УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Немања Милошевић

Сервер за транспорт порука од GNSS базних станица до покретних машина са кинематичким позиционирањем у реалном времену

ДИПЛОМСКИ РАД

- Основне академске студије -

Нови Сад, 2026.

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**:		**Немања Милошевић**	
Ментор, **МН**:		**проф. др Мирослав Поповић**	
Наслов рада, **НР**:		**Сервер за транспорт порука од GNSS базних станица до покретних машина са кинематичким позиционирањем у реалном времену**	
Језик публикације, **ЈП**:		Српски / ћирилица	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		2026	
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)		6 / 42 / 1 / 1 / 23 / 0 / 0	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница / Кључне речи, **ПО**:		GNSS, RTK, NTRIP, RTCM, Linux, multi-threading, e-poll	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		У овом раду је имплементиран интернет сервер за услуживање корисника којима су потребне корекционе поруке за прецизно позиционирање на Земљи. У раду су дате теоријске основе позиционирања са својим недостацима и подразумева упознавање са руковањем мрежних операција у Linux OS и начином на који је употребљено у развоју сервера. Интернет сервер са базним станицама чини окосницу мреже за услуживање корекција и представља критичну тачку у систему.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	проф. др Илија Башичевић	
	Члан:	доц. др Миодраг Ђукић	Потпис ментора
	Члан, ментор:	проф. др Мирослав Поповић	

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**:		**Nemanja Milosevic**	
Mentor, **MN**:		**Miroslav Popovic, PhD**	
Title, **TI**:		**A server for a message transport from GNSS base stations to a mobile rovers with real-time kinematic positioning**	
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**:		2026	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		6 / 42 / 1 / 1 / 23 / 0 / 0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		GNSS, RTK, NTRIP, RTCM, Linux, multi-threading, e-poll	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		As a part of the Bachelor's thesis, there is an implementation of internet server for users who need GNSS corrections for high-precision positioning on the ground. Document conveys theoretic basics of positioning with its flaws and encounters getting in touch with Linux networking and the way it's been used in the development of the server. Internet server is a backbone of the network for streaming corrections and it is a critical point in the system.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	Ilija Basicevic, PhD	
	Member:	Miodrag Djukic, PhD	Menthor's sign
	Member, Mentor:	Miroslav Popovic, PhD	

	УНИВЕРЗИТЕТ У НОВОМ САДУ • **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
		Датум:
	ЗАДАТАК ЗА ЗАВРШНИ РАД	

(Податке уноси предметни наставник - ментор)

Студијски програм:	Рачунарство и аутоматика		
Студент:	Немања Милошевић	Број индекса:	RA 200/2021
Степен и врста студија:	Основне академске студије		
Област:	Електротехника и рачунарство		
Ментор:	проф. др Мирослав Поповић		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ЗАВРШНИ РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;

НАСЛОВ ЗАВРШНОГ РАДА:

Сервер за транспорт порука од GNSS базних станица до покретних машина са кинематичким позиционирањем у реалном времену

ТЕКСТ ЗАДАТКА:

У оквиру овог дипломског рада потребно је урадити следеће:

1. Упознавање са GNSS системима и корекцијама грешке.
2. Упознавање са руковањем мрежних операција у Linux OS.
3. Израдити сервер за услуживање корисника.
4. Уочити предности и мане решења.
5. Валидирати имплементацију под оптерећењем симулацијом.
6. Написати текст дипломског рада.

Руководилац студијског програма:	Ментор рада:

21000 НОВИ САД, Трг Доситеја Обрадовића 6

ИЗЈАВА О НЕПОСТОЈАЊУ СУКОБА ИНТЕРЕСА

Изјављујем да нисам у сукобу интереса у односу ментор – кандидат и да нисам члан породице (супружник или ванбрачни партнер, родитељ или усвојитељ, дете или усвојеник), повезано лице (крвни сродник ментора/кандидата у правој линији, односно у побочној линији закључно са другим степеном сродства, као ни физичко лице које се према другим основама и околностима може оправдано сматрати интересно повезаним са ментором или кандидатом), односно да нисам зависан/на од ментора/кандидата, да не постоје околности које би могле да утичу на моју непристрасност, нити да стичем било какве користи или погодности за себе или друго лице било позитивним или негативним исходом, као и да немам приватни интерес који утиче, може да утиче или изгледа као да утиче на однос ментор-кандидат.

У Новом Саду, дана _____

Кандидат

Ментор

_____ _____

Захвалност

Захвалан сам Богу, својој породици и пријатељима на подршци и проведеном времену у току студирања као и професорима и другим наставницима од којих сам стекао знање као најдрагоценији алат на путу до испуњења својих животних жеља. Била ми је част и увек уживам у сарадњи са сјајним кадровима, посебно са ментором овог дипломског рада, проф. др Мирославом Поповићем.

Захвалност компанији Swift Navigation која ми је уступила право на коришћење врхунских илустрација у теоријским основама рада.

Немања Милошевић, 31.08.2026.

САДРЖАЈ

Скраћенице

GNSS -*Global Navigation Satellite System*, глобални навигациони сателитски систем

GPS -*Global Positioning System*, глобални систем за позиционирање

RTK -*Real Time Kinematic*, кинематичко позиционирање у реалном времену

NTRIP -*Networked Transport of RTCM via Internet Protocol*, протокол за пренос RTCM порука у интернету

RTCM -*Radio Technical Comission for Maritime Services*, радио-техничка комисија за поморске услуге

TCP -*Transmission Control Protocol*, протокол за контролу преноса

MEO -*Medium Earth Orbit*, средња орбита Земље

LEO -*Low Earth Orbit*, ниска орбита Земље

ECEF -*Earth Centered – Earth Fixed*, 3D координатни систем са центром у средини Земље

Cs -*Цезијум*, хемијски елемент чија особина атома служи као дефиниција за секунду као мерну јединицу

CG-NAT -*Carrier-Grade Network Address Translation*, пресликавање мрежних адреса на нивоу мобилног оператера

DDNS -*Dynamic Domain Name System*, динамичко освежавање сервера са адресама домена у интернету

UDP -*User Datagram Protocol*, протокол за пренос датаграма

MSC -*Message Sequence Chart*, графикон узастопних порука

LAN -*Local Area Network*, локална мрежа повезаних рачунара

RAM -*Random Access Memory*, меморија случајног приступа

CPU -*Central Processing Unit*, јединица за извршавање програма (инструкција)

DDOS -*Distributed Denial of Service*, напад на интернет сервер преоптерећењем

TLS -*Transport Layer Security*, протокол за обезбеђивање шифрованог преноса

HTTP -*HyperText Transfer Protocol*, апликативни протокол за пренос садржаја страница

ssh -*Secure Shell*, протокол сигурног преноса и пријављивања на удаљене сервере

DGNSS -*Differential GNSS*, диференцијални GNSS

1. Увод

Циљ пројекта је изградња сопственог решења интернет услужиоца (енгл. server) за пренос порука од GNSS базних станица до покретних машина (енгл. rover) са високо-прецизним кинематичким позиционирањем у реалном времену. Домен реалног времена задаје посебне захтеве за исправан рад система. Важно је сагледати учестаност испоруке података као и количину података који се шаљу и уочити потенцијалне тачке загушења које би могле пореметити рад читавог система. Као и сви други системи који функционишу у домену реалног времена, главни захтев овог система је пријем временски најсвежијих података и у оптималном случају, пријем свих података. Уколико се десе губици података у преносу из било ког разлога, ситуација може бити санирана испоруком најсвежијих података без форсирања за пријемом без губитака. Важно олакшање конкретног случаја је чињеница да је за разлику од аудио и видео преноса који имају велики пропусни опсег, није у питању интензивна комуникација са великом количином података.

Брзина одзива сервера је важан параметар у комуникацији између базних станица и корисника, поготово када је сервер под већим оптерећењем. Из тог разлога је одабран C++ програмски језик и рад са мрежном софтверском инфраструктуром оперативног система Линукс. C++ је компајлирани језик што значи да је након процеса компајлирања и асемблирања створена извршна датотека у којој се налази машински код односно низ инструкција које ће се извршити у процесору тог рачунара као и неке друге информације попут табеле симбола, адреса и вредности глобалних променљивих итд. У програмирању је омогућен приступ и манипулација меморијом и регистрима директно без надзора што индукује врло повољне околности по питању брзине извршавања програма, али и носи ризике попут неконзистентности и неправилног приступа меморији што може имати озбиљне последице по систем. Зато се овај језик сматра инжењерски изазовним и његове

предности доприносе томе да је широко употребљен у критичним системима и домену реалног времена [1].

У поглављу рада о теоријским основама долази се до детаљног упознавања са радом сателитских навигационих система, њиховим предностима и манама. Описују се одређене методе прецизнијег позиционирања, а потом је описана стандардизација и протоколи јавне мреже као темељ сваке интернет комуникације, тако и преноса у ком доприноси сервер који је тема овог рада. Поглавље о пројектовању система описује техничке детаље развоја сервера, предности и мане датих решења. Веома важна ставка је тестирање решења и евалуација рада. У завршној дискусији се акценат ставља на сигурност мреже и сервера, као и на начине приступа и реализације рачунара домаћина (енгл. host).

2. Теоријске основе

2.1 Увод у GNSS

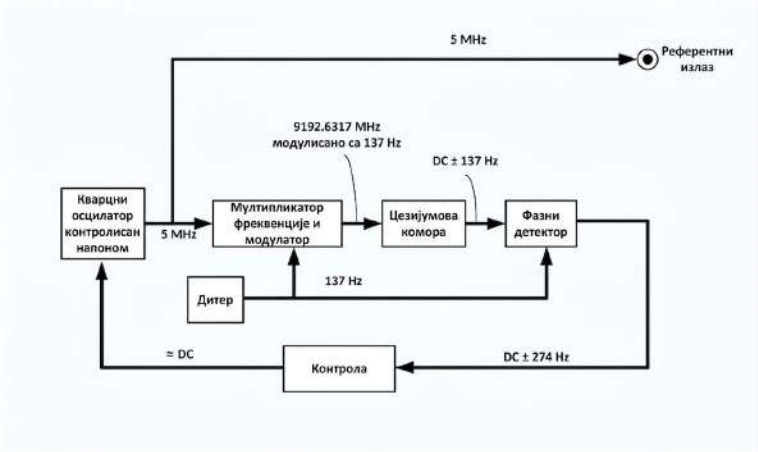
GNSS (енгл. Global Navigation Satellite System) представља навигационе системе за сателитско позиционирање. Постоји неколико имплементација као што су GPS, GLONNAS, Galileo и BeiDou у зависности из које државе потичу. Пријемници могу бити пројектовани тако да ослушкују све врсте GNSS система, а могу бити специфично пројектовани за, на пример GPS. GPS (енгл. Global Positioning System) је прва и најпознатија имплементација GNSS система те када се каже GPS, најчешће се мисли на цео GNSS скуп.

