INFORMATION SYSTEM PROPOSAL FOR CLOUD BASED FILE SYSTEM

Александар Соколовски[1], Сашо Гелев[1]
[1] Европски Универзитет Република Македонија" – Скопје,
aleksandar.sokolovski@eurm.edu.mk saso.gelev@eurm.edu.mk

Абстракт-Област на истражување на овој труд е една доста важна ИКТ технологија, технологијата која ја овозможува работата на интернтет базираните системи на датотеки Cloud Based File Systems (CDFS) . Оваа технологија е клучна за работењето на било која ИКТ компанија во денешното информатичко општество. Овој труд ќе се обиде да ги истражи можните бенефити од користењето на CDFS и да предложи нов алгоритам за принципот за размена на датотеки помеѓу повеќе корисници (multiple-read, single-write, file sharing principal, MRSW). Главната истражувачка цел ќе биде да се проучат неколку од најкористените постоечките CDFS (Dropbox, SkyDrive, Google File System, Folio Cloud) и истите да се споредат со традиционалните системи за размена на датотеки (File Transfer Protocol based systems). Примарната цел е да се најде или предложи најдобриот можен алгоритам за MRSW. Ова ќе се потврди со тестирање на предложениот алгоритам и споредба на резулатите со веќе постоечки системи за размена на датотеки (bencmarking).

Клучни Зборови: cloud computing, File Transfer Protocol, on the cloud file systems, multiple-read single-write, file sharing principals

I. Вовед

Постои акутна потреба за безбедно складирање, организирање, споделување и анализирање на огромни количества комплексни (полу-структурирани или неструктурирани) податоци, со цел да се определат насоките и трендовите за подобрување на квалитетот на здравствената заштита, за подобро да се заштити државата или, пак, за подобро да се истражат можностите врзани за алтернативните извори на енергија. Заради итната природа на можните примени, важно е интернет-облакот да биде безбеден. Најголемиот безбедносен предизвик врзан за интернет-облакот е тоа што корисникот може да нема контрола врз локацијата на складирање на податоците. Ова се должи на фактот што ако некој сака да ги ужива придобивките од употребата на интернет-облак, тој мора да ги користи и дистрибуцијата и распоредот кои ќе ги одреди облакот. Затоа, треба да ги заштитиме податоците во рамките на несигурните процеси. Интернет-облак се нарекува и апликацијата што се нуди како услуга преку интернет, како и хардверот и системскиот софтвер во центрите за податоци кои ја нудат таквата услуга. Самите услуги веќе одамна се нарекуваат Софтвер како услуга (SaaS), па и ние го користиме тој термин. Кога интернет-облакот се нуди на јавноста со плаќање според употребата, ние го нарекуваме Јавен интернет-облак, а услугата што се продава ја викаме компјутерска јавна услуга. Постојните примери на јавни услужни интернет-облаци ги вклучуваат услугите AmazonWeb, AppEngine и Microsoft Azure. Го користиме терминот Приватен облак за означување на внатрешните податочни центри на фирмите или другите организации кои не се достапни за јавноста. Следствено, интернет-облакот е збир на SaaS и компјутерска јавна услуга, кој во принцип не ги вклучува и приватните интернет-облаци.

Вовед во TCP (Transmission Control Protocol)

TCP или Transmission Control Protocol претставува протокол кој се наоѓа во четвртиот слој на OSI моделот (Транспортниот слој) и има улога да обезбеди сигурен пренос на податоците во IP средината. Од повеќето сервиси кој овој протокол ги нуди можат да се издвојат: сигурност, ефикасна контрола на текот на податоци, оперирање во целосен дуплекс (full duplex – истовремено испраќање и примање на податоци) и мултиплексирање кое овозможува истовремено работење на повеќе процеси од високи слоеви преку една конекција. [1] Со TCP се извршува трансферот на податоците како неструктурирана низа од бајти кои се идентифицираат со секвенца. Овој протокол ги групира бајтовите во сегменти, им доделува број на секвенците, им доделува број на порта на апликациите и ги проследува низ IP протоколот. Овој протокол обезбедува безбедност така што користи алгоритми кои при размената на податоците прво ја воспоставуваат врската меѓу корисниците, а потоа обезбедува и низа на механизми каков што е праќањето на ACK броеви. Страната која ги прима податоците го праќа бројот на секвенцата на бајти кој ги примила. Доколку примачот не прати ACK дека ја примил пратената секвенца на бајти во одреден временски интервал тогаш секвенцата повторно ќе се испрати. Механизмите за безбедност на TCP протоколот овозможува уредите да се справат со загуби, доцнење, дуплирање или погрешно читање на пакети. Time-out механизмот овозможува уредот да детектира изгубени пакети и да побара нивно *повторно праќање. TCP претставува еден дел од* групата во која исто спаѓа и Интернет Протоколот (Internet Protocol), а се нарекува TCP/IP. Со користењето на TCP протоколот апликациите можат да комуницираат безбедно, независно од долните

