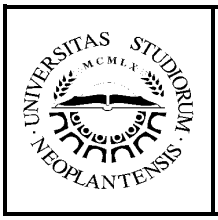


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: **Марко Савић**
Број индекса: **E12734**

Тема рада: **Реализација прототипа компоненте за управљање извршењем задатака у контексту грид инфраструктуре засноване на SIP протоколу**

Ментор рада: **доц. др Илија Башичевић**

Нови Сад, септембар, 2012. година

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**:		Завршни (Bachelor) рад
Аутор, **АУ**:		**Марко Савић**
Ментор, **МН**:		**Доц. др Илија Башичевић**
Наслов рада, **НР**:		**Реализација прототипа компоненте за управљање извршењем задатака у контексту грид инфраструктуре засноване на SIP протоколу**
Језик публикације, **ЈП**:		Српски / латиница
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)		**7/31/0/0/6/0/0**
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника
Предметна одредница/Кqучне речи, **ПО**:		**SIP протокол, реализација прототипа компоненте грид система**
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		У овом раду ће бити представљена имплементација подсистема за управљање извршењем задатака грид система. Подсистем се заснива на коришћењу SIP-а за преговарање корисника и послужиоца о потребним ресурсима за извршење задатака.
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	др Јелена Ковачевић, доцент
	Члан:	др Иштван Пап, доцент — Потпис ментора
	Члан, ментор:	др Илија Башичевић, доцент

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**:		Bachelor Thesis	
Author, **AU**:			
Mentor, **MN**:			
Title, **TI**:		**Realization of task execution manager for SIP based grid system**	
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**:			
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		**7/31/0/0/6/0/0**	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		SIP protocol, realization of task execution manager for SIP based grid system	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		A realization of task execution manager for grid system is presented. Manager relies on SIP protocol for negotiation between client and server about required resources for task execution.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	Jelena Kovacevic, PhD	
	Member:	Istvan Pap, PhD	Menthor's sign
	Member, Mentor:	Ilija Basicevic, PhD	

Захвалност

Захваљујем се асистенту Ненаду Четићу и професору Илији Башичевићу на несебичној стручној помоћи и личним саветима.

Такође се захваљујем породици, колегама и пријатељима на подршци не само током израде овог рада, већ и на подршци за време студирања.

САДРЖАЈ

Списак слика

Скраћенице

SIP – *Session Initiation Protocol*, протокол за успостављање, модификацију и раскидање мултимедијалних сесија

JSDL – *Job Submission Description Language,* мета језик за опис задатка

WWW – *World Wide Web,* светска мрежа, сервис намењен прегледу веб страница

SETI – *Search for ExtraTerrestrial Intelligence,* пројекат Универзитета у Берклију за проналазак интелигенције ван планете Земља

VoIP – *Voice over Internet Protocol,* апликациони протокол намењен преносу говора у интернет мрежи

MGCP – *Media Gateway Control Protocol,* протокол апликационог нивоа намењен управљању мултимедијалних мрежних пролаза, гејтвеја (енг. *Gateway*)

PSTN – *Public Switched Telephone Network,* мрежа фиксне телефоније

RTP – *Real-time Transport Protocol,* протокол апликационог слоја, намењен преносу видео и аудио секвенци

RTCP – *RTP Control Protocol,* протокол који се користи за надзор преноса и комуникацију изван опсега код RTP комуникације

SMTP – *Simple Mail Transfer Protocol,* протокол намењен размени поште

HTTP – *HyperText Transfer Protocol,* протокол намењен размени WWW садржаја

1. Увод

У овом задатку потребно је имплементирати подсистем за управљање извршењем задатака грид система. Подсистем се заснива на коришћењу SIP-а за преговарање корисника и послужиоца о потребним ресурсима за извршење задатака. За опис задатака, њихових компоненти, захтеваних и доступних ресурса користи се JSDL тип спецификације.

Идеја је да се направи такав подсистем који кориснику омогућава да релативно једноставно створи или измени опис задатка, пошаље га и у најкраћем року добије резултат извршавања.

Подсистем се састоји из три типа апликација:

1. Спрега са корисником, омогућава стварање, измену и покретање (енг. *Submit*) задатака.
2. Посредничка апликација (енг. *Proxy*), представља посредника између корисника и послужилаца задужених за извршавање задатака.
3. Послужилачка (енг. *Server*) апликација, преговара и извршава тражени задатак.

Ове апликације су написане у програмском језику Јава и базиране су на Еклипс платформи (енг. *Eclipse Based Applicaiton*). Као што је већ напоменуто, за преговарање се користи SIP протокол, из тог разлога све апликације користе “mjSip” библиотеку.

2. Теоријске основе

У основи ове теме, сусрећемо се терминима који заслужују посебну пажњу, тако да ће бити представљени у посебни поглављима.

2.1 Грид рачунарски систем

Тешко је дати прецизну дефиницију грид рачунарког система, али просто говорећи једна од дефиниција би била:

Грид рачунарски систем је скуп дистрибуираних рачунарских система, кластера на удаљеним локацијама који пружа скалабилан, заштићен и ефикасан механизам проналажења и дељења дистрибуираних ресурса и координисан приступ њима у оквиру динамичних, више-институционалних виртуелних организација.

Другим речима идеја грид рачунарства представља уопштење глобалне рачунарске мреже, тј веба (енг. *WWW*). За разлику од веба који првенствено омогућава презентовање информација и размену података путем Интернета, грид представља сервис за дељење процесорске снаге и меморије рачунара на мрежи. Таквим приступом локалну или глобалну мрежу претвара у један огроман рачунарски ресурс.

Термин грид рачунарства настао је раних деведесетих година прошлог века као метафора за једноставан приступ рачунарским ресурсима као што је приступ мрежи електричне енергије. Као што се види идеја грид рачунарства није нова, али се тек развојем рачунарства, а посебно мрежне инфраструктуре у читавом свету, дошло до могућности реализације ове идеје.