Сателити периодично емитују поруке на дефинисаним фреквентним опсезима (L1, L2, L5) и најчешће се налазе у средњој орбити Земље – MEO. Прва очигледна разлика између различитих имплементација GNSS технологије је формат порука, пријемници морају бити дизајнирани да ослушкују сигнале на овим фреквентним опсезима и да анализирају (енгл. parsing) поруке које шаље циљана GNSS имплементација. Сателити шаљу у оквиру тих порука своју позицију и време. На основу тих података, пријемници израчунавају своју позицију у простору и за то су им потребна бар четири сателита. Сателит шаље временски печат када је сигнал послат, заједно са својом позицијом у ECEF координатном систему. Електромагнетни таласи се крећу брзином светлости, тако да пријемник има све потребне параметре за израчунавање удаљености до свих сателита. Када се оне израчунају, добија се уски круг могућих вредности које представљају позицију пријемника (геог. ширина, геог. висина и надморска висина). Пошто је растојање огромно (~25000km), а пријемнику је потребна прецизна локација (~cm), и најмања грешка времена (~ns) доводи до велике грешке позиционирања, више стотина метара нпр. Зато је потребно да пријемник може да ухвати сигнал од што више сателита [3].

Слика 1.

Илустрација GNSS пријемника

Пошто је временско одступање критично у навигационим системима, сателити су опремљени са тзв. атомским сатовима, најпрецизнијим сатовима који постоје. Обичан кварцни осцилатор није довољно добар за прецизно мерење времена пошто има одступања и она зависе од спољашњих услова, а грешка се после неког времена акумулира. За високо прецизне осцилације се користе особине атома. За сваки атом је својствена резонантна фреквенција. Ствар је у томе што је фреквенција побуђивања електрона и њиховог повратка на нижи ниво фиксна и веома прецизна. Обичан кварцни осцилатор генерише сигнал који се користи за побуђивање електрона *Cs* при чему постоји детектор јачине деловања фотона. Јачина деловања фотона биће максимална само при идеалној фреквенцији осцилатора, уколико дође до њеног пада, кориговаће се фреквенција осцилатора. За атом *Cs* је својствена резонатна фреквенција од 9 192 631 770 Hz што представља 1 секунду у међународном систему за мерне јединице SI [4].



Слика 2.

Блок дијаграм атомског сата

Узгред, зарад одржања функције сваког GNSS сателита, потребно је да постоје земаљске станице које имају могућност слања података ка сателитима са циљем њиховог одржавања. Њихова позиција је тачно и прецизно утврђена, оне ослушкују сателите и израчунавају позицију и тако добијају одступање сателита, затим шаљу сателиту корекцију орбиталних података као што је путања сателита и сл. Такође, шаљу и корекцију времена, веома је важно да сви сателити имају синхронизовано време. Овај процес се обавља у просеку 1-2 пута дневно за сваки сателит [5].

Слика 3.

Земаљске GNSS станице

Као што је речено, GNSS сателит шаље временски печат и своју позицију у ECEF координатном систему. Пријемнику су потребна најмање четири сателита да разреши систем једначина и добије своје 3 позиције: геог. ширина, геог. висина и надморска висина. Пријемник коригује своје време на основу сателитског времена и због тога му треба тај четврти сателит. Сат пријемника и сателита нису савршено синхронизовани што уноси грешку у позиционирању. Та грешка се минимизује кроз ове једначине, стога пријемник захтева четири сателита.

$$p_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}$$

Формула 1.

Систем једначина за израчунавање позиције пријемника

i - један од присутних сателита

(x,y,z) - сопствена позиција пријемника

(x_i, y_i, z_i) - позиција сателита i

p_i - удаљеност до сателита i

$$p_i = c \cdot (t_i - \Delta t)$$

Формула 2.

Израчунато време и временско одступање

c - брзина светлости

Δt - разлика сателитског времена и времена пријемника

t_i - израчунато време путовања сигнала од сателита i, узимајући у обзир локални сат

У овим једначинама непознате су три координате пријемника и временска разлика сателитског сата и сата пријемника. Систем једначина је нелинеаран и решава се итеративним нумеричким методама попут Гаус-Њутновог поступка [2].

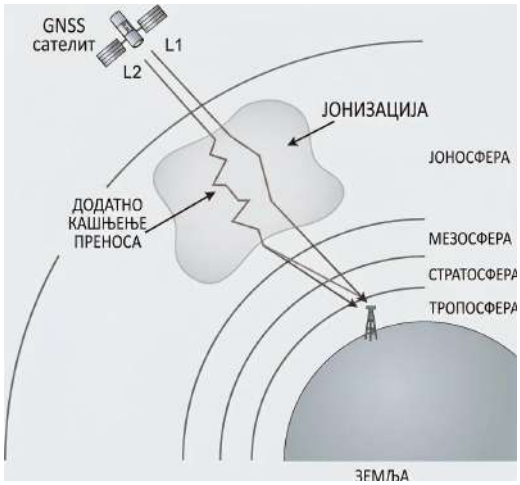
2.2 GNSS одступање и побољшање квалитета

По овом приступу, не постоји савршено решење које може довести до израчунавања идеално тачне позиције пријемника, увек постоји грешка на нивоу ~5-10m због успоравања сигнала у атмосфери при чему брзина кретања таласа није више једнака брзини светлости. За многе примене, овај резултат је сасвим довољан, међутим код навигација покретних машина, у грађевини, геодезији и сл. грешка у позиционирању мора бити готово у потпуности елиминисана.

Одступање локалног сата од прецизног сата сателита највише има утицаја када је кашњење сигнала веће, теже је елиминисати ту грешку у једначинама. Одступање од само 1 микросекунде доводи до грешке од чак 300 метара у позиционирању.

Највећи допринос кашњењу сигнала је успоравање сигнала у атмосферском слоју јоносфере. Управо је због тога тај слој атмосфере и добио назив - јоносфера. Слој у ком је енергија јонизације умногоме већа од енергије рекомбинације. У преводу, на надморској висини од око 50km-500km ваздух је ређи него у тропосфери. Простирање ултраљубичастог Сунчевог зрачења изазива избацивање електрона из атома (услед деловања енергије) што за последицу даје електромагнетно зрачење у нанометарским таласним дужинама што наше очи иначе детектују као светлост. То зрачење даље побуђује електроне у другим атомима и њихово хаотично кретање. У тропосфери, густина ваздуха је највећа и електрони убрзо по побуђивању и напуштању свог атома бивају привучени од стране позитивног јона, односно шупљине коју могу попунити у другом атому. Напуштање атома и постанак слободним електроном је процес јонизације, а обрнут процес попуњавања позитивног јона од стране електрона је процес рекомбинације. Можемо рећи да је у тропосфери, енергија јонизације превладана енергијом рекомбинације, слободни електрони убрзо по јонизацији бивају рекомбиновани. То наравно није случај у јоносфери где је густина ваздуха мања, тиме слободни електрони остају неко време слободни пре њихове рекомбинације. Тако да можемо рећи да је енергија јонизације у овом слоју већа од енергије рекомбинације. Појава да постоји већа

количина слободних електрона у ваздуху узрокује успоравање електромагнетних таласа које сателити емитују [7].



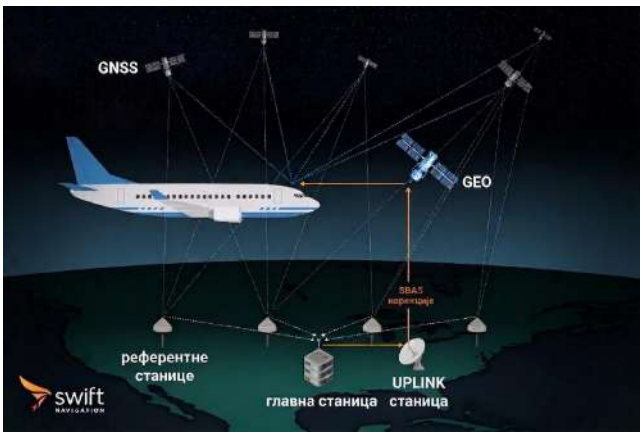
Слика 4.

Илустрација јоносферског кашњења

У мањем обиму постоји и утицај тропосферског кашњења као што су временски услови, притисак, влажност ваздуха итд. Такође сложеност терена је још један изазов, одбијање сигнала од зграде и друге објекте на Земљи прави ефекат одјека и може да збуни пријемнике.

DGNSS (енгл. Differential GNSS) је једна од метода исправке GNSS одступања. Подразумева постојање референтне станице у близини која познаје своју позицију и за сваки сателит израчунава грешку и шаље то пријемнику. Пријемник прима те податке и примењује их на израчунату позицију и успева да сведе грешку позиционирања на <1m. Имплементација није скупа и систем је релативно једноставан [8].

SBAS (енгл. Satellite-Based Augmentation System) је још један систем у ком постоје референтне станице на Земљи које се налазе на различитим локацијама и прикупљају податке од GNSS сателита и израчунавају грешку. Постоји централни ентитет који прикупља податке од референтних станица и потом шаље корекције ка гео-стационарним сателитима који се налазе у високој орбити Земље. Потом они емитују корекције које су доступне у различитим регионима света. Ово је грубља корекција грешке, јер су у питању велики региони, утицај атмосфере није свуда исти и грубља је геометрија до сателита ако је референтна станица далеко од пријемника коме требају корекције. Предност је доступност и систем је за кориснике једноставан. Генерално је употребљен у авијацији [8].



Слика 5.

Илустрација SBAS система

слика преузета уз дозволу од компаније SwiftNavigation

GNSS сателити уводе додатну имплементацију са циљем смањења одступања позиције а то је слање сигнала на више фреквенција. Фреквентни канали су означени као L1, L2 и L5. Ствар је у томе што се атмосферски ефекат успоравања сигнала не одражава једнако на све фреквенције електромагнетног зрачења које емитују сателити. Пријемник је имплементиран тако да има могућност да прати сигнале на тим фреквенцијама. Затим упоређује кашњење између њих и тиме ефикасније израчунава одступање позиције [3].

