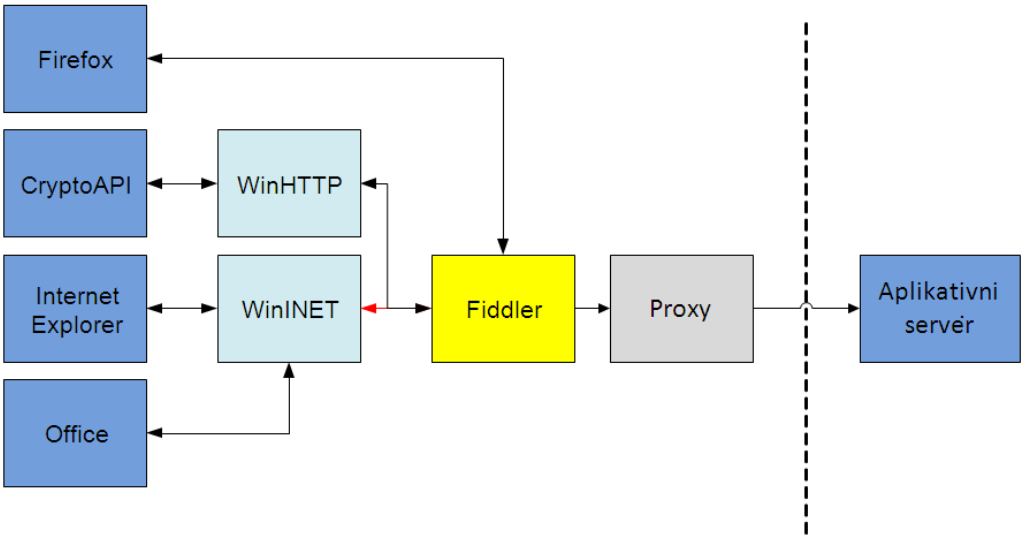
За обављање описаних мерења, развијена је посебна мерна апликација које имплементира основне референтне технике преноса података и додатну логику потребну за извршење мерења. Слика 8 приказује структуру коришћеног мерног окружења.



Слика 8 - начин мерења протока основних техника преноса података

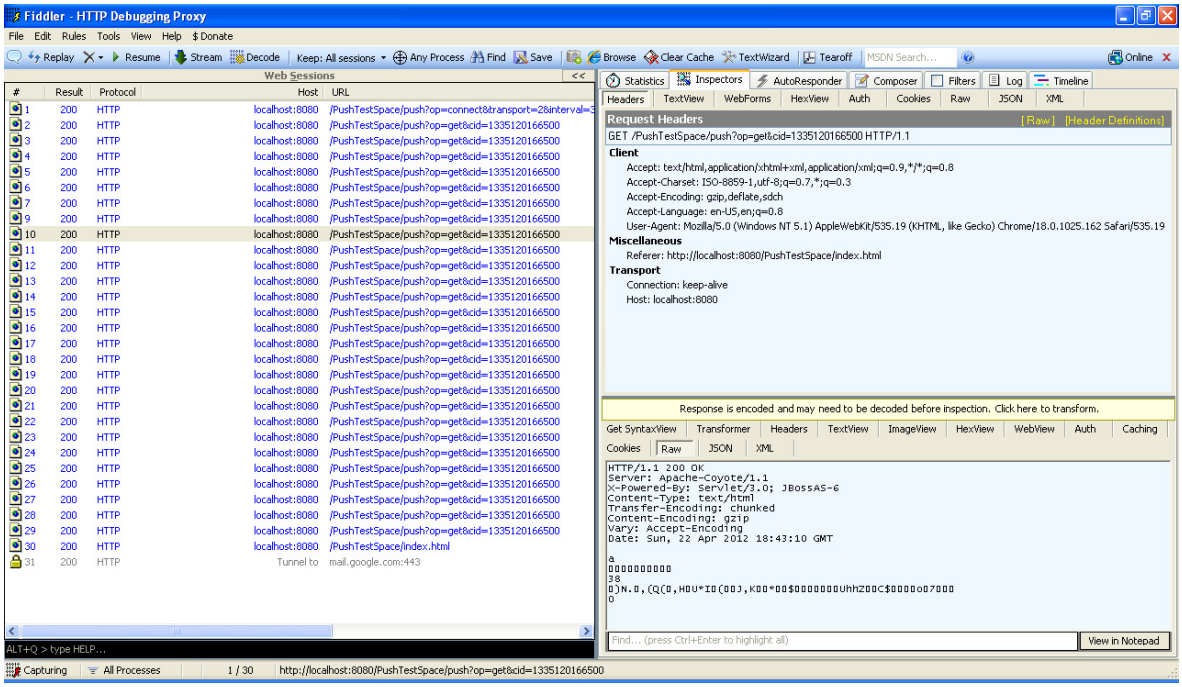
Свака од поређених техника преноса захтева посебну *JavaScript* имплементацију на клијентској страни (КП – клијентски протокол) и *Java* имплементацију на серверској страни (СП – серверски протокол), што је приказано одговарајућим модулима на слици. Функционалност потребна за мерење протока и времена преноса података имплементирана је у оквиру Тест *JavaScript* апликације која се извршава у веб претраживачу и у тест *servlet* класи која се извршава на апликативном серверу.

Као платформа за извршавање веб апликације коришћена су три најпопуларнија веб претраживача – *Firefox, Google Chrome* и *Internet Explorer*. За мерење протока података коришћена је апликација *Fiddler* ([Fiddler]), која омогућава праћење свих детаље комуникације између веб претраживача и апликативног сервера, укључујући комплетна заглавља захтева (енгл. *Request header*) и одговора (енгл. *Response header*), као и количину података пренету у сваком приступу апликативном серверу. Током рада, ова апликација се региструје као системски *proxy* за све апликације које користе *WinINET API* ([Wininet]), док се код апликација које не користе овај *API Fiddler* уланчава са системским *proxy*-јем, што је приказано на слици 9. На овај начин, сав саобраћај између веб апликације и апликативног сервера пролази кроз *Fiddler* апликацију, чиме се њој омогућава детаљна анализа саобраћаја.



Слика 9 - начин рада *Fiddler* апликације

Слика 10 приказује један од прозора *Fiddler* апликације у коме се виде детаљи размене података веб претраживача и апликативног сервера – листа сесија са леве стране и низ прозора за анализу сесија са десне стране.

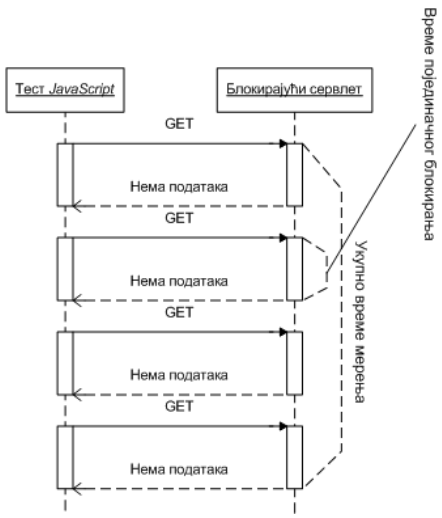


Слика 10 - прозор *Fiddler* апликације

4.1.1.2 Поступак мерења

Код првог мерења у овој групи, циљ је одређивање укупне количине података која се пренесе у сврху одржавања комуникационог канала, без преноса икаквих корисних података. Стога је развијена посебна тест апликација која одржава комуникациони канал у отвореном стању за сваку од поређених комуникационих техника. Слика 11 приказује ток комуникације који се понавља жељени број пута (на слици означено као укупно време мерења). Тест апликација приступа сервлету *GET* методом, сервлет блокира упит одређено време (на слици означено као *време појединачног блокирања*) и потом обавештава позивајући скрипт да нема нових података. Скрипт одмах по пријему информације о непостојању података креира нови упит ка серверу и поступак се понавља читаво време мерења. *Време појединачног блокирања* у мерењима ће бити 45 секунди, при чему се одговарајућим трансформацијама резултати мерења могу скалирати за било коју другу вредност времена појединачног блокирања.

Потребно је имати у виду да техника комуникације која се ослања на скривени *iFrame*, комуникациони канал одржава на другачији начин од друге две технике. Наиме, време између упућивања упита серверу и завршетка тог упита је далеко дуже (један сат у конкретним мерењима), с тим што се веза одржава периодичним слањем *script* тагова који у себи садрже позив повратне функције на главној страни веб апликације и то са периодом који одговара времену појединачног блокирања код друге две технике.



Слика 11 - одржавање комуникационог канала техником дугог повлачења

Табела 2 показује да све технике прослеђују исте параметре код *GET* упита којим приступају сервлету на серверској страни, док свака од техника добија различито тело одговора које назначава да сервер нема корисних података за веб апликацију. Тело одговора код техника 1 и 2 представља *JavaScript* код који се извршава по пријему од стране веб претраживача, и у конкретном примеру код обе технике своди се на позив повратних функција *i* (техника 1) и *x* (техника 2) објекта *$pm* комуникационе класе. Имена повратних функција су намерно означена само једним словом како би се минимизовала величина тела одговора које не носи корисну информацију. У случају технике 3, тело одговора је потпуно празно, што комуникациона класа третира као непостојање корисне информације на серверској страни.

Техника	***GET* параметри**	**Тело одговора**

1. Скривени iFrame	`o=g;`
	`c=timestamp;`
		`<script type="text/JavaScript">`
		`parent.$pm.i();`
		`</script>`

2. Убачени *script* таг са дугим повлачењем	`o=g;` `c=timestamp;`	`parent.$pm.x()`
3. HTTPXMLRequest са дугим повлачењем	`o=g;` `c=timestamp;`	

Табела 2 - параметри позива и тело одговора за различите технике

Код другог мерења, циљ је одредити време које протекне од тренутка када се порука пошаље из тест веб апликације до тренутка када она буде испоручена регистрованој повратној функцији. За потребе овог мерења основна мерна апликација је проширена на следећи начин:

- Креирана је посебна тест апликација „Тест пошиљалац“ која периодично шаље жељени број порука са временским печатом тренутка слања у телу поруке
- Тест апликација на уобичајен начин испоручује поруку функцији која је претплаћена на мерну поруку
- Слушалац порука у тренутку пријема поруке одузима вредност тренутног временског печата од временског печата у мерној поруци и тако добија време испоруке.

Слика 12 - време испоруке мерне поруке

Како би се минимизовао утицај тренутног оптерећења система на резултате мерења, мерење се понавља 100 пута и средња вредност се узима као крајњи резултат. Понављање се постиже уносом броја слања у „Тест пошиљалац“ апликацији, која жељени број порука шаље у секвенци са размаком од 0.5 секунди.

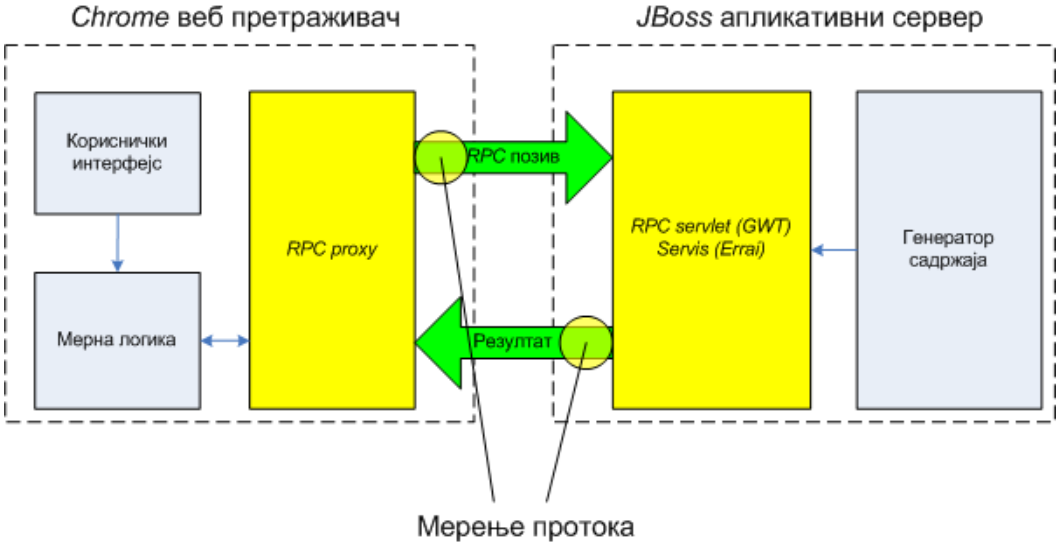
4.1.2 Мерење карактеристика *GWT* и *Errai RPC* механизама

Циљ овог мерења је поређење карактеристика *GWT* и *Errai* имплементације *RPC* механизма са аспекта потрошње пропусног опсега и брзине извршења позива. У односу на мерење потрошње пропусног опсега описано у секцији 4.1.1 које се односило на потрошњу пропусног опсега код преноса некорисних података, ово мерење се бави потрошњом пропусног опсега код преноса

корисних података. Модел података коришћених у мерењу направљен је по узору на модел података који се користи у стварном систему за пренос низова података који се приказују у листама. Код генерисања тест података, водило се рачуна о томе да генерисани подаци не буду погодни за компресију (нема понављања истих садржаја) што би могло да доведе до лажних резултата мерења.

4.1.2.1 Мерно окружење и алати

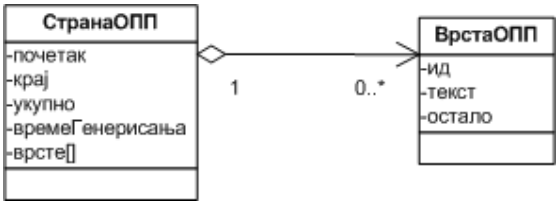
Као и код претходног мерења, за потребе овог мерења направљене су две једноставне мерне апликације којима се симулирају услови коришћења *RPC* механизма у пројектованом систему. Слика 13 приказује структуру модула која је идентична за обе мерне апликације. Модули чије се имплементације разликују зависно од *RPC* механизма који се користи означени су на слици жутом бојом.



Слика 13 - блок дијаграм система за мерење карактеристика *RPC* позива

Кроз једноставан кориснички интерфејс мерне апликације могуће је унети жељени број објеката који се генеришу на серверској страни помоћу *генератора садржаја* и преносе на клијентску страну и покренути мерење. Слика 14 приказује структуру објекта који се преноси у току мерења. Мерна апликација примењује *DTO* образац (енгл. *design pattern*) за пренос података између клијентске и серверске стране, који представља често коришћен образац у случају апликација које комуницирају путем удаљених интерфејса (енгл. *remote interface*)[Fowler]. *DTO* је енглеска скраћеница за *Data Transfer Object* (објекат за

пренос података - ОПП), објекат који се може серијализовати ради преноса одговарајућим комуникационим каналом. У конкретном случају, користи се објекат СтранаОПП, који у себи садржи податке о распону података који се преносе (у случају када се скуп свих података на серверу дели на мање целине, тзв. стране – енгл. *Page*), податке о укупном броју објеката на страни која се тренутно преноси, времену које је било потребно да се дата страна података креира као и низ ВрстаОПП објеката који садрже конкретне податке. Објекат класе ВрстаОПП представља податке који репрезентују линију у типичној листи у апликацији (на пример, листа удаљених уређаја за репродукцију и њихових стања се са серверске стране у веб апликацију преноси на овај начин). Овај објекат садржи свега три поља од којих прво представља јединствени кључ (ид) објекта, друго је текст којим се објекат репрезентује у листи, а треће поље (остало) се користи за додатне податке везане за објекат (на пример, тренутни статус удаљеног уређаја за репродукцију).



Слика 14 - ДТО класа за пренос података код *RPC* позива

Апликација *Fiddler* се и у овом случају користе за одређивање количине пренетих података у оквиру *HTTP* позива којим се преноси *RPC* позив.

4.1.2.2 Поступак мерења

Прво мерење у овој групи односи се на одређивање количине података која се пренесе у једном *RPC* позиву, која укључује податке упита (енгл. *request*) и одговора (енгл. *response*). Будући да је количина података која се преноси у току позива константна, мерење коришћења пропусног опсега за пренос података није потребно понављати. У корисничком интерфејсу мења се број објеката који се преносе са серверске стране и обавља се четири мерења и то за 10, 100, 500 и 1000 објеката. Код *Errai RPC* позива, мерење се обавља у случају постојања компресије података на страни сервера и у случају непостојања компресије, што се постиже подешавањем *HTTP* конектора *JBoss* апликативног сервера. Будући

да *GWT RPC* интерно већ укључује GZIP компресију података, мерење је вршено само једном и то са искљученом компресијом података на страни сервера. Количина пренетих података у сваком мерењу се очитава из *Fiddler* апликације, чиме је део мерења који се односи на коришћење пропусног опсега завршен.

Слика 15 - ток мерења протока података и брзине *RPC* позива *Errai* и *GWT* библиотеке

Слика 15 приказује секвенцу позива која се одвија током мерења. Процес мерења започиње се из корисничког интерфејса, при чему мерна логика иницира *RPC* позив са бројем садржаја као параметром и бележи тренутак позива. *RPC proxy* серијализује параметре позива које *RPC servlet* (код *GWT* мерења) односно сервис (код *Errai* мерења) десеријализује на серверској страни и позива функцију генератора садржаја који припрема СтранаОПП објекат са задатим бројем ВрстаОПП објеката. Припремљени садржаји се потом серијализују као резултат позива и достављају *RPC proxy*-ју. *RPC proxy* врши десеријализацију података и резултат доставља регистрованој повратној функцији *RPC* позива унутар блока мерне логике, који у том тренутну одређује време протекло од покретања позива, чиме се циклус завршава. Од укупног времена позива, на слици означеног зеленом бојом, мерна логика одузима време генерисања мерног садржаја (поље ***времеГенерисања*** класе ***СтранаОПП***), на слици означено црвеном бојом, чиме се добија време утрошено на припремне радње самог позива. Измерено време се приказује у корисничком интерфејсу апликације, као и средња вредност задњих 100 позива. Мерна апликација омогућава ресетовање свих измерених вредности и израчунатих средњих вредности како би мерење било могуће поновити у

сваком тренутку. За сваки скуп од 10, 100, 500 и 1000 пренетих објеката, са и без компресије података (у случају *Errai* библиотеке), мерење се обавља по 100 пута што је постигнуто аутоматизованим понављањем позива на сваких 500 милисекунди. Осим средњег времена позива за 100 мерења, мерна апликација креира и хистограме који показују варирање времена позива у свим случајевима.

4.2 Поступак за мерење параметара комуникације између серверске и клијентске контролне апликације

Мерења описана у овом поглављу везана су за комуникацију између серверске апликације и клијентске контролне апликације. Део мерења се бави понашањем система у случајевима отежане комуникације између серверске и клијентске апликације, док се други део мерења бави мерењем оптерећења сервера у случају комуникације са великим бројем клијената базиране на техници периодичног повлачења.

4.2.1 Одређивање параметара преузимања мултимедијалних датотека у условима отежане комуникације

Пренос рекламних садржаја са централног сервера на удаљене платформе за репродукцију представља једну од кључних функционалности платформе описане у овом раду. Стога је одређивање оптималних параметара преноса података у различитим мрежним окружењима, посебно у условима постојања различитим врста грешака и сметњи, од изузетног значаја за пројектовање поузданог и ефикасног система. Мерења описана у овој секцији извршена су у тест-окружењу пројектованом за симулирање критичних мрежних услова и циљ им је да одреде оптималан број симултаних преузимања датотека са сервера започетих од стране клијентске контролне апликације.

4.2.1.1 Мерно окружење и алати

Успостављено мерно окружење је пројектовано тако да омогући поновљива мерења понашања система у условима постојања различитих врста грешака на комуникационом каналу. Слика 16 приказује мерни систем који се састоји од

сервера и контролне клијентске апликације између којих је комуникација успостављена преко мрежног моста – рачунара са инсталираном *Dummynet* апликацијом [Dummynet]. Рачунар који представља мрежни мост има две мрежне картице, од којих се путем једне повезује на локалну мрежу преко које комуницира са сервером, док се преко друге повезује са рачунаром на коме је инсталирана клијентска контролна апликација и то преко *switch* уређаја који припада изолованој мерној подмрежи. Комплетна комуникације између локалне мреже и мерне подмреже контролише се путем *Dummynet* апликације, која се користи за постављање различитих врста параметара комуникационог канала као што су ширина пропусног опсега (енгл. *bandwidth*), кашњење (енгл. *latency*), губитак пакета (енгл. *packet loss ratio*) итд.

Слика 16 - окружење за мерење понашања система код постојања грешака на комуникационом каналу

Dummynet се у свом раду ослања на *ipfw* – један од бесплатних *freeBSD* заштитних мрежних програма (енгл. *firewall*). *Dummynet* пресреће мрежне пакете коришћењем *ipfw*-а, пропушта их кроз један или више објеката званих *queues* или *pipes* који уносе ефекте ограничења мрежног опсега, кашњења, губитака пакета и слично и на крају их испоручује примаоцу, чиме се постижу жељени ефекти грешака у мрежној комуникацији. За потребе мерења, направљени су скриптови који симулирају жељене мрежне параметре.

Клијентска контролна апликација је направљена тако да се може подесити број истовремених преузимања датотека са сервера (број нити које преузимају датотеке). Мерења су вршена за 5, 10 и 20 симултаних преузимања. Времена преузимања и остали параметри су одређивани из логова клијентске контролне апликације, која је податке бележила коришћењем *Log4J* библиотеке.

Током мерења, са серверске апликације је преузиман временски распоред са пуно мањих датотека (398) укупне величине 322MB.

4.2.1.2 Поступак мерења

Поступак мерења се своди на мењање претходно описаних параметара система и покретање процеса инсталације временског распореда репродукције на страни клијентске контролне апликације. Поступак се састоји од следећих корака:

1. Подешавање броја истовремених преузимања датотека код клијентске контролне апликације на 5, 10 или 20
2. Подешавања параметара *Dummynet* мрежног моста:
    a. 512 Kbit/s са кашњењем од 10ms,
    b. 1Mbit/s са кашњењем од 5ms,
    c. 2Mbit/s са кашњењем од 1ms,
    d. без ограничења
3. Покретање преузимања временског распореда репродукције и рекламних садржаја
4. Одређивање мерених вредности анализом лога
5. Брисање преузетих података на страни клијентске контролне апликације и довођење у почетно стање ради следећег мерења.

4.2.2 Мерење оптерећења сервера у случају великог броја клијентских контролних апликација

Циљ овог мерења је одређивање броја клијентских контролних апликација које истовремено могу одржавати везу са серверском апликацијом техником дугог повлачења без угрожавања осталих функционалности серверске апликације.

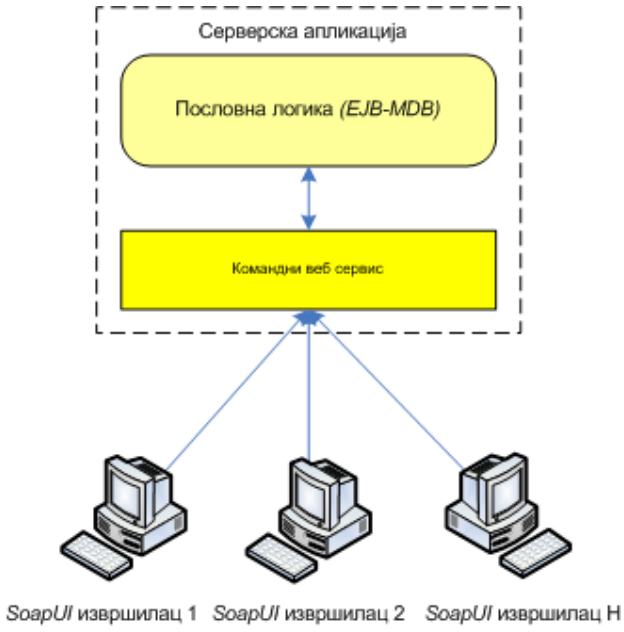
4.2.2.1 Мерно окружење и алати

Обзиром да се комуникација између клијентске контролне апликације и сервера одвија преко веб сервиса, уместо коришћења великог броја клијентских контролних апликација, за тестирање је коришћена *SoapUI* апликација специјализована за тестирање веб сервиса [SoapUI]. Ова апликација осим извршавања функционалних тестова омогућава и извршавање тестова оптерећења, што је функционалност коришћена код овог мерења. Слика 17

приказује конфигурацију мерног окружења, код кога се један или више рачунара користе као извршиоци *SoapUI* тестова, при чему сваки од извршилаца симулира мноштво клијентских контролних апликација.