слоеви. Ова значи дека рутерите (кои работат на интернет слојот) ќе мора да рутираат податоци во форма на датаграми, без да бидат задолжени со надгледување на податоците т.е. дали тие се исправни. [2]

Вовед во SCTP (Stream Control Transmission Protocol)

SCTP или Stream Control Transmission Protocol е протокол кој се наоѓа во Транспортниот слој и има улога слична на останатите протоколи кои исто така се наоѓаат во Транспортниот слој – такви се: TCP и UDP (User Datagram Protocol) протоколите. Овој протокол претставува мешавина од двата, ориентирано е на пораки како UDP протоколот и користи безбеден, сигурен секвенцијален транспорт на пораки/пакети со контрола на протокот како TCP протоколот. Создавањето на овој протокол било мотивирано од потребата за да се пренесуваат телекомуникациски сигнали (пораки) преку IP базирана мрежа. Системот за сигнализација SS7 бил доминантен носач на контролните информации кај телекомуникациските мрежи каде сигурноста и перформансот се критични за постоечките сервиси и апликации. SS7 сигналната мрежа е посебна мрежа која има потреба од посветена мрежна инфраструктура и само споделува некои физички ресурси со регуларен кориснички сообраќај. [2] IETF групата SIGTRAN предложила различен пристап за преносот на пораките/пакетите: Stream Control Transmission протоколот кој претставувал IP транспортен протокол на исто ниво како и UDP и TCP протоколите. На овој начин, сигналите се разменуваат преку заедничка IP базирана мрежа која користи пакети наместо користењето на посебна мрежа. [3]