Типичан пример коришћења грид система је потреба за интензивним нумеричким израчунавањем или за складиштењем и обрадом огромних количина података. Такве обраде нема смисла извршавати на једној машини већ се посредством програмске подршке грид система, обављају пријаве на систем и подносе захтеви са

спецификацијом проблема. Систем ће у зависности од расположивих ресурса одредити где и како ће се проблем решавати. На тај начин, у зависности од расположиве рачунарске снаге, послови који могу трајати сатима могу постати чак и интерактивни.

Један од најпознатијих примера грид система развијен је за потребе пројеката SETI на Универзитету у Берклију. Неколико година касније та идеја уопштена је у тзв. BOINC (http://boinc.berkeley.edu) који представља оперативни систем за инернет рачунарство. Потребно је редовне програме прилогадити да се извршавају на BOINC-u и на тај начин искористити снагу великог броја рачунара. Тренутно поред SETI пројекта, BOINC извршава пројекте за истраживање климатских промена, обраду електормагнетних таласа добијених мерењем пулсара, истраживање из области биомедицине и многа друга истраживања.

2.2 SIP

Са развојем мрежа за пренос података и широком применом IP протокола скренута је пажња на пренос говора Интернетом и другим пакетским мрежама односно на VoIP технологију. Иако уведена да омогући јефтиније телефонске разговоре IP телефонија постаје све атрактивнија због могућности да пружи нове мултимедијалне сервисе. Да би се пакетска комутација могла применити у реалном времену било је потребно увести адекватне протоколе. Један од протокола који се најчешће помиње у вези са IP телефонијом је SIP који се користи за контролу позива.

2.2.1 Намена

SIP је сигнални протокол намењен за успостављање, управљање и раскидање говорних и видео сесија у пакетским мрежама. Развијен је да омогући напредне телефонске сервисе путем Интернета и других IP мрежа. Користи предности постојећих протокола за управљање појединим деловима процеса (на пример упошљава MGCP за повезивање на VoIP капије на PSTN мрежу) и сарађује са протоколима за пренос мултимедијалних информација (RTP, RTCP). Развијен је под окриљем IETF-a организације за стандардизацију Интернета на основу два Интернет протокола SMTP и HTTP. Као и ова два протокола и SIP користи симболичке адресе да репрезентује људе који желе да комуницирају. Обзиром да је то IETF стандард има отворену архитектуру која омогућује да се убрза његово прихватање.

SIP је развијен да служи као механизам за успостављање разноврсних сесија, које обухватају једног или више учесника, а може се користити "*unicast*" и "*multicast*" комуникација. Сесије укључују сваки облик интерактивне комуникације као што су: Интернет мултимедијске конференције, Интернет телефонски позиви или

мултимедијална дистрибуција. SIP позив (INVITE) који се користи за успостављање сесије носи опис сесије што даје могћност учесницима да ускладе типове медија. SIP не диктира детаље о сесији, већ усклађује интеракцију и базира је на могућностима учесника, тако да учесници исте сесије могу користити различите уређаје.

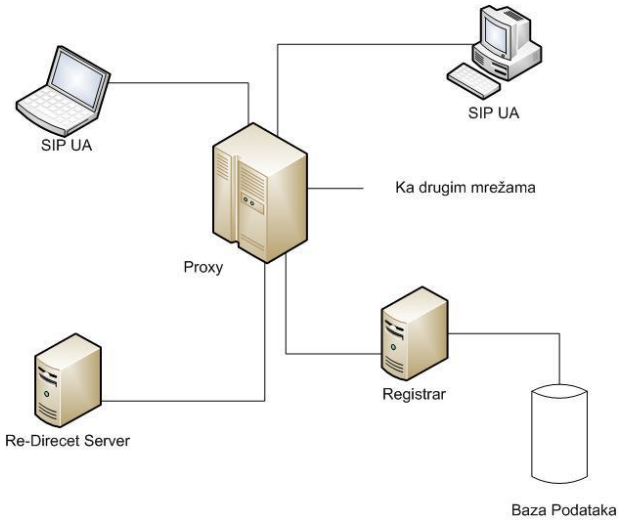
Протокол подржава функционалности персоналног рачунара, тако да корисници могу (уз одговарајуће апликације) да деле интерактивну белу таблу (енг. *White board*), тренутно размењују фајлове или шаљу URL адресу која отвара веб страницу код примаоца.

SIP није везан ни за један одређен програм за контролу конференције. То је протокол апликационог нивоа OSI модела и дизајниран је да буде независтан од коришћених протокола нижег нивоа. На нижем нивоу се може користити и UDP и TCP протокол, као и ATM.

SIP користи једноставну протокол структуру која дозвољава брже операције, флексибилност, скалабилност и мултисервисну подршку и може лако бити проширен додатним могућностима.

2.2.2 Ентитети

SIP мрежа је састављена од четири типа логичких SIP ентитета.



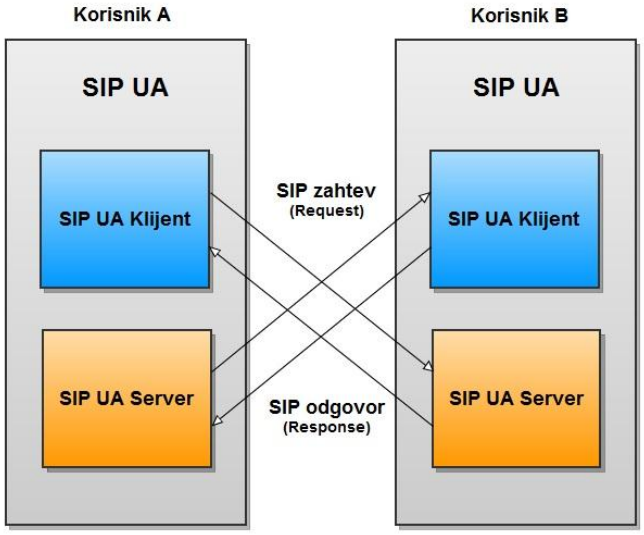
Слика 2.1 *Основни SIP мрежни елементи*

Сваки ентитет има специфичну функцију и учествује у SIP комуникацији као клијент (иницира захтев), као послижилац (одговара на захтев) или као и клијент и послужилац. Један "физички уређај" може функционисати као више логичких SIP ентитета (на пример мрежни послужилац ради у исто време и као послужилац посредник и као послужилац за преусмеравање (енг. *Redirect Server*)). Типови логичких ентитета су:

- Кориснички агент (*UA – User Agent*),

- Послужилац посредник (*Proxy Server*),
- Послужилац за преусмеравање (енг. *Redirect Server*) и
- *Registrar*.