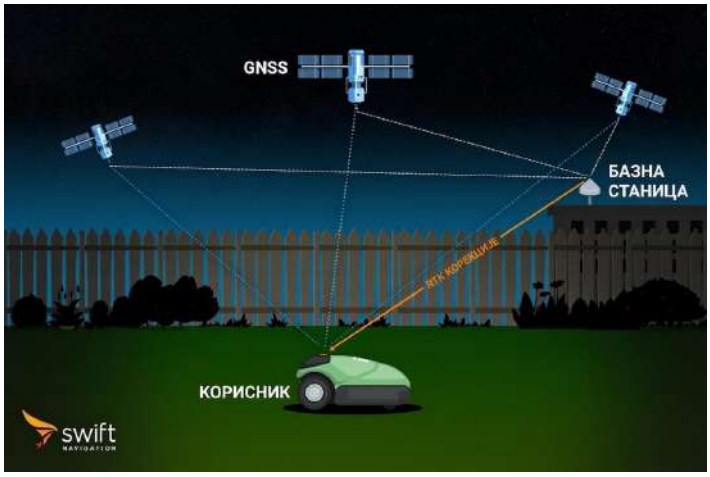
2.3 RTK корекције

RTK (енгл. Real-Time Kinematic) важи за методу најпрецизније исправке GNSS одступања, сводећи га на ниво 1-2cm. Надограђује DGNSS поступак, такође подразумева референтну (базну) станицу која пружа корекције. Базна станица познаје своју позицију, ослушкује сигнале сателита, израчунава GNSS позицију и пореди је са познатом. Тиме се добија одступање и грешка за сваки сателит, то је операција над подацима које сателит шаље и грубо коригује грешку.

Сателити константно шаљу сигнал носилац који не носи никакве корисне податке, а периодично на тој фреквенцији шаљу поруке, односно сигнал носилац модулисан подацима. У RTK приступу, након пријема поруке од сателита, пријемник наставља да ослушкује тај сигнал носилац који он емитује и броји периоде таласа за прецизно рачунање удаљености до њега. Ако је фреквенција L1 опсега једнака 1.57GHz, таласна дужина је 19cm. Бројањем периода таласа добија се прецизнија удаљеност до сателита који емитује тај сигнал. И даље постоји потреба за базном станицом која познаје своју тачну позицију, познавањем своје позиције она може да израчуна тачан број периода таласа и тиме удаљеност до сателита. Покретни пријемници не могу да одреде тачан број периода таласа до сателита, јер увек постоје кашњења и потенцијални одјек сигнала у

атмосфери и на површини, с друге стране базне станице које познају своју тачну позицију могу да исправе ту грешку и одреде тачан број периода таласа за сваки сателит. Још један изазов је целобројна неодређеност - број пуних (правих) периода таласа између сателита и пријемника. Важно је одредити егзактан број периода таласа јер грешка за само 1 период таласа носи грешку у позиционирању за 19cm. Овај процес подразумева поређење мерења између пријемника и базне станице и узима у обзир неколико сателита.

Добра страна ове методе је максимално елиминисање грешке позиционирања и брзо достизање високе прецизности. Примењује се у навигацији пољопривредних машина, беспилотних летелица и других аутономних возила, грађевини, геодезији и осталим системима где је таква прецизност неопходна. Систем је сложен, потребне су базне станице у максималној удаљености од 20km као и повезивање са њима. Базне станице морају бити у близини како због различитих атмосферских услова у удаљеним областима, тако и због остваривања приближне геометрије сателита са пријемницима да се тачан број периода таласа које пријемник прима ефикасно одреди [8].



Слика 6.

Илустрација RTK система

слика преузета уз дозволу од компаније SwiftNavigation

RTCM (енгл. Radio Technical Commission for Maritime Services) поруке су стандардне корекционе поруке које се преносе од базних станица према покретним машинама и свим другим пријемницима којима су потребне RTK корекције. Преносе све потребне информације пријемницима попут грубих корекција за сателите и финијих корекција сигнала носиоца тако да пријемници могу израчунати своју позицију прецизно [9]. Стандард је развијен 1990их година, а данас је најшире у употреби верзија 3.x. У складу са стандардом постоје различите врсте порука попут: 1001, 1002, 1005, 1230 и сл. Обично се шаљу једном у секунди. На пример, једна порука 1002 представља детаље у вези L1

мерења за GPS. Физички пренос ових порука од базних станица је данас путем интернета, а некада је био реализован путем радио везе [19].

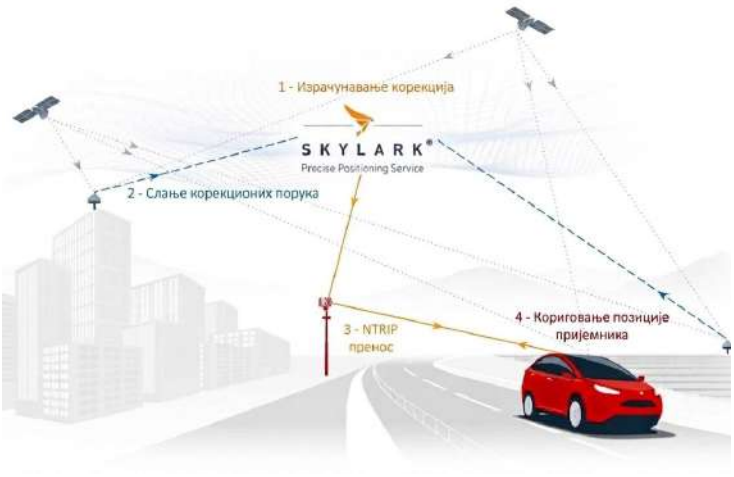
NTRIP (енгл. Networked Transport of RTCM via Internet Protocol) је протокол апликативног нивоа који се ослања на HTTP протокол и стандардизује комуникацију између базних станица, централног сервера и уређаја пријемника којима су потребне корекције. Настао је у новије доба са развојем интернет инфраструктуре и тиме заменио некадашњу комуникацију базних станица са пријемницима преко радио везе.

У NTRIP свету постоје три улоге рачунарских система:

1) NTRIP Caster - централни сервер који служи као главна тачка, односно мост у комуникацији базних станица са пријемницима. Све базне станице из мреже шаљу корекционе поруке на овај сервер а сервер их даље шаље према пријемницима који их захтевају од конкретне базне станице. Сервер не обрађује податке, него их усмерава. Ова улога је управо тема овог дипломског рада.
2) Базна станица (енгл. source) - најчешће микроконтролер који преузима готове RTCM корекционе поруке од GNSS/RTK модула који је извршио сва мерења и дигиталну обраду сигнала сателита и израчунао корекције које су потребне пријемницима. Те поруке шаље путем интернета ка централном серверу.
3) Пријемник (енгл. rover) - GNSS модул коме су потребне корекције да би елиминисао одступање позиције и потребна му је висока прецизност у навигацији покретних машина, возила итд.

Изазови у овом приступу могу бити: захтев за сталним присуством стабилне интернет везе, софтверска и хардверска компатибилност. На пример, изузетно је важно при развоју NTRIP система испратити стандардизацију, јер у супротном може доћи до ограничења употребљивости истог [10].

Постоје две верзије NTRIP-а и обе су у употреби. V2 је више везан за HTTP користећи његове стандардне захтеве као што је POST.



Слика 7.

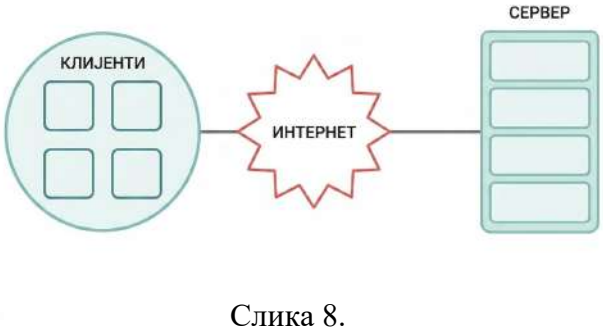
Пример NTRIP сервиса

слика преузета уз дозволу од компаније SwiftNavigation

2.4 Интернет архитектура

Модерна стандардна интернет архитектура подразумева клијент-сервер везу. Клијенти су уређаји односно апликације на њима којима је потребна услуга или пружају одређену услугу. Имају или динамичке јавне IP адресе, или приватне унутар локалних мрежа ради уштеде адресног простора у глобалној мрежи. Све адресе у мобилном интернету су такође приватне пресликаване са CG-NAT техником коју изводе мобилни оператери. [17] Реализација да клијенти траже једни друге у целој глобалној мрежи би била изузетно непрактична и скупа, неупоредиво је једноставније да се обраћају на једном рачунару односно једној IP адреси у интернету. Приступ споља у приватну мрежу је сложен, могућ је у ограниченом обиму под условом подешавања усмеривача који је излазни чвор у јавни интернет.

Сервери су рачунари који пружају одређене услуге клијентима. Често имају јавну статичку IP адресу да би били јавно видљиви. У супротном морају бити одржавани преко DDNS сервиса. [18] Ако је у локалној мрежи, подешава се прослеђивање на бази пролаза (енгл. port-forwarding) опција у усмеривачима који су излаз у јавну мрежу. Углавном су засновани на захтевима, чекају захтеве који су дефинисани апликативним протоколом и у складу са истим одговарају клијентима. Представљају мост између клијената и контролишу њихову комуникацију. Мана овог приступа је чињеница да је сервер тачка отказа и искључивањем рачунара сервера искључена је цела мрежа. Такође, уколико мрежа расте, сервер се налази под све већим оптерећењем и потребно га је на неки начин појачати ресурсима [15].



Слика 8.

Клијент-сервер архитектура

С обзиром на различитости рачунара и мрежа у јавном интернету, постоји потреба за чврстом стандардизацијом комуникације у интернету. На сцену ступа OSI модел који слојевито описује протоколе и енкапсулацију података између њих. Сваки протокол мора да дефинише скуп порука и њихов формат, једнозначно и формално опише обраду порука између ентитета и да опише обраду грешака [16].

Слика 9.

Слојеви интернет протокола

1. Апликациони протоколи подразумевају поруке које носе податке потребне самој апликацији, углавном је на нивоу захтев-одговор с тим да апликација зна шта значе који захтеви и шта ће одговорити на њих. Ту спадају HTTP, FTP, као и NTRIP. Убрајају још једну групу протокола који служе за асистенцију при анализи података, синхронизацији, безбедности података итд.
2. Транспортни протоколи представљају контролу комуникације с краја на крај. Ту спада протокол сигурне комуникације TCP који подразумева обезбеђивање информације да је порука стигла на одредиште и корака шта радити даље ако није стигла у одређеном временском интервалу, као и протокол брже комуникације UDP који је једноставан, без успоставе везе, али ентитети немају потврду да ли су сви подаци стигли на одредиште или не. Адреса транспорног нивоа је пролаз, то је један број који идентификује процес, односно апликацију на одредишном рачунару.

