ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

Факултет по телекомуникации
Специалност: Телекомуникации

Курсова проект по Мрежови Архитектури и Сигурност

Тема:

Разработване на решение за балансиране на натоварването в SDN мрежа

Студент: инж. Николай Проданов Фак. №: 111321045 Група: 232

Преподавател: доц. д-р инж. Камелия Николова

Дата: ..... Подпис: .....

София, 2021

Съдържание

Част 1: Увод

С експоненциалното увеличение на възможностите на изчислителните машини, много големи компании в сферата на информационните технологии и бизнеса започват да интегрират в своите сървърни решения различни методи на виртуализация, при което ресурсите на една физическа машина се използват за създаването на логически машини, които действат като самостоятелни единици. Покрай този тренд, рязко нашумява и идеята за софтуерно дефинираните мрежи (Software Defined Network - SDN). SDN технологията представя решение на проблемите в традиционните компютърни мрежи.

При традиционните мрежи, системата (визирайки протоколи като STP (Spanning Tree Protocol), OSFP (Open Shortest First Protocol)) няма централна видимост над цялата мрежова топология, което налага постоянната обмяна на съобщения. Същевременно мрежата трябва да има възможност да се разширявана и да не влошава качеството на обслужване. Тези изисквания са довели до сложни по архитектура и скъпи мрежови устройства, имащи за цел да интегрират възможно най-голяма част от функциите на контролната равина в хардуера на устройствата, като при това увеличат бързодействието на системата. Този подход създава различни технически ограничения в развитието на традиционните мрежи и изгражда монопол на пазара, при които само компании, с големи инвестиции в производството на мрежови устройства, могат практически да разработват нови поколения мрежови устройства [1].

Част 2: Теория

1. Архитектура на софтуерно дефинираната мрежа

За разлика от IP (Internet Protocol) базираните мрежи, които работят децентрализирано, SDN (Software Defined Network) е централизирана равнината на управление на различните мрежови устройства. Освен това SDN позволява да се използват същите протоколи както в традиционно базираните IP мрежи (IP, ARP (Address Resolution Protocol), VLAN (Virtual Local Area Network), Ethernet, TCP (Transmission Control Protocol), UDP (User Datagram Protocol)).

Архитектурата на SDN мрежата може да се раздели на три равнини. Най-горната равина представлява приложната равнина (application plane). Тя дефинира правилата в мрежата и може да предоставя услуги като защитни стени, динамично маршрутизиране, проксита, контрол на достъпа, мониторинг на мрежата и балансиране на натоварването. Под приложната равнина е разположена контролната равнина (control plane). Основното устройство в нея е контролерът. Той може да бъде най-обикновен персонален компютър или сървър. Софтуерът, който върви на него, отговаря за създаването и поддръжката на маршрутните таблици и изпълняването на правилата, подадени от приложната равнина. Чрез това той абстрахира приложната равнина от спецификите на мрежата. Най-долната равнина се нарича равнина за данни (data plane). Там се разполагат устройствата, които приемат и комутират пакетите през мрежата. Те отговарят за сегментирането, предаването и ре асемблирането на пакета. Те се конфигурират и управляват от контролната равнина. Приложният интерфейс, който контролната равнина предоставя на приложенията в приложната равнина се нарича northbound API (Application Programming Interface) или северен интерфейс, а интерфейсът между контролната

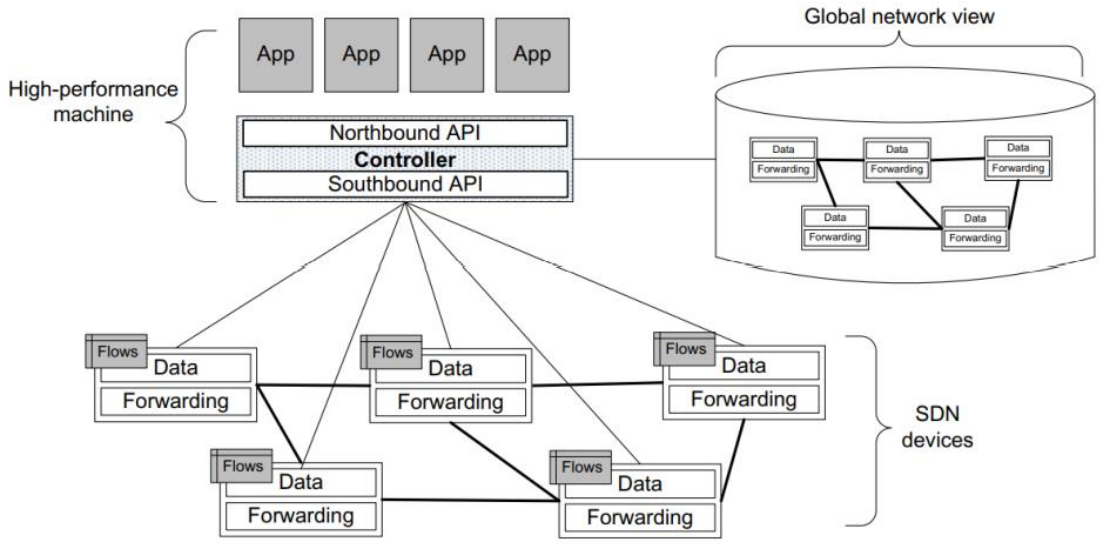
равнина и комутаторите в равнината за данни,southbound API или южен интерфейс [2].