У SIP-у кориснички агент је крајњи ентитет који успоставља и раскида сесије тако што размењује захтеве и одговоре. Функцију корисничког агента могу да имају радне станице, IP телефони, телефонски гејтвеји, аутоматизовани сервиси за одговор, UMTS мобилни телефони итд.



Слика 2.2 *Кориснички агент (клијент и послужилац)*

Кориснички агент UA је дефинисан као апликација која садржи: UA клијент (енг. *UAC – User Agent Client*) и UA послужилац (енг. *UAS – User Agent Server*). UAC клијент апликације иницира SIP захтеве, а UAS послужилац апликације контактира корисника када је примљен SIP захтев и враћа одговор у име корисника. На овај начин се комуникација две стране остварује коришћењем клијент/послужилац протокола.

Послужилац посредник је одговоран за усмеравање и испоруку поруке позваној страни. То је посреднички ентитет који зависно од сврхе захтева (који прави у име осталих клијената) делује и као послужилац и као клијент. Захтев се обрађује или интерно или се, уз могуће превођење прослеђује даље до других послужилаца. Посредник интерпретира, а када је неопходно и саставља поново захтев пре него што га проследи даље.

Аутентификација, ауторизација и контрола приступа мрежи су такође под надзором посредника, као и заштита података и захтев за ретрансмисијом. Постоје две врсте посредничког ентитета: са и без памћења стања (“*statefull”* и “*stateless”*). *Statefull”* подразумева да се позив или цела трансакција памти од почетка до краја, а у “*stateless”* моду сваки захтев и одговор се обрађују засебно.

Послужилац за преусмеравање враћа нову локацију за захтев тј. даје информације о наредном скоку SIP поруке. Он растерећује усмеравање посредника тако што прихвата SIP захтев, мапира адресу позване стране у нула (ако није позната друга адреса) или више нових адреса и враћа их клијенту. За разлику од послужиоца посредника, редирект послужилац не прослеђује захтеве другим послужиоцима.

Регистрар је послужилац који прихвата захтев за регистрацију и има приступ бази података (енг. *Location Service*) која садржи информације које повезују корисникову SIP адресу и његове контакт (физичке) адресе. Корисник може да региструје један или више уређаја на мрежи и тиме постаје доступан за контакт без обзира где се налази и то независно од појединости у вези мреже или уређаја (PC, мобилни или IP телефон, PDA) који користи. Захваљујући регистрару, позиви се упућују на име особе, уместо да се примењује компликована нумеричка шема. Он утврђује мапирање од имена до адресе.

Захваљујући редирект послужиоцу и регистрару SIP подржава мод за преусмеравање јер се позив може преусмерити у складу са тренутном локацијом корисника. На овај начин SIP подржава мобилност корисника. Ово је нови персонализовани модел комуникација – уместо да се акценат ставља на уређај у нади да ће и особа бити доступна, сада мрежа може да лоцира особу.

2.2.3 Адресирање

SIP користи Интернет URL (енг. *Uniform Resourse Locator*) за адресирање. Адреса корисника уопштено има форму "ime@domen" и личи на "*e-mai"l* адресу. Због ове сличности SIP URL се лако може придружити корисниковој "*e-mail"* адреси. SIP подржава и Интернет и PSTN адресе.

Обзиром да SIP URL описује корисников домен поруке се прво рутирају на домену, а затим их ентитети домена прослеђују до терминала, послужиоца других мрежа или апликација.

Везе између јавних SIP URL-а и URL-а одредишта корисника се налазе у базама података локација и могу се динамички мењати SIP порукама.

2.2.4 Поруке

SIP поруке су текстуалне, а по синтакси и по пољу заглавља су сличне HTTP-у. У њима се не преноси садржај сесије већ само URL-ови и други подаци које крајњи ентитети користе да прибаве садржај. Могу се поделити на два типа:

- захтев (енг. *Request*) – шаље се од клијента ка послужиоцу

- одговор (енг. *Response*) – који се шаље од послужиоца ка клијенту. Поруке за одговор садрже нумеричке кодове, делимично засноване на HTTP одговор – кодовима.

Без обзира да ли је у питању захтев или одговор порука садржи заглавље и тело. У заглављу се преносе:

- информације о SIP адреси позваног корисника,
- информације о SIP адреси позивајућег корисника,
- индентификатор сесије,
- информације о дужини тела поруке и типу информација које се преносе у телу поруке,
- информације о путањи поруке и
- редни број захтева.

Тело поруке се користи да опише сесију коју треба иницирати (у мултимедијалној сесији ово може укључити тип аудио и видео кодека или на пример учестаност) али може бити искоришћено и за обичне текстуалне или бинарне податке повезане са сесијом.

SIP ентитети примају поруке у име корисника у циљу обезбеђивања услуге, при чему SIP прави јасну разлику између сигнализационих информација, које се преносе у заглављу и информација сесије које су ван надлежности SIP-а.

Могући типови тела поруке су SDP (енг. .*Session Description Protocol*), MIME (енг. *Multipurpose Internet Mail Extensions*) или друго (дефинисано од стране IETF-а или кроз конкретну имплементацију).

2.2.5 Закључак

SIP постаје прави покретач у мрежама следеће генерације и софтверског инжењерства. Његово прихватање ипак зависи од економских фактора иако је мањи, компактнији и делотворнији, боље прилагођен Интернет технологији од старијег H.323 протокола.

Вредност SIP-а лежи у комбиновању услуга које подржава. Сличности између SIP-а и Интернет технологије помоћи ће бржем развијању нових услуга, што ће довести до развоја Интернет телефоније.