Кораци у креирању мерне апликације у *SoapUI* алату су били следећи:

- На основу *WSDL* описа командног веб сервиса, *SoapUI* генерише низ функција којима се сервису може једноставно приступати уз задавање параметара функција у аутоматски генерисаном XML телу *SOAP* поруке
- Коришћењем овако дефинисаних функција, креира се тест случај (енгл. *test case*) који представља секвенцу операција попут позива функција командног веб сервиса, обраде резултата позива коришћењем *XPath* и *Groovy* језика ([XPath], [Groovy]) или извршавања комплексније тест логике у виду скриптова у *Groovy* језику.
- Овако дефинисан тест случај се потом покреће коришћењем алата за тест оптерећења, који покреће *н* изолованих тест случајева у виду нити, које у паралели приступају циљном серверу

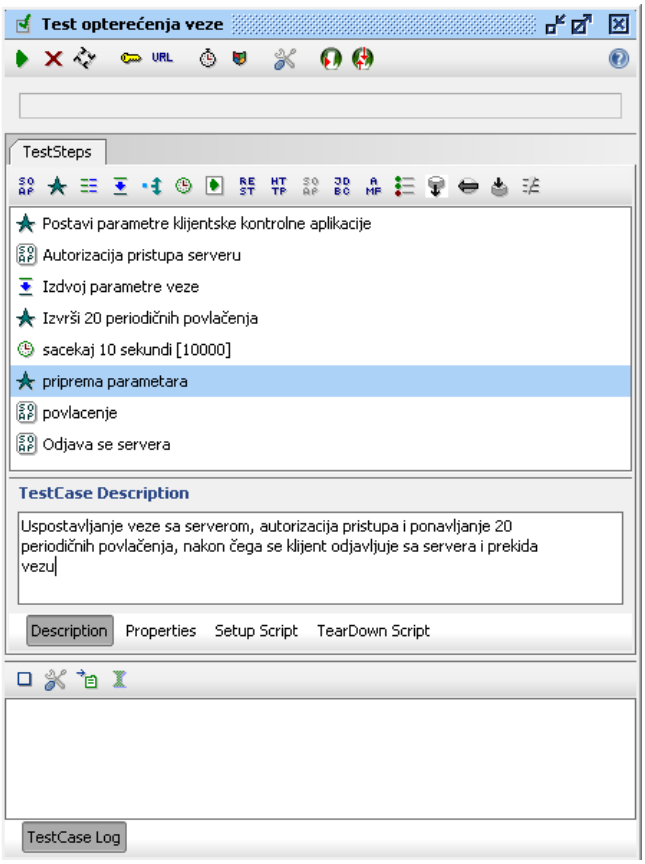


Слика 17 - Окружење за тестирање оптерећења сервера

Слика 18 приказује дијалог *SoapUI* апликације у коме се дефинише тест случај који симулира успоставу, одржавање и раскидање везе појединачне

клијентске контролне апликације и сервера. Тест случај се састоји од следећих корака:

- Постављање параметара клијентске контролне апликације, којима се описује једна инстанца тест клијентске контролне апликације. Овај корак се своди на извршење *Groovy* скрипта, који на основу параметара које извршилац тестова оптерећења доставља конструише јединствене параметре клијентске контролне апликације
- Коришћењем параметара конструисаних у првом кораку, у другом кораку се позива функција веб сервиса којом се тест клијент ауторизује на сервер
- У трећем кораку се коришћењем XPath модула издвајају параметри везе (попут стринга сесије) из *SOAP* одговора добијеног код позива функције за ауторизацију.
- У наредном кораку се помоћу једноставне петље у *Groovy* језику врши двадесет повлачења команди којима се уједно и одржава веза се сервером. *Groovy* скрипт је дат у табели 3.
- Последњи корак представља одјављивање клијентске контролне апликације са сервера. До овог корака се стиже одмах после извршења *Groovy* скрипта, прескакањем наредне три команде коришћењем *Groovy* наредбе *testRunner.gotoStepByName*



Слика 18 - кораци којима се симулира веза појединачног клијента

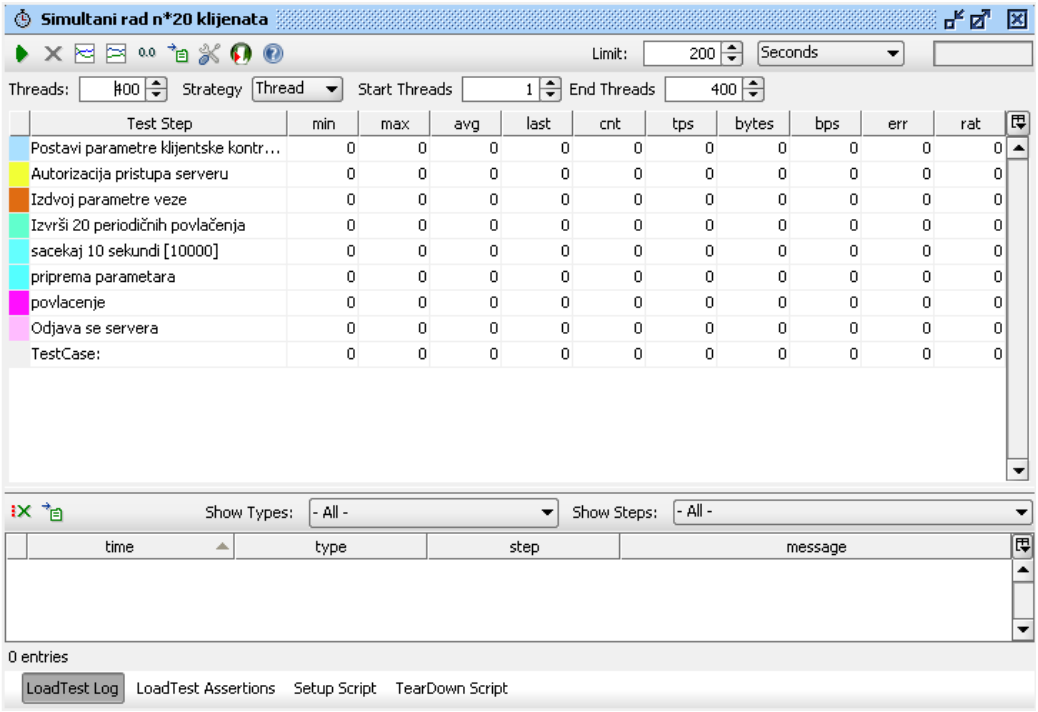
```
for( i in 1..20 )
{
     testRunner.runTestStepByName( "sacekaj 10 sekundi");
     testRunner.runTestStepByName( "priprema parametara");
     testRunner.runTestStepByName( "povlacenje");
}
testRunner.gotoStepByName( "odjava sa servera");
```

Табела 3 - *Groovy* скрипт за 20 понављања периодичних повлачења

4.2.2.2 Поступак мерења

Тест оптерећења се врши постепеним повећавањем броја симулираних клијентских контролних апликација које одржавају везу са сервером, уз истовремено мерење заузећа процесора сервера. Тестирање је подељено на фазе, при чему се у свакој наредној фази повећава број симулираних клијентских контролних апликација за 20, почевши од 20 у првој фази. Слика 19 приказује дијалог *SoapUI* апликације у коме се подешавају параметри теста оптерећења,

при чему се број симулираних контролних клијентских апликација дефинише кроз број нити, док се време потребно за покретање датог броја нити дефинише као параметар *limit*. Приликом извршавања тестова, вођено је рачуна о томе да се параметар *limit* постави тако да се не покреће више од две нити у секунди. По извршењу теста, прикупљају се статистички подаци, попут минималног, средњег и максималног кашњења код извршења функције веб сервиса као и број грешака које су се догодиле током извршења теста оптерећења. На серверу се током извршења сваке фазе мерења покреће шел-скрипт (енгл. *shell script*) који коришћењем *Linux top* команде креира лог датотеку која садржи промене заузетости процесора од стране *Java* процеса, чиме се процењује оптерећење система у датој фази мерења.



Слика 19 - креирање теста оптерећења у *SoapUI* апликацији

ПОГЛАВЉЕ 5.

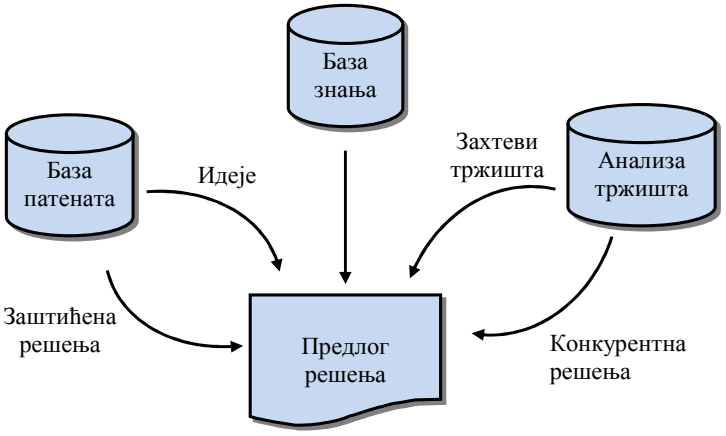
СИСТЕМ ЗА КОНТРОЛУ ДИГИТАЛНИХ ОГЛАСНИХ ПАНЕЛА И ДИСТРИБУЦИЈУ РЕКЛАМНИХ САДРЖАЈА

Мотивација за развој система за контролу дигиталних рекламних панела и дистрибуцију рекламних садржаја потиче од чињенице да оваква врста оглашавања постаје изузетно популарна захваљујући комбинацији већег броја фактора[Schaeffler]:

- Широка доступност *LCD* панела високе резолуције
- Широка доступност мрежа, укључујући Интернет, којима се рекламни садржаји брзо и јефтино достављају
- Доказани ефекат реклама на потрошаче
- Далеко мањи трошкови овакве врсте оглашавања у односу на статичне видове реклама, попут билборда

Критични аспекти оваквих система које треба оптимизовати су цена експлоатације, флексибилност и поузданост контроле система на даљину, једноставност инсталације и одржавања, проширивост система.

Проучавањем релевантних извора информација из различитих извора, као што је то приказано на слици 20 [Papp], долази се до предлога решења.



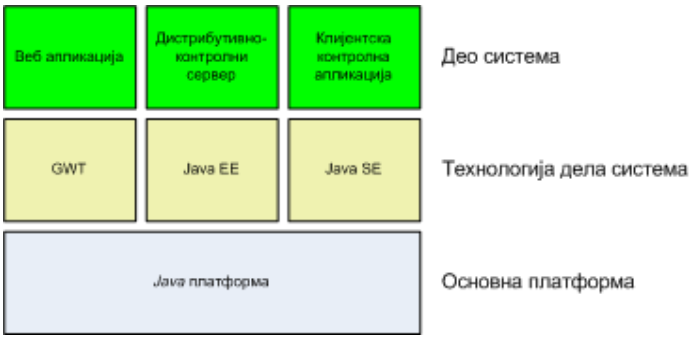
Слика 20 - анализа релевантних информација

Претрага базе патената указала је на релевантне патенте и дала увид у савремена истраживања, уз пружање увида у савремене трендове у области интереса овог рада. Показало се да се одређени број патената концентрише и на аспекте интеракције система са корисником, што је тренд који је узет у обзир код развоја система. Један од патената се фокусира на начине оптимизације саобраћаја између централног сервера и клијената, што је такође узето у обзир код развоја система, будући да је то аспект који директно утиче на скалабилност система.

У оквиру анализе тржишта анализирано је више решења система за контролу дигиталних рекламних панела и дистрибуцију рекламних садржаја и на тај начин је утврђен скуп неопходних функционалности које предложени систем мора да имплементира. Поред неопходног скупа функционалности, утврђени су и недостаци постојећих решења који ће бити уграђени у предложено решење. Анализа је такође показала да већи део постојећих решења нуди само део потребних функционалности (нпр. могућност дистрибуције садржаја без могућности контроле репродукције, непостојање могућности дијагностике система и слично). Такође, установљено је да већи део постојећих решења уводи платформску зависност, и то најчешће везано за специфичан вид садржаја који се дистрибуира.

5.1 Примењене технологије

У тези се као предлог система за контролу дигиталних рекламних панела и дистрибуцију рекламних садржаја нуди систем базиран на *Java* платформи, при чему су различите *Java* технологије коришћене за различите делове система (Слика 21), при чему је детаљна анализа технологија дата у секцији 3.4.



Слика 21 - преглед коришћених технологија према деловима система

Коришћење *Java* платформе као основе система обезбеђује платформску независност свих делова система – од веб апликације за приступ систему, преко дистрибутивно-контролне апликације која складишти и дистрибуира садржаје и контролише клијентске апликације, па до клијентске контролне апликације која прихвата и извршава команде дистрибутивно-контролне платформе. Различити делови система користе различите *Java* технологије које су најбоље прилагођене потребама тог дела система:

- Веб апликација се заснива на *GWT* технологији [GWT] која нуди следеће предности:
  - Веб апликација се развија у Јави а компајлира се у оптимизован *JavaScript* програмски код
  - Креиране апликације су компатибилне са најчешће коришћеним веб претраживачима
  - Интеграција креираних апликација са серверском страном је једноставна и заснива се на *RPC* позивима[GWTAction]
  - Развој апликација је брз захваљујући интегрисаним развојним окружењима
- Серверска дистрибутивно-контролна апликација представља језгро система, које комуницира са великим бројем веб клијената и *Java*

клијентских контролних апликација. Као таква, заснива се на *Java EE* технологији, која нуди следеће предности:

- *Java EE* омогућава развој дистрибуираних апликација базираних на модуларним компонентама
- *Java EE* дефинише специфичне компоненте, као што су *EJB* компоненте (енгл. *Enterprise Java Bean*), које представљају градивне блокове серверске апликације у које су по стандарду укључене функционалности као што су подршка за трансакције, *RPC* позиви, сигурност, рад са базама података (енгл. *Persistence*) [EJB3Action].
- *Java EE* као део спецификације доноси подршку за веб сервисе, који представљају популаран вид флексибилног повезивања компоненти у дистрибуираним системима
- *Servleti* су део *Java EE* спецификације који омогућавају једноставну интеграцију серверског дела апликације са веб апликацијама развијеним у *GWT*-у

- Клијентска контролна апликација прима и извршава команде од серверске апликације путем веб сервиса. Ова апликација је имплементирана као конзолна *Java* апликација из следећих разлога:
  - Захтева минимум ресурса на одредишној платформи на којој се извршава
  - Може се вршити аутоматско преузимање најновије верзије апликације коришћењем *JWS* (енгл. *Java Web Start*) технологије
  - Може се извршавати на свакој платформи која подржава *Java SE* технологију

5.2 Изоловани проблеми и предложена решења

На основу истраживања различитих извора информација везаних за системе којима се ова теза бави (секције 3.1, 3.2 и 3.3), идентификован је низ проблема и предложена су решења за њихово превазилажење. Резултати су приказани у табели 4.

Проблем / Решење	Брзина реаговања веб апликације	Минимизација протока веб апликација - сервер	Минимизација протока сервера-клијент	Зависност садржаја од платформе	Приступ клијентима иза заштитног програма	Преоптерећење пропусног опсега	Контрола и дијагностика уређаја	Статистика коришћења система
Комбиновани модели комуникације	■	■						
Коришћење СОВ сервиса за сигнализацију			■					
Модел садржаја независног од платформе				■				
Коришћење веб сервиса за комуникацију					■			
Контролор проп. опсега						■		
Протокол за контролу и дијагностику							■	
Модули за прикупљање и обраду стат. података								■

Табела 4 - приказ уочених проблема и предложених решења

Следи кратак опис наведених проблема заједно са појашњењем предложених решења.

Једно од ограничења апликација које се извршавају у веб претраживачу је начин комуникације са сервером, који је увек иницран од стране претраживача. Стога је најчешћи начин достављања информација са сервера на веб клијент коришћењем механизма повлачења података од стране веб клијента (енгл. *pooling*). Будући да клијентска веб апликација поседује низ листа са подацима који потичу са сервера, уобичајен начин да се листе држе у ажурном стању је њихово периодично освежавање, чиме се повећава количина пренетих података између сервера и клијента и оптерећење сервера. Уместо непрекидног освежавања листа, уведен је механизам доставе података са сервера (енгл. *server push*), којим се клијентској веб апликацији доставља информацију о променама података на серверу чиме се омогућава минимално коришћење пропусног опсега.

Комуникација између клијентске апликације и сервера је слична претходно описаној, уз ту разлику што се комуникација врши путем веб сервиса, који уносе повећано коришћење пропусног опсега због природе коришћеног *SOAP* протокола. Стога се пренос података путем веб сервиса у овом приступу

комбинује са техником достављања информације о доступности података кроз помоћни комуникациони канал малог пропусног опсега као начин за минимизацију искоришћености пропусног опсега. Помоћни комуникациони канал се такође користи и за праћење постојања везе са сваком контролном клијентском апликацијом појединачно, и у спрези са специјализованим COB сервером ослобађа централни сервер оптерећења везаног за симултано праћење стања великог броја мрежних веза.

Већина разматраних система захтева одређен формат рекламних садржаја који је у стању да репродукује и није у могућности да преноси произвољан тип садржаја који би се могао приказати алтернативним системом за репродукцију. Предложено решење уводи модел преноса рекламних садржаја који се може искористити за пренос произвољног типа рекламног садржаја. Осим тога, клијентска контролна апликација је у овом решењу раздвојена од апликације за репродукцију, чиме се ствара независност преноса рекламних садржаја од њихове репродукције.

Дигитални огласни панели су често постављени иза заштитних програма (енгл. *firewall*), што отежава њихову комуникацију са сервером. Коришћењем веб сервиса за комуникацију између клијената и сервера, као и помоћних комуникационих канала који се заснивају на *TCP* протоколу овај проблем се решава, уз све остале предности које веб сервиси доносе.

У случајевима када је потребно дистрибуирати велику количину рекламних садржаја великом броју клијената, постоји могућност преоптерећења сервера. Предложено решење обезбеђује начине да се преоптерећење спречи и расподели на већи број сервера.

Тренд у модерним системима за оглашавање је повећавање интеракције са корисником кроз коришћењем других уређаја осим рекламних панела (контрола светла, звука, мириса, реаговање на покрет корисника...). Оваква поставка система уводи потребу за контролом и дијагностиком више уређаја, што је могуће уз унификовани протокол за контролу и дијагностику уређаја предложен у овом решењу.

Прикупљање информација о понашању потрошача у објектима у којима су системи за репродукцију рекламних садржаја инсталирани од изузетног је

значаја за дизајнере рекламних садржаја, који на основу прикупљених информација могу да побољшају ефекат рекламних садржаја на којима раде. Мулти-сензорски систем какав је описан у овом раду у сваком тренутку прикупља мноштво података о целокупном окружењу, који се помоћу модула за логовање статистичких информација уписују у фајлове који се потом архивирају и периодично шаљу на централни сервер ради даље обраде. Централни сервер преузима архиве статистичких података и помоћу модула за креирање извештаја генерише специјализоване извештаје прилагођене потребама аналитичара.

5.3 Основни елементи система за контролу дигиталних рекламних панела и дистрибуцију рекламних садржаја

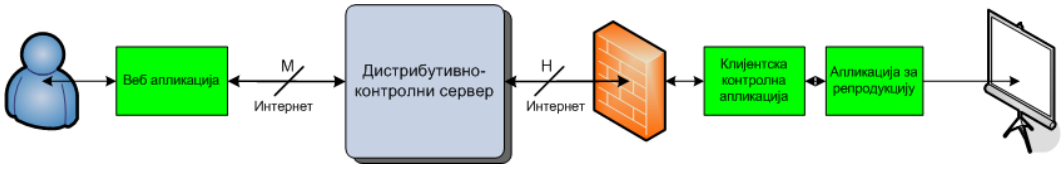
У типичном систему за контролу дигиталних рекламних панела и дистрибуцију рекламних садржаја постоји низ фактора који утичу на ефикасност система што опет највише утиче на цену експлоатације система. Минимизација количине пренетих података између сервера и веб апликације, као и минимизација количине пренетих података између сервера и клијентске Java апликације су задаци које треба решити, при чему се тежи ка систему који брзо реагује на команде корисника и брзо доставља одговоре контролисаног система кориснику. Поменути захтеви су међусобно супротстављени, будући да се комуникација између појединих делова система често заснива на периодичном повлачењу података. Осим утицаја на количину пренесених података, поменути начин комуникације утиче и на оптерећење сервера, о чему је такође потребно водити рачуна код дизајна система.

Савремени рекламни садржаји који се користе у системима дигиталног оглашавања по правилу захтевају пуно простора за складиштење, а њихов пренос на сервер и дистрибуција ка одредишним клијентима захтевају значајно коришћење пропусног опсега. Систем се стога мора дизајнирати тако да се складишни простор може лако проширити, а дистрибуција и пријем садржаја се могу контролисати како би се могло спречити преоптерећење сервера.

Спрега између централног сервера и клијената је типично путем Интернета, а због природе услова у којима се клијенти инсталирају не може се гарантовати

непрекидно постојање везе. Стога је битно да систем буде флексибилан по питању присутности клијената и да гарантује одложену испоруку садржаја и команди. Такође, неочекивани прекиди везе у току комуникације су могући и неопходно је да систем буде отпоран на њих.

Слика 22 приказује блок дијаграм система за контролу дигиталних рекламних панела и дистрибуцију рекламних садржаја у очекиваном радном окружењу. Главни део система је дистрибутивно-контролни сервер. Са једне стране, овај сервер комуницира са M веб апликација које служе као графичка корисничка спрега за управљање и надзор система. Са друге стране, серверска апликација комуницира са H клијентских контролних апликација, које прихватају рекламне садржаје и команде и управљају репродукцијом. Ове апликације се могу налазити иза заштитних програма те је неопходно да буду дизајниране тако да могу комуницирати са сервером у таквом окружењу. Треба приметити да је *Java* клијентска контролна апликација одвојена од апликације за репродукцију, чиме се обезбеђује независност пренесеног садржаја од типа апликације за репродукцију.



Слика 22 - структура система за дистрибуцију и контролу

5.3.1 Архитектура дистрибутивно-контролног сервера

Пословна логика дистрибутивно-контролног сервера имплементирана је као низ *EJB*-ова, при чему је сваки *EJB* специјализован за одређене задатке у оквиру свог подсистема. Слика 23 приказује скуп основних подсистема пословне логике који обезбеђују комплетну функционалност дистрибутивно-контролног сервера, при чему се сваки подсистем састоји од више *EJB*-ова. Инструментација пословних метода *EJB*-ова ради извршења комплексних задатака врши се из командног веб сервиса, RPC сервлета и из самих *EJB*-ова. У наставку ће бити дат опис функционалности сваког од приказаних подсистема.

Слика 23 - подсистеми пословне логике

5.3.1.1 Командни подсистем

Овај подсистем обезбеђује функционалности за слање команди клијентским контролним апликацијама, као и обраду резултата извршења команди. Структура команди и резултата описана је у секцији 5.3.3.3.1, а начини доставе команди и одговора у секцијама 5.3.3.1 и 5.3.3.2. Најбитније функције које обезбеђује командни подсистем су:

- Достава команди појединачним клијентским контролним апликацијама
- Достава команди групама клијентских контролних апликација
- Ажурирање статуса извршења команди сходно дијаграму стања датом на слици 31

Пословне методе командног подсистема користе се од стране командног веб сервиса (преузимање команди и ажурирање стања извршења) и *RPC* сервлета из којих се команде иницирају на захтев корисника веб апликације.

5.3.1.2 Подсистем за праћење везе

Овај подсистем обезбеђује функционалност којом се у сваком тренутку прати стање везе клијентских контролних апликација, као и основни параметри стања (време последњег пријављивања, тренутно стање репродукције и слично). Механизми праћења везе описани су у секцији 5.3.3.4. Најбитније функције овог подсистема су:

- Ажурирање стања везе у бази података сходно променама добијеним разним механизмима праћења стања везе
- Обезбеђивање листа статуса везе за приказ у веб апликацији

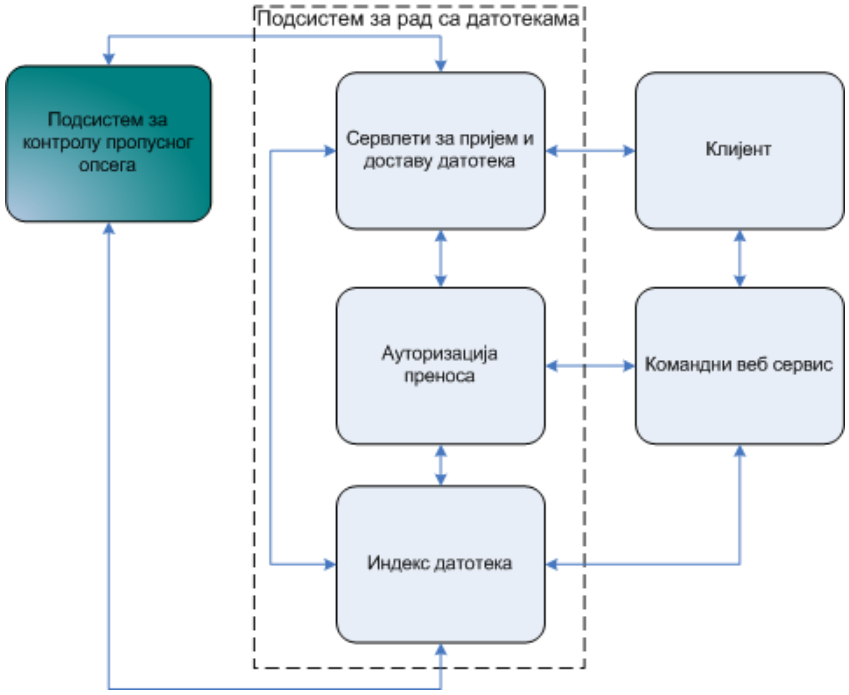
5.3.1.3 Подсистем за руковање временским распоредима

Овај подсистем обезбеђује креирање и ажурирање временских распореда, као и функције потребне за њихову доставу клијентским контролним апликацијама. Детаљи концепата везаних за временске распореде дати су у секцији 5.3.4.2. Најбитније функције овог подсистема су:

- Креирање, ажурирање и брисање временских распореда
- Доставу листа временских распореда, као и појединачних распореда
- Аутоматска редистрибуција промењених распореда

5.3.1.4 Подсистем за рад са датотекама

Овај подсистем је задужен за преузимање датотека од клијентских апликација као и за доставу датотека клијентским апликацијама, при чему се појам клијентске апликације односи како на веб апликацију, тако и на клијентску контролну апликацију. Као што је наведено секцији 5.3.1, *API* овог подсистема је исти и за веб апликацију и за клијентску контролну апликацију. Слика 24 показује детаљнију структуру подсистема за рад са датотекама као и његову спрегу са осталим подсистемима.