II. *СПОРЕДБЕНА АНАЛИЗА НА TCP И STCP* ПРОТОКОЛИТЕ

Нешто повеќе за TCP протоколот

TCP обезбедува сервис за комуникација на средно ниво помеѓу апликацијата и Интернет Протоколот (IP). TCP протоколот го обезбедува овој сервис кога една апликација сака да прати голема количина на големи податоци низ интернетот користејќи го IP протоколот, наместо да ги раздели податоците на помали делови со IP големина и да изврши серија од IP пребарувања, софтверот може да изврши само едно побарување до TCP протоколот и да го остави да ги среди IP деталите. Интернет протоколот работи на тој начин што разменува делови од информацијата наречени пакети. Еден пакет претставува секвенца од октети (единица која се состои од 8 бита) и содржи header или глава и тело – body. Главата ја опишува дестинацијата на пакетот до која е пратен, а опционално и за рутерите да можат да ги проследуваат пакетите се додека не пристигнат до дестинацијата. Во телото се наоѓаат податоците кои IP протоколот ги праќа. Поради загушување на мрежата, балансирање на оптовареноста на сообраќајот или друга непредвидлива ситуација која може да настане кај мрежата, IP пакетите можат да се изгубат, да се дуплицираат или да не се пренесат исправно. TCP протоколот ги пронаоѓа овие проблеми и бара изгубените податоци повторно да се пратат доколку станува збор за изгубени пакети, ако пак податоците не се добро подредени TCP протоколот е задолжен да ги подреди, дури и помага да се минимизира загушувањето на мрежата за да го спречи настанувањето на останатите проблеми. Штом TCP примачот ги состави секвенците од октети кои биле испратени, ги предава на апликациското ниво. TCP протоколот значително е искористен од повеќето популарни интернет апликации, вклучувајќи ги World Wide Web (WWW), Емејл, File Transfer Protocol (FTP), Secure Shell, peer-to-peer споделувањето на податоци и некои апликации за стримување на медиа. Овој протокол е оптимизиран за прецизна достава, а не за брза достава на пакети па затоа можат да настанат релативно долги одложувања при чекањето на пораки за неподредени пакети или пак за одново праќање на изгубени пораки. Ова не е погодно за апликации кои праќаат пакети во реално време каква што е Voice over IP. За такви апликации препорачливо е да се користи Real-time Transport Protocol (RTP) кој работи над UDP протоколот. TCP протоколот е сигурен сервис за пренесување кој гарантира успешен пренос на податоци од еден хост до друг без да настане дуплицирање на тие податоци или да дојде до нивно губење. За да го гарантира ова TCP протоколот содржи низа од правила кои се користат со Интернет Протоколот. Кога Интернет Протоколот се грижи за праќањето на податоците, TCP се грижи за индивидуалните единици при преносот на податоците наречени сегменти. На пример, кога HTML документ е пратен од Веб сервер, TCP софтверскиот слој на тој сервер ја поделува секвенцата од октети на документот во сегменти и ги проследува до Интернет слојот еден по еден. Интернет слојот го енкапсулира секој TCP сегмент во IP пакет со додавање на глава (header) во која се вклучени дестинациската IP адреса. Иако секој пакет ја има истата дестинациска адреса, тие можат да бидат рутирани на различни патеки низ мрежата. Кога компјутерот кон кој се пратени тие пакети ќе ги прими, TCP слојот ќе ги состави сегментите и ќе се осигура дека тие се правилно подредени и без грешки.

Операции на протокол

TCP операциите на протоколот може да се поделат во три фази. Конекциите мора да се успешно остварени во повеќечекорен handshake процес пред да се влезе во фазата за пренос на податоците. Откако ќе се оствари преносот на податоците, се терминира (завршува) врската т.е. конекцијата и се ослободуваат сите алоцирани ресурси. При едно конектирање една TCP конекција поминува низ следниве промени: [4]

- LISTENING: Доколку станува збор за сервер тој чека за да се побара барање за конекција од клиент.
- SYN-SEND: Чека далечинскиот врсник да прати назад TCP сегмент со поставени SYN и ACK знаци.

- SYN-RECEIVED: Чека за далечинскиот врсник да прати назад потврда веднаш по праќањето на потвра до далечинскиот врсник
- ESTABLISHED: Портот е спремен за да прима или праќа од или до далечинскиот врсник.
- FIN-WAIT-1:
- FIN-WAIT-2: Индицира дека клиентот ги чека серверите за последниот (fin) сегмент.
- CLOSE-WAIT:
- CLOSING:
- LAST-ACK: Индицира дека серверот моментално испраќа негов последен (fin) сегмент.
- TIME-WAIT: Се претставува времето кое мора да помине за да се осигура успечното стигање на потврдата кај далечинскиот врсник за неговото барање за терминација на конекцијата.
- CLOSED: Конекцијата се затвора.

Нешто повеќе за SCTP протоколот

Апликациите кои го користат SCTP протоколот ги испраќаат своите податоци во вид на пораки (групи од бајти) до SCTP транспортното ниво. SCTP ги поставува пораките и информациите за контрола во посебни парчиња (*chunks*), секој со своја глава (*chunk header*). Една порака може да биде фрагментирана преку повеќе податочни парчиња, но секое податочно парче содржи податоци само од една порака од корисникот. SCTP парчињата се спакувани во SCTP пакети кои пак се носат до Интернет Протоколот и содржат глава на пакетот, SCTP контролни парчиња кога се потребни, проследено со SCTP податочни парчиња кога се достапни.
Овој протокол може да се карактеризира како ориентирано на пораки што значи дека пренесува секвенца од пораки, а не стрим од бајти како што пренесува TCP протоколот. Како и кај UDP протоколот и кај SCTP протоколот праќачот праќа порака во една насока и таа порака се пренесува до примачкиот SCTP кој пак ја пренесува до апликацискиот процес во една операција. Напротив, TCP протоколот е стрим ориентиран протокол кој пренесува стримови на бајти сигурно и под ред. TCP протоколот не дозволува примачот да знае колку пати апликацијата која го користи овој протокол на праќачот го повикала TCP транспортот предавајќи групи од бајти кои се за праќање. TCP од страна на праќачот ефективно и едноставно додава повеќе бајти во редицата на бајти кои што чекаат да бидат иратени. [6]