3. Мрежни протокол је IP и описује начин како се пакети података усмеравају у мрежи. Адреса овог протокола је јединствена адреса у целој мрежи и на основу те адресе се врши усмеравање према уређају који је поседује. Углавном су IP адресе хијерархијски подељене и уређаји близу једни других имају сличну адресу. На основу тога се и усмеравају кроз мрежу.
4. Протоколи везе обезбеђују комуникацију између два уређаја у мрежи, често између два усмеривача негде у интернету или усмеривача и одредишног уређаја нпр. Потребно је да два уређаја која комуницирају познају MAC адресу један од другог да би поделили податке између себе. То су протоколи Ethernet, WiFi, Bluetooth и други.
5. Физички слој представља медијуме преко којих се преносе подаци. Могу бити бакарне жице, оптички каблови или електромагнетни таласи у случају бежичне комуникације. Тражи се јасна дефиниција шта су нуле и јединице на нивоу физичког сигнала [14], [16].

Оваквом поделом се поједностављује развој сваког ентитета у мрежи и обезбеђује независност између њих. Јасно је да апликација не жели да зна начин преноса сигнала, нити да брине о томе да ли су подаци стигли, исто тако транспортни протокол не брине о томе шта се налази у пакетима података и о начину преноса сигнала, а с друге стране слој везе једино треба да обезбеди да подаци стигну у суседни уређај и познаје начине како то да обезбеди и сл. [14].

Сви ови протоколи се реализују од стране рачунара у мрежи и од стране различитих софтверских модула унутар истог рачунара који су специјално намењени за своју функцију и независни су од осталих модула. Исто тако су и пакети података независни једни од других, сваки слој надоле енкапсулира пакет у садржај свог пакета и надогради оквир који значи другом уређају односно софтверском модулу који прима ту поруку и обавља исту функцију на истом нивоу [14].

Слика 10.

Нивои протокола и капсулација података

3. Пројектовање система

3.1 NTRIP (Broad)Caster

Као што је објашњено у уводном делу, NTRIP централни сервер је једна од три улоге у NTRIP свету за пренос корекционих порука за исправку GNSS грешке. Главна је тачка којој приступају базне станице које пружају и пријемници којима су потребне RTK корекције. Погодно тло за развој оваквог сервера је Linux OS (Ubuntu) и C++ програмски језик са својим g++ компајлером. Излазна датотека компајлирања програма је извршна датотека.

Мрежне операције се ослањају на Линуксов мрежни стек увозом заглавља <*sys/socket.h*>, <*netinet/in.h*>. У оквиру њихових функција имплементирана је TCP/IP подршка којом рукује језгро оперативног система, а кориснички процес позива те операције системским позивима. Утичница (енгл. socket) је датотека која представља један пар "IP адреса - пролаз" и садржи све потребне информације које се тичу једне TCP везе. Позивањем системских позива *socket(), bind()* и *listen()* ствара се утичница намењена за долазне захтеве која представља пар сопствене IP адресе рачунара и пролаза на ком овај кориснички процес ослушкује захтеве. Позивањем *accept()* системског позива процес се блокира на неодређено време док не стигне нови захтев за повезивањем на тај пролаз. Када језгро оперативног система детектује TCP захтев за повезивањем на тај пролаз, процес ће бити одблокиран, стављен у спремне и при поновном активирању приступиће баферу у ком се налази пакет података који је стигао и копирати га у свој кориснички

простор, тј. локални бафер. Потом се извршава стандардна TCP успостава везе, рукована од стране језгра односно мрежне подршке ОС и отвара се веза.

Поред тога, пре него што почне главна петља програма, сервер чита и анализира (енгл. parsing) датотеке (sources.txt, clients.txt) у којима се налазе регистроване базне станице и регистровани корисници са њиховим корисничким именима и шифрама. Смешта их у посебан вектор објеката и приступа му приликом сваког захтева за повезивањем и проверава интегритет корисника. Сваки корисник мора да приступи са корисничким именом и шифром, а базне станице са шифром у складу са NTRIP v1 протоколом. База података корисника је једноставна, на нивоу текстуалне датотеке са форматом: *name_identification username:password*. Ради једноставности, постоји глобална статичка листа (низ) свих базних станица, са максимално 100 база. Када се нова база прикључи, њен индекс у овој листи се ажурира и користи се надаље та референца, односно адреса тог објекта и све се извршава над њим.

Базне станице и корисници су описани структурама података. За базне станице постоји *struct source* која садржи променљиве везане за име базе, синхронизацију приступа подацима, бафер података, сигнал да ли је тренутно повезана, веза са статичком листом база. Корисник је описан структуром *struct rover* и за њега је такође везано име, бафер, показивач на базу коју жели да слуша, њен индекс у статичкој листи база.

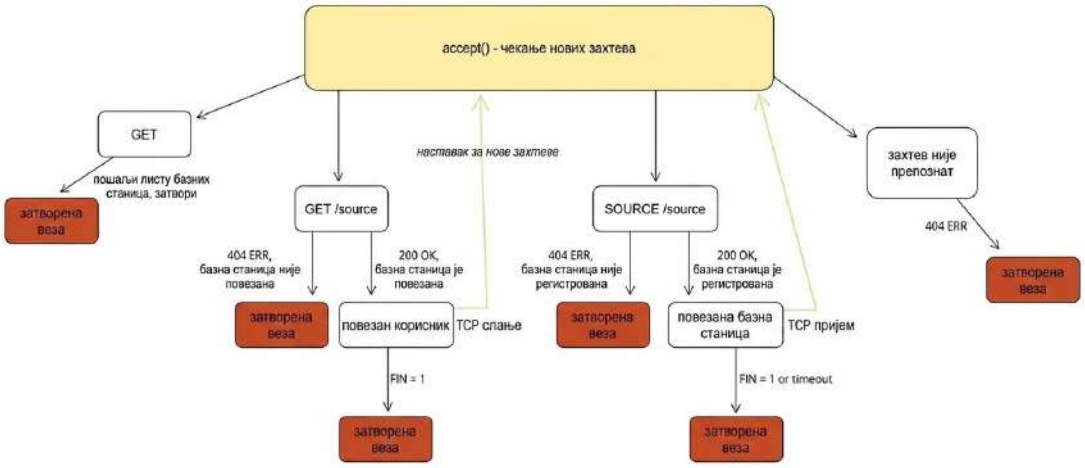
```
typedef struct sourceStruct{    // bazne stanice
    char name[256];
    mutex m;    // medjusobna iskljucivost niti za pristup istom baferu
    condition_variable cv; // uslovna promenljiva za obavestenje ostalih niti rovera koje cekaju te podatke
    uint8_t recvBufferSource[RECV_BUFFER_SIZE] = {0};          // bafer podataka koji se puni sa jedne baze i salje se roverima koji tu bazu slusaju
    uint64_t data_ready_counter = 0;  // brojac koji ukazuje nitima rovera da sad ima raspolozivih podataka
    ssize_t bytesReceived = 0;   // koliko je podataka stiglo od baze
    atomic<bool> connected {false};
    uint16_t index; // id u listi registrovanih baznih stanica (veza izmedju te 2 strukture)

}source;

typedef struct roverStruct{ // roveri
    char base[256]; // na koju se bazu kaci
    uint8_t sendBuffer[RECV_BUFFER_SIZE] = {0}; // bafer koji treba poslati roveru.
    source* s;      // bazna stanica na koju se rover pretplacuje
    bool connected_to_source = false;
    int16_t index;  // indeks korisnika u listi registrovanih korisnika
    int16_t index_at_registered_sources;  // indeks bazne stanice na koju je povezan rover (u listi registrovanih baznih stanica)
}rover;
```

Слика 11.

Структуре које описују базе и кориснике

Почетни захтев када се нови корисник или базна станица повежу на сервер након успоставе TCP везе јесте NTRIP/HTTP захтев. У NTRIP v1, базна станица шаље **SOURCE/base_name_id** а корисник приступа са **GET/** да добије листу база односно **GET/base_name_id** за претплату на конкретну базну станицу. У NTRIP v2, дошло је до приближавања HTTP протоколу, па тако базне станице шаљу **POST/base_name_id** када се

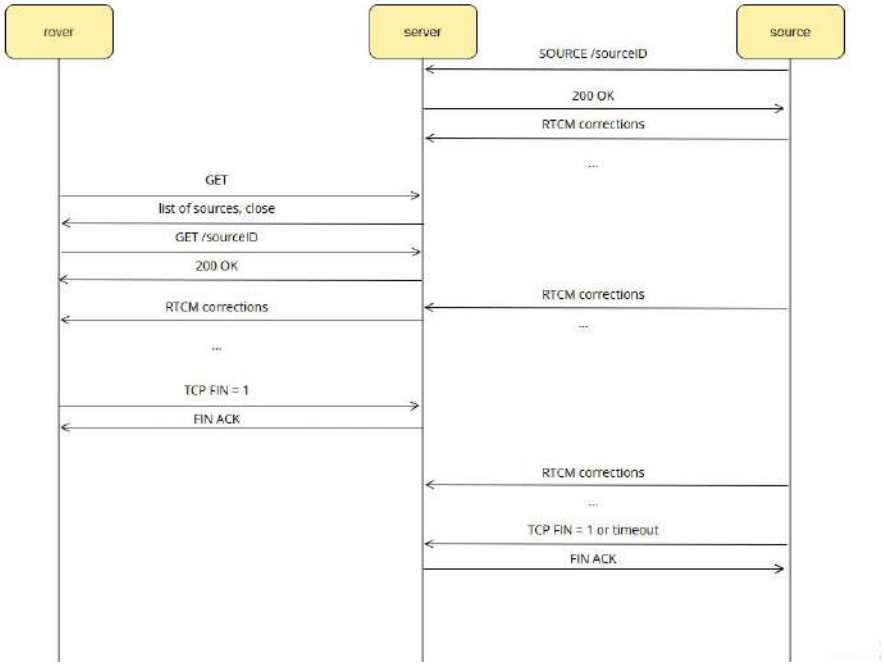
прикључују на сервер као и корисничко име и шифру. Надаље се наставља размена података преко обичног TCP/IP слања и примања, односно базне станице шаљу ка серверу, а корисници примају податке од сервера. Сваки протокол и комуникациони систем је потребно описати формалном спецификацијом. Протоколи и комуникациони системи се могу представити као аутомати са коначним бројем стања (енгл. Finite State Machine). Сва стања и промене између њих морају бити програмски покривени током развоја телекомуникационог система. Блок је један део система, описује шта ради једна целина.



Слика 12.