Слика 24 - структура подсистема за рад са датотекама

Сама достава датотека или њихов пријем врши се помоћу сервлета који могу бити физички лоцирани на више различитих сервера. Блок са називом ***Индекс датотека*** чува информације о датотекама, у шта спадају њихове путање на диску, подаци о величини и израчунатим *hash* вредностима, као и *URL* сервера на коме се датотеке физички налазе. Блок за ауторизацију преноса чува информације о ауторизованим клијентима. Корак који претходи сваком преузимању или достављању датотеке је позивање функције ***захтевПреноса*** командног веб сервиса, која клијенту даје све релевантне параметре преноса (*URL* сервера и параметре за ауторизацију) уз упис потребних информација у базу података од стране блокова за ауторизацију и индекс датотека. Клијент потом користи добијене параметре код позива одговарајућег сервлета, при чему сервлет дате податке верификује позивом блока за ауторизацију преноса и индекса датотека. Подсистем за контролу пропусног опсега води рачуна о тренутном броју трансфера и сходно томе дозвољава и не дозвољава нови трансфер датотеке. Уколико је у тренутку покушаја покретања новог трансфера већ достигнут максималан број дозвољених трансфера, подсистем за контролу пропусног опсега на основу стања тренутних трансфера и броја већ одбијених трансфера израчунава време после кога клијент треба да понови трансфер и

доставља га клијенту уз *HTTP* код грешке 503 (енгл. Service Unavailable – сервис није на располагању).

5.3.1.5 Подсистем за контролу пропусног опсега

Сврха подсистема за контролу пропусног опсега је праћење броја активних преноса датотека на свим серверима и спречавање преоптерећења појединих сервера услед превеликог броја истовремених преноса датотека. Као што је приказано на слици 24, овај блок је у непосредној вези са индексом датотека, који користи информације о оптерећењу појединих сервера приликом одређивања сервера на који ће се доставити датотеке у захтеваном трансферу.

Сваки од сервлета за доставу или преузимање датотека шаље следеће информације овом подсистему:

- Почетак трансфера
- Напредак трансфера (променљива грануларност, нпр. на сваких 1MB промене)
- Завршетак трансфера

Код сваког покушаја трансфера, сваки од сервлета позива подсистем за контролу пропусног опсега, који трансфер дозвољава или даје процену времена када покушај трансфера треба поновити.

5.3.1.6 Подсистем за рад са рекламним садржајима

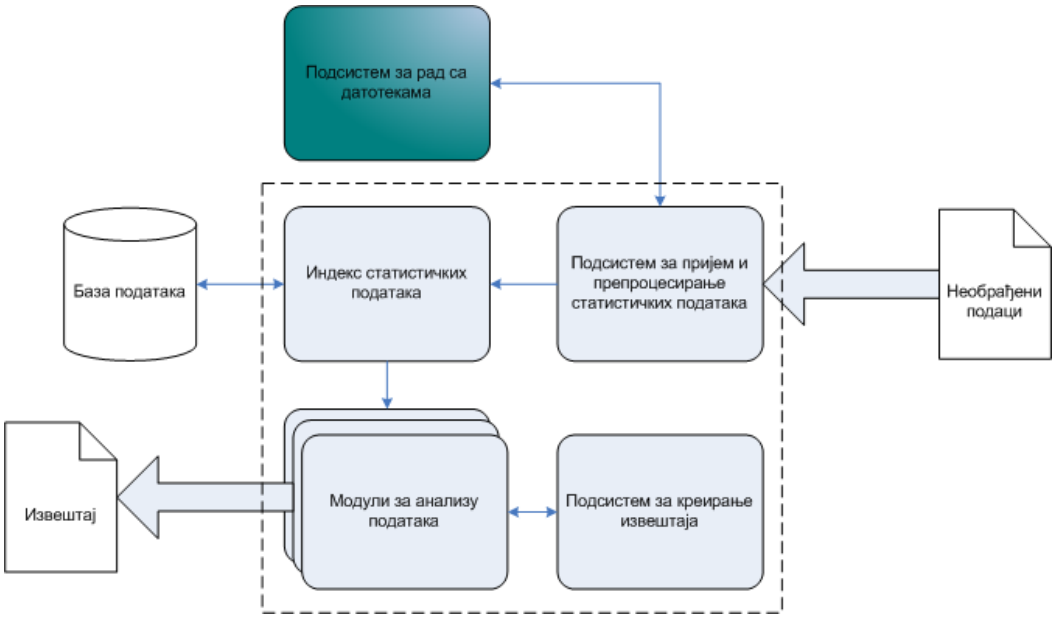
Модел рекламних садржаја описан је у секцији 5.3.4.1, при чему је сврха подсистема за рад са рекламним садржајима успостављање везе између наведеног модела и физичких датотека којима манипулише подсистем за рад са датотекама. Овај модул омогућава операције додавања, мењања, брисања и листања (енгл. *CRUD - Create, Read, Update and Delete*) рекламних садржаја, ослањајући се на подсистем за рад са датотекама код приступа самим рекламним датотекама.

Подсистем успоставља релације између рекламних пројеката и рекламних садржаја, водећи рачуна и о дељењу рекламних садржаја између рекламних пакета. Зависно од ових релација и промена дељених садржаја, подсистем врши и обележавање рекламних садржаја који захтевају редистрибуцију.

5.3.1.7 Подсистем за извештавање

Овај подсистем је задужен за пријем, складиштење и обраду статистичких информација добијених од стране клијентских контролних апликација. Детаљи унутрашње структуре овог подсистема приказани су на слици 25.

Статистички подаци се од клијентске контролне апликације добијају у виду датотека у *CSV* формату, које садрже статистичке податке у предефинисаном облику. Датотеке се на дистрибутивно-контролни сервер достављају као и све друге датотеке, и чувају се у оквиру подсистема за рад са датотекама. По испоруци датотека на дистрибутивно-контролни сервер, клијентска контролна апликација позива функцију ***пријавaПромене*** командног веб сервиса пријављујући доставу статистичких датотека, чиме се иницира подсистем за пријаву и пре-процесирање статистичких података. Овај подсистем преузима пристигле датотеке из подсистема за рад са датотекама, врши њихово пре-процесирања ради издвајања основних информација о садржаним подацима (обухваћени временски период, количина података и слично) и дате информације доставља индексу статистичких података, који их бележи у базу података.



Слика 25 - блокови подсистема за извештавање

Из добијених статистичких података могу се издвајати различите врсте информација па самим тим и креирати различите врсте извештаја. Сходно томе,

за сваки појединачни извештај креира се посебан модул за анализу података. Модулима за анализу података рукује подсистем за креирање извештаја, представљајући приступну тачку за веб апликацију преко које се иницира генерисање различитих врста извештаја. Додавање нових типова извештаја у систем се своди на додавање нових модула за анализу података, при чему их подсистем за креирање извештаја региструје и ставља на располагање корисницима веб апликације, покрећући их по потреби и достављајући генерисане извештаје корисницима.

5.3.1.8 Подсистем за дијагностику

Модел класа којима се представља стање уређаја и његових параметара дат је у секцији 5.3.4.4. Подсистем за дијагностику добија резултат комплетне дијагностике система од клијентске контролне апликације тако што клијентска контролна апликација позива функцију ***достваДијагностике*** командног веб сервиса. Ова функција се позива у следећим случајевима:

- Код покретања клијентске контролне апликације
- Код детектовања неисправног рада неког уређаја
- На експлицитан захтев (команда КМД_ПУНА_ДИЈАГНОСТИКА) услед захтева корисника веб апликације за освежавањем листе стања уређаја

Са стране веб апликације, овај подсистем се користи за преузимање задњег забележеног стања уређаја на клијенту, или за задавање експлицитне команде за добијање тренутног стања уређаја у случајевима када се сумња да стање листе у бази података не одговара стварном стању уређаја на клијенту.

5.3.1.9 Подсистем за периодичне задатке

Сврха овог подсистема је извршавање периодичних задатака везаних за одржавање стања система. У ове задатке спада:

- уклањање привремених датотека
- уклањање застарелих команди
- слање статусне електронске поште администраторима система

Систем се базира на коришћењу стандардно *Timer* сервиса *Java EE* платформе, који омогућава једноставно извршавање периодичних задатака у *Java EE* окружењу.

5.3.1.10 Подсистем за манипулацију листама

Кориснички интерфејс контролне веб апликације садржи већи број листа, попут листа рекламних платформи, листа рекламних пакета, листа распореда репродукције и слично. Поменуте листе често имају више стотина па и хиљада елемената, за које је битно да у сваком тренутку буду ажурни, то јест да стање елемента приказаног у листи одговара стању елемента у систему. Илустрација ове потребе може бити приказ листе рекламних платформи, од којих су неке у вези са дистрибутивно-контролним сервером, а неке нису, што је у листи приказано одговарајућом статусном иконицом. Очекивано понашање је да ће се у тренутку успоставе везе између претходно неповезане клијентске контролне апликације рекламне платформе и дистрибутивно контролног сервера статусна иконица променити тако да рефлектује нови статус везе.

Класичан приступ који се користи за ажурирање листа је периодично освежавање листа превлачењем комплетних листа са сервера, што има више негативних аспеката:

- Значајна потрошња пропусног опсега сервера
- Оптерећење сервера периодичним упитима
- Заузимање комуникационог канала са сервером
- Брзина приказа промене зависи од учестаности повлачења листе

Подаци који се преносе за приказ једне листе су организовани на начин приказан на слици 14. Мерења дефинисана у секцији 4.1.2 а чији су резултати дати у секцији 6.1.3 показују количину података која се преноси зависно од броја елемената листе. Наведена мерења такође показују да *GWT* платформа показује боље карактеристике од *Errai* платформе код преноса листа, што је узето у обзир код дефинисања унапређеног решења за освежавање листа.

Подсистем за манипулацију листама унапређује механизам освежавања листа тако што омогућава да се једном учитане листе мењају само када до промене приказаних података заиста дође, без потребе за константним

освежавањем. Ово се постиже комбиновањем *GWT RPC* и *Errai* технологија на следећи начин:

1. *GWT RPC* механизам се користи за иницијално учитавање листе у веб апликацију. Овај механизам је изабран због ефикаснијег коришћења пропусног опсега и веће брзине у односу на *Errai RPC* механизам, као што је показано у секцији 6.1.3.
2. Веб апликација се потом претплаћује код подсистема за манипулацију листама за добијање информација о промени података у листи коришћењем *Errai* механизма издавач/претплатник
3. Делови дистрибутивно-контролног сервера који врше измене над подацима садржаним у листама емитују информације о променама података (укључујући и саме промене) преко *Errai* сервиса
4. Емитоване промене података достављају се претплаћеним класама веб апликације, које освежавају листу уколико су емитовани подаци део приказане листе.

Описани механизам значајно смањује коришћење пропусног опсега сервера као и оптерећење сервера, при чему је време од настајања промене до приказа у веб апликацији изузетно мало захваљујући константној вези коју *Errai* механизам одржава између веб апликације и сервера.

5.3.2 Архитектура клијентске контролне апликације

Клијентска контролна апликација представља посредника између дистрибутивно-контролног сервера и апликације за репродукцију рекламних садржаја. Основне функционалности ове апликације су следеће:

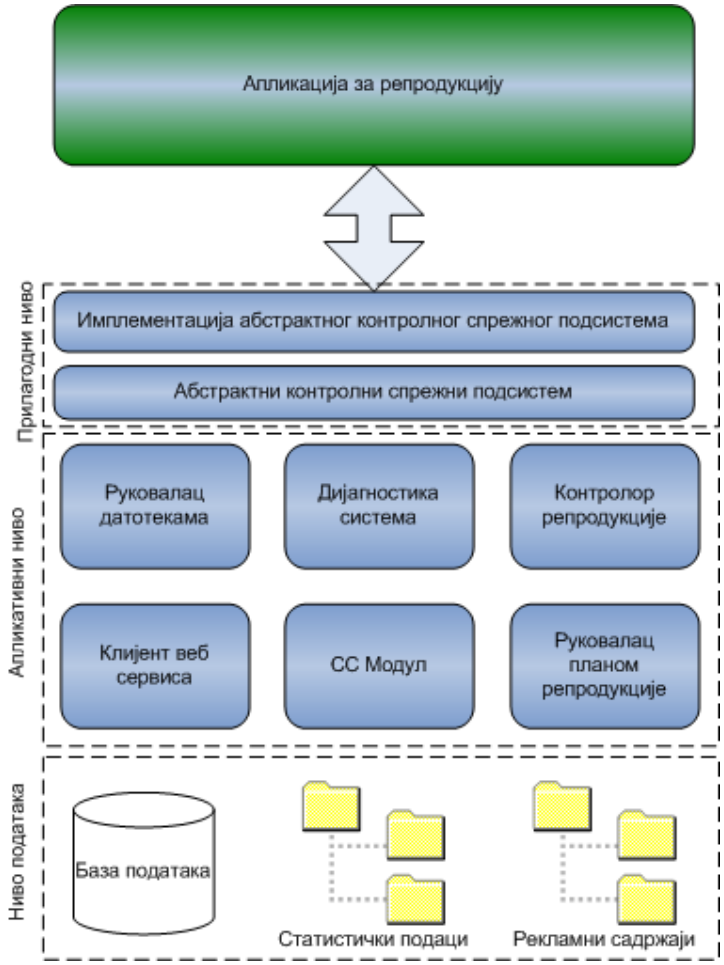
- Преузимање рекламних садржаја са дистрибутивно-контролног сервера и њихова достава апликацији за репродукцију рекламних садржаја
- Управљање радом и контрола апликације за репродукцију рекламних садржаја и то:
    - покретање, заустављање, поновно покретање (енгл. *restart*)
    - контрола репродукције према плану репродукције
    - дијагностика уређаја који су део рекламног система

- Достава статистичких и других датотека дистрибутивно-контролном серверу

Структура клијентске контролне апликације приказана је на слици 26. Апликација се, како је приказано на слици 26, састоји од 3 нивоа:

1. Ниво података, где спадају база података и одговарајући систем складиштења датотека, како долазних тако и одлазних
2. Апликативни ниво, у коме је имплементирана логика клијентске контролне апликације која управља апстрактним моделом апликације за репродукцију
3. Прилагодни ниво, који од стране апликативног нивоа прима команде из стандардног сета команди за контролу репродукције и дијагностичких команди, прилагођава их расположивом *API*-ју апликације за репродукцију и извршава.

Детаљнији опис делова клијентске контролне апликације дат је у наредним секцијама.



Слика 26 - структура клијентске контролне апликације

5.3.2.1 Структура нивоа података

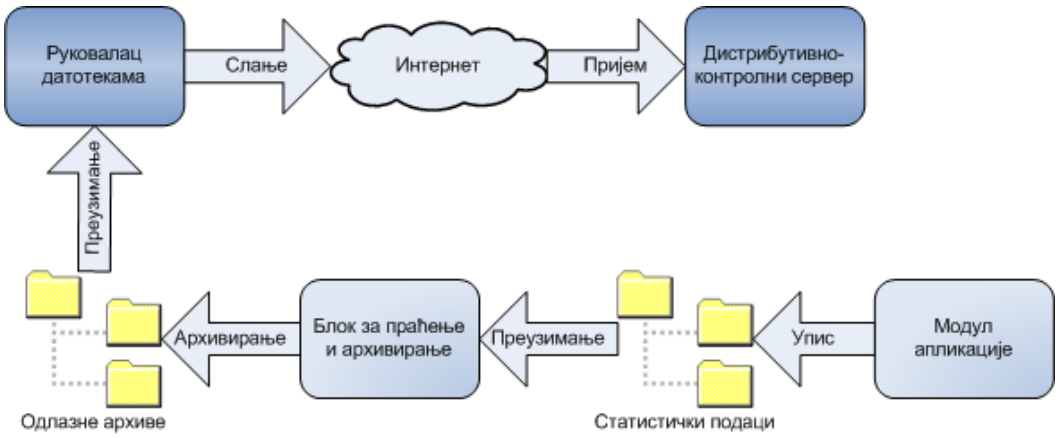
Ниво података клијентске контролне апликације састоји се од базе података и адекватно организованог система датотека. Узевши у обзир потребу за независношћу од платформе и минималним ангажовањем ресурса клијентског рачунара, база података на коју се ослања ниво података је *SQLite* ([SQLite]). У бази података чувају се следеће информације:

- Модел плана репродукције, који укључује временски распоред репродукције и информације о припадајућим рекламним садржајима
- Листу долазних команди
- Листу одлазних резултата и промена

- Податке о тренутном стању апликације потребне за успостављање претходног стања у случају неочекиваног прекида рада (услед грешке у апликацији, нестанка струје, искључења рачунара и слично).

Систем датотека се састоји од две целине – прва се састоји од скупа директоријума са датотекама које садрже статистичке податке, док друга садржи организован скуп директоријума у којима се налазе рекламни садржаји преузети са дистрибутивно-контролног сервера.

Слика 27 показује животни циклус статистичких података. Модули апликације уписују статистичке податке у одговарајуће датотеке, које у тренутку достизања одређене величине бивају означене као спремне за слање. Директоријуме у којима се налазе датотеке са статистичким подацима надгледа блок за праћење и архивирање, и у тренутку проналажења датотека спремних за слање, врши њихово архивирање, пребацивање архиве у директоријум одлазних архива и брисање из полазног директоријума. Руковалац датотекама бива обавештен о постојању нове архиве, преузима је из директоријума одлазних архива и шаље на дистрибутивно-контролни сервер, где она бива прихваћена и обрађена од стране подсистема за извештавање (секција 5.3.1.7)



Слика 27 - креирање, прикупљање и слање статистичких података

Рекламни садржаји преузети од дистрибутивно-контролног сервера чувају се у једноставно организованом скупу директоријума. Сваки рекламни пакет се налази у посебном директоријуму, при чему су директоријуми свих преузетих рекламних пакета на истом хијерархијском нивоу. Сви припадајући рекламни садржаји рекламног пакета налазе се у истом директоријуму.

5.3.2.2 Блокови апликативног нивоа

Овај ниво клијентске контролне апликације имплементира суштинску функционалност апликације, при чему је очувана потпуна независност од специфичности *API*-ја апликације за репродукцију.

5.3.2.2.1 Руковалац датотекама

Овај модул је задужен за размену датотека са дистрибутивно-контролним сервером. Руковалац датотекама имплементира комплетан протокол слања и пријема датотека, од захтева преноса преко командног веб сервиса (секција 5.3.3.3.3) па до физичког трансфера према подсистему за рад са датотекама на страни дистрибутивно-контролног сервера (секција 5.3.1.4). У свим случајевима када остали модули захтевају трансфер датотека, користи се овај модул. Један од таквих примера је приказан на слици 27.

5.3.2.2.2 Дијагностика система

Овај модул прати параметре система, како клијентске контролне апликације тако и апликације за репродукцију и доставља информације о стању система дистрибутивно контролној апликацији. Достава дијагностичких података се врши на два начина:

- У случају промена које нису везане за уређаје већ за стање апликације, модул иницира слање података о промени (функција ***пријавaПромене*** командног веб сервиса описана у секцији 5.3.3.3.3).
- Достава листе уређаја и њихових параметара у случајевима наведеним у секцији 5.3.1.8).

Информације о уређајима овај модул добија посредством прилагодног нивоа, при чему је коришћени модел уређаја и њихових параметара стања описан у секцији 5.3.4.4.

5.3.2.2.3 Контролор репродукције

Сврха овог модула је извршење команди за контролу репродукције рекламних садржаја директно добијених од стране дистрибутивно-контролног сервиса. Команде за контролу репродукције које овај модул може добити на

поменути начин описане су у табели 6 у секцији 5.3.3.3.2. Добијене команде овај модул доставља апликацији за репродукцију посредством прилагодног нивоа описаног у секцији 5.3.2.3.

5.3.2.2.4 Клијент веб сервиса

Овај модул преузима команде (по потреби периодичним повлачењем), шаље резултате и иницира остале позиве функција контролног веб сервиса по налогу других модула. Све команде добијене путем контролног веб сервиса овај модул додаје у листе чекања, које се ради опоравка стања апликације у случајевима прекида рада чувају у бази података. Модул потом добијене команде доставља одговарајућим модулима, који након извршења достављају резултате извршења. Клијент веб сервиса резултате извршења команди чува у одлазним листама које су такође смештене у бази података, где остају до успешне доставе дистрибутивно-контролном серверу.

5.3.2.2.5 Сигнално-статусни (СС) Модул

У спрези са СОВ сервисом, овај модул обезбеђује брзу доставу информације о постојању команди за клијентску контролну апликацију на дистрибутивно-контролном серверу, као и праћење статуса везе клијентске контролне апликације од стране СОВ сервиса а самим тим и дистрибутивно-контролног сервера. Модул успоставља трајну *TCP* везу са СОВ сервисом преко које се врши сигнализација. По пријему сигналних порука, овај модул иницира преузимање команди преко командног веб сервиса. У случају неуспешног успостављања *TCP* везе са СОВ сервисом, овај модул сигнализира клијенту веб сервиса да пређе у режим пријема команди методом периодичног повлачења.

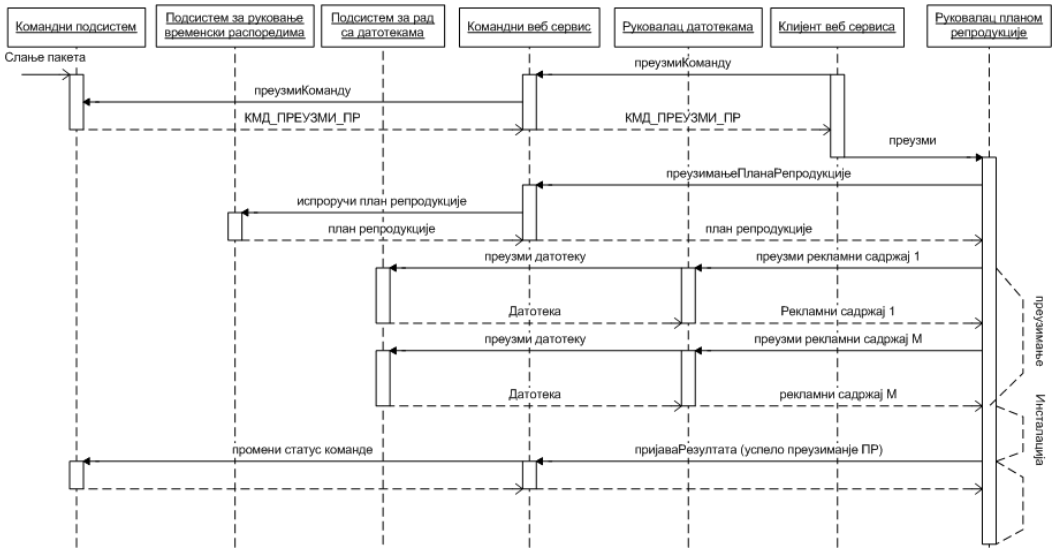
5.3.2.2.6 Руковалац планом репродукције

Овај модул је задужен за обављање свих функција везаних за план репродукције:

- Руковођење преузимањем плана репродукције са пратећим садржајима
- Извршење плана репродукције
- Уклањање плана репродукције

Извршења плана репродукције се врши постављањем временских окидача (енгл. *timer*) који у задатим тренуцима покрећу или заустављају жељене рекламне пројекте.