Терминот мулти-стриминг се однесува на способноста на SCTP да пренесува повеќе независни стримови од парчиња паралелно еден на друг, на пример, пренесување на слики од Веб страна заедно со текстот на таа Веб страна. [7]

Особини на SCTP

Примарни карактеристични особини кај SCTP протоколот се multi-homing и multi-streaming.

Безбедност

SCTP протоколот бил дизајниран за да биде безбеден. Со 4-страниот handshake, овој протокол спречува SYN flooding напади и користи “cookies” за верификација и автентикација. Сигурноста исто така бил клучен аспект при дизајнирањето на протоколот. Multi-homing-от овозможува конекцијата помеѓу две крајни точки да остане отворена дури и ако некој патеки и интерфејси се паднати. Во оргиналниот дизајн на SCTP не била вклучена енкрипцијата. SCTP понекогаш е добар кандидат за fingerprinting (пасивно собирање на податоци за конфигурацијата од далечинска направа). Дури и некои оперативни системи доаѓаат со овозможена подршка на SCTP протоколот и бидејќи овој протокол не е познат како што е на пример, TCP и UDP понекогаш се занемарува кај firewall-от и системот за детекција на напади. [5]

Споредба помеѓу TCP и SCTP

За мрежните апликации изборот на транспортен протокол е доста важен. Денеска повеќето апликации го користат TCP протоколот или UDP протоколот. На апликациите на кои им е потребен сигурен, подреден пренос на бајти го користат TCP протоколот, додека пак кај апликациите кои можат да толерират одреден степен на загуби преферираат UDP, главно бидејќи UDP овозможува побрз пренос на пакети. Повеќето апликации преферираат да го користат TCP протоколот наместо UPD. Ова е поради тоа што апликациите кои што користат TCP се популарни – FTP, емејл и World Wide Web (WWW). Кај TCP протоколот поимите стрим на бајт и конекција се еквивалентни. SCTP протоколот е нов транспортен протокол кој овозможува сигурен пренос на податоци во вид на пораки. Овој протокол е многу сличен на останатите транспортни протоколи какви што се TCP и UDP и е дизајниран за да ги крие апстракностите на мрежните слоеви од апликациите кои го користат. Во некои случаи, TCP или не ја овозможува саканата функционалност која е потребна од апликацијата или пак добавува повеќе функционалност отколку што е потребно. Додека SCTP транспортниот слој е ориентиран на пораки TCP протоколот е ориентиран кон стримувањето на бајти. Како што веќе напоменав SCTP протоколот е сличен со TCP протоколот. Една негова сличност е контрола на протокот и контрола на загушувањето на сообраќајот т.е. стратегиите применети кај SCTP. Други сличности со TCP протоколот се: користењето на checksum и број на секвенцата, имплементира TCP механизми за:

- Сигурен пренос
- Подредена достава на податоци
- Алгоритми за контрола на тек и спречување на загушување
- Бавен старт
- Брзо закрепнување во случај на пад
- Брзо препраќање на податоци

Исто така постојат и разлики меѓу TCP и SCTP протоколите. Како најважни можат да се издвојат:

- SCTP протоколот користи 32 битен checksum додека пак TCP користи 16 битен
- SCTP може да има повеќе стрима во конекција помеѓу две крајни точки
- SCTP протоколот го дефинира стримот како секвенца од пораки (парчиња), а не бајти како кај TCP
- Кај SCTP во еден пакет се вклучени главата плус еден или повеќе парчиња кои можат да бидат контролни или податоци
- SCTP протоколот може да користи multi-homing за редундантност
- SCTP користи "cookie" за да спречи SYN flooding напади

III. Виртуелен фајл систем

Frangipani е скалабилен дистрибуиран систем гледајќи ги податоците кои се колекција на дискови на повеќе машини, како едно податично складиште, со цел да се има централизирано администрирање.