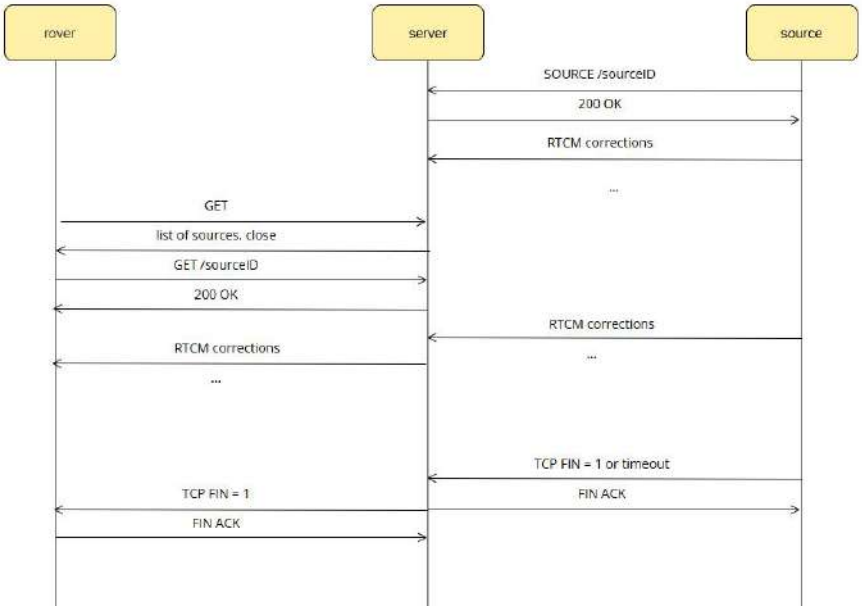
Дијаграм стања (енгл. State diagram) NTRIP сервера

Постоји још један формални језик, за опис комуникационог система MSC (енгл. Message Sequence Chart) који приказује размену сигнала. Његова презентација је графичка. Омогућава да се догађаји у времену боље сагледају. Дијаграм стања нам олакшава да видимо сва стања у систему и прелазе између њих, док MSC омогућава боље сагледавање догађаја и размену сигнала (порука) у времену по корацима.



Слика 13.

MSC дијаграм - случај 1



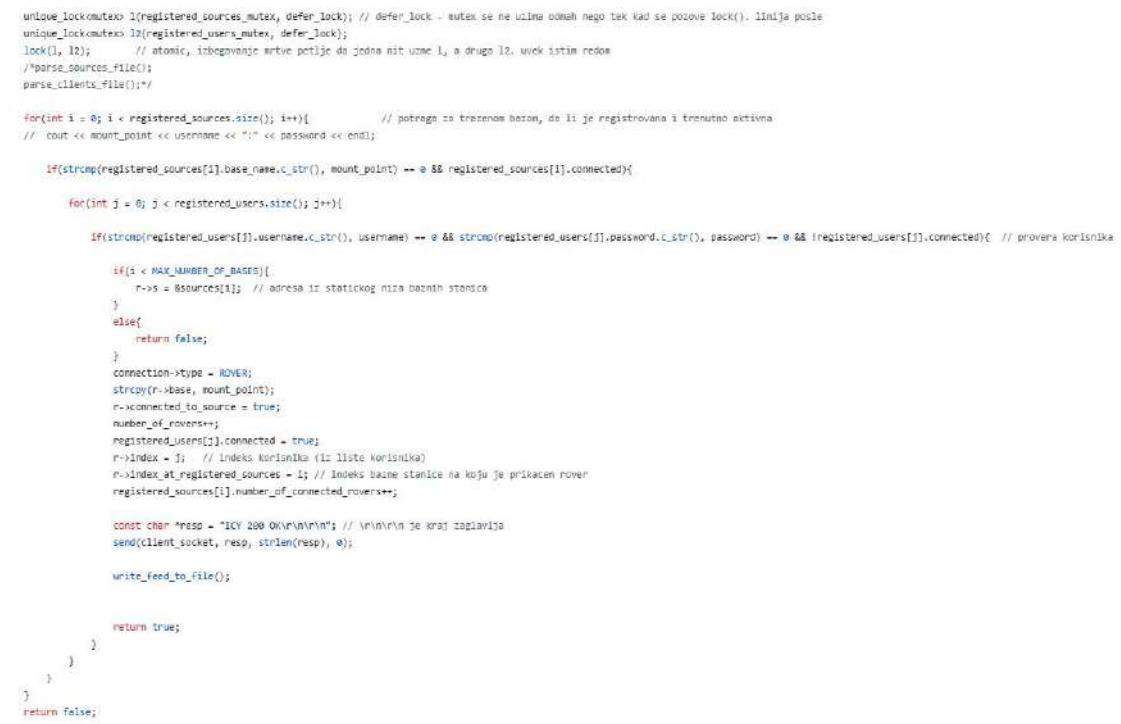
Слика 14.

MSC дијаграм - случај 2

Анализу HTTP захтева обавља функција <bool *HTTP_recv_analyze(char* recvBuffer, int client_socket, connStruct* connection, rover* r, uint8_t* ind)*> која враћа да ли је захтев препознат и успешан или не. Све информације о врсти захтева и иденфитикацији ко га је послао се налази у заглављу HTTP пакета. *<sscanf(recvBuffer, "%15s %255s %255s", method, data1, data2)>* линија ради читање прве линије заглавља. Тиме функција добија врсту захтева и два податка - име базне станице и шифру (уколико га шаље базна станица, NTRIP v1). Уколико корисник шаље захтев, корисничко име и лозинка се налазе у пољу "Authorization" заглавља HTTP пакета. Оно је енкриптовано тзв. B64 енкрипцијом. Тиме се добијају сви параметри за идентификацију захтева и корисника. B64 енкриптује сваких 6 бита података са неким бројем 0-63. Смисао кориштења ове енкрипције података у HTTP је омогућавање преноса бинарних података, да се неки бајт не би погрешно протумачио од текстуалних анализатора, тако се онда сви бајтови преносе сигурно у вредностима 0-63 који не могу бити неки специјални карактери који би потенцијално збунили текстуалне анализаторе.

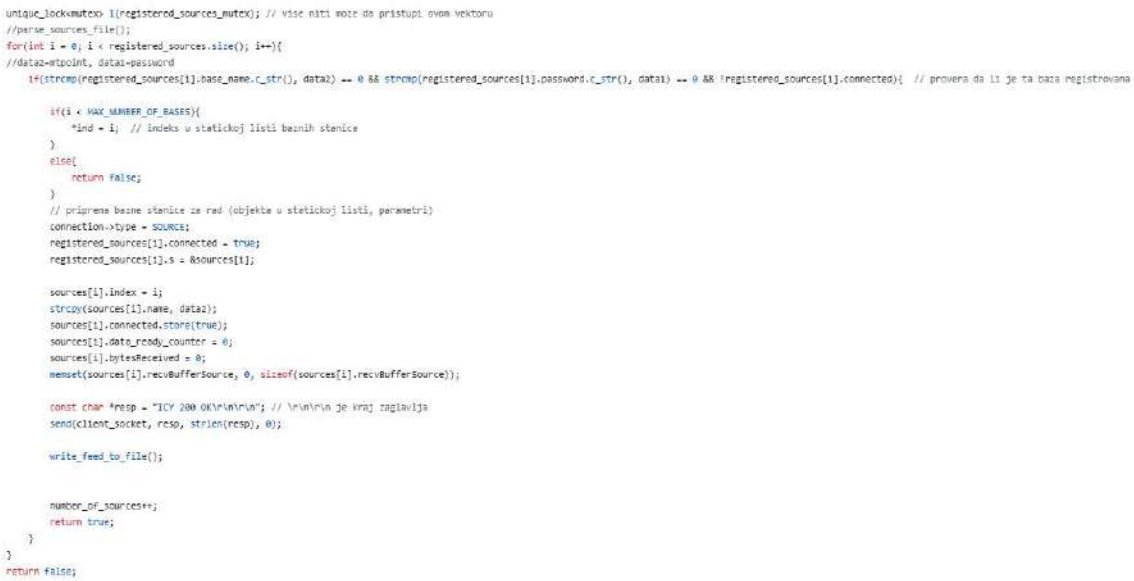
У сваком догађају успешног повезивања/искључивања на сервер од стране базне станице или корисника, та информација се исписује у "stdout" излазни бафер процеса у виду броја прикључених базних станица и корисника. Такође постоји посебна датотека (logs/connections.txt) у коју се исписује које су базне станице прикључене (њихова имена) и број корисника који тренутно слушају конкретну базну станицу.
Када корисник приступа са GET/ да би добио листу базних станица, сервер шаље одговор у складу са обе верзије NTRIP, наизменично по један одговор. Уколико клијенту не одговара тај одговор, он ће сигурно покушати поново и добиће исправан одговор сервера. Тако да постоје две датотеке које сервер шаље као одговор на GET/ захтев: "sourcetable.txt", "sourcetable_ntripv2.txt". У њима је у складу са форматом који дефинише NTRIP протокол дата листа регистрованих базних станица, врсте подржаних RTCM порука, њихове локације уколико клијентски софтвер има функционалност да се аутоматски повеже на најближу базну станицу и неки други подаци у вези базних станица.

У случају непознатог захтева или базна станица/корисник није регистрован, шаље се грешка HTTP404 и прекида веза.



Слика 15.

Обрада захтева за прикључењем који је послао корисник



Слика 16.

Обрада захтева за прикључењем који је послала базна станица

3.2 N веза - N нити

Првобитно је развијена верзија сервера која ради на принципу да свакој новој вези односно кориснику и базној станици придода једну посебну системску нит обраде. Нити се извршавају паралелно и конкурентно. Систем је релативно једноставан јер се главна обрада налази у оквиру једне функције и адреса те функције је предата при прављењу нити, све што се тиче те везе се обавља у тој функцији. Оно што је мана овог приступа је ризик од трке до података (енгл. Data race) између нити што може имати фаталне последице по извршавање програма, тиме и стабилност мреже. Мора се пажљиво водити рачуна о безбедности података и синхронизацији приступа критичним секцијама. То се решава или међусобном искључивошћу користећи "*mutex*" објекте или атомским приступом. У овом решењу су кориштена оба приступа. Без адекватне бриге о синхронизацији приступа критичним секцијама долази до случаја када нити које се истовремено извршавају на различитим процесорским језгрима истовремено приступају истим меморијским локацијама, односно променљивама и низовима. Та дељена меморија између нити којој оне приступају је угрожена, а делови програмског кода где се извршавају такве операције се зову критичне секције. Уколико би оставили да им нити приступају спонтано, долазило би до корупције података и неконзистентног стања ако оне истовремено приступе и покушају да упишу две различите вредности у исту меморијску локацију. Коришћењем "*mutex*" објеката испред критичних секција се решава овај проблем, то су такође системски позиви. Језгро води рачуна које све нити желе да приступе критичној секцији и формира ред чекања, и како која нит заврши рад са дељеном меморијом, тако следећа улази (бива одблокирана и распоређена у процесор) и обавља своје операције. За једноставне операције над дељеним променљивама се може користити атомски приступ. Обезбеђује се да се цела операција над променљивом обави у оквиру једне инструкције, а тиме у току извршавања те инструкције друга процесорска језгра нису у могућности да приступе тој локацији у меморији. То су специјалне инструкције и морају бити хардверски подржане од стране архитектуре на којој се извршава програм. Први приступ преко "*mutex*" објеката интерно користи други (атомски) приступ за обезбеђивање основне пропуснице којој приступају нити.