Низ корака који резултују преузимањем и покретањем плана репродукције приказан је на слици 28. Процес је иницран од стране корисника, који из веб апликације задаје команду за дистрибуцију плана репродукције. На страни клијентске контролне апликације, команду преузима клијент веб сервиса, који покреће руковаоца планом репродукције на основу добијене команде. У првом кораку, руковалац планом репродукције преузима план репродукције заједно са информацијама о свим укљученим рекламним садржајима. Руковалац планом репродукције одређује скуп садржаја које треба преузети (искључује оне који су раније преузети) и започиње фазу од M преузимања датотека, ослањајући се на руковалац датотекама, који врши преузимање датотека са дистрибутивно-контролног сервера.



Слика 28 - кораци преузимање плана репродукције са рекламним пакетима

По завршетку фазе преузимања, рекламни садржаји се из привремених директоријума пребацују у одредишне директоријуме и подаци о њима и плану репродукције уписују се у базу података, чиме се комплетира корак инсталације. Успех инсталације се пријављује кроз командни веб сервис, а потом се у одговарајућем тренутку, дефинисаном у плану репродукције, почиње са репродукцијом рекламних пакета.

5.3.2.3 Прилагодни ниво

Сврха увођења прилагодног нивоа је постизање што већег степена независности клијентске контролне апликације од апликације за репродукцију. Ово се постиже дефинисањем апстрактног контролног интерфејса који клијентска контролна апликација користи за управљање било којом апликацијом за репродукцију, док се имплементација овог интерфејса мења зависно од конкретно коришћене апликације за репродукцију. Апстрактни контролни интерфејс дефинише функције приказане у табели 5.

Функција	Опис
провераПодршке	За сваку од функција наведених испод, ова функција проверава да ли је имплементирана у датом систему.
репродукуј	Започињање/наставак претходно паузиране репродукције
заустави	Потпуно заустављање репродукције.
паузирај	Паузирање репродукције уз могућност настављања
преузмиСтањеРепродукције	Даје тренутно стање репродукције (у току, паузирано, заустављено)
идиНаТренутак	Скок на одређени тренутак у времену репродукције, уколико је подржано од стране уређаја за репродукцију
преузмиВремеРепродукције	Преузимање тренутка репродукције (време од почетка репродукције, брзина репродукције урачунана)
поставиБрзинуРепродукције	Убрзавање/успоравање репродукције
преузмиБрзинуРепродукције	Преузимање тренутне брзине репродукције садржаја
брзиСтатусУређаја	Преузимање сажетог статуса стања уређаја, који показује да ли има неисправних уређаја или проблема у току репродукције

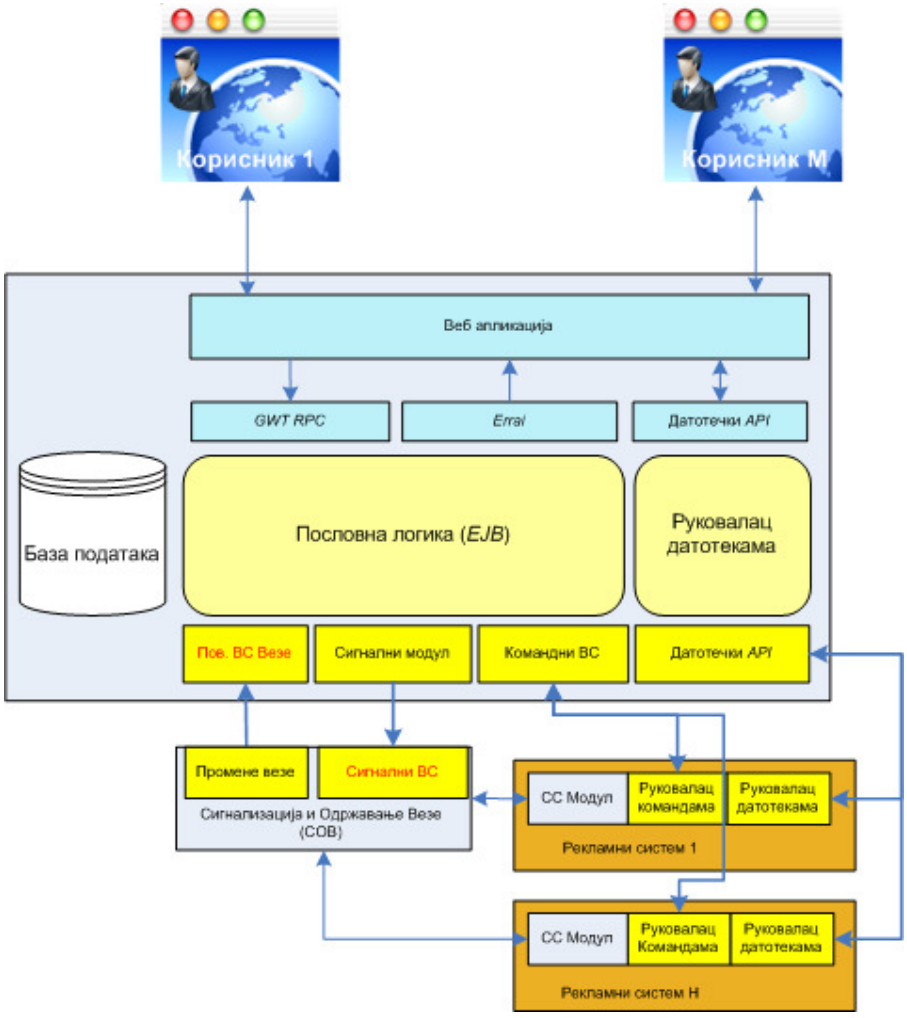
преузмиЛистуУређаја	Преузимање комплетне листе уређаја са тренутним параметрима.

Табела 5 - списак функција апстрактног контролног интерфејса

Имплементација апстрактног контролног интерфејса једини је део клијентске контролне апликације који треба мењати зависно од апликације за репродукцију. Зависно од функционалности које подржава апликација за репродукцију, поједине функције дате у табели 5 могу бити недоступне. Провера доступности сваке појединачне функције постиже се функцијом ***провераПодршке***, којој се као параметар прослеђује име функције која се проверава. Тако, на пример, уколико би апликација за репродукцију био прегледач *PDF* докумената, функција ***идиНаТренутак*** не би била доступна, док би била доступна у случају прегледача видео садржаја.

5.3.3 Архитектура спрежних подсистема

Слика 22 показује да дистрибутивно-контролни сервер има два основна спрежна подсистема: са веб апликацијом и клијентском контролном апликацијом. Слика 29 даје детаљан приказ свих спрежних подсистема дистрибутивно-контролног сервера са осталим компонентама система.



Слика 29 - детаљан приказ спрежних подсистема

Основа комуникације са удаљеним клијентским контролним апликацијама се заснива на шаблону захтев-одговор (енгл. *request – response pattern*). Код овог шаблона комуникације, дистрибутивно-контролни сервер доставља удаљеним клијентским контролним апликацијама посебно форматиране команде које оне извршавају и потом враћају резултате извршења. Због природе система у коме клијентске контролне апликације не морају бити у константној вези са

дистрибутивно-контролним сервером, команде се до испоруке складиште у одлазним листама које су имплементиране у виду табела у бази података. Исти принцип важи и за одговоре клијентских контролних апликација, које у тренутку завршетка извршења команде не морају бити у вези са дистрибутивно-контролним сервером али морају бити у могућности да резултате извршења команде доставе у тренутку када дође до успоставе везе.

Систем је пројектован тако да се достава команди клијентским контролним апликацијама може вршити на два начина:

1. Периодичним повлачењем команди од стране клијентских контролних апликација путем веб сервиса (енгл. *web service pooling*). Ова комуникација се одвија између блока ***Командни ВС*** на страни дистрибутивно-контролног сервера и блока ***Руковалац командама*** на страни клијентске контролне апликације.
2. Коришћењем трајне везе између клијентске контролне апликације и сервиса за сигнализацију и одржавање везе. Ова веза се користи за испоруку сигнала о постојању команде коју клијентска контролна апликација потом преузима путем веб сервиса (COB сигнализација). Везу са COB сервером одржава ***CC модул*** који уједно и прихвата сигнал постојања нове команде, док са стране дистрибутивно-контролног сервера ова информација стиже до COB сервера кроз ***сигнални веб сервис***.

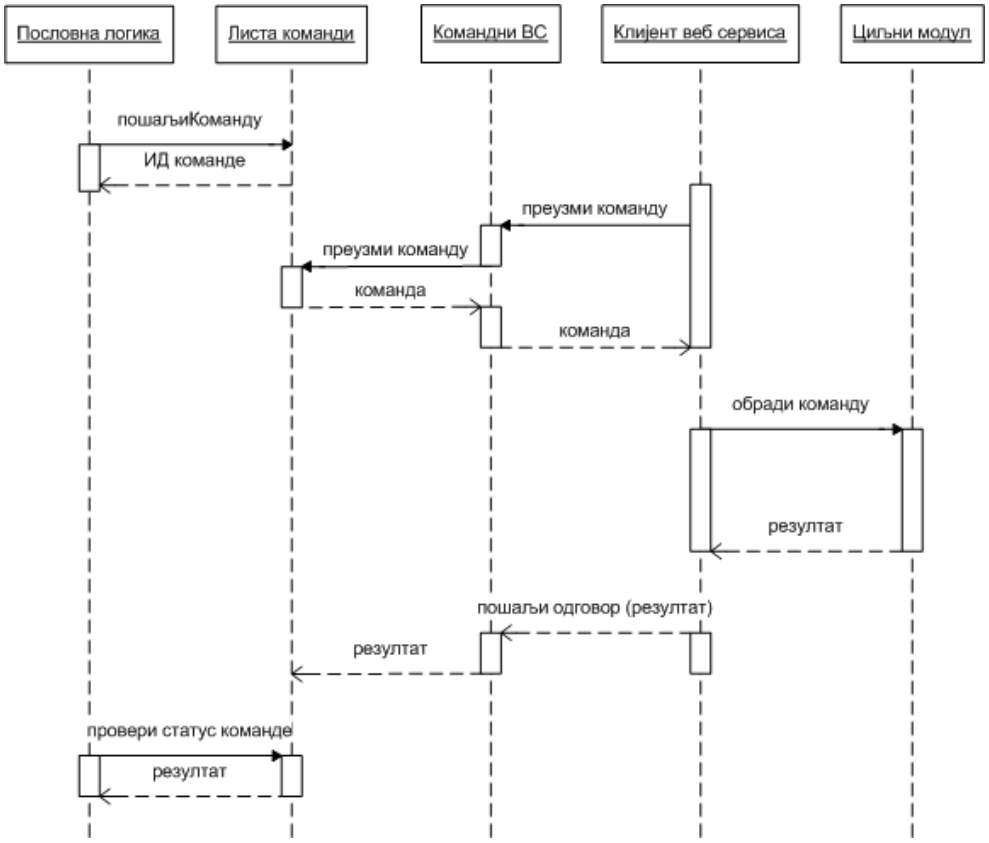
Комуникација између пословне логике имплементиране на дистрибутивно-контролном серверу и веб апликације је такође двојака и укључује:

1. Комуникацију засновану на позивима удаљених функција (*RPC* имплементација *GWT* развојног окружења)
2. Комуникација заснована на размени порука која се базира на моделу издавач-претплатник (енгл. *publish/subscribe*) која је део *Errai* комуникационе библиотеке [ERRAI]

Заједнички спрежни подсистем који се користи и од стране веб апликације и од стране клијентске контролне апликације је руковалац датотекама, чија је сврха да омогући достављање датотека на сервер и њихово преузимање са сервера. У наредним поглављима биће дати детаљи свих спрежних подсистема.

5.3.3.1 Достављање команди клијентској контролној апликацији методом периодичног повлачења

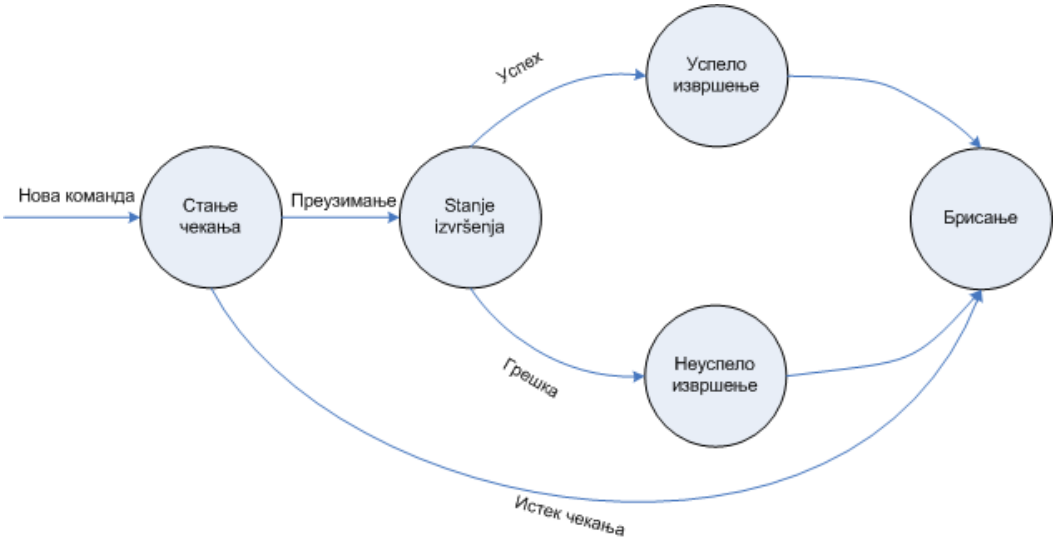
Слика 30 приказује секвенцу доставе и извршења једне команде методом периодичног повлачења команди од стране клијентске контролне апликације.



Слика 30 - секвенца команда-одговор методом периодичног повлачења

У првом кораку, један од серверских модула додаје команду (***пошаљиКоманду***) у листу команди, при чему је команда адресирана на одређену клијентску контролну апликацију. Као резултат додавања команде у листу добија се јединствени идентификатор команде преко кога се може пратити статус њеног извршења. Клијентска контролна апликација из своје главне комуникационе петље периодично позива веб сервис и проверава има ли команди за преузимање (преузми команду). Веб сервис преузима команду адресирану на дату клијентску контролну апликацију из листе команди и доставља је. Главна петља клијентске контролне апликације (која је део клијента веб сервиса) доставља примљену команду модулу који је задужен за њено извршење. По извршењу команде, модул формира одговор на команду, при чему

се референцирање на команду врши преко јединственог идентификатора (ИД) команде. Формирани одговор се потом доставља дистрибутивно-контролном серверу позивом одговарајуће функције командног веб сервиса (функција ***пријавaРезултата*** описана у секцији 5.3.3.3.3). Командни веб сервис по пријему одговора проналази изворну команду, мења јој статус зависно од резултата извршења, чува резултат у бази података и евентуално обавештава одређене модуле серверске апликације о пристизању резултата извршења.



Слика 31 – дијаграм стања команди

Слика 31 показује дијаграм стања команди, од тренутка креирања до брисања после успешног или неуспешног извршења. Новокреирана команда у листи команди се налази у стању чекања до тренутка када буде преузета од стране клијентске контролне апликације. Од тренутка преузимања, команда прелази у стање извршења, и зависно од резултата извршења достављеног од стране клијентске контролне апликације, прелази у стање успелог или неуспелог извршења. По обради резултата извршења, команда се брише из листе. Команда може бити избрисана из листе и у случају када истекне дозвољено време чекања на преузимање команде, које зависи од врсте команде.

Описани начин достављања команди је поуздан и једноставан за имплементацију, али има и следеће недостатке:

- Постоји кашњење у испоруци команде које зависи од периода повлачења комади

- Периодично повлачење од стране великог броја клијената представља значајно оптерећење за сервер (резултати мерења оптерећења дати су у секцији 6.2.3).
- Периодично повлачење захтева пренос значајне количине података, од које је веома мали део користан.

Решење поменутих проблема постиже се ослањањем на COB сервис описан у наредној секцији.

5.3.3.2 Достављање команди путем COB сервиса

COB платформа је специјализовани сервис за комуникацију са великим бројем клијената који обезбеђује следеће функционалности:

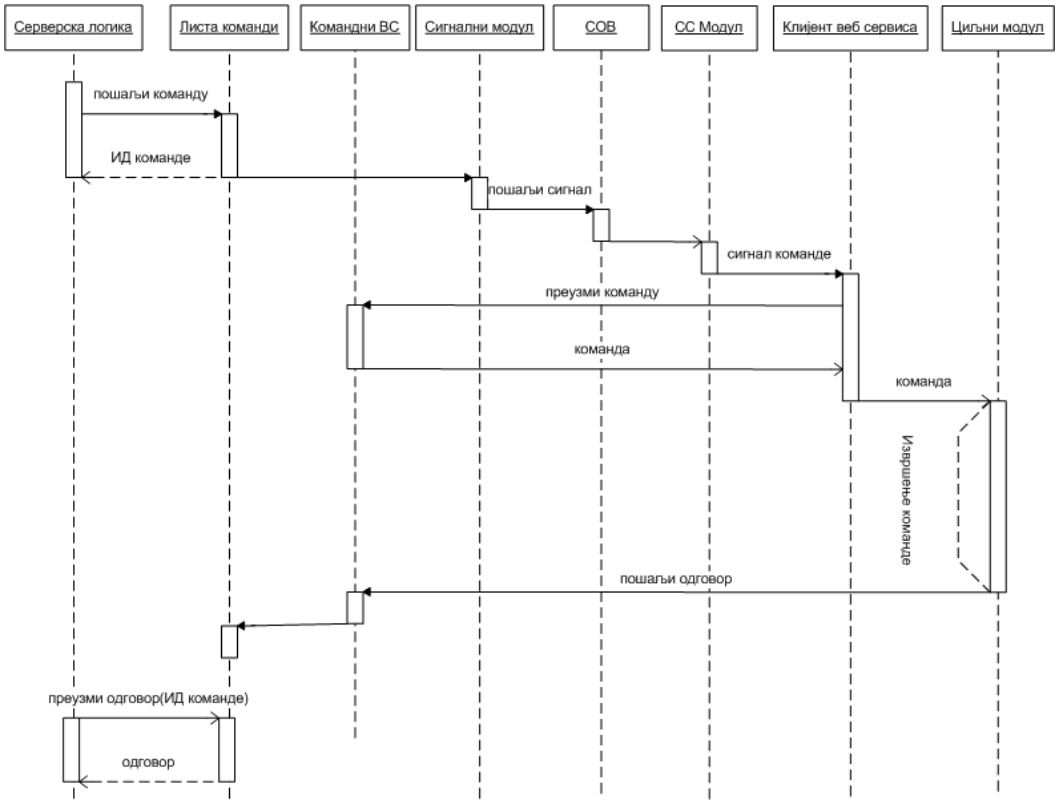
- Брзу испоруку сигналних порука великом броју клијената
- Доставу информација о стању везе клијената претплаћеним серверским платформама путем веб сервиса

COB платформа се састоји од серверске (COB сервис) и клијентске компоненте (Сигнално Статусни - CC модул).

Сигнално статусни модул је интегрисан у клијентску контролну апликацију као додатна библиотека која успоставља везу са COB сервисом и доставља сигналне информације клијентској контролној апликацији.

Спрега између COB сервиса и дистрибутивно-контролног сервера је нешто сложенија и успоставља се преко једног веб сервиса на страни COB сервера и једног веб сервиса на страни дистрибутивно-контролног сервера.

У контексту доставе команди од дистрибутивно-контролног сервера до клијентске контролне апликације, COB омогућава елиминисање периодичног повлачења команди од стране клијентске контролне апликације на тај начин што се путем COB сигнализације она обавештава о постојању команде. Клијентска контролна апликација стога позива командни веб сервис само у случају када је добила сигнал постојања команде, чиме се елиминише непотребно оптерећење дистрибутивно-контролног сервера услед периодичног повлачења. Такође, будући да CC модул успоставља трајну *TCP* везу са COB сервисом, сигнализација функционише веома брзо, чиме се време испоруке команди драстично скраћује.



Слика 32 - испорука команди уз сигнализацију путем СОВ сервиса

Слика 32 приказује секвенцу испоруке команде уз ослањање на сигнализацију од стране СОВ сервиса. У овом случају, серверски модул на идентичан начин као и код првог метода комуникације доставља команду у листу команди, уз додатни корак слања сигнала постојања команде путем сигналног модула. Сигнални модул позива сигнални веб сервис на страни СОВ сервера, који сигнал доставља СС модулу адресираног клијента. СС модул иницира позив командног веб сервиса којим се преузима команда. Обрада команде и слање одговора је надаље исто као и у случају комуникације са периодичним повлачењем команди.

Овај начин комуникације решава све претходно побројање недостатке првог начина комуникације, уз нешто сложенију имплементацију пословне логике на страни дистрибутивно-контролног сервера. У случају престанка рада СОВ сервера, дистрибутивно-контролни сервер једноставно може прећи у режим одржавања везе методом периодичног повлачења команди.

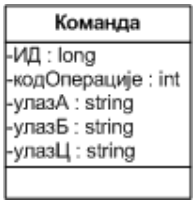
5.3.3.3 Командни веб сервис

Осим преузимања команди и достављања одговора на команде, командни веб сервис се користи и за размену додатних информација између клијентске контролне апликације и дистрибутивно-контролног сервера:

- Ауторизацију слања/преузимања датотека
- Дијагностику уређаја на страни клијентске контролне апликације
- Доставу планова репродукције
- Праћење тренутног стања клијентске контролне апликације

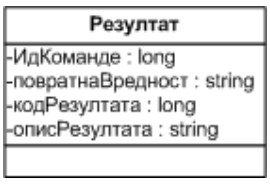
5.3.3.3.1 Структура серверске команде и клијентског одговора

Команда која се испоручује клијентској апликацији представљена је једноставном класом ***Команда*** која омогућује довољан степен апстракције како би се њоме могле представити све потребне команде и пренети сви потребни параметри.



Слика 33 - структура серверске команде

Слика 33 приказује сва поља од којих се команда састоји. Поље ***ИД*** представља јединствени идентификатор команде, који се користи кад је потребно успоставити референцу на команду (на пример, у случају доставе резултата извршења команде). Поље ***кодОперације*** дефинише операцију коју треба извршити. Листа свих команди које систем подржава дата је у секцији 5.3.3.3.2. Преостала три поља, ***улазА***, ***улазБ*** и ***улазЦ*** се користе за пренос параметара команде и њихов садржај у потпуности зависи од саме команде.



Слика 34 - структура резултата извршења команде

Свака команда која се достави клијентској контролној апликацији се извршава од стране одговарајућег модула клијентске контролне апликације и

резултат извршавања се смешта у објекат класе ***Резултат*** која се доставља дистрибутивно-контролном серверу (Слика 34). Поље ***ИдКоманде*** се користи за референцирање на команду чији је резултат извршења садржан у објекту и једнако је вредности поља ***ИД*** референциране команде. Сам резултат извршења команде садржан је у пољу ***повратнаВредност***. Уколико је код извршења команде дошло до грешке, њен код се смешта у поље ***кодРезултата***, уз детаљан опис грешке који се смешта у поље ***описРезултата***. Уколико је извршење команде било успешно, код грешке је постављен на вредност 0 а опис резултата се може користити за пренос додатних вредности.

5.3.3.3.2 Листа команди

Управљање радом клијентске контролне апликације је у потпуности изведено помоћу команди. Табела 6 приказује скуп свих команди груписаних према намени.

Команде за контролу рада апликације за репродукцију	
Команда	Опис
КМД_ЗАУСТАВИ_АР	Потпуно заустављање апликације за репродукцију
КМД_ПОКРЕНИ_АР	Покретање апликације за репродукцију
КМД_СТАТУС_АР	Захтева слање тренутног стања апликације за репродукцију
Команде за контролу репродукције рекламних садржаја	
КМД_ПОКРЕНИ_РП	Покретање репродукције одређеног рекламног пакета, који не мора бити на реду по плану репродукције
КМД_НАСТАВИ_ПР	Настављање (ако је претходно паузиран) или прво покретање плана репродукције који је тренутно активан
КМД_ПАУЗИРАЈ	Паузира репродукцију тренутно репродукованог рекламног пакета, која се потом може наставити. После ове команде игнорише се активни план репродукције. Шаље команду паузе апликацији за репродукцију.
КМД_ЗАУСТАВИ	Зауставља тренутну репродукцију (шаље команду заустављања апликацији за репродукцију). После ове команде игнорише се активни план репродукције. Поновно покретање рекламног садржаја подразумева репродукцију од почетка.
Команде за дистрибуцију садржаја	
КМД_ПРЕУЗМИ_ПР	Иницира преузимање задатог плана репродукције од

	стране клијентске контролне апликације. Тек по преузимању свих рекламних садржаја и плана репродукције прекида се извршење тренутног плана репродукције и покреће се преузети.
КМД_УКЛОНИ_ПР	Уклањање тренутног плана репродукције, подразумева његово заустављање и уклањање свих рекламних садржаја.
Команде за дијагностику и опоравак система	
КМД_ПУНА_ДИЈАГНОСТИКА	Захтева слање листе свих уређаја и њихових статуса на дистрибутивно-контролни сервер
КМД_РЕСТАРТ	Захтева поновно покретање рачунара на коме се клијентска контролна апликација извршава
КМД_РЕСЕТ	Захтева брисање свих рекламних садржаја, привремених датотека, логова и довођење система у иницијално стање (стање након прве инсталације система).
КМД_ПОШАЉИ_ДАТОТЕКУ	Захтева слање наведене датотеке на дистрибутивно-контролни сервер. Обично се користи за преузимање логова клијентске контролне апликације или апликације за репродукцију.
Помоћне команде	
КМД_ПОСТАВИ_ВРЕДНОСТ	Користи се за постављање неког од параметара клијентске контролне апликације (на пример, период повлачења команди)
КМД_ПОШАЉИ_ВРЕДНОСТ	Захтева слање вредности неког од конфигурационих параметара клијентске контролне апликације.
КМД_ПРЕУЗМИ_ДАТОТЕКУ	Захтева од клијентске контролне апликације да преузме наведену датотеку и сачува је на задатој локацији

Табела 6 - скуп подржаних команди

5.3.3.3.3 Функције командног веб сервиса

Детаљи свих функција којима командни веб сервис располаже дати су у овој секцији.