Frangipani е изграден над Petal едноставен дистрибуиран менаџер на складиште со потадоци (storage). Постоеле обиди за вакви дистрибуирани системи како Frangipani, но неговата уникатност се должи на едноставната внатрешна структура, која овозможува лесен system recovery, лесно реконфигурирање, балансирано вчитување (load balancing), како и едноставното користење и администрирање на целиот систем. Во тој труд е објаснет нов тип (во периодот кога е пишуван трудот) и структура на (фајл систем) систем за управување со датотеки, кој има голема предност за корисниците и за администраторот на системот. Скалабилен дистрибуиран систем наречен Frangipani, менаџира колекција на дискови на повеќе машини, како едно податично складиште. Системот е поделен на две нивоа првото е Frangipani, дистрибуираниот фајл систем, второто ниво или долниот дел е Petal, менаџер на складиште со потадоци (storage).Главните предности за систем администраторот се: можноста да додава и отстранува лесно сервери без менување на конфигурацијата на постоечките сервери, лесно може да се прави конзистенет backup додека системот работи, администраторот на системот може да додава многу лесно нови корисници без загриженост за тоа кој корисник ќе биде авторизиран за кој сервер, бидејќи Frangipani ги гледа сите дискови на посебните сервери како една меморија, а Petal се грижи за тоа на кој вирутелен диск што ќе биде сместено. [8]

Backup-ите, предходните верзии од фајловите се чуваат online. Со цел да може некој од корисниците да пристапи до избришан документ, во случај документот ненамерно да се избришал или по одредено време избришаниот документ повторно бил потребен. Во дистрибуираниот фајл системот е размислувано и за безбедноста, корисниците мора да го користат само препорачаниот оперативен систем. Не може да користат модифицирани верзии на оперативниот систем и да бидат во истата доверлива мрежа на системот со сите привилегии како и другите компјутери во кои целиот систем има максимална доверба и на тие корисници им доделува најголеми привилегии. Овозможено е и далечинско поврзување на клиенти од било кој оперативен систем, бидејќи во овие клиентски оперативни системи Frangipani нема доверба, тие ќе комуницираат преку друга мрежа со Frangipani серверите. Со ова немаат директен пристап до Petal виртелните дискови, ниту пак максимална контрола. Ова е направено поради безбедносни причини со цел некој од далечинските клиенти да не ги злоупотреби своите кориснички привилегии, бришејќи системски датотеки или пристапувајќи во фолдери на други корисници. [9]

Во дистрибуираниот фајл системот, дисковата структура е различна од работењето на другите оперативни системи. Не им се доделува однапред фиксна меморија на сите корисници подеднакво бидејќи со тоа може да имаме еден корисник кој ќе искористи 5% од доделената меморија, а друг корисник би искористил 80% од меморија и тој ќе мора да отстранува работи со цел да не остане без доволно меморија. Меморијата се алоцира-доделува кога корисникот ќе почне да запишува, 2 на степен 64 бајти меморија, кога корисникот ќе ја исполни оваа меморија и повторно ќе запише фајл кој заедно со предходно запишианиот фајл ја надминува меморијата од 2 на степен 64 бајти, повторно ќе му биде доделана меморија од 2 на степен 64 бајти итн. Со ова се оптимизира менаџментот со меморија според потребите на корисниците, и од страна на администраторите значи штедење на вкупниот мемориски простор, на оваа логика е базирано креирањето на виртуелни дискови кај сите денешни модерни софтвери за виртуелизација на оперативни системи (VirtualBox, VirtualPC,Hyper-V, ESX). Логовите за метадата записите, тоа се всушност така наречени покажувачи кон податоците, кој податок каде е запишан, кој го креирал податокот, кој има право на пристап итн. Со ова се прави индексирање на податоци во случај ако некој сервер престане со работи, побрзо да може да се направи recovery процеруда на друг сервер. Логовите се окулу 128KB во големина, но тоа според политиката на запишување во Petal е два дела од по 64KB, кога ќе се наполни логот, најстариот дел од логот ¼ од него се алоцира за повторно запишување. Во recovery процедура само еден recovery демон (позадински процес кој работи без знаење на корисникот) може активно да работи. Со ова се гарантира конзистентност на податоци и се елиминира можноста да настане грешка, пребришување или на некој метаподаток, или загуба на некоја информација. За корисниците главен бенифит е погледот кон сите податоци во системот (секако со пермисии за пристап), главна предност е што наместо еден документ да го препраќаат преку најразличен тип на мемории и е-маил (електронска пошта), секогаш последната верзија на документот е достапна за двајца или повеќе корисници, но кога некој корисник го едитира менува документот тој е заклучен за другите (locked). Ова е така наречената логика на пристап до дистрибуирани документи, multiple read /