Слика 17.

Незаштићена критична секција и конкурентни приступ

У развоју NTRIP сервера, за примитивне промењљиве типа "*bool*", "*char*" и сл. користи се атомски приступ, једноставан је и нема потребе за додатним објектима. Тако је имплементирана синхронизација за промењљиве које прате број прикључених базних станица и корисника, заставицу (енгл. flag) да ли је базна станица присутна итд. Сложеније критичне секције у којима постоје петље или више промењљивих које треба заштитити се штите "*mutex*" објектима, најчешће "*unique_lock*" приступом који закључава блок програмског кода.

```
cout << "Server pokrenut... " << endl;
while (true) {  // ceka na pojavu zahteva od nekog klijenta, kad se pojavi, pravi nit i obradi

    // proces se stavlja u red na tom portu, i blokira se, od OS
    // ako se pojavio neki klijent koji zeli pristup serveru, OS obavestava ovaj proces i on nastavlja
    // prihvata init pakete (TCP handshake) i dobija dodatne info o njemu, i salje svoje info klijentu
    int client_socket = accept(listen_socket, (sockaddr*)&clientAddr, &clientAddrSize);
    clientAddrSize = sizeof(clientAddr);  // novi klijent, accept() moze promeniti ovo
    if (client_socket < 0) {
       // cout << "greska accept";
        continue;
    }
    cout << "neko zove"<<endl;
    thread t(client_service, client_socket, clientAddr);    // nova nit za novog klijenta. client_service - funkcija niti i prima argumente info o novokonektovanom klijentu
    t.detach(); // nit radi samostalno, main nit nastavlja sa izvrsavanjem
}
close(listen_socket);
```

Слика 18.

Стварање нове нити по повезивању новог корисника

Као што видимо у слици 18, након што је потврђена нова веза и створена нова мрежна утичница у системском позиву *accept(),* ствара се и нова нит за ту везу и предаје јој се функција "*client_service(int fd)*" којој се као аргумент прослеђује идентификатор мрежне утичнице за ту везу и даље та нит самостално ради са њом, прима или шаље податке. Прва ствар коју ради та функција је чекање HTTP/NTRIP захтева да анализира која је врста везе у питању и шта даље да ради са њом. Ако је успешно иденфитикован

корисник или базна станица, функција се даље грана на случај базне станице и случај корисника. За случај базне станице, ставља се временска контрола (енгл. timeout) на максимално 55 секунди блокирања нити док не стигну подаци, уколико премаши тај период, сматра се да се нешто догодило са базном станицом и она се искључује. Користе се условне променљиве (енгл. condition variable) за сигнализацију када су подаци стигли од базне станице. Када су подаци стигли, повећава се бројач *data_ready_counter* који је део структуре базне станице и својствен је свакој бази и сигнализира се условна променљива са *notify_all()* позивом. У петљи се враћа на *recv()* системски позив и чекају се нови подаци. За то време, нити корисника који слушају ту базну станицу долазе до критичне секције и једна по једна позивају *wait_for()* чекање максимално 55 секунди и блокирају се на улазу у критичну секцију до доласка сигнала условне променљиве. Када подаци стигну, копирају их у свој интерни бафер и то шаљу кориснику кроз TCP *send()* системски позив. Функција брине о томе да су послати сви подаци који су копирани у интерни бафер. Након тога, понављају се претходни кораци. Када се напусти бесконачна *while(true)* петља из било ког разлога (нпр. истекло време чекања) освежава се број прикључених базних станица и корисника, бришу се заставице о статусу везе и сл. и позива се системски позив *close(int fd)* за брисање мрежне утичнице и ослобађање њене меморије у оквиру језгра оперативног система и евентуално искључење TCP везе у складу са тим протоколом. Када корисник шаље захтев за прикључењем на неку базну станицу, при провери валидности, потребно је приступити и вектору који садржи регистроване кориснике и вектору који садржи регистроване базне станице. Пошто постоје два објекта "*mutex*" који штите та два ресурса, зарад избегавања међусобног блокирања нити користи се "*defer_lock*" механизам који заузима пропусницу увек истим редоследом. То је један од начина избегавања међусобног блокирања у конкурентном програмирању.

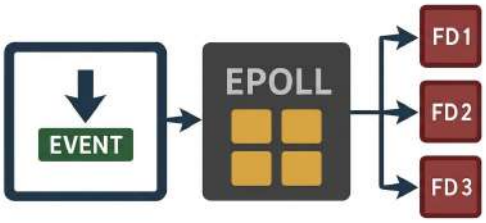
Као што је речено, предност овог приступа је концептуална једноставност, постоји једна функција, свака нит је извршава и све је на једном месту у програмском коду. Када се једном обезбеди међусобна искључивост нити и заштита критичних секција, систем ће радити без проблема. Оно што је интригирајуће у овом приступу, а о томе ће бити више речи у поглављу о тестирању, јесте шта ће се десити са више стотина или хиљада TCP веза. То значи да ће постојати по једна нит за сваку везу, која мора бити распоређена да би се та веза одржала и обавила коректно, у реалном времену. Повољна чињеница је да RTK ток података није интензиван, али и даље поставља се јасно питање каква је перспектива оваквог приступа са толико много веза. Иза системских позива се крије слој софтвера језгра оперативног система који троши време и тај фактор не сме бити занемарен. Распоређивање нити у процесор, обрада системских позива, драјверске операције слања за

сваку нит могу да изазову велику системску режију (енгл. overhead) на великом оптерећењу.

3.3 e-poll приступ

Са циљем повећања скалабилности и перформанси система на великом броју веза, развија се верзија која користи уграђени Линуксов механизам за руковање У\И операцијама - epoll. То је део језгра, надгледа мрежне утичнице и бива обавештен да ли су могуће некакве операције над њима. Овај приступ је секвенцијалан, у оквиру једног процеса / системске нити. Тај главни процес прво позива *epoll_create()* системски позив за иницијализацију epoll инстанце у језгру, затим *epoll_ctl()* за додавање мрежних утичница које требају да буду праћене и коначно у петљи *epoll_wait()* и бива блокиран док се не појави неки нови догађај. Када пакет података стигне на неки мрежни улаз дешава се хардверски прекид, језгро га обрађује у оквиру прекидне рутине, идентификује који је тачно мрежни улаз у питању, проверава табелу уређаја и проналази адекватан драјвер за дату комуникацију који комуницира са мрежном картицом [20], односно другим уређајем у мрежи и идентификује мрежну утичницу на основу пристиглог пакета. Наставља се извршавање у оквиру системског простора, затим epoll механизам има увид у то које су мрежне утичнице спремне. Уколико је стигао нови пакет података на одређени пролаз, односно мрежну утичницу од интереса, процес бива враћен у спремне процесе и наставља се његово извршавање где је стао. Преузима податке који су стигли и све информације у вези тог пакета и враћа се у кориснички простор. Могуће је да процес покупи и више пакета уколико су стигли у међувремену док је процес ре-активиран односно распоређен. Ово је типичан механизам комуникације базиран на догађајима, без периодичног проверавања (енгл. polling) за разлику од неких претходника попут “poll” или “select” који периодично проверавају све утичнице од интереса, што на великом броју веза није ефикасно [11].

Као апстракција рада са epoll механизмом, користи се софтверска библиотека *libevent*. Заснована је на почетној иницијализацији и функцијама повратног позива (енгл. callback). Намењена је за асинхроди рад са дескрипторима датотека, односно мрежним утичницама.



Слика 19.

Приказ рада epoll

Потребно је прво дефинисати структуре *struct event_base* и *struct evconlistener* из *libevent* библиотеке. Библиотека се увози преко заглавних датотека "*event2/...*". Иницијализација библиотеке се ради преко *event_base_new()* функције која у позадини подешава epoll, затим се подешава и слушалац (енгл. listener) са *evconnlistener_new_bind()* за слушање нових веза и покреће се петља са *event_base_dispatch()* [12]. Преко "callback" функција библиотека обавештава апликацију о догађајима. Тако постоји *accept_conn_cb()* која се позива након успоставе нове TCP везе. Овде се и на апликативном нивоу прави нова веза, упарује се са мрежном утичницом и прави се објекат структуре *struct bufferevent* који представља чвор комуникације са том мрежном утичницом. Меморија за ове објекте се заузима динамички.

```
// callback kad se prikljuci novi klijent, TCP handshake ( jos uvek nema podataka )
static void accept_conn_cb(struct evconnlistener* listener, evutil_socket_t fd, struct sockaddr* address, int socklen, void* ctx){
    // ctx - proizvoljni (user-defined) kontekst
    struct event_base* base = evconnlistener_get_base(listener);


    connStruct* conn = (connStruct*)malloc(sizeof(connStruct)); // identifikacija nove konekcije
    if(!conn){
        close(fd);
        return;
    }

    conn->type = NONE; // jos uvek nepoznat korisnik
    struct bufferevent* bev = bufferevent_socket_new(base, fd, BEV_OPT_CLOSE_ON_FREE);  // registracija novog klijenta i uparivanje sa njegovim socket-om, dinamicka alokacija
    bufferevent_setcb(bev, read_cb, NULL, echo_event_cb, conn);
    bufferevent_enable(bev, EV_READ | EV_WRITE);

}
```

Слика 20.