Користећи SIP, провајдери и њихови партнери могу да прилагоде и испоруче скуп услуга базираних на SIP-у који ће клијентима омогућити употребу више услуга у оквиру једне комуникационе сесије. Сервис провајдери могу да креирају једну

флексибилну сет апликацију која задовољава потребе корисника уместо инсталирања специјализованих појединачних апликација.

SIP обећава значајна побољшања у области бизнис комуникација, прво као VoIP протокол. а такође и зато што се добро интегрише са осталим пословним комуникационим апликацијама. Коришћен заједно са осталим Интернет технологијама, SIP ће донети нове комуникационе сервисе корисницима канцеларијских и мобилних рачунара и тиме повећати продуктивност запослених док у исто време смањује оперативне трошкове.

2.3 JSDL

Представља прошириву XML спецификацију која је направљена од стране "*Global Grid Foruma"* за опис једноставних задатака намењених не интерактивном, најчешће удаљеном извршном рачунару.

JSDL описује задатак са нагласком на аспекте подношења, не описује начин извршавања ни историју подношења. Предвиђа следеће описе:

- Име задатка
- Потребне ресурсе
- Ограничења приликом извршавања
- Опис улазно излазних датотека
- Команде извршавања, аргументе командне линиије итд.

2.4 Eclipse RCP апликације

Иако је Еклипс платформа направљена са циљем да служи као скуп отворених алата, тако је пројектована да се компоненте од којих је сачињена могу искористити за прављење било какве друге клијентске апликације. Минималан скуп додатака (енг. *Plugins*) који је потребан за прављење сложене клијентске апликације (енг. *Rich Client Application*) је познат као RCP (енг. *Rich Client Platform*).

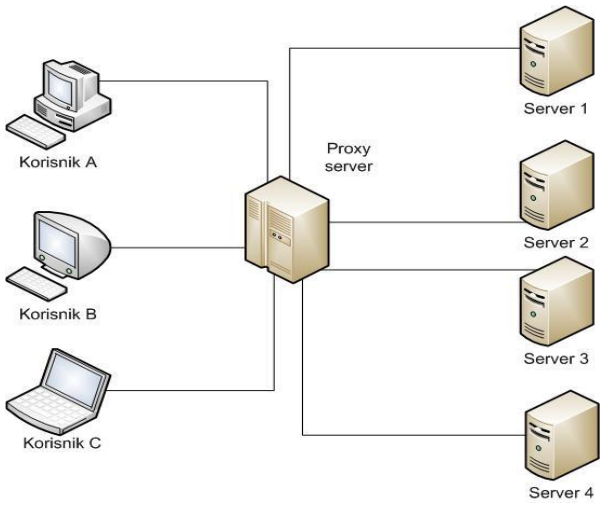
Еклипс је проширива платформа и као таква пружа основни скуп услуга који контролише додатке како би они функционисали као једна целина. Основни механизам који Еклипс пружа је могућност да сваки додатак буде проширив. На тај начин је могуће градити инетегрисано развојно окружење са деловима од неког већ постојећег развојног окружења, што знатно скраћује трајање развоја. Иако је Еклипс платформа специјализована за изградњу интегрисаних развојних окружења Еклипс платформа се може искористити за изградњу RCP-а које као своје основно језгро користе Еклипс и граде се као скуп Еклипс додатака.

Дакле, RCP апликација представља Еклипс извршно окружење (енг. *Runtime*) са једне стране и скуп само оних додатака који су потребни да би апликација функционисала.

3. Концепт решења

3.1 Типови апликација

Решење обухвата реализацију три типа апликација које међусобну комуницирају у циљу реализације грид инфрастуктуре. Као што је већ споменуто постоји: клијентска апликација, затим посредничка и послужилачка апликација, Слика 3.1 показује организацију оваквог грид решења.

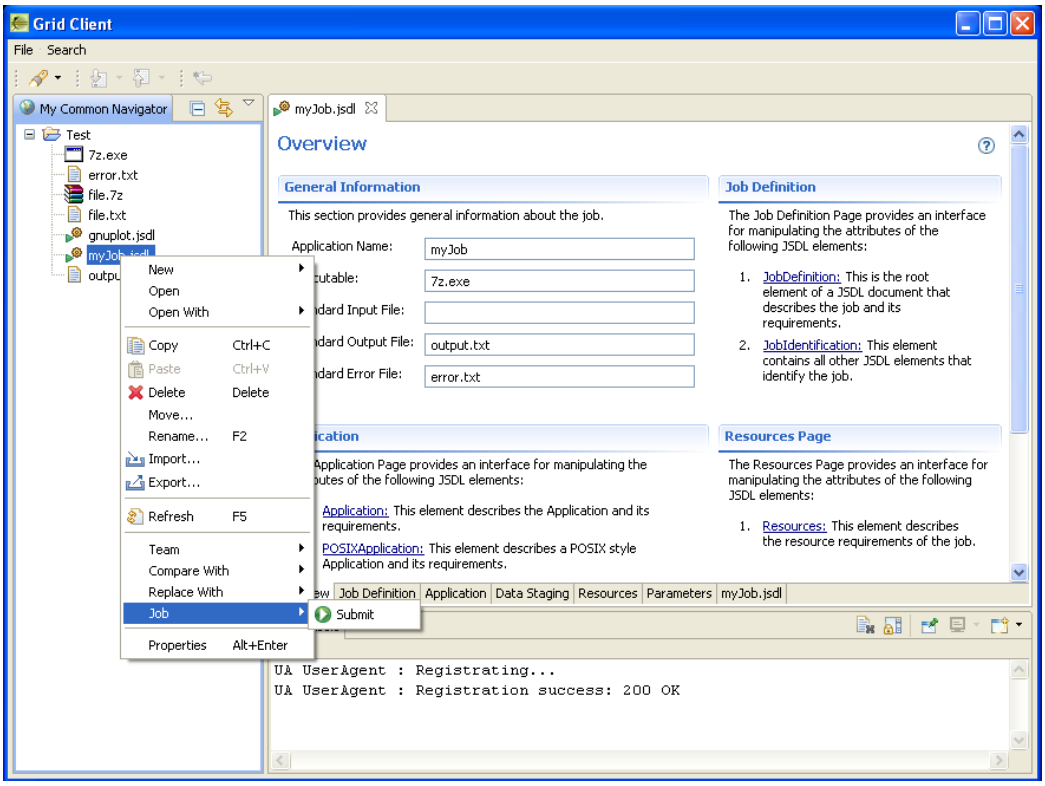


Слика 3.1 *Организација ентитета у грид систему*

Укратко клијентска апликација је задужена за стварање и подношење задатака, посредничка апликација служи као посредник порука а послужилачке апликације су задужене за извршавање задатака.