single write, повеќе можат да читаат но само еден да менува-запишува. На оваа логика денес се базирани голем број на online storage сервиси (dropbox, skydrive). Со процедурата на заклучување се гарантира дека на ист документ во исто време нема да има два различни записи. Секогаш кога некој сервер ќе почне да чита во одреден дел од дискот (read lock), се прави заклучување за читање, серверот прави копија во својата кеш меморија од прочитаниот податок и истиот го ослободува за понатамошно читање.

Ако некој сервер нешто запише во кеш меморијата (ова подразбира некој клиент да запише или измени некој документ), серверот својата кеш меморија (кога корисникот ќе заврши со запишувањето) или write lock-от кога ќе заврши, мора променетата верзија од својата кеш меморија да ја измени и на централниот диск или во read lock-от. Со тоа сите сервери што имаат read lock на истиот мемориски дел да можат да ја обноват својата кеш меморија (со ова на сите корисници кои го читаат истиот документ можат да ги видат промените кои ги направил последниот корисник). Во поглед на backup-ите можноста за креирање целосен backup и чување online веќе ја споменавме како и предностите од тоа (клиентот може да врати избришани документи). Функционалноста наречена snapshot која ја прави Petal, корисникот со неа може да направи целосен backup на своите податоци (целиот негов виртуелен диск). Во случај да падне серверот каде е хостиран овој виртуелен диск, оваа копија може да се покрене и корисникот да си продожи со работа на својот виртуелен диск, без губење на податоци. Во случај во периодот на правење на backup на еден виртуелен диск се активира локална бариера која спречува било кој сервер да активира write-lock или да проба да запишува врз податоциите кои што во моментот се backup-ираат. Постои global lock поддржан од сервисите за заклучување кој спречува било кој сервер да пристапи до било кој виртуелен диск кога се прави целосен backup на сите виртуелни системи од страна на администраторот. На крајот се прикажани резултати кои ги потврдуваат горните тврдења за оптимизирана работа на системот и исто така ги покажуваат неговите одлични перформанси. Во десеттата и последна глава Frangipani е спореден со други негови претходници и покажани се подобрувањата имплементирани во него. Најсличен на него е фајл системот xFS, но има две клучни разлики кои го прават Frangipani подобар од xFS. Првата е внатрешната организација на фајл системот Frangipani каде што секој кориник може да пристапи и да ги гледа само своите податоци (ова се постигнува со виртуелните дискови менаџирано од Petal), а кај xFS секој кориник може да ги види сите податоци. Втората предност е system recovery што е уште отворено и нерешено прашање кај xFS, а кај Frangipani е решено со можноста за креирање backup на виртуелниот диск на секој корисник.