Функција	**пријава**
Параметри	
Стринг ауторизационИКод	Ауторизациони код клијентске контролне апликације који омогућава приступ систему

Стринг клијентИд	Јединствени ИД клијента који одређује одређену инстанцу клијентске контролне апликације. Једна од могућности је коришћење серијског броја хард диска рачунара.
Стринг име	Име под којим ће се клијент појављивати у листи активних клијената
Пов. вредност	Стринг
Опис	Пре могућности обављања било какве операције са веб сервисом, клијентска апликација мора да се аутентификује и добије важећи идентификатор сесије. Надаље се идентификатор сесије користи код позива свих других функција веб сервиса.

Функција	**одјава**
Параметри	
Стринг идСесије	Идентификатор сесије добијен позивом функције ***пријава***
Пов. вредност	нема
Опис	Позива се у тренутку прекидања клијентске контролне апликације или у тренутку преласка у самостални режим рада. Стање везе клијентске контролне апликације се мења у бази.

Функција	**преузмиКоманду**
Параметри	
Стринг идСесије	Идентификатор сесије добијен позивом функције ***пријава***
Пов. вредност	Објекат класе ***Команда*** описане у секцији 5.3.3.3.1, уколико има следеће команде. Уколико команде нема, *null* вредност
Опис	Преузимање задате команде и одржавање статуса везе (у случају одржавања везе методом периодичног повлачења команди)

Функција	**пријаваРезултата**

Параметри	
Стринг идСесије	Идентификатор сесије добијен позивом функције ***пријава***
Резултат резултат	Резултат извршења задате команде
Пов. вредност	Стринг
Опис	Достављање резултата извршења преузете команде, на начин описан у секцији 5.3.3.3.1

Функција	**пријаваПромене**
Параметри	
Стринг идСесије	Идентификатор сесије добијен позивом функције ***пријава***
Промена промена	Подаци о промени насталој на страни клијентске контролне апликације. Класа ***Промена*** је веома слична класи ***Команда*** и садржи код промене уз два параметра опште намене у којима се, зависно од типа промене, преносе вредности специфичне за ту промену.
Пов. вредност	нема
Опис	Достављање информације о битној промени на страни клијентске контролне апликације. Примери промене могу бити неочекиван прекид репродукције рекламног садржаја или појава грешке на неком од уређаја клијентског система.

Функција	**захтевПреноса**
Параметри	
Стринг идСесије	Идентификатор сесије добијен позивом функције ***пријава***
логичка слање	*true* за слање, *false* за преузимање датотеке
логичка група	*true* за преузимање/слање групе датотека. У том случају, добијени ауторизациони стринг траје предефинисани период времена.
Пов. вредност	Стринг Ауторизациони стринг са којим је дати пренос могуће обавити преко датотечког *API*-ја
Опис	Свако достављање или преузимање датотеке преко

	датотечког API-ја мора се на овај начин ауторизовати пре самог трансфера.

Функција	**преузимањеПланаРепродукције**
Параметри	
Стринг идСесије	Идентификатор сесије добијен позивом функције ***пријава***
лонг идПлана	ИД плана који треба преузети, задат командом КМД_ПРЕУЗМИ_ПР дефинисаном у табели 6
Пов. вредност	ПланРепродукције – Објекат који у себи садржи комплетан план репродукције са листом коришћених рекламних садржаја
Опис	Преузимање детаља плана репродукције након добијања команде за инсталацију.

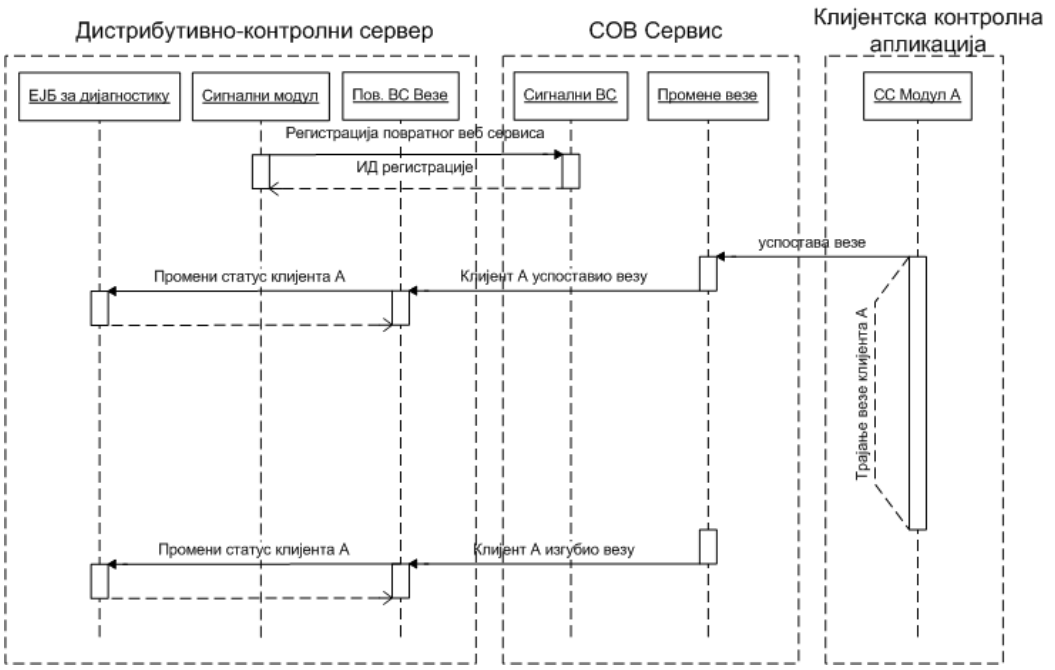
Функција	**достава**Дијагностике
Параметри	
Стринг идСесије	Идентификатор сесије добијен позивом функције ***пријава***
Уређај[] листаУређаја	Листа објеката који описују стање уређаја. Детаљи коришћеног модела дати су у секцији 5.3.4.4
Пов. вредност	нема
Опис	Комплетно стање свих уређаја у систему. Шаље се код покретања клијентске контролне апликације, одмах иза пријаве на сервер или на експлицитан захтев сервера (команда КМД_ПУНА_ДИЈАГНОСТИКА)

5.3.3.4 Праћење статуса везе клијентских контролних апликација

Један од елемената дијагностике система је и континуирано праћење расположивости клијентских контролних апликација (клијената). Као и у претходно описаном случају доставе команди, праћење расположивости клијената се може базирати на периодичном повлачењу команди или се ослонити на СОВ сервис.

У случају периодичног повлачења команди, сваки позив функције веб сервиса за проверу расположивости команди истовремено представља и потврду расположивости клијента у датом тренутку (функција ***преузмиКоманду*** описана у секцији 5.3.3.3.3). Стога се за сваку клијентску контролну апликацију бележи време задњег позива командног веб сервиса (било које функције). Успостављено је правило да се сматра да клијент није расположив уколико је од задњег позива веб сервиса прошло више од два интервала повлачења команди. Овакав приступ праћења расположивости клијената не уноси додатну комплексност у апликацију, али сви претходно наведени недостаци методе периодичног повлачења важе и у овом случају.

Коришћење СОВ сервиса за праћење расположивости клијената значајно смањује оптерећење серверске апликације уз тачније и брже праћење расположивости. Слика 35 приказује блокове који учествују у праћењу расположивости клијентске контролне апликације као и редослед позива код праћења стања једног клијента. СОВ сервис кроз сигнални веб сервис даје могућност регистрације повратних веб сервиса за праћење стања СС модула. Стога у првом кораку дистрибутивно-контролни сервер региструје једну или више клијентских контролних апликација чију расположивост треба пратити уз прослеђивање адресе свог веб сервиса (*Пов. ВС Везе*) који ће СОВ сервис позивати у случају промене расположивости посматраних клијената (принцип повратног позива – енгл. *CallBack*). СОВ сервис у сваком тренутку поседује информације о стању везе сваког СС модула, и само у случајевима промене статуса везе (успостављање или раскидање везе) путем повратних веб сервиса обавештава регистроване апликације о промени расположивости клијената.



Слика 35 - праћење везе преко СОВ сервиса

У датом примеру показано је да непосредно по успостављању и раскидању везе од стране СС клијента А, модул за праћење промена везе позива ***Пов. ВС везе***, који информацију о промени расположивости клијента доставља EJB-у за дијагностику који ову информацију уписује у базу података. На страни веб апликације намењеној дијагностици система, ова информација се користи за приказ статуса везе клијентских контролних апликација.

5.3.3.5 Комуникација са веб апликацијом

Комуникација између контролне веб апликације и серверске апликације се зависно од потребе одвија једним од два расположива канала комуникације – путем *RPC* (*GWT* спрежни подсистем) спреге или спреге засноване на размени порука (*Errai* спрежни подсистем).

Комуникација путем позива удаљених функција [GWTRPC] се користи у следећим случајевима:

- Када је потребно са серверске стране допремити иницијалне податке (листе садржаја, уређаја, планове репродукције...).

- Када је потребно на акцију корисника иницирати пословни процес на серверској страни (дистрибуцију плана репродукције, креирање извештаја...).
- Када је потребно урадити *CRUD* операције над подацима над којима апликација манипулише.

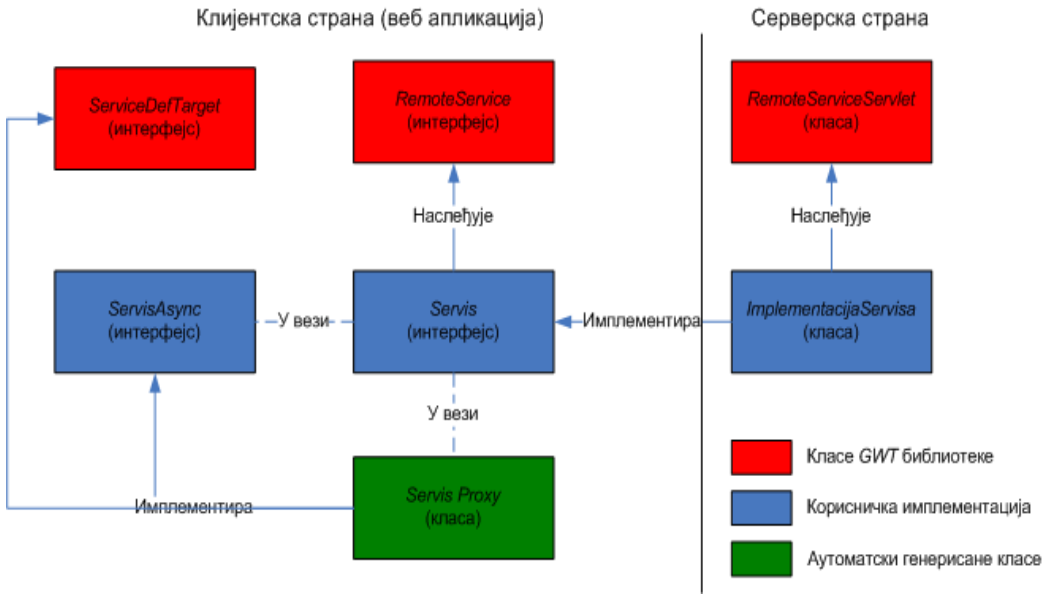
Највећи недостатак *RPC* комуникације је чињеница да се ради о једносмерној комуникацији која је увек иницирана од стране веб апликације. Стога се у случајевима када је потребно комуникацију иницирати са серверске стране користи *Errai* комуникациони подсистем. Типичне ситуације у којима се користи *Errai* комуникациони подсистем су следеће:

- Када је у веб апликацији потребно пратити напредак дуготрајних операција на серверској страни (креирање извештаја, пренос великих датотека)
- Када је потребно променити иницијално учитане податке у веб апликацији због промена на серверској страни
- Када је потребно пратити промене статуса уређаја за репродукцију у реалном времену.

Детаљи оба коришћена спрежна подсистема дати су у наредним подсекцијама.

5.3.3.5.1 GWT RPC механизам комуникације

Слика 36 приказује детаљан дијаграм релација класа и интерфејса којима се постиже *RPC* комуникација између веб апликације и серверске апликације. Полазна тачка у дефиницији *RPC* спреге је креирање *Java* интерфејса (енгл. *intrerface*) који декларише листу функција које ће бити позиване из веб апликације а имплементиране на серверској страни. На дијаграму приказаном на слици 36 то је интерфејс *Servis*. На серверској страни, њега имплементира класа *ImplementacijaServleta* која уједно наслеђује класу *RemoteServiceServlet*, која обезбеђује функционалност којом се из *HTTP* позива иницираног од стране веб апликације десеријализују подаци и прослеђују одговарајућој функцији класе која имплементира интерфејс.

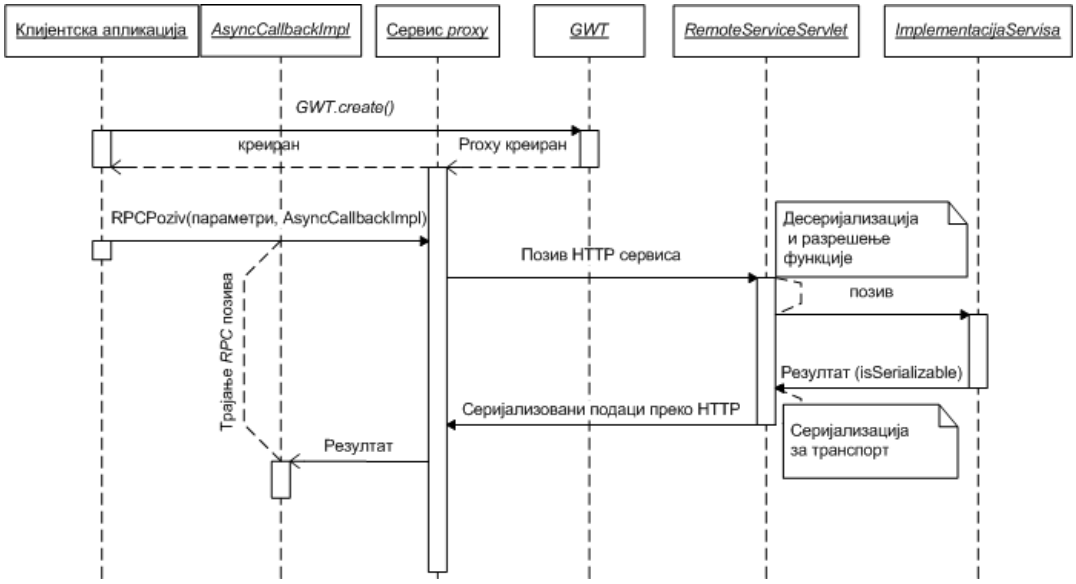


Слика 36 - дијаграм веза код *RPC* позива (извор [*GWTRPC*])

На клијентској страни, за сваки сервис интерфејс се креира и асинхрона верзија, која се разликује по томе што ни једна његова метода нема повратних вредности, већ се резултати враћају путем класе која имплементира *AsyncCallback* интерфејс и која се у свим функцијама асинхроног интерфејса прослеђује као задњи параметар.

Слика 37 приказује типичан низ корака код припреме и извршења *RPC* позива код *GWT* развојне библиотеке. Први корак је креирање сервис *proxy*-ја, класе која имплементира комуникациони протокол којим се физички спроводе *RPC* позиви. Инстанцирање ове класе врши се коришћењем статичке методе *GWT.create*, која као параметар прима сервис интерфејс а као резултат враћа *proxy* класу која имплементира асинхрони сервис интерфејс и која се даље користи за *RPC* позиве. У тренутку *RPC* позива, сервис *proxy* врши серијализацију параметара и осталих компоненти позива и тако серијализоване податке путем *HTTP* позива преноси сервлету који наслеђује *RemoteServiceServlet* а имплементира сервис интерфејс дефинисан на клијентској страни. Класа *RemoteServiceServlet* обезбеђује функционалност којом се серијализовани подаци прослеђени путем *HTTP* позива са клијентске стране десеријализују и разрешава се функција имплементираног интерфејса која се позива са задатим параметрима. Након извршења позване функције, резултат њеног извршења се поново серијализује и враћа као резултат *HTTP* позива.

Ограничење које овај начин комуникације је да сви параметри функција као и повратне вредности морају имати могућност серијализовања. На клијентској страни, по окончању *HTTP* позива и пријему резултата, *proxy* класа врши поновно десеријализовање резултата у објекат и прослеђује га класи која имплементира *AsyncCallback* интерфејс а која је прослеђена као параметар *RPC* позива, чиме се позив окончава. Уколико је током позива дошло до грешке и појаве изузетка (енгл. *Exception*), позива се одговарајућа функција класе која имплементира *AsyncCallback* уз прослеђивање детаља грешке.

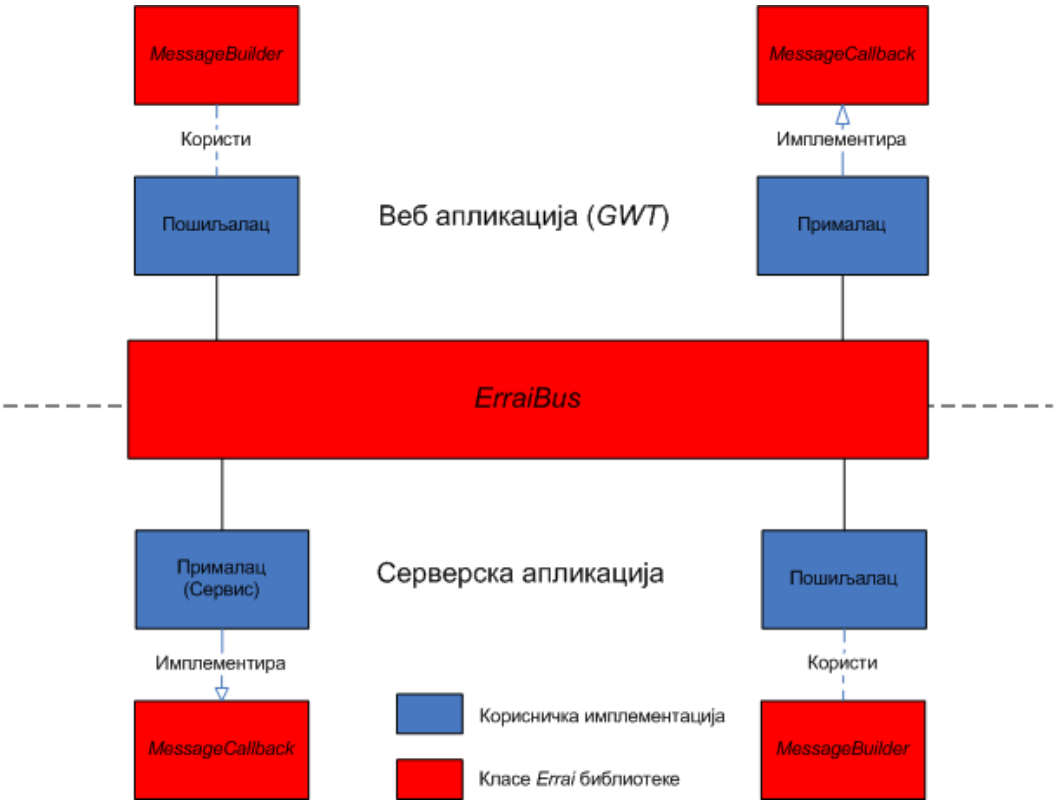


Слика 37 - ток *RPC* позива

5.3.3.5.2 Комуникација базирана на Errai библиотеци

За разлику од *GWT* библиотеке, код које се комуникација између серверске и клијентске апликације одвија преко *RPC* позива, *Errai* библиотека комуникацију заснива на размени порука по шаблону издавач – претплатник (енгл. *publisher-subscriber*). Слика 38 приказује основне елементе који се јављају у комуникацији заснованој на *Errai* библиотеци. *Errai bus* представља језгро система за размену порука, нудећи јединствени *API* и на серверској и на клијентској страни. Пошиљалац порука, било да је на серверској или клијентској страни, користи класу *MessageBuilder* за креирање порука које се потом шаљу коришћењем инстанце *MessageBus* класе, која се креира од стране *ErraiBus* класе. Подједнака униформност постоји и код пријема порука и на клијентској и

серверској страни – прималац поруке (сервис) имплементира интерфејс *MessageCallback* и прима све поруке на задату тему (енгл. *topic*).

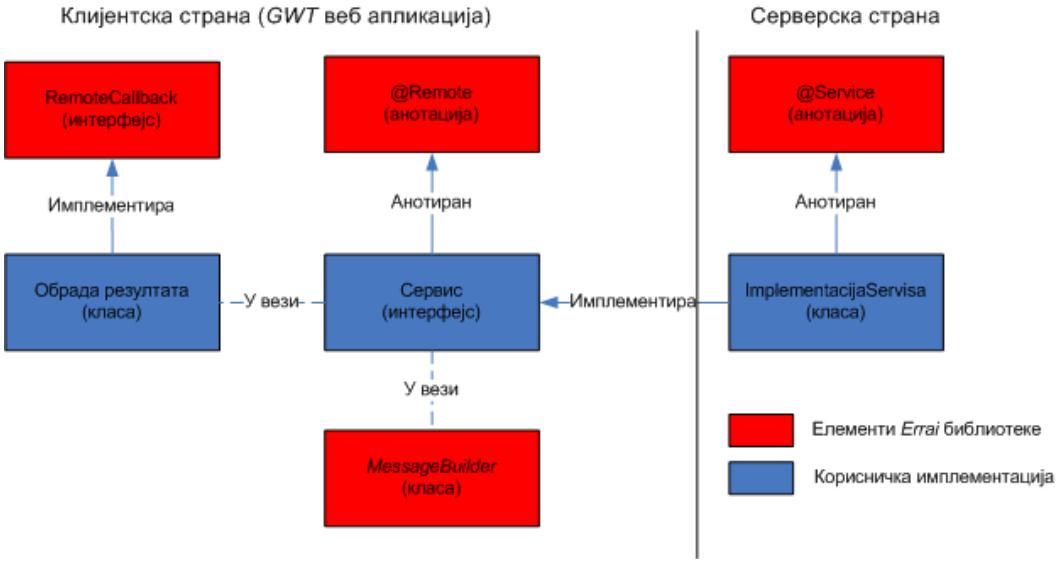


Слика 38 – дијаграм комуникације путем порука код *Errai* библиотеке

На бази основног комуникационог шаблона заснованог на порукама, *Errai* библиотека нуди надоградњу која омогућава *RPC* комуникацију између клијента и сервера. За разлику од комуникације порукама, која се може одвијати у било ком смеру, *RPC* комуникација је увек усмерена од стране веб апликације ка серверској апликацији. Резултати мерења перформанси *RPC* функционалности *Errai* библиотеке наспрам *GWT* библиотеке дати су у секцији 6.1.3.

Слика 39 приказује дијаграм класа, интерфејса и анотација које се користе за имплементацију *RPC* позива коришћењем *Errai* библиотеке. Као и код *GWT RPC* механизма, и код дефинисања *Errai RPC* позива полазна тачка је креирање сервис интерфејса, који дефинише све функције које се са стране веб апликације могу позвати на страни серверске апликације. Овај интерфејс се мора анотирати анотацијом *@Remote* дефинисаном у пакету *org.jboss.errai.bus.server.annotations Errai* библиотеке. За иницирање *RPC* позива се користи класа *MessageBuilder*, док се обрада резултата позива врши у класи која имплементира интерфејс

RemoteCallback. На серверској страни, класа која имплементира сервис интерфејс мора бити анотирана *@Service* анотацијом.