Имплементација функције повратног позива након прихвата везе

Затим постоји *read_cb()* у ком као аргумент добија *struct bufferevent* објекат као идентификацију на којој мрежној утичници се десио догађај, односно пријем података. Као и кориснички (енгл. context user-defined) показивач који је искориштен да се пренесе информација о тој вези на апликативном нивоу, односно да утврди да ли је то корисник,

базна станица или још увек непознат клијент. Ако је још увек непознат, онда се очекује пријем HTTP/NTRIP захтева за идентификацијом клијента. Након тога, та информација се пакује у објекат на апликативном нивоу и сваки следећи пут при пријему података, има увид у то каква је то врста клијента. У овој имплементацији сервера, свака базна станица има вектор објеката корисника (енгл. rover) односно листу корисника који је слушају. При пријему података од базне станице, пролази се кроз ту листу и подаци се шаљу свим корисницима који је слушају. Зато је *struct bufferevent* објекат својствен сваком кориснику, односно део је структуре корисника.

```
typedef struct roverStruct{ // roveri
    char base[256]; // na koju se bazu kaci
    uint8_t send0uffer[RECV_BUFFER_SIZE] = {0}; // bafer koji treba poslati roveru.
    bool connected_to_source = false;
    int16_t index;  // indeks korisnika u listi registrovanih korisnika
    int16_t index_at_registered_sources;  // indeks bazne stanice na koju je povezan rover (u listi registrovanih baznih stanica)
    struct bufferevent* bev;
}rover;

typedef struct sourcestruct{    // bazne stanice
    char name[256];
    mutex m;    // medjusobna iskljucivost niti za pristup istom baferu
    condition_variable cv; // uslovna promenljiva za obavestenje ostalih niti rovera koje cekaju te podatke
    uint8_t recv0ufferSource[RECV_BUFFER_SIZE] = {0};           // bafer podataka koji se puni sa jedne baze i salje se roverima koji tu bazu slusaju
    ssize_t bytesReceived = 0;   // koliko je podataka stiglo od baze
    atomic<bool> connected {false};
    uint16_t index; // id u listi registrovanih baznih stanica (veza izmedju te 2 strukture)
    std::vector<rover*> connectedRovers;

}source;
```

Слика 21.

Структуре које описују базе и кориснике

Структуре података које описују базе и кориснике су сличне онима у првој имплементацији са мањим разликама које се односе на “libevent” библиотеку.

Још једна значајна функција повратног позива је она која се позива у случају прекида везе или истека временске контроле - *echo_event_cb()*. Као и у неким другим неуобичајеним догађајима. Проверава се који је то клијент, ако је корисник, мора да се освежи вектор базне станице коју је слушао, као и да се ослободи меморија коју су заузели његови објекти. Ако је искључена базна станица, морају да се раскину везе са свим тренутно повезаним корисницима који је слушају. Позивањем *bufferevent_free()* за неког клијента позива у позадини исти овај *echo_event_cb()* за њега тако да се и његови подаци бришу а меморија ослобађа. На крају се освежава број базних станица и корисника, и ти подаци за праћење рада сервера се уписују у излазну датотеку.

3.4 Тестирање

У развоју софтвера, један од најважнијих корака је верификација и валидација. То је процес провере да ли софтвер испуњава критеријуме захтева и спецификације. Верификација је провера да ли софтвер извршава задатак без компромиса, а валидација је провера да ли је тај задатак то што је тражено. Иако су веома слични термини и често помешани, још једна разлика је "Верификација је питање: Да ли развијамо софтвер добро? А валидација је питање: Да ли развијамо прави софтвер?". Фокус је на софтвер у рачунарским мрежама, односно у овом случају на серверски софтвер. Верификација подразумева исправност самог програмског кода, са акцентом на конкурентност. Валидација се више односи на понашање система у току рада на великом оптерећењу, пропусну моћ, губитак података и сл. [13].

Тестирање у оквиру рачунарских мрежа може подразумевати, за почетак, проверу понашања система при повезивању клијената у виду других процеса у истом рачунару. То је одлична почетна одлука, највише се односи на верификацију кода. Уколико постоје логички или технички проблеми са програмским кодом, програмске грешке (енгл. bugs) ће убрзо изаћи на видело, без да је домен тестирања изашао изван рачунара на ком се извршава сервер. Може се лако пратити понашање система у раду и исправност података при прикључивању клијената као посебних процеса. Наравно, комуникација унутар истог рачунара је веома брза и на нивоу приступа меморији и не приказује реално стање у коме серверу приступају корисници из удаљених локација, кашњење кроз интернет мрежу је главни фактор. Право оптерећење се огледа када се укључе други рачунари који комуницирају са сервером преко интернета.

Типови тестова могу бити:

- **Провера тачке** - фокус на проблематичан случај
- **Тест оптерећења** - симулација дуготрајног рада
- **Комбинација** - провера граничних случајева
- Стварни сценарио рада

Евалуација је процес оцењивања резултата испитивања у складу са стандардима. Циљ је проверити исправност система, усклађеност са корисничким захтевима и тачност рада. Кључни параметри евалуације код NTRIP сервера би био квалитет рада покретне машине којој су потребне RTCM корекције, уколико је сервер проблематичан чвор у мрежи, уноси кашњења и сл. проблеми ће се показати у реалном раду. Ако софтвер који контролише аутономно возило, исправно управља са њим, држи правац и суседни трагови којима пролази се граниче на нивоу 1-2cm односно онако како је тражено захтевима аутономног управљања, може се закључити да и сервер који пружа податке ради исправно. У

инжењерству рачунарских система, потребно је проверити и испитати рад сваке компоненте и модула система. Најчешће се то ради рачунарским симулацијама које могу да имитирају реалан сценарио са којим ће се систем сусрести у раду.

Покретање извршне датотеке сервера се ради или преко командне линије у терминалу као "./caster2026vX" или као "systemd" позадински сервис. "systemd" је системски сервис у Линуксу који покреће и надгледа извршни програм. Извршава се као део корисничког простора. Покреће извршни програм као посебан процес, али му преусмерава стандардни излаз у своју датотеку коју форматира тако што додаје информације попут времена и сл. Аутоматски прати потрошњу RAM меморије, CPU времена, број активних системских нити и неке друге информације.

За потребе тестирања написани су посебни програми који симулирају рад базне станице и корисника. Потребно је да симулирани корисници и базна станица буду регистровани у систему и да се програму дају корисничка имена и шифре, у чему највише помажу аргументи командне линије при покретању. Такође је написана посебна "bash" скрипта која покреће већи број корисника са својим креденцијалима. Сваки процес који симулира корисника исписује у излазну датотеку количину информација коју је примио и тест се сматра успешним ако сви корисници имају релативно идентичан број примљених података. RTCM ток података није значајно интензиван, али је важно да поруке нису старије од 1-2 секунде. Пропусна моћ је процењена на максимално ~1kB/s. То се узима као сценарио најгорег случаја. Коначно је узето да симулатори базних станица шаљу 1000 B/s.

Првобитно се тестира понашање система када се са истог рачунара прикључи 1 базна станица и 300/500/1000 корисника који је слушају. Комуникација је интензивнија од најгорег случаја. Шаље се 1000 бајтова, сваких 100ms. Овај тест је успешан, сви корисници су примили подједнаку количину података и очекивану у складу са временом рада. Важи за обе верзије сервера. Узимамо случај једне базне станице и N корисника јер је то најгори случај, то значи да N нити / дескриптора чекају на ту једну базну станицу.

Најзначајније тестирање је тест оптерећења у реалним условима у интернет мрежи. Уводи се други рачунар на ком се покрећу симулатори базне станице и корисника. Тај рачунар се прикључује на другу локалну мрежу, односно приступну тачку (енгл. hotspot) тако да подаци морају проћи пут кроз интернет док не стигну до рачунара на ком се извршава сервер. Уско грло код овог приступа тестирања јесте то што велики број корисника потиче са истог рачунара и сви пакети морају проћи кроз исту приступну тачку и може доћи до загушења или тог рачунара или те приступне тачке. У реалном раду, велики број корисника може приступити, али су сви са различитих локација и њихове приступне тачке нису под оптерећењем с обзиром да само њих услужују. Поновљивост

ове ситуације у тестирању је врло тешко остварива, најреалнији тест би био случај те ситуације у раду. Најважнији параметри тестирања мреже под оптерећењем:

- Да ли су сви корисници одржани на вези, није било губитака корисника?
- Да ли су сви примили подједнаку количину података и количина података је у складу са протеклим временом?

Број корисника / приступ	**multi-thread**	**e-poll**
300	OK	OK
500	OK*	OK*
1000	OK*	OK*

* - два рачунара у истој локалној мрежи

Управо због проблема велике системске режије приступне тачке, са 500 и више корисника, рачунар сервер и рачунар који симулира базну станицу и клијенте су стављени у исту локалну мрежу, дели их усмеривач. Наравно, нису исти услови у локалној мрежи и у интернету, али с друге стране, сценарио да 500 корисника приступа са исте приступне тачке такође није могућ.

Поврх ових тестова, сервер је тестиран у реалном раду, пружајући 24/7 корекције са максималном дужином рада од око 45 дана пре рестартовања. Прикључене су 4 базне станице и максималан број корисника у једном тренутку је био око 15. Задовољени су захтеви за пријемом RTCM порука, то се највише доказује чињеницом да су навигациони уређаји задовољили своје захтеве, а који зависе од порука које добијају од базних станица преко NTRIP сервера.

Слика 22.

Приказ рада навигационог система у пољопривреди

Ердевик, 15.4.2026.