Главна предност на Frangipani е (транспарентното) бришење и додавања на сервери, на системот не му менува ништо бројот на сервери. Со ова е олеснета recovery процедурата, овие работи се овозможени поради комбинираните, write-aheaed loggings и заклучувања на фајловите, со цел да се спречи запишување кога се backup-ираат документите. [10]

Идентификувани проблеми се: Користењето на реплицирани Petal виртуелни дискови (logging-от запис може два пати да се повтори, еднаш да биде запишан во Petal, еднаш во Frangipani логот).Не користи локација на дискот кога зачувува податоци, бидејќи дисковите се виртуелизирани со Petal, Frangipani не ја знае точната локација на секоја датотека, туку знае кој податок на кој вирутелен диск е запишан. Frangipani заклучува со lock процедурата цели фајлови и дирекоториуми, наместо само блок на податоци, поради тоа што податоците се сместени на виртуелен диск. Ова не е добро решение ако повеќе клиенти делат податоци од еден ист виртуелен диск. Кога еден клиент ќе прави backup, другите не можат да ги менеуваат податоците на целиот виртуелен диск. Иако може да биде случај клиентот да сака да backup-ира податок, а тој не е поделен со другите клиенти.

IV. Предлог Решение

На секои 5 или 10 пакети се менува содржината на 3 ниво на пакетите IPv6 [11] . Глобален Рутирачки Префикс (Слика бр.01) полињата во Изворна IP и дестинациска IP адреса се менува со податоци од Виртуелното ОС НИВО. Од програмерски аспект ќе направиме всушност предифинирање напреоптоварување на конструкторот од која се изведуваат објеките - IPv6 мрежните пакети како во делот код 2, во код 1 е дадека оргиналната структура на IPv6 класата. Во код 3 е даден пример имплементација за предлог класата во мрежнит симулатор ns-3[12]. Во Глобален Рутирачки Префикс полето за изворна IP адреса се додава ProcessID апликацијата која го праќа пакетот, UserID, OsID. Горните податоци мора да останат исти информации треба да останат исти, ако се променат не е ист праќач. Во Global Prefix полето за дестинациска IP адреса се запишува системското SystemRunTime време на оперативниот систем, истото од еден пакет до друг треба да се разликува во онолку минути и секунди колку што поминале на компјутерот примач, ако тоа не е така тогаш не е ист праќач.

Со ова се прави детекција на аномалија на 3 ниво, што значи многу побрза детекција на упад/напад отколку на 7 ниво. Како што кажавме погоре со скенирање на секое погорно ниво времето на процесирање и детекција се намалува.

Код 1. IPv6 Оригинална Класа

Слика 1. Полиња во IPv6

Код 2. IPv6 Пример во ns-3

```
#include "ns3/core-module.h"
#include "ns3/simulator-module.h"
#include "ns3/node-module.h"
#include "ns3/helper-module.h"
using namespace ns3;
NS_LOG_COMPONENT_DEFINE
("FirstScriptExample");
  int main (int argc, char *argv[])
{ LogComponentEnable("UdpEchoClientApplication",
LOG_LEVEL_INFO);
LogComponentEnable("UdpEchoServerApplication",
LOG_LEVEL_INFO);

  NodeContainer nodes;  nodes.Create (2);
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate",
StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay",
StringValue ("2ms"));
  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);
  InternetStackHelper stack; stack.Install (nodes);
  Ipv6AddressHelper address;
  address.SetBase ("AutoAssign(IPv6Sender();)");
  Ipv4AddressHelper address;
  address.SetBase ("AutoAssign(IPv6Recever();)");
  Ipv6InterfaceContainer interfaces = address.Assign
(devices);
  UdpEchoServerHelper echoServer (9);
  ApplicationContainer serverApps = echoServer.Install
(nodes.Get (1));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));
  UdpEchoClientHelper echoClient
(interfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue
(1));
  echoClient.SetAttribute ("Interval", TimeValue
(Seconds (1.)));
  echoClient.SetAttribute ("PacketSize", UintegerValue
(1024));
  ApplicationContainer clientApps = echoClient.Install
(nodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  Simulator::Run (); Simulator::Destroy (); return 0;
}
```

Код 3. Предложена Класа

```
Class IPv6(); { Int Global Routing Prefix; Int SubnetID;
Int InterfaceID; }

Class IPv6Sender();
{
        Int ProcessID;
        Int UserID;
        Int OsID;
        Int SubnetID;
        Int InterfaceID;
}


Class IPv6Recever ();
{
        Int SystemRunTime;
        Int SubnetID;
        Int InterfaceID;
}
```

V. Заклучок и Идна работа

Со нивиот додаток на STPC протоколот се овозможуваат две работи или се разрешуваат два проблеми: безбедната комуникација и колфиктнет запис.
Безбедната комуникација се овозможува со додаденото TCP/IP ниво ново виртуелно ниво со предефинирање на IPv6 адресата,ова предлог решение може да го забрза процесирањето на податоците и пред сè анализата на пакетите со цел на време да се отстрани некоја потенцијална закана.