Слика 39 - дијаграм *RPC* позива коришћењем *Errai* библиотеке

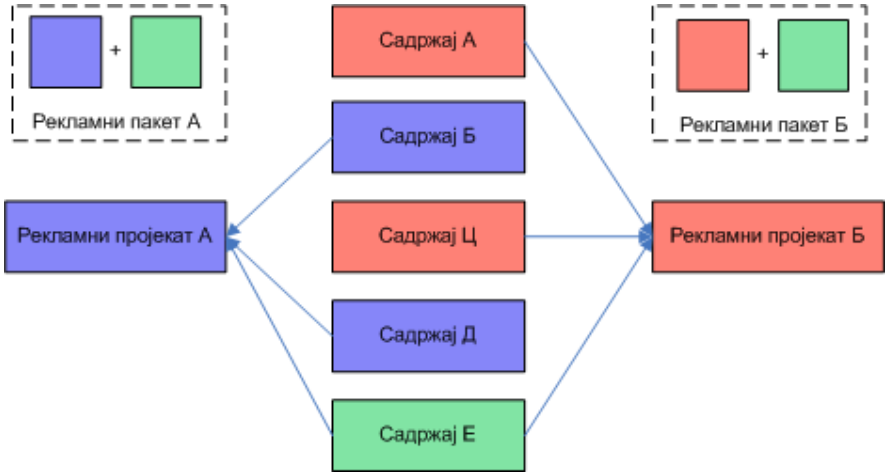
5.3.4 Основни модели података

Приликом креирања модела података вођено је рачуна о томе да се креира модел који на што апстрактнији начин репрезентује реалне податке, при чему опет ниво апстракције не би смео да смањи ефикасност коришћења пропусног опсега. Наредне секције представљају моделе података који су креирани тако да реше неке од постављених проблема, као што су независност рекламних садржаја од платформе, флексибилан план репродукције и платформски независан систем за дијагностику.

5.3.4.1 Структура дистрибуираних садржаја

Током развоја и експлоатације, захтеви и могућности рекламних система се мењају, те је један од битних циљева које треба остварити флексибилност система у погледу садржаја који се дистрибуирају. Платформа приказана у овом раду дистрибуиране садржаје организује користећи релациони модел, у коме се успоставља релација између једне главне датотеке (***рекламни пројекат*** у терминологији конкретног система) и пратећих датотека (***пратећи мултимедијални садржаји***). Рекламни пројекат и његови пратећи мултимедијални садржаји заједно чине ***рекламни пакет***. Код складиштења од

стране клијентске контролне апликације, све датотеке рекламног пакета су увек у истом директоријуму. Код репродукције рекламних пакета, апликацији за репродукцију се као параметар увек прослеђује рекламни пројекат. Овај приступ ће бити илустрован у два примера. У првом случају, уколико је апликација за репродукцију веб претраживач, рекламни пројекат је главна *HTML* страница, док пратеће мултимедијалне садржаје чине слике, видео записи и уопште све датотеке које се учитавају код прилаза странице. У другом случају, уколико би апликација за репродукцију била приказивач видео садржаја, рекламни пројекат би могао бити видео садржај, док титлови представљају пратећи мултимедијални садржај.



Слика 40 - релациони модел дистрибуираних садржаја

Слика 40 приказује два рекламна пројекта са везаним пратећим мултимедијалних садржајима (дакле два рекламна пакета) и илуструје концепт дељења пратећих мултимедијалних садржаја. Наиме, исти пратећи мултимедијални садржај може бити у релацији са више рекламних пројеката – може се делити између рекламних пакета. Предности оваквог приступа су следеће:

- Уштеда складишног простора се постиже избегавањем чувања копија истог мултимедијалног садржаја који се користи у више рекламних пакета.
- Постиже се уштеда пропусног опсега код дистрибуције рекламних пакета са дељеним садржајима

- Измена дељеног мултимедијалног садржаја на једном месту се рефлектује на све рекламне пакете у којима се он користи

Рекламни пројекат и његови пратећи мултимедијални садржаји (рекламни пакет) чине недељиву (атомску) целину у следећим случајевима:

- Код планирања временског распореда репродукције
- Код репродукције

У случају дистрибуције рекламних пакета путем клијентских контролних апликација, дистрибуирају се само датотеке из рекламног пакета које нису присутне на клијентској платформи, или се њихов садржај разликује од садржаја датотеке на клијентској платформи. Поређење садржаја датотека своди се на поређење њихових израчунатих *MD5* сума (енгл. *MD5 checksum*) које се чувају у базама података дистрибутивно-контролне и клијентске контролне апликације.

5.3.4.2 Временски распоред репродукције рекламних пакета

Модел временског распореда репродукције рекламних пакета развијен је сходно захтевима постављеним од стране корисника система. Основни захтеви су:

- У једном тренутку могућа је репродукција само једног рекламног пакета
- Појединачни рекламни пакети се могу репродуковати периодично, са периодом од једног дана или недељу дана
- Потребно је омогућити бесконачно дуго приказивање појединачних рекламних пакета
- Осим репродукције рекламних пакета, потребно је да постоји могућност временског планирања тзв. ***контролних*** **пакета** (нпр. укључивања и искључивања рачунара за репродукцију)

Осим ограничења везаних за начин планирања репродукције, пред систем се постављају и ограничења везана за начин примене плана репродукције у систему:

- План репродукције се везује за групу, која репрезентује скуп уређаја за репродукцију
- Једна група може имати само један план репродукције

На основу претходно дефинисаних захтева и ограничења а по узору на модел који дефинише [Harrison], креиран је модел података који описују табела 7 и табела 8. Принципи коришћења дефинисаног модела биће дати у два примера.

Поље	**Опис**
ИД	Примарни кључ табеле
ИДПројекта	Јединствени идентификатор рекламног пројекта који треба репродуковати, чиме се јединствено идентификује и рекламни пакет. Негативне вредности се користе за контролне пакете.
тренутакАктивације	Тренутак почетка репродукције рекламног пакета(релативно у односу на почетни тренутак плана репродукције)
трајање	Време репродукције. Ако се постави на -1, траје до почетка следећег рекламног пакета
периодПонављања	Уколико се периодично понавља, период између две активације
ИДПлана	Јединствени идентификатор плана репродукције коме овај план репродукције рекламног пакета припада.

Табела 7 - модел плана репродукције појединачног рекламног пакета (табела пр_пакета)

Поље	Опис
ИД	Примарни кључ табеле. На ово поље се референцира ИДПакета из табеле 7
име	Име под којим се овај пакет приказује у листама.
почетакВажења	Тренутак у коме се активира план репродукције – датум и време. Од овог тренутка се релативно представљају сви припадајући планови репродукције рекламних пакета
крајВажења	Тренутак када план репродукције престаје да важи. Уколико је празан, план репродукције се одвија без временских ограничења

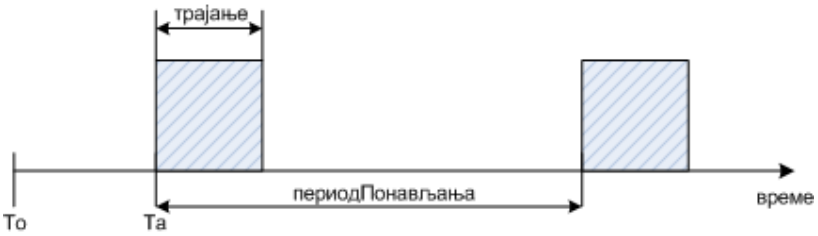
Табела 8 - модел плана репродукције (табела план_репродукције)

Први пример демонстрира план репродукције који се састоји од само једног рекламног пакета, који се репродукује сваког дана од 10 до 15 часова, почевши од 01.01.2012. године. Претходно описане табеле модела у том случају имају вредности поља дате у табели 9.

Подаци у табели **план_репродукције**	
ИД	1
име	Пример дневно периодичног понављања
почетакВажења	01.01.2012 00:00:00
крајВажења	нема
Подаци у табели **пр_пакета**	
ИД	1
ИДПројекта	1
тренутакАктивације	10*60*60
трајање	5*60*60
периодПонављања	24*60*60
ИДПлана	1

Табела 9 - пример периодичног понављања рекламног пакета

Слика 41 графички илуструје први пример, при чему је ***To*** време дефинисано у пољу ***почетакВажења***, ***Ta*** је време дефинисано као ***тренутакАктивације***. Сва релативна времена су у табели 9 дата у секундама. Обзиром да крај важења временског распореда није дефинисан, периодична репродукција рекламног пакета ће се понављати бесконачно.



Слика 41 - илустрација начина функционисања периодичног понављања

Други пример демонстрира комплекснији план репродукције, који садржи три рекламна пакета:

- први се понавља сваког дана између 10 и 13 часова
- други се приказује само 01.01.2012. између 8 и 9 часова
- трећи почиње од 14 часова и траје до почетка репродукције следећег рекламног пакета

Подаци у табелама модела којима се постиже претходно описани план репродукције дати су у табели 10.

Подаци у табели **план_репродукције**			
ИД	2		
име	Пример комплексног распореда		
почетакВажења	01.01.2012 00:00:00		
крајВажења	нема		
Подаци у табели **пр_пакета (3 врсте)**			
	Врста1	Врста 2	Врста 3
ИД	1	2	3
ИДПројекта	1	2	3
тренутакАктивације	10*60*60	8*60*60	14*60*60
трајање	3*60*60	1*60*60	-
периодПонављања	24*60*60	-	-
ИДПлана	2	2	2

Табела 10 - пример комплексног плана репродукције

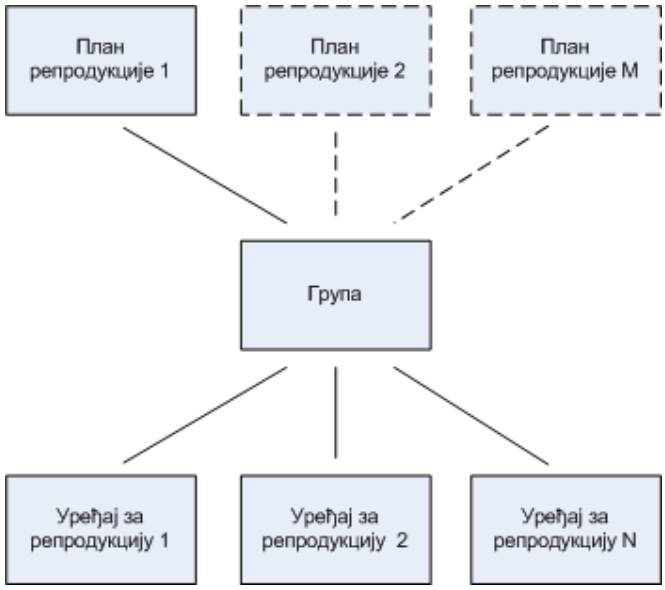
Илустрација другог примера дата је на слици 42. Осим рекламних пакета 1 и 2, код којих су дефинисани тренутак активације и трајање репродукције, слика показује и рекламни пакет 3 коме није дефинисано трајање репродукције. У случају таквог пакета, прекид репродукције се врши у тренутку почетка репродукције неког другог пакета – у примеру то је тренутак Tx, у коме почиње други период репродукције периодичног рекламног пакета 1.

Слика 42 - илустрација комплексног плана репродукције

5.3.4.3 Појам групе

Већи број уређаја за репродукцију сродних по одређеном критеријуму (географска локација, на пример) се сврстава у ***групе***. У контексту референтног система описаног у секцији 3.3.1 појам групе одговара појму канала. Слика 43 приказује релацију 1:H у којој се налазе група и њој придружени уређаји за репродукцију. Тренутна имплементација система подржава придруживање

једног плана репродукције једној групи, али се захваљујући чињеници да сваки план репродукције може имати временски оквир важења систем може лако проширити тако да више планова репродукције буде асоцирано једној групи.

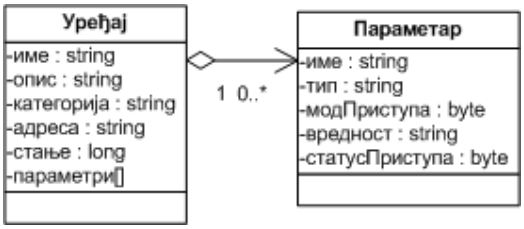


Слика 43 - релација између групе и уређаја за репродукцију

5.3.4.4 Модел података за дијагностику система

Савремени системи оглашавања на јавним местима постају све сложенији, при чему су уређај за репродукцију и дигитални екран само делови система који може укључивати камере ([PatPinpoint][PatSonyUK]), сензоре, светла и друге уређаје. Стога је дијагностика рада овако сложеног система веома битна како би се на време уочиле и кориговале неисправности у раду било ког дела система.

Један од битних захтева које модел класа којима се репрезентују уређаји у рекламном систему мора да испуни је да буде довољно генералан да се њиме може репрезентовати било који тип уређаја који се може појавити у рекламном систему. Модел коришћен у овом раду је дат на слици 44.



Слика 44 - класе за моделовање генералног модела уређаја

Понуђени модел се састоји из две класе, при чему класа ***Уређај*** даје податке о уређају и његовом тренутном стању (недоступан, ради исправно, ради са грешком, у квару). Ова класа у себи садржи и низ објеката класе ***Параметар***, у којима се чувају очитане вредности битних параметара датог уређаја (уколико је очитавање било могуће, што је забележено у пољу ***статусПриступа***).

ПОГЛАВЉЕ 6.

РЕЗУЛТАТИ МЕРЕЊА ПЕРФОРМАНСИ РЕШЕЊА

6.1 Измерене вредности параметара комуникације између веб апликације и серверске апликације

6.1.1 Измерене референтне вредности базних *COMET* техника

Као основа за процену ефикасности коришћених библиотека за комуникацију између веб апликације и апликативног сервера, извршен је низ мерења параметара базних *COMET* техника на начин описан у поглављу 4.1.1. Ова мерења су подељена у две групе:

1. Мерења потрошње пропусног опсега канала
2. Мерења брзине размене података.

6.1.1.1 Детаљи мерног окружења

За мерења су коришћени следећи алати (у складу са описом у поглављу 4.1.1.1):

- *JBoss 6.1.0 Final* апликативни сервер
- *Fiddler web debugging proxy 2.3.9.5*
- *Firefox 11.0* веб претраживач
- *Google Chrome 18.0.1025.162* веб претраживач

- *Internet Explorer 8.0.6001.18702* веб претраживач

Мерења су обављена на рачунару *Dell Latitude E6400* базираном на двојезгарном процесору *P8700* на радној фреквенцији од 2.53GHz и опремљеном са 4GB меморије. Оперативни систем рачунара је *Microsoft Windows XP Professional, Service pack 3*.

6.1.1.2 Измерени саобраћај без компресије тела одговора

Код ове групе мерења, *HTTP* конектор *JBoss* веб сервера је био подешен тако да је компресија података послатих од стране сервера ка веб апликацији увек искључена. Интервал дугог повлачења код мерења је износио 45 секунди, а поред вредности измерених по једном повлачењу, у резултатима су дате и израчунате вредности саобраћаја у случају када би систем радио један сат без преноса корисних података (на основу задатог интервала повлачења од 45 секунди, у току сат времена се обави 80 повлачења). Сва мерења су обављена коришћењем сва три наведена веб претраживача како би се одредила и зависност количине података од веб претраживача. Количине података приказане у свим табелама дате су у бајтовима.

Табела 11 показује резултате мерења код *COMET* технике која се заснива на дугом повлачењу путем *HTTPXMLRequest* асинхроних позива сервера.

Параметар/Претраживач	Firefox	Chrome	Internet Explorer
Заглавља захтева	430	498	444
Заглавља одговора	174	174	174
Тело одговора	5	5	5
Укупно	609	677	623
Процена за 1час	48720	54160	49840

Табела 11 - количина података код технике *HTTPXMLRequest* дугих повлачења

Табела 12 Приказује резултате мерења количине пренетих података код *COMET* технике која се заснива на преносу података кроз повратне функције позване из *script* тагова који се динамички додају на веб страницу.

Параметар/Претраживач	Firefox	Chrome	Internet Explorer
Заглавља захтева	338	423	412
Заглавља одговора	174	174	174
Тело одговора	25	25	25
Укупно	537	622	611
Процена за 1час	42960	49760	48880

Табела 12 - количина података код технике преноса кроз *Script* таг

Табела 13 даје резултате мерења количине пренетих података *COMET* технике код које се подаци преносе кроз скривени *iFrame*. За разлику од претходне две технике, где је количина пренетих података за сат времена била прорачуната на основу података добијених за појединачно повлачење, код ове технике је симулирано повлачење у току једног сата (80 повлачења са интервалом од 1с). Овакав приступ је примењен због специфичности технике, која свих 80 повлачења обавља у једном дугом упиту, при чему у овом случају под повлачењем подразумевамо позив повратне функције кроз све време отворени комуникациони канал, без упита за свако повлачење. Укупна количина података пренета кроз *iFrame* за један сат је 4885 бајтова, што у збиру са заглављима захтева и одговора даје измерене вредности у табели.

Параметар/Претраживач	Firefox	Chrome	Internet Explorer
Заглавља захтева	381	466	659
Заглавља одговора	174	174	174
Тело одговора	66	66	66
Укупно	621	706	899
Измерено за 1час	5440	5525	5718

Табела 13 - количина података код технике преноса кроз скривени *iFrame* елемент

6.1.1.3 Измерени саобраћај са компресијом тела одговора

Код ове групе мерења, *HTTP* конектор *JBoss* веб сервера је био подешен тако да је компресија података послатих од стране сервера ка веб апликацији увек укључена, при чему је коришћена *GZIP* техника компресије података. Сви остали параметри мерења су идентични параметрима мерења описаних у секцији 6.1.1.2.

Табела 14 приказује резултате еквивалентне онима које приказује табела 11, при чему се поређењем двеју табела може уочити да коришћење компресије у овом случају увећава количину пренетих података.

Параметар/Претраживач	Firefox	Chrome	Internet Explorer
Заглавља захтева	430	498	444
Заглавља одговора	221	221	221
Тело одговора	35	35	35
Укупно	686	754	700
Процена за 1час	54880	60320	56000

Табела 14 - количина података код технике *HTTPXMLRequest* дугих повлачења са компресијом

Табела 15 даје резултате еквивалентне онима које приказује табела 12, при чему је и у овом случају евидентно да је количина пренетих података повећана у случају када се користи компресија података у телу одговора.

Параметар/Претраживач	Firefox	Chrome	Internet Explorer
Заглавља захтева	338	423	412
Заглавља одговора	221	221	221
Тело одговора	51	51	51
Укупно	610	695	684
Процена за 1час	48800	55600	54720

Табела 15 - количина података код технике преноса кроз *Script* таг са компресијом

Табела 16 приказује резултате еквивалентне резултатима које даје табела 13, при чему се може запазити да је количина пренетих података драстично смањена у случају када се тело одговора компресује коришћењем *GZIP* алгоритма на страни веб сервера. Тело поруке, које је у првом мерењу без компресије заузимало 4885 бајтова, коришћењем компресије сведено је на 118 бајтова.

Параметар/Претраживач	Firefox	Chrome	Internet Explorer
Заглавља захтева	381	466	659
Заглавља одговора	221	221	221
Тело одговора	82	82	82
Укупно	684	769	962
Измерено за 1час	720	805	998

Табела 16 - количина података код технике преноса кроз скривени *iFrame* елемент са компресијом

6.1.1.4 Брзина преноса података без компресије тела одговора

Код ове групе мерења, *HTTP* конектор *JBoss* апликативног сервера је конфигурисан да не користи компресију код слања података. Како је објашњено у секцији 4.1.1.2, мерење времена преноса података за сваки протокол и сваки веб претраживач вршено је по 100 пута, при чему су као резултат мерења узимане минималне, средње и максималне вредности времена испоруке. Табела 17 приказује измерене вредности времена испоруке у милисекундама.

Претраживач	*Firefox*			*Chrome*			*Internet Explorer*		
Метод/Време	мин	сре	макс	мин	сре	макс	мин	сре	макс
HTTPXMLRequest	4	10.1	32	0	8.2	41	0	12.8	125
Script таг	4	12.2	43	2	8.5	39	0	15	47
Скривени *iFrame*	3	9.63	28	0	13.24	58	0	6.24	16

Табела 17 - времена испоруке без компресије података

6.1.1.5 Брзина преноса података са компресијом тела одговора

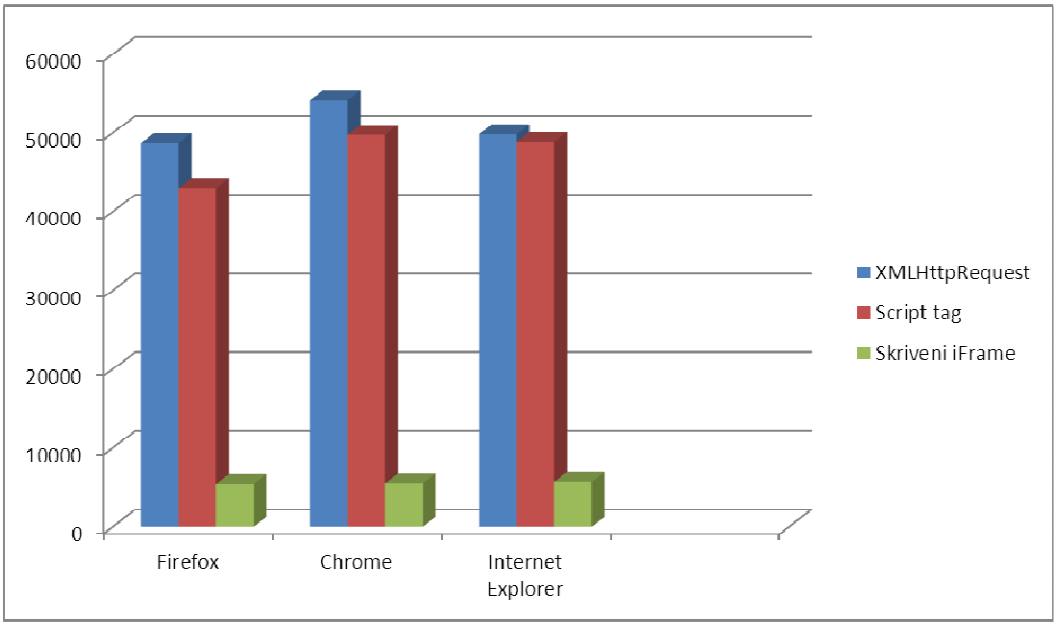
Ова група мерења обављена је на идентичан начин као и мерења описана у секцији 6.1.1.4, уз ту разлику што је *HTTP* конектор *JBoss* веб сервера био подешен да податке компримује коришћењем *GZIP* компресије. Табела 18 приказује измерене вредности времена испоруке представљена у милисекундама, при чему се може приметити да су све измерене вредности значајно веће од еквивалентних вредности у случају када компресија података није коришћена.

Претраживач	*Firefox*			*Chrome*			*Internet Explorer*		
Метод/Време	мин	сре	макс	мин	сре	макс	мин	сре	макс
HTTPXMLRequest	6	15	63	4	14.6	74	0	18.9	125
Script таг	6	16.9	111	2	13	49	0	16.7	62
Скривени *iFrame*	4	14.2	54	2	15.1	82	0	9.5	62

Табела 18 - времена испоруке са компресијом података

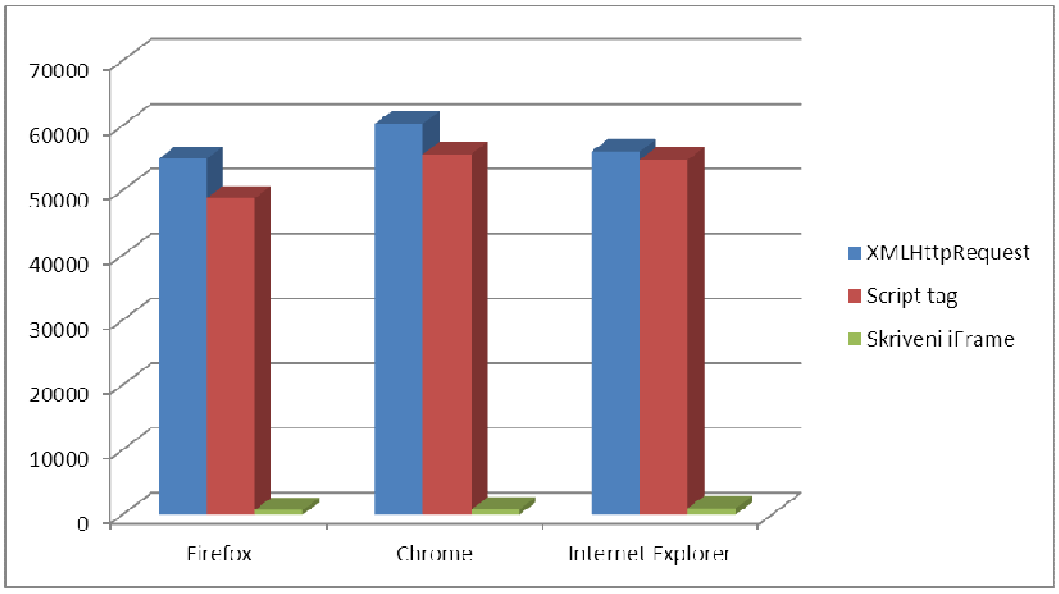
6.1.2 Анализа измерених референтних вредности *COMET* техника

Слика 45 и слика 46 графички приказују количине некорисних података пренетих у току једног сата код различитих *COMET* техника са и без коришћења компресије. Са слика се може уочити да је количина пренетих података порасла са коришћењем компресије код техника *HTTPXMLRequest* са дугим повлачењем и *script* таг са дугим повлачењем, док је код технике скривеног *iFrame*-а коришћење компресије довело до значајног смањења количине пренетих података. Са слика је такође уочљиво да техника скривеног *iFrame*-а у оба случаја, са и без компресије података, захтева преношење далеко мање количине података за одржавање везе са сервером него друге две технике.