Има много имплементации на SDN контролер, разработени на различни програмни езици като NOX (C\C++), POX (Python), Floodlight (Java) и други.

2. OpenFlow

OpenFlow е общо приет стандарт, предложен от ONF (Open Networking Foundation) за имплементация на SDN мрежа и разработка на SDN устройства и контролери. OpenFlow протоколът представлява реализация на южния интерфейс и определя комуникацията между SDN устройствата и контролерът.

Според OpenFlow стандарта, задачата на SDN устройствата в равнината за данни е да вземат решение какво действие трябва да се предприеме за всеки входящ пакет. Поради това те съдържат таблици с маршрутизираща информация под формата на потоци. Един поток представлява съвкупност от пакети, които трябва да се прехвърлят от една или повече входни точни до една или повече изходни точки.



Фиг. 1

Точките могат да бъдат уникално дефинирани в мрежата чрез IP адрес и TCP/UDP порт, VLAN точки или всякакъв друг мрежови протокол.

Структурно потокът може да бъде представен като условие, чрез което всеки пакет се проверява дали принадлежи и към този поток.

Всяко SDN устройство в равнината за данни съдържа една или повече таблици с потоци. Таблицата с потоци може да бъде представена като таблица с две колони. В първата колона е поставен потокът (или условието за този поток). Втората колона представлява списък от действия, които SDN устройството трябва да изпълни за входящия пакет, ако той принадлежи към този поток. Тези таблици са предварително създадени и качени на SDN устройствата от управляващия ги SDN контролер. Действията най-често са указание към кой изход трябва да се изпрати пакета, но може и да е указание пакета да се отхвърли без да бъде предаван. Могат и да се описват и доста по-сложни действия като смяна на полета в етикета. Ако даден пакет не принадлежи на никой поток SDN устройството може да го пренасочи към SDN контролера за обработка или да го отхвърли. Това поведение зависи от настройките на SDN устройството, качени от контролера. SDN контролерът абстрахира използващите го приложения от структурата и спецификите на SDN устройствата отдолу и превръща мрежата в единна единица. SDN контролерът трябва да има пълен поглед над мрежата и да знае за състоянието на всяко устройство в нея, за да може да прави сложни изчисления като определяне на най-краткият път през мрежата (особено, ако взима в предвид и натоварянето във възлите). Това създава известни трудности да се използват повече от един SDN контролер (понеже кешираната информация за устройствата трябва да се споделя от устройствата по някакъв начин) и налага сегментиране на мрежата.

Приложенията или по-специфично SDN приложенията (приложения, които си комуникират с SDN контролера) използват северния интерфейс на контролера. Тези приложения не бива да се бъркат с приложенията, вървящи в приложния слой на OSI (Open Systems Interconnection) модела. Понеже SDN архитектурата обхваща предимно каналния и мрежовия слой

на OSI модела (макар че един SDN контролер може да бъде програмиран да взима в предвид TCP/UDP портове). Приложенията могат да изискват от SDN контролера да изпраща входящите пакети от устройствата към тях и след обработка, те могат да инструктират контролера да приложи съответни правила за пакети, принадлежащи към същия поток. Тази форма на управление на SDN мрежата се нарича проактивна или динамична, защото действията се решават и приемат в момента, в който пакет от потока влезе в мрежата. Друга форма на управление е, когато SDN приложението, когато се стартира, инструктира контролера да инсталира предварително определени правила (поточни таблици) върху устройствата. Това се нарича пре-дефинирано или статично управление. Обикновено SDN приложенията използват комбинация от статично и динамично управление, инсталирайки някакви начални правила и след това променяйки ги или добавяйки нови, спрямо нуждите на мрежата. По този начин SDN приложението може да имплементира маршрутизация с най-различни протоколи (IP, MPLS (Multiprotocol Label Switching), OSPF (Open Shortest Path First)) както и да прилага ACLs (Access Control Lists) правила върху мрежата. Един от протоколите, които дефинират комуникацията между контролера и SDN устройствата е OpenFlow. Въпреки че той е широко използван стандарт в SDN общността, има разработени алтернативни решения част, от които са фирмени [3].