Додека колфиктнет запис се решава слично како и логиката на отварање на датодеките кај PETAL дискот, но секоја страница има посебен механизам за заклучување, и ако двајца корисници гледаат ист документ, мод само за читање на вториот корисник нема да биде целиот документ туку само страната која е еден од корисниците ја едитира.

Со овие две решенија се гаранитра автентичност на корисникот, и се отстранува проблемот на конфликтен запис.

VI. Библиографија

[1] RFC: TCP
[2] RFC: STCP
[3] Chandramohan A. Thekkath, Timothy Mann, Edward K. Lee, *Frangipani: A Scalable Distributed File System*
[4] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, The Google File System
[5] William Stallings, *Operating Systems Internals and Design Principles* (5th Edition) , 2009
[6] Larry L. Peterson and Bruce S. Davie, *Computer Networks, a System Approach*, Edition 3
[7] From Brian X. Chen, From iCloud to Dropbox: 5 Cloud Services Compared, http://www.wired.com/gadgetlab/2011/06/cloud-services-compared/, [visited: july 2011]
[8] Christian Braun and Robert Winter, *Integration of IT Service Management intoEnterprise Architecture*, *SAC'07*, March 11-15, 2007, Seoul, Korea.
[9] Henk Jonkers, Marc Lankhorst, René van Buuren, Stijn Hoppenbrouwers, Marcello Bonsangue,Leendert van der Torre, *Concepts for Modelling Enterprise Architectures*, 2008

[10] Andrew S. Tanenbaum, *Computer Networks,* 4th Edition

[11] Cisco Press, *Cisco's Self-Defending Network Architecture Reference Model*, january 2010

[12] NS-3 network simulator, http://www.nsnam.org/ [visited: mart,2011]

Aleksandar Sokolovski was born in Skopje, R. Macedonia on August, 03rd 1984. Between 1999 and 2003 attended the High School "Rade Jovcevski – Korcagin" in Skopje. He was a student at the Faculty of electronics and information technology, Ss. Cyril and Methodius University, Computer Science Department in Skopje, from 2003, in 2007 received Bachelor in Science degree from the same University. Aleksandar Sokolovski in 2009 receives a Master Degree in Science in Technology, Innovation, and Entrepreneurship from the University of Sheffield, The Master's Dissertation topic was "The Usage of Collaborative ICT Tools in Higher Education", the field of Research was Education and ICT Technology. He is a MSc Graduate from the European University in Skopje, at the Faculty of Informatics with his Master's Dissertation in the fields of computer network and intrusion detection systems He is a member of the IEEE organization.

Information System Proposal for Cloud Based File System

Aleksandar Sokolovski, Saso Gelev,
European University of Republic of Macedonia-Skopje,
aleksandar.sokolovski@eurm.edu.mk saso.gelev@eurm.edu.mk

Abstract-The scope of this paper is one of the most important aspects nowadays in ICT, the cloud computing technologies, more specifically the distributed file solution systems which are a key feature that any ICT Enterprise need in order to be competitive this world of Information Age. This paper attempts to investigate the possible benefits of using the Internet Based File Systems, also known as Cloud Based File Systems (CDFS), and proposes a new CDFS and algorithm for file sharing between multiple users (multiple-read, single-write, file sharing principal, MRSW). The main research aim of this paper will be investigating few of the most used CDFS existing today (Dropbox, SkyDrive, Google File System, Folio Cloud) and comparing them with traditional File Transfer Protocol systems in use today. The primary objective is to find/propose the best possible sharing algorithm for MRSW. This will be achieved by benchmarking the proposed file sharing algorithm.

Keyword: cloud computing, File Transfer Protocol, on the cloud file systems, multiple-read single-write, file sharing principals