Слика 45 - приказ количине пренетих података без компресије код референтних *COMET* техника

Поменуто повећање количине података са коришћењем компресије података код *HTTPXMLRequest* и *script* таг техника је последица чињенице да је тело одговора код ових техника у случају непостојања корисних података веома мало и самим тим није погодно за компресију. Са друге стране, код технике скривеног *iFrame*-а, веза између сервера и клијента се одржава низом идентичних порука које се од стране сервера шаљу ка веб клијенту преко једног дуготрајног одговора, што овај одговор чини идеалним за компресију.

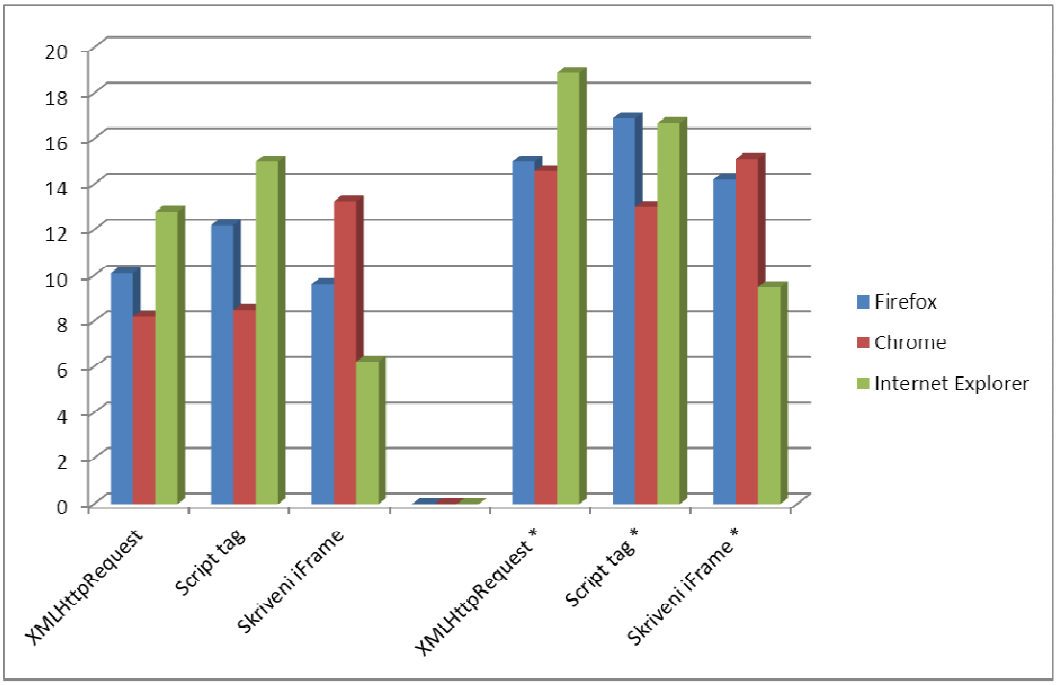


Слика 46 - приказ количине пренетих података са компресијом код референтних *COMET* техника

Анализом резултата мерења приказаних у секцијама 6.1.1.2 и 6.1.1.3 уочава се да у циклусу захтев/одговор највећа количина података припада заглављима захтева и одговора, док тело одговора представља само незнатан део размењених података. Стога је јасно да нема значајније промене количине података код техника које се заснивају на великом броју периодичних повлачења (*HTTPXMLRequest* и *script* таг) са укључењем/искључењем компресије података, посебно ако се узме у обзир да се компресија не примењује на заглавља упита и одговора. Са друге стране, код технике скривеног *iFrame*-а се веза одржава једним упитом који се не прекида дуже времена (у обављеним мерењима, дуже од сат времена), због чега ова техника има далеко мање оптерећење око преноса података заглавља и самим тим је далеко ефикаснија.

Мерења показују разлику у пренетој количини података од претраживача до претраживача, при чему *Firefox* у свим мерењима користи најмање пропусног опсега за пренос заглавља захтева, док *Chrome* користи највише пропусног опсега. Измерене вредности показују да су ова одступања ипак релативно мала.

Слика 47 пореди времена доставе поруке код различитих *COMET* техника, са и без коришћења компресије тела одговора. Мерења извршена са коришћењем компресије података означена су знаком *.



Слика 47 - Време доставе поруке са и без коришћења компресије тела одговора

Са слике се види да код свих *COMET* техника долази до значајног повећања времена преноса поруке са коришћењем компресије података. Ово повећање времена преноса је и очекивано, обзиром да компресија података на страни сервера и њихова декомпресија на страни веб претраживача захтева додатно процесирање.

Измерене вредности времена доставе порука треба користити само као оријентационе вредности, будући да су настале коришћењем *JavaScript Date* објекта, који зависно од претраживача обезбеђује ограничену прецизност мерења времена. Током мерења показало се да поједини претраживачи (нарочито *Internet Explorer*) дају резултате доста грубе грануларности, због чега нема услова за детаљније поређење брзине доставе порука код појединих претраживача.

6.1.3 Измерене карактеристике *GWT* и *Errai RPC* библиотека

Поређење ефикасности *GWT* и *Errai RPC* механизама за комуникацију између веб апликације и серверске апликације извршено је на начин дефинисан у секцији 4.1.2. Резултати приказани у овој секцији нуде поређење *GWT* и *Errai RPC* механизама по два основа:

- Ефикасности коришћења пропусног опсега за пренос података
- Брзине извршења *RPC* позива

6.1.3.1 Детаљи мерног окружења

Елементи мерног окружења, како рачунарске компоненте тако и програмске компоненте, идентични су елементима набројаним у секцији 6.1.1.1, уз ту разлику што су сва мерења у овом случају обављена коришћењем само једног веб претраживача, *Firefox 11*. Обзиром да резултати описани у секцији 6.1.1 дају довољно детаља о разликама у перформансама три најкоришћенија веб претраживача које би се у сличном односу пресликале на мерења обаљена у овој групи, мерења су обављена коришћењем само једног веб претраживача.

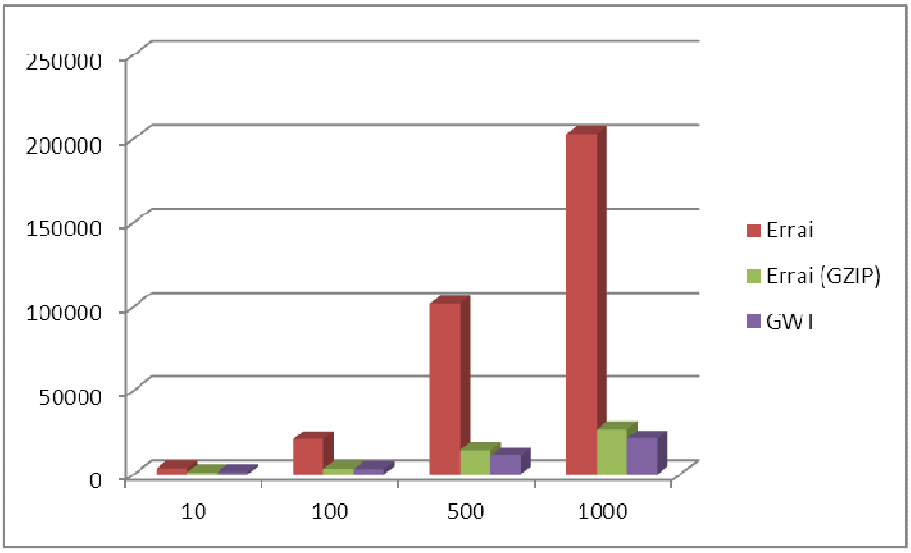
6.1.3.2 Измерени саобраћај *GWT* и *Errai RPC* позива

Резултати приказани у овој секцији пореде количину података измерену код преноса 10, 100, 500 и 1000 ВрстаОПП објеката (секција 4.1.2.1) у случају коришћења *GWT и Errai RPC* комуникационог протокола, при чему су за *Errai RPC* протокол мерења извршена са и без коришћења *GZIP* компресије тела *HTTP* одговора. Табела 19 приказује резултате свих мерења, при чему је количина пренетих података дата у бајтовима.

Biblioteka/Broj objekata	10	100	500	1000
Errai	3538	20972	101740	202954
Errai (GZIP)	1262	3306	14428	26777
GWT	1329	3096	11186	21350

Табела 19 - количина пренетих података према броју објеката

Слика 48 даје графички преглед података из табеле 19, при чему вертикална оса представља количину пренетих података у бајтовима, док хоризонтална оса групише резултате мерења оба протокола (уз варирање компресије код *Errai* протокола) према броју пренетих ВрстаОПП објеката.



Слика 48 - поређење количине пренетих података код *RPC* позива

6.1.3.3 Измерена времена извршења *GWT* и *Errai RPC* позива

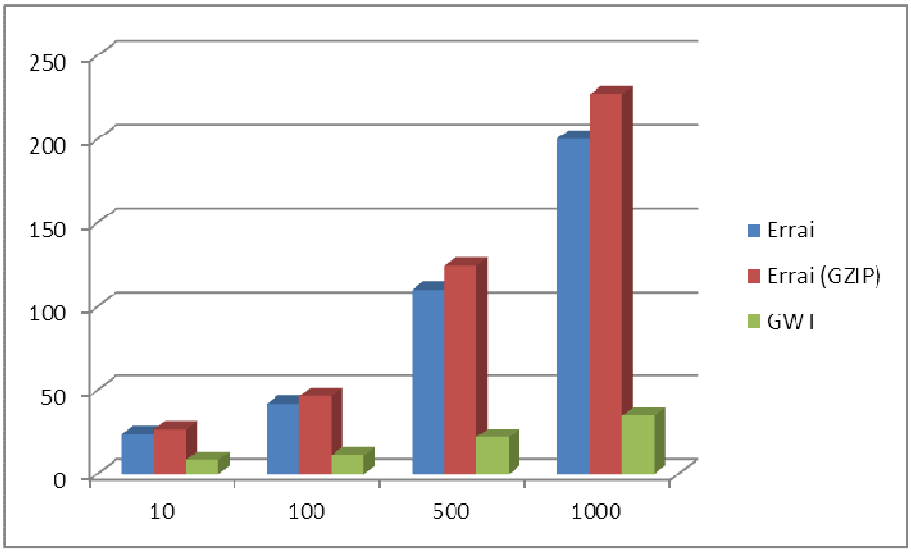
Резултати приказани у овој секцији извршени су у 12 група мерења, при чему је за сваку групу извршено по 100 понављања. Наведених 12 група настало је варирањем броја пренетих ВрстаОПП објеката између 10, 100, 500 и 1000 за *Errai*, *Errai* са *GZIP* компресијом и *GWT RPC*. Свака од 12 група мерења дала је две врсте резултата – средњу вредност времена извршења *RPC* позива, као и хистограм који показује учестаност понављања појединих резултата.

Табела 20 приказује измерене средње вредности времена *RPC* позива (вредности су у милисекундама).

Biblioteka/Broj objekata	10	100	500	1000
Errai	23.93	41.86	110.13	200.44
Errai (GZIP)	26.61	46.52	124.42	226.93
GWT	8.39	11.29	21.92	34.82

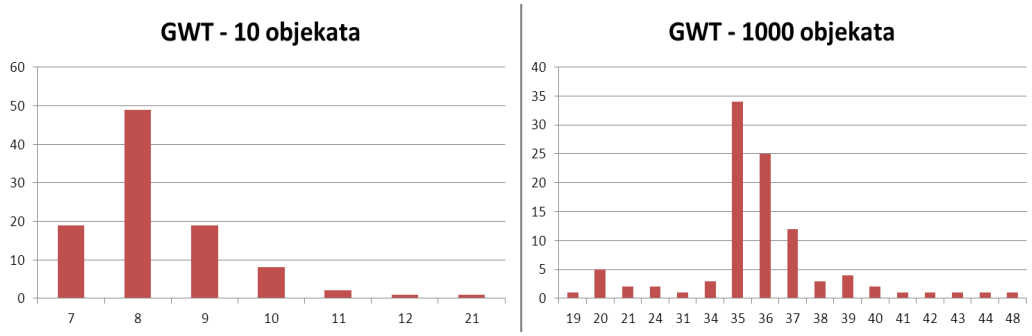
Табела 20 - времена *RPC* позива према броју објеката

Слика 49 представља илустрацију података датих у Табела 20.



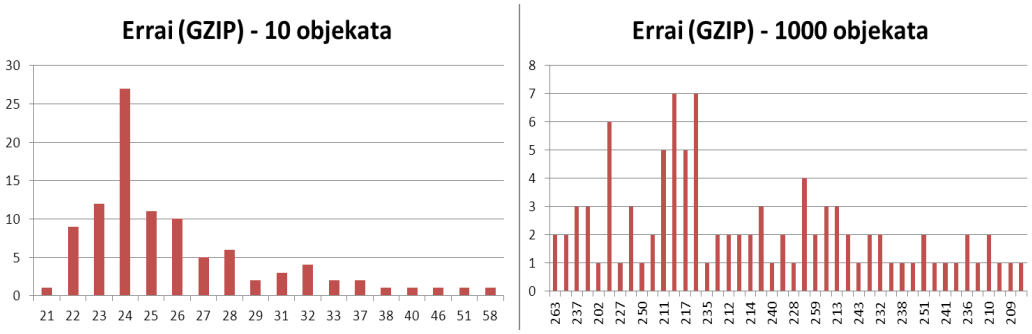
Слика 49 - поређење средњих времена *RPC* позива

Слика 50 пореди хистограме времена извршења *GWT RPC* позива у случајевима када се преноси 10 и 1000 ВрстаОПП објеката.



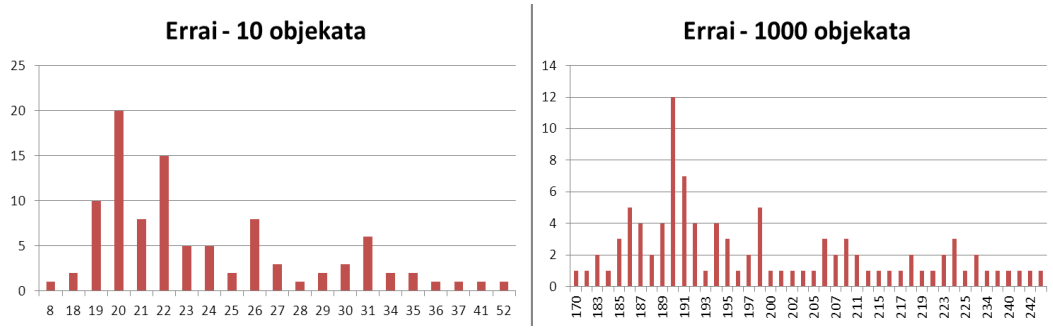
Слика 50 - хистограми времена извршења код *GWT RPC* механизма

Слика 51 даје поређење хистограма времена извршења *RPC* позива код *Errai* библиотеке за 10 и 1000 пренетих ВрстаОПП објеката, при чему се на страни апликативног сервера врши компресија тела *HTTP* одговора којим се *RPC* позив преноси.



Слика 51 - хистограми времена извршења код *Errai RPC* механизма са *GZIP* компресијом

Слика 52 пореди хистограме времена извршења *Errai RPC* позива за 10 и 1000 пренетих ВрстаОПП објеката, при чему је компресија података тела *HTTP* одговора на страни апликативног сервера искључена.



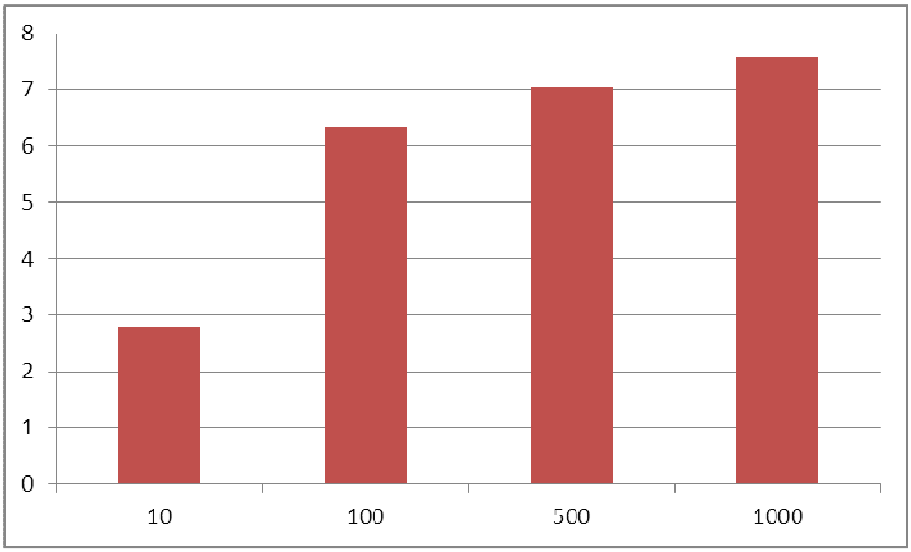
Слика 52 - хистограми времена извршења код *Errai RPC* механизма без компресије

6.1.4 Анализа резултата мерења

Анализа резултата мерења приказаних у секцији 6.1.3.2 даје две врсте информација:

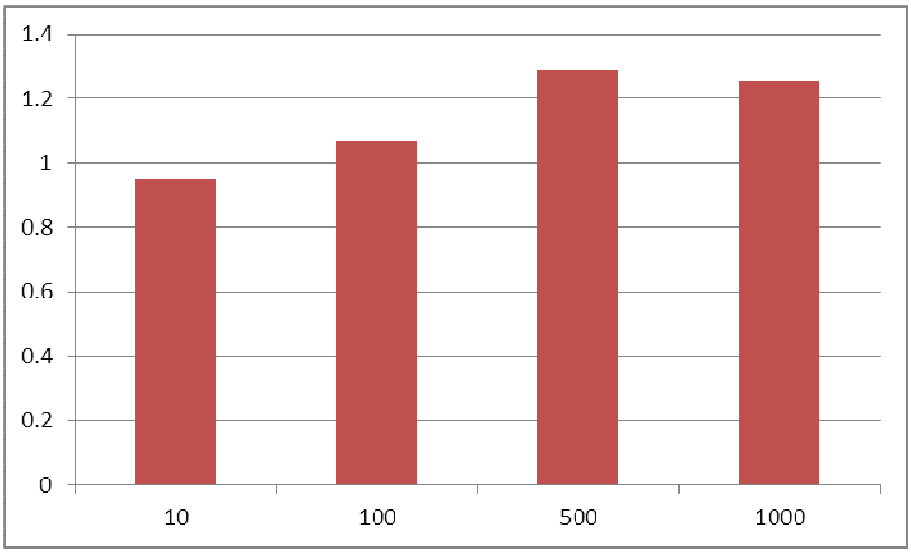
- Утицај компресије на количину пренетих података код *Errai RPC* позива
- Поређење ефикасности *GWT* и *Errai RPC* механизама у погледу коришћења пропусног опсега

Слика 53 показује однос количине података без и са коришћења *GZIP* компресија тела *HTTP* одговора код *Errai RPC* позива. Као што је и очекивано, *GZIP* компресија доноси значајно смањење количине пренетих података, при чему се уштеда пропусног опсега повећава са бројем пренетих објеката, тј. са повећањем количине података која се *RPC* позивом шаље са стране сервера. На основу овога може се закључити да би коришћење *Errai RPC* механизма без компресије било изузетно неефикасно по питању коришћења пропусног опсега, због чега карактеристике *Errai RPC* позива без компресије података неће бити поређене са карактеристикама *GWT RPC* позива.



Слика 53 - однос количина пренетих података са и без *GZIP* компресије код *Errai RPC* позива

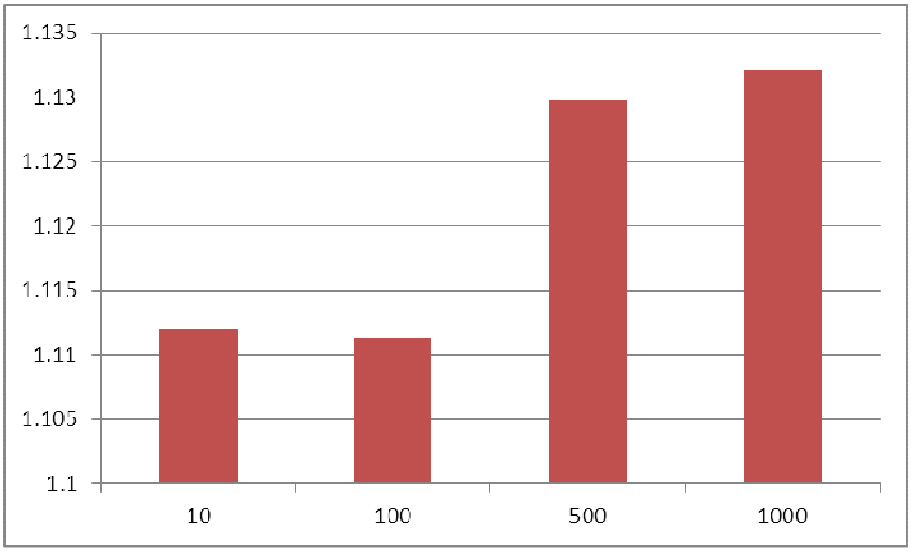
Слика 54 приказује промену односа количина пренетих података код *Errai* и *GWT RPC* позива са бројем пренетих ВрстаОПП објеката. Са слике се види да једино у случају малог броја објеката (10 у конкретном случају) *Errai RPC* механизам ефикасније преноси податке, док код већег броја пренетих објеката, *GWT RPC* механизам показује и до 25% већу ефикасност у преносу података.



Слика 54 - однос количина пренетих података *Errai* и *GWT RPC* позива

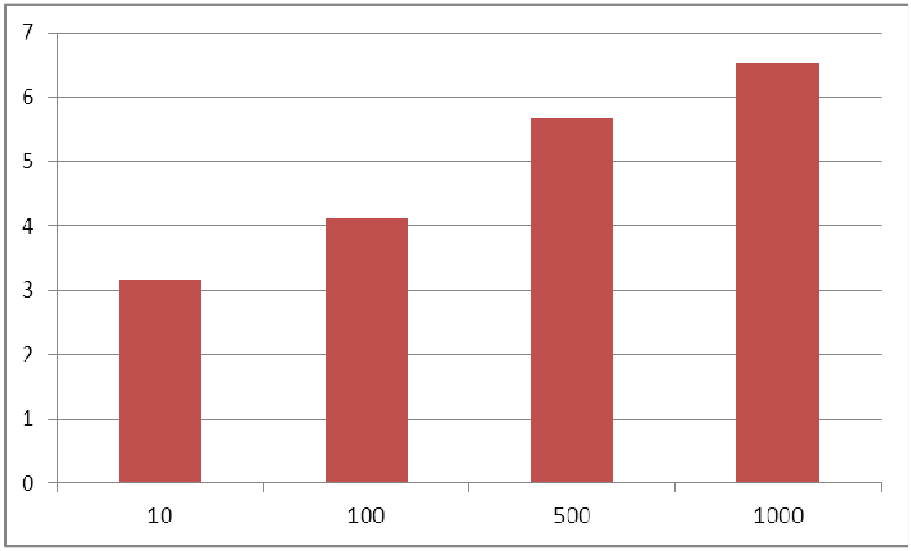
Утицај коришћења *GZIP* компресије код *Errai RPC* механизма на брзину извршења *RPC* позива приказан је на слици 55. Са слике се види да коришћење

компресије повећава време извршења *RPC* позива између 11% и 13.5%, уз тенденцију раста са повећањем количине података пренетих са серверске стране. Обзиром на постигнуту уштеду пропусног опсега, повећање времена извршења *RPC* позива има прихватљиву вредност.



Слика 55 - однос времена извршења *Errai RPC* позива са и без *GZIP* компресије

Разлика у временима извршења *RPC* позива много је више изражена ако се упореде резултати *GWT* и *Errai* механизама, при чему се са слике 56 може уочити да се *GWT RPC* позиви извршавају између 3 и 7 пута брже него еквивалентни *Errai RPC* позиви.