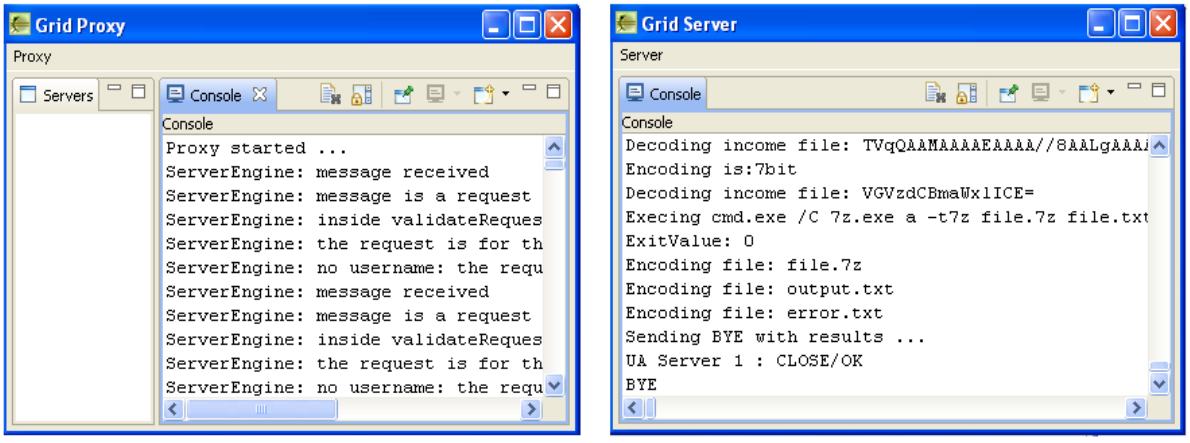
Све апликације су направљене као RCP апликације и као такве врло једноставно користе компоненте (енг. *Plugins*) које су већ направљене у оквиру неког другог пројекта. Тако на пример клијентска апликација користи едитор JSDL датотека из пројекта G-Eklipse и компоненту задужену за надгледање пројекта (енг. *Common*

Navigator) директно из Еклипс пројекта. Изглед клијентске апликације приказан је на слици 3.2.



Слика 3.2 *Графичка корисничка спрега клијентске апликације*

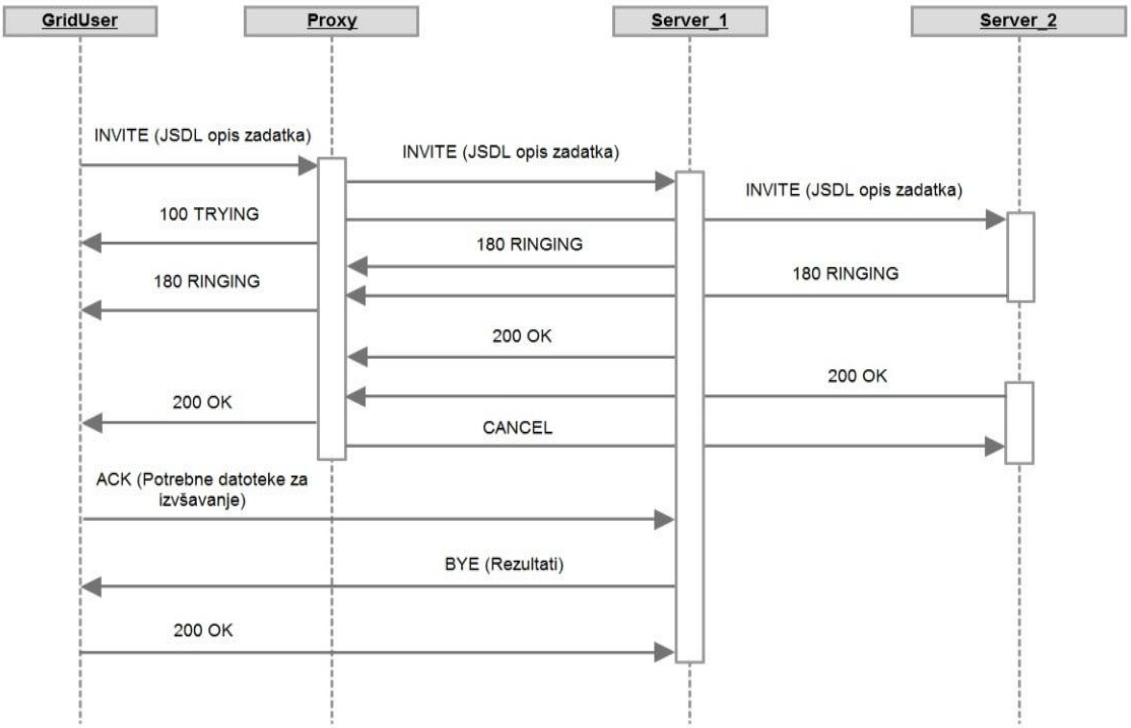
Посредничка и послужилачка апликација имају знатно једноставнију графичку корисничку спрегу из разлога што оне немају интеракцију са корисником у смислу управљања већ се та интеракција и ако постоји своди на надгледање. Изглед апликација приказан је на слици 3.3.