4. Дискусија и резултати

4.1 О систему, безбедност и правци развоја

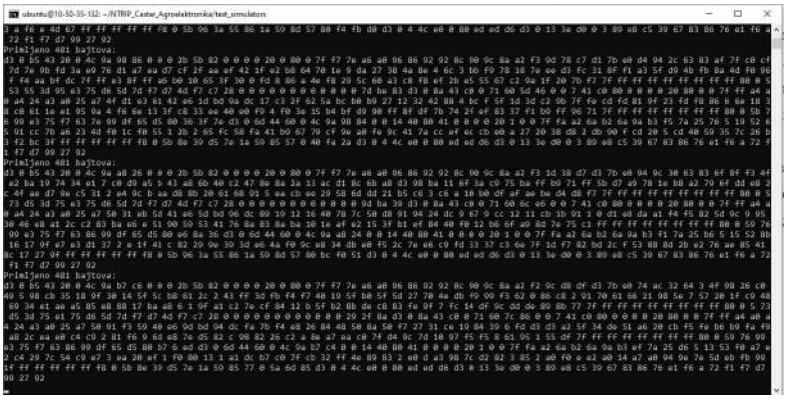
Као и у свакој другој клијент-сервер архитектури, тако и у NTRIP свету, сервер је централна тачка и тачка отказа система. При приступу великог броја корисника којима не само да тренутно требају корекције, него уопште њихов процес рада не може бити извршен без њих, важност стабилности и брзог одзива система постаје критична. Инжењерима који учествују у развоју таквих система је потребна свест о предностима и манама, односно потенцијалним узроцима нестабилности система како тренутно, тако и убудуће при ширењу мреже. Поред квалитетно израђеног софтвера за услуживање преко интернета, важни су и хардверски ресурси рачунара на ком се извршава сервер. Најважније особине би биле количина RAM меморије у коју је потребно сместити информације о системским нитима, дескрипторе мрежних утичница, бафере података који се преносе и сл.; процесорска снага и брзина, како радни такт тако и број процесорских језгара односно број системских нити које је могуће паралелизовати. Подразумевано, мрежни аспект као најважнији, најбоље је да тај рачунар комуницира са глобалном мрежом не-бежично односно да буде повезан LAN каблом до првог усмеривача а даље оптичким кабловима у остатак јавне мреже, због брзине преноса података и смањеног ризика од губитака пакета и колизије са другим бежичним рачунарским комуникацијама. С обзиром да је јавна IP адреса сервера статичка, потребно је и обезбедити сигурност система, отпорност на DDOS нападе и било какве покушаје диверзије над системом. У случају нестанка електричне енергије, обезбеђивање наставка рада рачунара је од критичног значаја. Једно од решења може бити употреба специјализованих усмеривача који имају уграђену и конфигурабилну подршку за наведене захтеве (попут “Teltonika”

усмеривача). Друго решење, оно које је и употребљено је коришћење услужног рачунара (енгл. Cloud Computing Services, Virtual Private Server - VPS) на коме се извршава сервер. Један пример таквог је “Surfercloud hosting”, постоји одабир спецификација жељеног рачунара: оперативни систем, ресурси. Укључена је аутоматска заштита мреже, додуше не постоји безусловна гаранција, али је врло значајно постојање заштите сервера.

Пошто се NTRIP увелико ослања на HTTP протокол, што значи пренос не-шифрованих података, тиме је ток података изложен трећим агентима у току преноса у инфраструктури мреже. Међутим, то се не сматра критичним јер подаци нису осетљиве природе, није важно да ли неко може да их пресретне или не. Коришћење обичног HTTP протокола у том случају поједностављује и убрзава успоставу везе јер нема потребе за криптографским корацима у складу са TLS (енгл. Transport Layer Security) протоколом. Регулацију и стандардизацију NTRIP протокола обавља RTCM организација.

Даљи развој овог решења NTRIP сервера може да узрокује нека нова верзија тог протокола, или могуће, неки ретки случајеви непокривених делова тренутне две верзије, односно појава неког навигационог уређаја који може да изазове неки непокривен сценарио од стране сервера. Тренутно, сервер не покрива случај базне станице са POST захтевом. Такође, могућ је развој естетских аспеката при одговарању на GET/ захтев, уколико је захтев стигао из веб претраживача. HTML (енгл. Hyper Text Markup Language) код се може уградити у HTTP одговор чиме би претраживач приказао кориснику одређени GUI (енгл. Graphic User Interface), то би могла бити мапа у којој су уцртане базне станице са својим дометима и неким другим детаљима.

Приступ VPS рачунару се обавља преко “ssh” протокола, даље се задају команде из терминала (енгл. shell) за приступ датотекама. За тестирање рада неке базне станице се користи извршна датотека клијент симулатора. Приказује број примљених података и њихов садржај.



Слика 23.

Пример теста базне станице на серверу

На слици 23 се види пријем бинарних RTCM порука у реалном времену, први бајт сваке поруке је 0xD3, у складу са форматом.

4.2 Развој GNSS и превазилажење RTK

Јавља се нова метода исправљања GNSS грешке, а то је PPP (енгл. Precise Point Positioning) која се ослања на референтне станице које су ређе расуте по свету. Те станице израчунавају временску и орбиталну грешку и достављају је централном ентитету који те податке шаље сателитима. GNSS сателити шаљу то као додатне информације. Пријемници процењују број периода таласа до сателита и временом га рафинишу и тиме достижу високу прецизност (~10cm), али време конвергенције је превише велико, потребно је 30мин - сат времена за постизање такве прецизности, што је резултат одсуства локалне базне станице. То је погодно за навигацију у поморству [8].

Развија се још једна метода, PPP-RTK која решаваману претходне методе и достиже високу прецизност у кратком времену. Овде постоје референтне станице које су гушће распрострањене него у претходном приступу, али и даље нису толико густе као код обичног RTK. Комбинује претходни приступ са RTK. Моделирано рачуна корекције за сваки регион. Предност над RTK је што геостационарни сателити могу да шаљу те корекције према пријемницима, тиме је за кориснике јефтинија изградња система, јер нема потребе за базном станицом и интернет приступом. Мана је, што прецизност ипак није толико висока као код RTK, достиже се ~5cm грешке у позиционирању што је врло висока прецизност али за неке примене и даље недовољна. Што не значи да у будућности неће достићи и највишу, потенцијалном изградњом још референтних станица. Традиционални RTK користи OSR (енгл. Observation Space Representation) приступ што значи слање корекција у облику који је везан директно за позицију. PPP-RTK користи SSR (енгл. State Space Representation) приступ који шаље корекције у одвојеним компонентама (атмосферски извор грешке, број периода таласа за сваки сателит и сл.) тако да корисници из ширег региона могу користити корекције [8].

5. Закључак

Развијен је NTRIP сервер са основном функцијом, а то је усмеравање порука од базних станица ка корисницима. Сервер је само један ентитет у целокупној мрежи рачунара који комуницирају. Да би крајња апликација била у могућности да испуни све захтеве аутономног управљања, потребно је да сви ентитети који је опслужују и који се налазе у њеним темељима испуњавају своје захтеве. Посебно важи за сервер јер је централна и критична тачка у систему без ког ниједан корисник не може да обавља послове. Из тог разлога је приступљено развоју што квалитетнијег софтвера у смислу стабилности, поузданости и брзини одзива. У таквим ситуацијама је потребно размишљање унапред о сценарију ширења мреже и перспективи рада сервера у таквим условима, те је развијена још једна верзија са повећаном скалабилности. Посебан акценат је стављен на тестирање и валидацију сервера, како у симулационом окружењу тако и у реалном раду. Рачунарске симулације су изузетно практичан и исплатив полигон за тестирање програма, међутим, права валидација долази из реалног рада. Реалан рад је валидација за све ентитете који учествују у обслуживању крајње апликације која врши аутономно управљање, почевши од базних станица и сервера укључујући и саму апликацију коју корисник види и чије резултате очекује.

Сопствено решење било ког рачунарског ентитета или апликације има своје предности и мане. У инжењерском смислу, познајемо решење и можемо пројектовати његово понашање у разним ситуацијама. У случају појаве програмских грешака, имамо могућност да их исправимо, а такође и могућност да мењамо систем у складу са будућим потребама окружења у ком систем ради. С друге стране, пројектовање сопственог решења захтева знање, издвојено време и ресурсе. Тестирање и валидација су велика препрека коју решење мора да савлада пре него што оно може бити проглашено као успешно и способно да обавља своје задатке.

6. Литература

[1] "rubenghosh968", The Pros and Cons of C++ Language, Medium platform [online]. Available at: medium.com/@rubenghosh968/the-pros-and-cons-of-c-programming-language-b19be005a772 (Accessed on Dec 23, 2023).

[2] Harnam Arneja, Andrew Bender, Sam Jugus and Tim Reid, Solving the GPS Equations, George Mason University [online]. Available at: mason.gmu.edu/~treid5/Math447/GPSEquations/

[3] GNSS and satellite navigation explained, Advanced navigation [online]. Available at: www.advancednavigation.com/tech-articles/global-navigation-satellite-system-gnss-and-satellite-navigation-explained/ (Accessed on Mar 8, 2023).

[4] Atomic clock, Wikipedia [online]. Available at: en.wikipedia.org/wiki/Atomic_clock (Accessed on Aug 26, 2026).

[5] GPS Ground Segment, The European Space Agency [online]. Available at: gssc.esa.int/navipedia/index.php/GPS_Ground_Segment (Accessed on Apr 17, 2026).

[6] Katie Baynes, GNSS, NASA [online]. Available at: www.earthdata.nasa.gov/data/space-geodesy-techniques/gnss (Accessed on Sep 1, 2026).

[7] Ionosphere, Wikipedia [online]. Available at: en.wikipedia.org/wiki/Ionosphere (Accessed on Aug 11, 2026).

[8] Devon Sharp, What are the different GNSS correction methods?, Swift Navigation [online]. Available at: www.swiftnav.com/resource/blog/what-are-the-different-types-of-gnss-corrections

[9] What is RTCM?, Swift Navigation [online]. Available at: www.swiftnav.com/glossary/what-is-rtcm-radio-technical-commission-for-maritime-services

[10] Devon Sharp, What is NTRIP and how does it work?, Swift Navigation [online]. Available at: www.swiftnav.com/resource/blog/what-is-ntrip-and-how-does-it-work

[11] Meriah Abderrahim, Mastering epoll - The Engine Behind High-Performance Linux Networking, Medium platform [online]. Available at: medium.com/@m-ibrahim.research/mastering-epoll-the-engine-behind-high-performance-linux-networking-85a15e6bde90 (Accessed on Aug 27, 2025).

[12] Libevent, Wikipedia [online]. Available at: en.wikipedia.org/wiki/Libevent (Accessed on July 3, 2026).

[13] Software Verification and Validation, Wikipedia [online]. Available at: en.wikipedia.org/wiki/Software_verification_and_validation (Accessed on June 19, 2026).

[14] Internet Architecture: A Layer-Based Analysis of Selected Internet Policy Issues, EveryCRSReport [online]. Available at: www.everycrsreport.com/reports/R48902.html (Accessed on Apr 10, 2026).

[15] Client-Server model, Wikipedia [online]. Available at: en.wikipedia.org/wiki/Client–server_model (Accessed on Aug 29, 2026).

[16] Miroslav Popovic, Communication Protocol Engineering, CRC Press, Taylor & Francis Group, Boca Raton, FL, USA, 2006, ISBN 0-8493-9814-2.

[17] Carrier-grade NAT, Wikipedia [online]. Available at: en.wikipedia.org/wiki/Carrier-grade_NAT (Accessed on June 21, 2026).

[18] Dynamic DNS, Wikipedia [online]. Available at: en.wikipedia.org/wiki/Dynamic_DNS (Accessed on Sep 19, 2025).

[19] RTCM SC-104, Wikipedia [online]. Available at: en.wikipedia.org/wiki/RTCM_SC-104 (Accessed on June 14, 2026).

[20] Radni okviri Kernela za rukovaoce uređaja, OSLuNR [online]. Available at: www.rt-rk.uns.ac.rs/sites/default/files/materijali/predavanja/P9.2.pdf