Слика 56 - однос времена извршења *Errai RPC* позива и *GWT RPC* позива

Претходна анализа показује да је *GWT RPC* механизам ефикаснији од *Errai RPC* механизма и по питању ефикасности коришћења пропусног опсега, као и по питању брзине извршења *RPC* позива, где је разлика у ефикасности далеко израженија.

6.2 Измерене вредности параметара комуникације серверске и клијентске контролне апликације

6.2.1 Паралелизација преноса датотека у условима отежане комуникације

Мерно окружење којим се мери ефикасност преузимања рекламних пакета са великим бројем рекламних садржаја у зависности од броја симултаних преузимања а у условима отежане комуникације описано је у секцији 4.2.1.

6.2.1.1 Детаљи мерног окружења

За мерење су коришћена три рачунара наведених карактеристика:

- Сервер
  - Персонални рачунар са процесором *Intel Pentium 4* на 3GHz са 2GB меморије
  - Оперативни систем *Ubuntu Linux 10.04.1*, *kernel 2.6.32-24-generic*
  - *JBoss 6.1.0 Final* апликативни сервер
- Мрежни мост:
  - Персонални рачунар са процесором *Intel ATOM N270* на 1.66GHz са 2GB меморије
  - Оперативни систем *Ubuntu Linux 10.04.4*, *kernel 2.6.32-38-generic*
  - *Dummynet*
- Клијентски рачунар
  - Персонални рачунар са процесором *Intel ATOM N270* на 1.66GHz са 2GB меморије

- Оперативни систем *Ubuntu Linux 10.04.1 LTS*, *kernel 2.6.35-25- generic*
- Клијентска контролна апликација

6.2.1.2 Резултати мерења

Оцена ефикасности преузимања врши се на основу два мерена параметра:

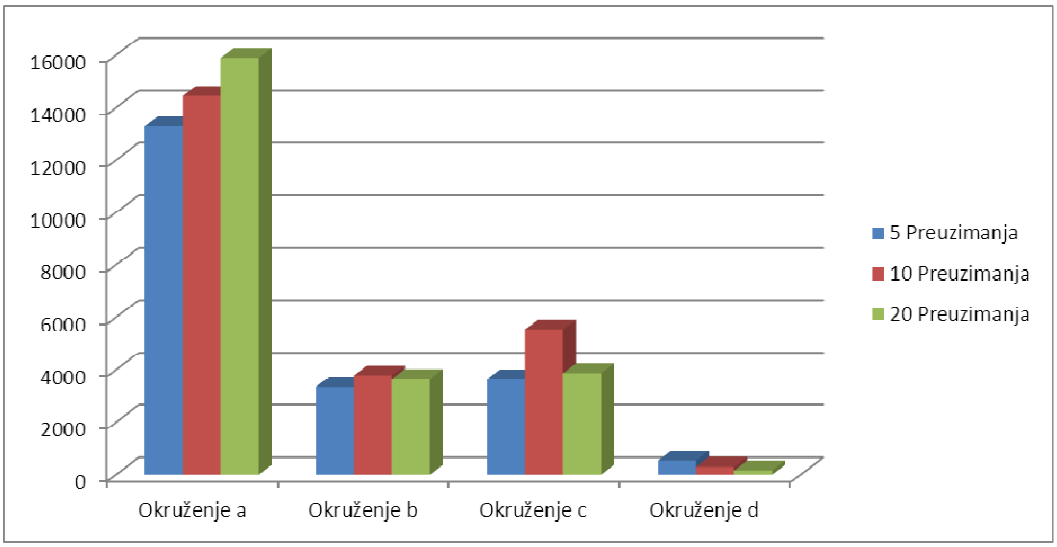
- Укупног времена потребног за преузимање комплетног рекламног пакета
- Укупног броја покушаја преузимања појединачних датотека

Табела 21 приказује измерена времена (дата у секундама) преузимања референтног рекламног пакета описаног у секцији 4.2.1.1. Детаљи мрежних окружења који су коришћени код овог мерења дати су у секцији 4.2.1.2.

Окружење/преузимања	5 Преузимања	10 Преузимања	20 Преузимања
Окружење а	13277	14446	15889
Окружење б	3345	3777	3645
Окружење ц	3635	5533	3864
Окружење д	530	297	159

Табела 21 - времена преузимања у различитим мрежним условима

Слика 57 илуструје податке из табеле 21.



Слика 57 - поређење времена преузимања рекламног пакета у разним мрежним окружењима

Током преузимања датотека са сервера у отежаним мрежним условима долазило је до прекидања преузимања на два начина:

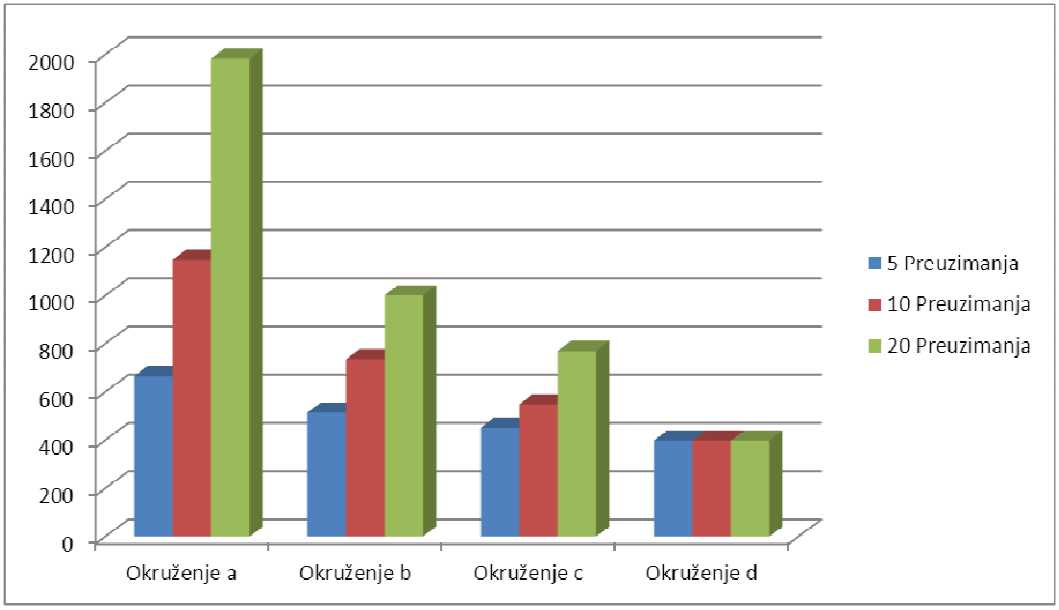
- услед истека времена чекања на успоставу везе (енгл. *connection timeout*)

- током преноса датотеке (енгл. *socket timeout*)

Табела 22 показује на који начин се мења број покушаја преузимања са променама мрежних услова и бројем истовремених преузимања.

Окружење/преузимања	5 Преузимања	10 Преузимања	20 Преузимања
Окружење а	666	1149	1983
Окружење б	515	737	1001
Окружење ц	452	549	770
Окружење д	398	398	398

Табела 22 - број покушаја преузимања у различитим мрежним условима



Слика 58 - поређење броја покушаја преузимања датотека у разним мрежним окружењима

6.2.2 Анализа резултата

Резултати мерења изложени у секцији 6.2.1.2 показују очекиван раст времена преузимања рекламног пакета као и броја покушаја преузимања са погоршањем мрежних услова. У окружењу локалне мреже без додатних сметњи (окружење д) није долазило до неуспелих преноса датотека и у том случају постоји тенденција побољшања резултата преузимања (тј. смањење времена преузимања) са повећањем броја истовремених преузимања датотека.

У осталим случајевима (окружења а, б и ц) добијају се другачији резултати, при чему се уочава јасна тенденција повећања укупног броја покушаја преузимања датотека са повећањем броја истовремених преузимања. Промена укупног времена преузимања рекламног пакета нема јединствену зависност од

броја истовремених преузимања у различитим мрежним конфигурацијама, али се из резултата може уочити да су најбољи резултати у свим случајевима постигнути у случају 5 истовремених преузимања.

Обзиром на природу пројектованог система и очекиваног рада у неповољним мрежним условима, а узевши у обзир претходне резултате, може се претпоставити да ће у већини реалних мрежних окружења клијентска контролна апликација најефикасније преузимати рекламне пакете уколико датотеке преузима у 5 нити.

6.2.3 Мерење перформанси система у условима великог оптерећења

Циљ овог мерења је био утврђивање начина на који расте оптерећење процесора дистрибутивно-контролног сервера зависно од повећања броја контролних клијентских апликација са којима сервер одржава везу методом периодичног повлачења. Додатни циљ мерења био је и приближно утврђивање максималног броја клијентских контролних апликација са којима је могуће одржавати везу.

6.2.3.1 Детаљи мерног окружења

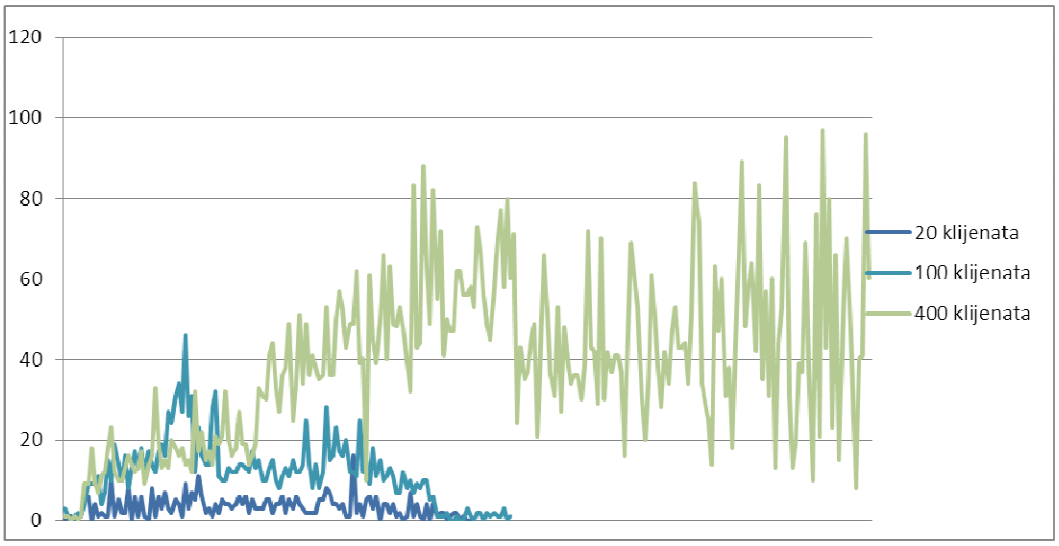
За мерење су коришћена два рачунара наведених карактеристика:

- Сервер
  - Персонални рачунар са процесором *Intel Pentium* 4 на 3GHz са 2GB меморије
  - Оперативни систем *Ubuntu Linux 10.04.1*, *kernel 2.6.32-24-generic*
  - *JBoss 6.1.0 Final* апликативни сервер
- *SoapUI* извршилац
  - Персонални рачунар *Dell Latitude* са процесором E6400 на 2.53GHz и 4GB меморије.
  - Оперативни систем *Microsoft Windows XP Professional, Service pack 3*.
  - *SoapUI 4.0.1*

Комуникација између рачунара вршена је кроз *LAN* мрежу брзине 100 MBit.

6.2.3.2 Резултати мерења

На основу логова оптерећења процесора креираних код одржавања везе са 20, 100 и 400 клијентских контролних апликација, креиран је график приказан на слици 59. График показује део оптерећења процесора које се односи на *Java* процес, који се у појединим тренуцима ближи граници од 100%. При томе треба имати у виду да је током мерења детектовано и значајно коришћење процесора од стране *Postgres SQL* процеса, што је последица интензивног приступа бази приликом читања и освежавања података о статусу везе клијентских контролних апликација.



Слика 59 - промена оптерећења *Java* процеса са бројем ККА

6.2.4 Анализа резултата

На графику приказаном на слици 59 се за случајеве од 20 и 100 клијентских контролних апликација јасно уочавају три дела:

- У првом делу, оптерећење процесора расте до максималне вредности, после чега пада. Раст оптерећења је последица постепеног додавања нових нити од стране *SoapUI* теста оптерећења, при чему свака симулирана клијентска контролна апликација врши ауторизацију која је процесорски захтевна

- У другом делу, оптерећење је релативно константно – све симулиране клијентске контролне апликације су ауторизоване и оптерећење је резултат периодичних повлачења
- У трећем делу долази до поновног скока оптерећења и потом његовог постепеног смањивања, што је последица чињенице да симулиране клијентске контролне апликације које су прве пријављене почињу да раскидају везу са серверском апликацијом, што је операција која је процесорски захтевнија од одржавања везе

Иста тенденција постоји и на графику који приказује случај 400 симулираних клијентских контролних апликација, с том разликом што је време креирање симулираних клијентских контролних апликација доста веће (200 секунди у односу на 50 секунди код прва два).

На основу приказаних резултата може се закључити да серверска платформа са једним процесором коришћена за ово мерење може истовремено одржавати везу са око 400 клијентских контролних апликација.

ПОГЛАВЉЕ 7.

ЗАКЉУЧАК

Дигитални огласни панели незаобилазан су део оглашавања на јавним местима, тако да је њихова контрола и достављање рекламних садржаја изузетно атрактивна област за нова програмска решења. У овој тези у фокусу истраживања је платформа за ефикасну и поуздану доставу рекламних садржаја до рекламних панела као и дијагностику њиховог рада, уз тежњу за постизањем што веће платформске независности, како од рачунарске платформе, тако и од типа рекламних садржаја и елемената система за репродукцију рекламних садржаја.

Део истраживања састојао се у проучавању постојећих програмских решења, како комерцијалних, тако и бесплатних. На основу постојећих решења дошло се до скупа типичних функционалности које један систем за дигитално оглашавање нуди. Проучавањем постојећих решења уочен је и низ недостатака које овај рад покушава да елиминише. Међу уочене недостатке постојећих решења спадају пре свега ригидна веза система за дистрибуцију и система за репродукцију. Ова веза успоставља се преко строго дефинисаног скупа и структуре дистрибуираних садржаја. Поред тога, типично за постојећа решења је и третирање уређаја за репродукцију као једноставног система за репродуковање аудио-видео садржаја.

Преглед базе патената је, за разлику од постојећих програмских решења, донекле показао и тенденције ка третирању уређаја за репродукцију као комплексних мултисензорских система, који доносе и одређени степен интеракције са публиком и самим тим иду корак даље од једноставних уређаја за аудио-видео репродукцију. Стога се у овом раду посебна пажња поклања имплементирању подсистема за дијагностику уређаја за репродукцију рекламних садржаја као делова мултисензорског окружења.

Претраживање научних радова резултовало је мањим бројем радова, при чему поједини радови нису уско везани за област дигиталног оглашавања већ се баве комуникационим моделима и техникама, као и начинима тестирања. Основне идеје за креирања модела плана репродукције преузете су из једног од проучених радова, док су одређене технике мерења перформанси појединих комуникационих подсистема такође инспирисане неким од радова.

Фаза припреме подразумевала је и претрагу стандарда у области дигиталног оглашавања, која је резултовала сазнањем да у овој области нема опште прихваћених стандарда. Осим *Popai* стандарда који доноси мањи скуп у великој мери недовршених докумената, нису нађени други релевантни стандарди.

Припремна фаза подразумевала је и проналажење одговарајућих технологија за имплементацију система. Резултат ове фазе је одлука да се користи *Java* платформа и низ технологија које се на њој базирају а добро су применљиве на поједине делове система.

Овај рад доноси решења више проблема, од којих су поједина концептуална и архитектурна и као таква нису погодна за мерење у класичном смислу, док су поједина решења и избори појединих технологија управо резултат конкретних мерења.

Од концептуалних решења која рад доноси, једно од битнијих је увођење клијентске контролне апликације као посредника између апликације за репродукцију и дистрибутивно контролног сервера. Минимизација зависности система од типа рекламних садржаја и од апликације (или уређаја) за репродукцију рекламних садржаја као и универзалан подсистем за дијагностику

и прикупљање статистичких података резултат су увођења и пажљивог дизајна клијентске контролне апликације.

Избор протокола за комуникацију између веб апликације и серверске апликације резултат је низа мерења брзине и ефикасности појединих библиотека. Хибридно решење које комбинује класичне *RPC* позиве *GWT* библиотеке и комуникацију засновану на принципу издавач-претплатник *Errai* библиотеке представљено у раду показује најбоље перформансе по питању оба мерена параметра. Начин паралелизације преноса датотека са дистрибутивно-контролног сервера на клијентску контролну апликацију такође је резултат мерења извршених у условима мрежних сметњи типичних за реалну примену система. Оцена понашања система са повећањем броја клијентских контролних апликација у случају коришћења технике периодичног повлачења може се донети на основу мерења перформанси система у условима великог оптерећења датим у овом раду.

Као доказ тезе реализован је систем у складу са описаном архитектуром, при чему је дистрибутивно-контролни сервер реализован као *Java EE* апликација која се ослања на *JBoss* апликативни сервер, клијентска контролна апликација је развијена као *Java SE* апликација, док је управљачка веб апликација развијена коришћењем *GWT* технологије. Развијени систем се успешно користи за контролу већег броја рекламних платформи на различитим географским локацијама. Контролисане рекламне платформе су веома различите по својој природи, од најједноставнијих, које врше једноставну видео репродукцију, до изузетно сложених, које укључују сложену интеракцију са публиком путем већег броја различитих сензора на основу које се динамички креира рекламно окружење са мноштвом аудио-визуелних стимуланса. Везе са рекламним платформама су успостављане различитим каналима, од *ADSL* и кабловских веза са Интернетом до непоузданих и спорих *3G* веза, при чему систем показује велику поузданост испоруке рекламних садржаја. Контролно-дистрибутивни сервер прикупља велике количине статистичких података са рекламних платформи, на основу којих систем генерише неколико врста различитих извештаја.

На основу резултата овог рада, могућа су даља истраживања у смеру унапређења појединих подсистема. Подсистем за извештавање, као подсистем који манипулише великим количинама података и захтева пуно процесорске снаге за своје функционисање, тренутно је у великој мери спрегнут са дистрибутивно-контролним сервером и оптерећује његов рад, па би његово издвајање као посебне целине која се лако интегрише у постојећи систем било једно од могућих унапређења. Дијагностички систем би такође могао бити унапређен у погледу могућности креирања страница за брзе прегледе статуса одабраних параметара удаљених система путем шаблона који се могу мењати, уместо тренутних прегледа који нуде само две врсте прегледа – преглед стања везе или детаљан преглед свих параметара (за сваки систем посебно). Ефикасност коришћења пропусног опсега за комуникацију између веб апликације и сервера могла би даље бити унапређена коришћењем *HTML5* стандарда који по први пут доноси могућност системски подржане двосмерне комуникације сервера и веб претраживача коришћењем *WebSockets* концепта.

ЛИТЕРАТУРА

[ProdXibo]	Xibo digital signage http://xibo.org.uk
[ProdConcerto]	Concerto digital signage http:// www.concerto-signage.com
[ProdWebSignage]	Web Signage http://www.websignage.eu
[PolizziFontana]	Vittorio Polizzi, Mario Fontana, Digital Signage content broadcasting through the cloud with Windows Azure, whitepaper, 2010
[ProdScala]	Scala Connected Signage Network http://www.scala.com
[ScalaWP]	Scala, Inc., „How Scala Works“, whitepapper
[PatStrand]	Strand; Michael J.,” Locally responsive kiosk signage from on-line source”,USPTO patent 7,685,259, 2006
[PatPinpoint]	Pinpoint Incorporated, “Location enhanced information delivery system”, USPTO patent 6,571,27, 2003.
[PatSonyUK]	Sony United Kingdom Limited, “Display arrangement including face detection”, USPTO patent 7,643,658, 2010
[PatCoreptronic]	CORETRONIC CORP [TW], "Management method for remote digital signage",USPTO Patent Application 20090276491 A1, 2008
[PatFactorx]	Factor Comm X, “Dynamic digital signage, customer content control portal and management system”, WIPO WO2009073624 A2, 2009
[PatChief]	CHIEF SYSTEM TECHNOLOGY CO LTD,”Method for transmitting files based on network digital signage system”,USPTO Patent Application Publication 2008/0313342 A1
[PatNCTS]	Patrick Julien, Bryan Mongeau, Daniel Parisien, ”Network control time spans”, USPTO Patent Application Publication 2008/0095052 A1
[Schaeffler]	Jimmy Schaeffler,“ Digital Signage: Software, Networks, Advertising, and Displays: A Primer for Understanding the Business“, Focal Press, 2008
[Papp]	Ištvan Papp, „Prilog rešenju obrade govornog signala korišćenjem mikrofonskog niza“, Doktorska disertacija, 2009.
[EPO]	*European Patent Office* – EPO, www.epo.org
[GWT]	Google Web Toolkit, http://code.google.com/webtoolkit
[GWTAction]	Robert Hanson i Adam Tacy, “GWT in Action”,Manning Publications Co., 2007

[EJB3Action]	Debu Panda, Reza Rahman, Derek Lane, “EJB 3 in Action”, Manning Publications Co., 2007
[JavaEE]	Oracle, Java EE documentation, http://docs.oracle.com/javaee
[AJAX]	Jesse James Garret,„Ajax: A New Approach to Web Applications“,www.adaptivepath.com/publications/essays/archives/000385.php, 2005
[DOMScripting]	Jeremy Keith,“DOM Scripting – Web Design with JavaScript and the Document Object Model“, Apress, 2005
[SOA]	Gopalan Suresh Raj, Binod PG, Keith Babo, Rick Palkovic “Implementing Service-Oriented Architectures (SOA) with the Java EE 5 SDK”, Sun Microsystems, Inc., 2006
[Harrison]	John V Harrison, “An Emerging Marketplace for Digital Advertising Based on Amalgamated Digital Signage Networks”, Proceedings of the IEEE International Conference on E-Commerce (CEC’03), 2003
[BozdagEtAl]	Engin Bozdag, Ali Mesbah, Arie van Deursen, „A Comparison of Push and Pull Techniques for AJAX”,Technical Report, Delft University of Technology, 2007
[GibsonRossiter]	Gibson Lam, David Rossiter,”A SOAP-based Streaming Content Delivery Framework for Multimedia Web Services”, 2008 IEEE Asia-Pacific Services Computing Conference, 2008
[Marcel]	Marcel-Cătălin Roşu,”*A-SOAP*: Adaptive SOAP Message Processing and Compression”, 2007 IEEE International Conference on Web Services (ICWS 2007), 2007
[Mono]	Mono Software platform, http://www.mono-project.com
[JSON]	The application/json Media Type for JavaScript Object Notation (JSON), http://tools.ietf.org/html/rfc4627
[HTTP1.1]	Hypertext Transfer Protocol -- HTTP/1.1, http://tools.ietf.org/html/rfc2616
[COMET]	Comet: Low Latency Data for the Browser, http://infrequently.org/2006/03/comet-low-latency-data-for-the-browser/
[DOJO]	The Dojo foundation, http://dojofoundation.org
[BAYEUX]	Bayeux Protocol - Bayeux 1.0.0, http://svn.cometd.com/trunk/bayeux/bayeux.html
[POPAI]	POPAI Digital Signage Standards Committee, http://popai.com/

[TermsPOPAI] Standard terminology, POPAI Digital Signage Standards Commitee, http://popai.com/docs/DS/POPAI%20DigSignage%20Standard%20TermsRev%201_0.pdf

[Jbossas] JBoss Application Server, http://www.jboss.org/jbossas

[Jbossclust] JBoss, Inc., "The JBoss 4 Application Server Clustering Guide", 2006

[ERRAI] JBoss Errai, http://www.jboss.org/errai

[GWTRPC] GWT - Remote Procedure Calls, https://developers.google.com/web-toolkit/doc/latest/DevGuideServerCommunication

[Fowler] Martin Fowler, "Patterns of Enterprise Application Architecture", Addison-Wesley, 2002

[Dummynet] Marta Carbone, Luigi Rizzo, "Dummynet Revisited", Dipartimento di Ingegneria dell'Informazione Universita' di Pisa, 30 november 2009

[Rizzo] Luigi Rizzo, "Dummynet: a simple approach to the evaluation of network protocols", ACM Computer Communication Review, Vol.27, n.1, Jan.97

[SoapUI] SoapUI projekat, http://www.soapui.org

[XPath] XML Path Language (XPath), Version 1.0, http://www.w3.org/TR/xpath, W3C Recommendation, 16 November 1999

[Groovy] Groovy projekat, http://groovy.codehaus.org

[Fiddler] Fiddler web debugging proxy, http://www.fiddler2.com/fiddler2

[Wininet] The Windows Internet application programming interface, http://msdn.microsoft.com/en-us/library/windows/desktop/aa383630%28v=vs.85%29.aspx

[SQLite] Strana SQLite projekta, www.sqlite.org