Слика 3.3 *Графичка корисничка спрега посредничке и послужилачке апликације*

3.2 Комуникација

Идеја је да се искористи SIP протокол за комуникацију међу апликацијама тако да су све поруке које се користи за комуникацију SIP поруке. Слика 3.4 илуструје комуникацију:



Слика 3.4 *Један пример комуникације у SIP грид систему*

Приликом покретања клијентске и посредничке апликације, обавља се аутоматско регистровање код послужиоца посредника. Посредник је *"stateful"* типа, што практично значи да ће он пратити стања сваког позива. Ово омогућава да се сви послужиоци региструју под једним именом. Захтев за позивом који је стигао од клијента посредник прослеђује свим послужиоцима, при чему послужилац који први одговори на захтев добија позив, док се комуникација са осталим прекида.

Након успешне регистрације, клијентска апликација може да подноси захтев послижиоцу посреднику. Кад корисник апликације поднесе захтев, створи се SIP порука типа INVITE. У телу ове INVITE поруки се дода опис задатка. Опис задатка је представљен XML форматом који у форми JSDL-а чува податке о задатку неопходне извршиоцу тј. послужилачкој апликацији.

INVITE поруку прихвата послужилац посредник и прослеђује је свим регистрованим послужиоцима.

Послужилац прихвата поруку и парсира тело поруке. На основу захтеваних ресурса из описа задатка (енг. *Job Requirements*) и стварних ресурса које има на располагању, послужилац одлучује да ли ће се кандидовати за извршавање или не. Уколико има задовољавајуће ресурсе послужилац одговара са поруком 200 OK, док у супротном игнорише INVITE захтев.

Уколико послужилац одговори са 200 OK поруком, поруку прихвата посредник и шаље је назад до клијента. Наравно, уколико се јави више послужилаца који могу да обаве задатак посредник дозвољава само једном од послужилаца комуникацију са клијентом, са осталим послужиоцима се прекида позив. Први послужилац који се јави посреднику добија пролаз односно успоставља везу са клијентом.

Када клијент добије поруку 200 OK, то значи да постоје одговарајући ресурси у систему и да може да пошаље задатак и потребне податке за извршавање. Клијент тада прикупи потребне податке, као што су извршна датотека и све остале пропратне датотеке које су наведене у опису задатка и спакује их у тело ACK поруке. Свака датотека се енкриптује са *"base64"* енкрипцијом и пакује у MIME поруку. Овако запакована порука шаље се назад посреднику послужиоцу.

Послужилац посредник, обзиром да је веза успостављена само прослеђује поруку послужиоцу.

Послужилац декодује поруку и извршава задатак. Целокупан резултат извршавања пакује се у тело BYE и то у MIME формату и шаље назад послужиоцу.

Клијент одговара потврдно и веза се раскида.

4. Програмско решење

Програмско решење је организовано као скуп Еклипс додатака, с тим што неки од додатака могу самостално да функционишу и практично предсатављају самосталну апликацију. Дакле, имамо додатке:

- Клијент (grid.client), клијентска апликација
- Посредник (grid.proxy), посредничка апликација
- Послужилац (grid.server), послужилачка апликација
- JSDL (grid.jsdl), подршка JSDL обради
- SIP (grid.sip), библеотека "*mjSIP"* са одређеним изменама

Свака од самосталних апликација (клијентска, посредничка и послужилачка) укључује по потреби остале додатке.

Оваквим приступом (енг. *Plugin Development*) остварују се значајне предности које се огледају у виду проширења апликације, додатка нових могућности, једноставне измене итд.

4.1 JSDL додак

Овај додатак намењен је обради JSDL података. Класе које су од значаја у овом додатку су:

- JSDLParser
- JSDLSystemCall
- JSDLResourceComparator

4.1.1 JSDLPasrser

Класа задужена за мапирање JSDL података у Јава класе и обратно. Битне методе које користе апликације су:

```
static public JobDefinitionType unmarshallFromFile(String filename)
```

Параметри:

filename – путања до JSDL датотеке

Повратна вредност:

Објекат класе JobDefinitoinType

Метода мапира податке из JSDL датотеке у јава објекте .

```
static public JobDefinitionType unmarshallFromString (String stream)
```

Параметри:

`stream` – стринг који чува JSDL опис задатка

Повратна вредност:

`JobDefinitoinType` класа

Метода мапира податке из JSDL низа карактера у јава објекте. Ова метода има примену приликом парсирања MIME поруке и преузимања података из ње.

```
static public ByteArrayOutputStream marshallToByteArray(JobDefinitionType
model)
```

Параметри:

`model` – објекат који садржи елементе JSDL описа

Повратна вредност:

Објекат класе `ByteArrayOutputStream`

Метода мапира податке из JSDL објекта у низ бајтова. Метода има примену приликом прављења MIME поруке.

4.1.2 JSDLSystemCall

Класа поседује само једну методу за покретање апликације на послужиоцу:

```
public static void run(String args)
```

Параметри:

`args` – аргументи командне линије

Повратна вредност:

Нема

4.1.3 JSDLResourceComparator

Класа задужена за поређење ресурса којима располаже послужилац и ресурса које захтева поднешен задатак. Поседује низ метода којима се проверава свако поље JSDL

спецификације. Да би задатак био прихваћен од стране послужиоца, послужилац мора задовољити све потребне ресурсе, информацију о томе добија позивањем методе:

```
public boolean compare (ResourcesType availableResource, ResourcesType requiredResource)
```

Параметри:

`availableResource` – објекат који садржи доступне ресурсе послужиоца

`requiredResource` – објекат који садржи захтеване ресурсе

Повратна вредност:

`True` или `False` у зависносто да ли послужилац задовољава тражене ресурсе.

Ова метода позива низ метода које проверавају свако поље JSDL спецификације и уколико бар једна од њих врати негативан одговор ова метода враћа негативан одговор.

4.2 SIP додатак

За реализацију комунакционе спреге грид клијента, послужиоца и посредника коришћене су већ постојеће реализације у оквиру *"mjSip"* библиотеке. Горе наведене апликације само инстанцирају потребне комуникационе класе, о томе ће бити више речи у наредном поглављу.

Битније измене у библиотеци *"mjSip"* су урађене над класом:

- `UserAgent`

Такође за потребе стварања и обраде MIME порука реализована је класа:

- `MimeUtility`

4.2.1 UserAgent класа

Методе намењене за комуникацију ентитета у SIP грид систему. У тексту који следи дат је преглед метода над којима су извршене измене.

4.2.1.1Успостављање сесије

Иницирање сесије се обавља позивањем методе:

```
public void call(String jobDescription)
```

Ова метода је задужена за иницирање комуникације тако што шаље INVITE поруку послужиоцу посреднику. Битан детаљ је да се у локалну поруку уписује дескриптор задатка који је прослеђен као стринг параметар `jobDescription`.

Пример позива методе из Грид клијента:

```
ua.call(JSDLParser.marshallToByteArray(job).toString());
```

Уписивање дескриптора у тело поруке је тривијално:

```
local_session = jobDescription;
```

4.2.1.2 Прихватање сесије

За прихватање сесије користи се метода која се извршава на послужиоцу и то након што посредник проследи INVITE поруку.

```
public void onCallIncoming(Call call, NameAddress callee, NameAddress caller, String sdp, Message invite)
```

У овој методи се обављају две битне фазе а то је: мапирање из JSDL форме у Јава класе (енг. *Unmarshalling*) и позивање методе која обавља проверу потребних ресурса. Део кода који је уметнут у постојећу функцију је следећи:

```
...
JSDLResourceComparator comparator = new JSDLResourceComparator();
JobDefinitionType required = null;
try{
        required = JSDLParser.unmarshallFromString(sdp); //mapiranje podataka, prva faza
}catch (Exception e){
        System.out.println("Error in unmarshalling sdp !");
}
If ( comparator.compare(availableResources.getJobDescription().getResources(), required.getJobDescription().getResources()) ) { //ako server zadovoljava kriterijum, druga faza
        System.out.println("Server accepted for the job !");
        setTempJob(required);
        accept(); //pozivanje metode koja odgovara sa 200OK
}
...
```

4.2.1.3 Одговор о успешном прихватању сесије

Након добијене поруке 200ОК од послижиоца посредника, клијент одговара са ACK поруком у коју се смешта потребне податке за извршавање. Метода која прихвата позив и обавља тражени задатак је:

```
public void onCallAccepted(Call call, String sdp, Message resp)
```

Промене које су од значаја у овој методи су: издвајање потребних датотека из JSDL спецификације, енкриптовање тих датотека, формирање MIME поруке и на крају попуњавање тела ACK поруке. Код изгледа овако:

```
...
List<String> inputFiles = new ArrayList<String>();
inputFiles.add(projectName+"\\"+tempJob.getJobDescription().getApplication().getPOSIXApplication().getExecutable().getValue()); //izdvajanje potrebnih datoteka iz specifikacije
for ( int i=0; i<tempJob.getJobDescription().getDataStaging().size(); i++ )
        if ( null != tempJob.getJobDescription().getDataStaging().get(i).getSource() )
```

```
                inputFiles.add(projectName+"\\"+tempJob.getJobDescription().getDataStag
                ing().get(i).getSource().getURI()); //izdvoj samo ulazne datoteke

local_session = grid.sip.tools.MimeUtility.createMime(inputFiles); //enkodovanje
datoteka, kreiranje MIME poruku i smestanje u telo ACK poruke
call.ackWithAnswer(local_session); // pozivanje metode koja salje ACK
...
```

4.2.1.4Обављање задатака и завршетак сесије

Послужилац прихвата ACK поруку методом `onCallConfirmed()`. Промене, односно додатак овој методи јесте низ операција којима се пристигла порука парсира, потом извршава и на крају резултати шаљу у телу BYE поруке.

Операције које се изводе у овој методи су: парсирање MIME поруке, декодовање и стварање датотека на локалном систему података, покретање извршне датотеке, енкодовање резултата, паковање у MIME поруку и враћање резултата у BYE поруци. Део кода задужен за извршавање горе наведеног је:

```
...
List<String> inputFiles = new ArrayList<String>();
inputFiles.add(tempJob.getJobDescription().getApplication().getPOSIXApplication().getExe
cutable().getValue());
for ( int i = 0; i < tempJob.getJobDescription().getDataStaging().size(); i++ ){
        if(null != tempJob.getJobDescription().getDataStaging().get(i).getSource()){
                //izdvoj samo ulazne datoteke
                System.out.println("Income file:
                "+tempJob.getJobDescription().getDataStaging().get(i).getSource().getUR
                I();
                inputFiles.add(tempJob.getJobDescription().getDataStaging().get(i).getS
                ource().getURI());
        }
}
grid.sip.tools.MimeUtility.extractMime(sdp, inputFiles); //parsiraj MIME poruku,
genirisi potrebne datoteke
String executableLine = new String();
executableLine =
tempJob.getJobDescription().getApplication().getPOSIXApplication().getExecutable().getVa
lue(); //izdvoj ime izrsne datoteke
String arguments = new String();
arguments =
tempJob.getJobDescription().getApplication().getPOSIXApplication().getArgument().get(0).
getValue();//izdvoj argumente komandne linije
if ( arguments.charAt(0) == ' ' ) //pobrisi suvisno prazno mesto ako postoji
        executableLine = executableLine.concat(arguments);
else
        executableLine = executableLine.concat(" ").concat(arguments);
JSDLSystemCall.run(executableLine); //izvrsavanje
List<String> outputFiles = new ArrayList<String>();
for ( int i=0; i<tempJob.getJobDescription().getDataStaging().size(); i++ )
        if(null != tempJob.getJobDescription().getDataStaging().get(i).getTarget())
        //izdvoj samo izlazne datoteke
```

```
                outputFiles.add(tempJob.getJobDescription().getDataStaging().get(i).get
                Target().getURI());
String jobResults = grid.sip.tools.MimeUtility.createMime(outputFiles); //enkoduj
datoteke i formiraj MIME poruku
call.bye(jobResults); // pozivanje metode koja salje BYE sa potrebni izlaznim datotekama
...
```

4.2.2 MimeUtility класа

Ова класа је написана као подршка парсирању MIME порука као и енкодовању и декодовању датотека из поруке. Поседује две битне методе:

```
public static String createMime(List<String> files)
```

Параметри:

`files` – листа са путањама до датотека које је потребно енкодовати и убацити у MIME поруку

Повратна вредност:

Низ карактера који представљају MIME поруку

Метода задужена за енкодовање задатих датотека и формирање MIME поруке.

```
public static boolean extractMime(String stringMsg, List<String>
inputFiles)
```

Параметри:

`stringMsg` –MIME порука представљена као низ карактера

`inputFiles` – листа са именима датотека које је потребно декодовати и потом створити на локалном систему података

Повратна вредност:

`True` или `False` у зависности да ли је успешно завршена метода

Метода задужена за распакивање MIME поруке, декодовање датотека и њихово стварање на систему података.

4.3 Клијент, посредник и послужилац додатак

Као што је напоменуто у тексту, ови додаци представљају додатке који могу самостално да се извршавају, тако да практично представљају апликације. Свака од ових апликација поред *"sip.grid"* и *"jsdl.grid"* укључује специфичне додатке као што су додатак за навигацију пројеката, едитор JSDL датотеке, конзолни испис итд.

Клијентска и послужилачка апликација поред графичког окружења које нуде Еклипс додаци инстанцирају класу `GraphicalUA` из *"mjSip"* библиотеке. која обезбеђује комуникацију.

Са друге стране посредничка апликација инстанцира `StatefulProxy` класу која практично преузима комуникацију на себе.

Програмско решење ових апликација своди се на укључивање додатака и подешавање XML конфигурациоих датотека, тако да детаљније разматрање програмског решења излази из опсега овог рада.

5. Испитивање и верификација

Испитивање је обављено коришћењем JUnit библиотеке. Испитане су функционалности *"grid.jsdl"* додатка, *"grid.sip"* додатка и провера да ли је извршен задатак коректан. Класе који су задужене за испитивање налазе се у оквиру додатка који се испитује.

Прликом испитивања *"grid.jsdl"* додатка проверена је класа `JSDLParser` и методе које се користе у оквиру апликација. Испитивање је обављено тако што је створена JSDL датотека за испитивање а потом су позиване методе које су обављале обраду користећи ту датотеку. Резултати су поређени са очекиваним вредностима и уколико су једнаки тест је сматран успешним. Проверено је мапирање из JSDL у класе и из класа у JSDL.

У оквиру овог додатка испитана је и класа `JSDLResourceComparator` тако што су проверавани случајеви да ли послужилаца може да изврши тражени задатак. Треба напоменути да је овај део кода посебно подложан грешкама јер дозвољава велики број комбинација, из тога разлога потребно је обавити додатна испитивања уз комбиновање тестних случајева а све у циљу повећања поузданости одговора.

Испитивање додатка *"grid.sip"* своди се на испитивање метода класе `MimeUtility`. Испитано је стварање и парсирање MIME поруке. Прво је створена тестна датотека познатог садржаја и кодована познатом енкрипцијом. Затим је створена MIME порука и датотека је убачена у поруку. Испитивањем да ли порука поседује кључне елементе MIME формата, закључује се да ли је створена исправна порука или не. Порука се затим парсира посебном методом и из ње декодује датотека. Уколико су декодована датотека и створена тестна датотека једнаког садржаја можемо сматрати да се компонента понаша у складу са очекивањем.

Последња фаза испитивања је провера да ли је успешно извршена обрада. Прво се ствара тестна датотека, тј. задатак се прво извршава на локалној машини. Након успешног подношења задатка послужиоцу и обраде, провера се да ли је датотека која је примљена једнака тестној датотеци. На тај начин утврђујемо да је систем коректно извршио тражени задатак.

Након обављеног тестирања елементи ситема су показали очекивану функционалност.

Такође, систем успешно обавља подношење и извршавање задатка. Међутим са функционалне стране постоји проблем у томе што је могуће само једанпут урадити подношење јер послужилац посредник не прекида позиве ка осталим сервирама и тиме их оставља у неконзистентном стању са становишта комуникације.

6. Закључак

У овом раду приказано је једно решење грид система заснованог на SIP протоколу. Након резултата испитивања можемо сматрати да је систем у складу са спецификацијом. Дакле након пријаве задатака грид систему, уколико си сви захтеви испуњени тражени задатак ће бити извршен а резултат враћен клијенту.

Систем обезбеђује минималне функционалности и са тог аспекта постоји простор за даље унапређење. На пример, потребно је обезбедити конфигурисање SIP протокола у току извршавања апликације а не из конфигурационе датотеке. Такође, корисничка спрега нуди само основне команде кориснику, што у комерцијалном систему није довољно.

Када су перформансе у питању примућује се да *"base64"* енкрипција траје релативно дуго у односу на остале компоненте комуникације, тако да би један вид убразања био или убрзати коришћену енкрипцију или користити неку алтернативну.

Битан детаљ који није реализован услед обима рада а свакако би представљао знатно убрзање је компонента за одлучивање који послужилац ће добити задатак. Та компонента би била практично диспечер на послужиоцу посреднику који услед квалитета ресурса послужиоца, расположивости послужиоца, брзине и других параметара одлучује који послужилац ће добити задатак.

7. Литература

[1] http://www.jwork.net/articles/grid/

[2] Telfor rad, SIP (Session Initiation Protocol), Bogović Vesna i Janković Željko, 2004 godina

[3] http://en.wikipedia.org/wiki/Job_Submission_Description_Language

[4] I. Basicevic, M. Popovic,"Session Initiation Protocol", "Encyclopedia of Internet Technologies and Applications", ed. Mario Freire, Manuela Pereira, Information Science Publishing, 2007

[5] Ilija: I.Basicevic, M. Popovic, D. Kukolj, "Comparison of SIP and H.323 Protocols", ICDT 2008, Bucharest, Romania, June 29- July 5, 2008.

[6] http://www.ietf.org/rfc/rfc3261.txt