3. Поточни таблици (flow tables)

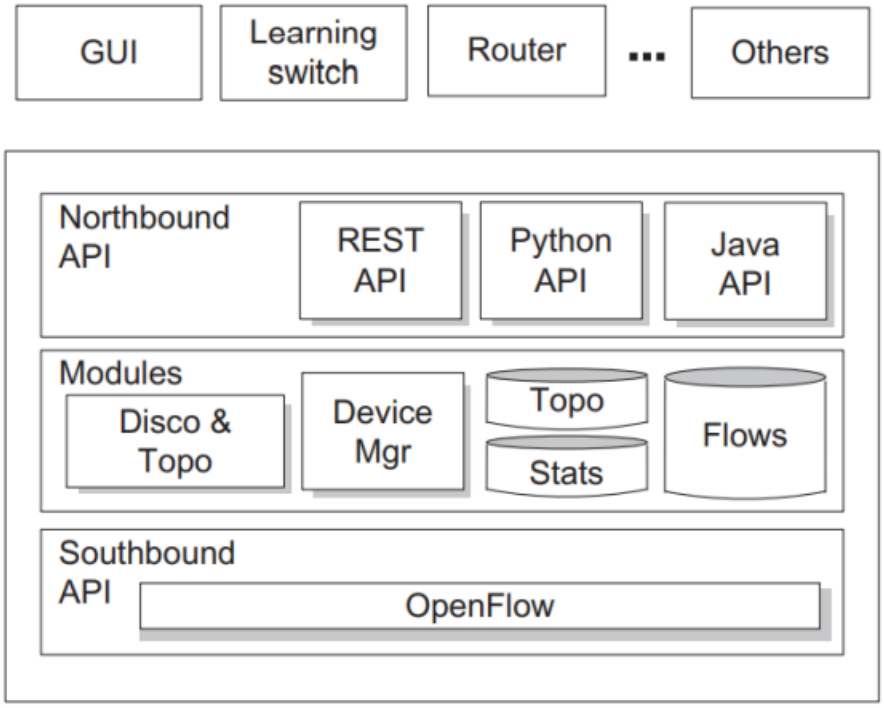
Поточните таблици позволяват на комутатора да определи входящите пакети към кои потоци спадат и да приложи необходимите действия. Тези действия могат да бъдат препращане на пакета през съответния порт (или портове), отхвърляне на пакета, наводняване с пакета през всички портове на устройството, промяна на някое поле в етикета и др. Поточните таблици представляват таблици с две колони, условието за поток и действия.

Редовете в поточна таблица се нарича поточен ред. Когато пристигне пакет по даден порт, неговите данни (основно от етикета) се сравняват с всеки ред от поточната таблица и ако се установи, че пакетът принадлежи на потока в поточния ред, действията дефинирани в този ред се изпълняват. Този процес, при който един пакет се прехвърля през таблица може да се нарече обхождане на таблица. Действията сами по себе си се делят на два типа, терминиращи – такива, които прекратяват последващите сравнения и не терминиращи – такива, след които обхождането на таблицата продължава. При първия тип, след изпълнението на действието пакетът вече не се обработва. Това са действия, при които пакетът се изпраща през порт или се отхвърля. Някои терминиращи действия могат да прехвърлят пакета към следваща таблица, в която процесът обхождане на таблица се изпълнява отново за избраната таблица. При втория тип, след изпълнението на действието пакетът продължава да се сравнява с долните редове и да изпълнява действията им, т.е. процесът на сравнение не се прекратява. Не терминиращи процеси може да са процеси, които променят етикета или информацията в пакета или сигнализират контролера. Условията за поток могат да описват множество критерии за стойността на дадено поле. Например, когато условието е точен IP адрес на източник, критериите за другите полета могат да се дефинират като множество от всички възможни стойности. В друга ситуация, в която трябва да се гледат само първите два октета от IP адреса, критериите за останалите два октета може да се дефинират като множество от всички възможни стойности. Тази гъвкавост в условията позволява на SDN разработчици, инженери и програмисти голяма свобода и възможности [3] [4].

4. SDN контролер

SDN контролерите често идват със свои собствени приложения (модули), които имплементират функционалности на „учащ“ комутатор (learning

switch), маршрутизатор, проста защитна стена и прост разпределител на натоварването (load balancer).



Фиг. 2 Структурна схема на SDN контролер

Фиг. 2 изобразява компонентите, от които се състои един SDN контролер. Компонентът с модулите дефинира основната функционалност на контролера. Южният интерфейс дефинира връзката между контролера и SDN устройствата. Протоколът, използван за този интерфейс най-често е OpenFlow, но се допускат и алтернативи като BGP (Border Gateway Protocol) или друго SDN решение. Северният интерфейс дефинира връзката между контролера и SDN приложенията. Тук, за разлика от южния интерфейс, няма точно определен стандарт. Най-често се използва RESTful (REpresentational State Transfer) интерфейс, който позволява SDN приложенията да се имплантират на всеки език, който подържа TCP/HTTP (HyperText Transfer Protocol) приложен интерфейс и може да върви на отделен процес от контролера на същата машина или на отдалечена. Други имплементации на този интерфейс включват Java API, Python API и други, които позволяват по-качествена имплементация от REST. Но за разлика от

REST, за да се използват подобни интерфейси, SDN приложението трябва да живее в процеса на контролера (или обратното). Това намалява гъвкавостта при използване на интерфейса, но предоставя значително по-голямо бързодействие. Възможно е този интерфейс да се имплементира, използвайки и други протоколи, които често се използват в комуникация между уеб сървъри, като SOAP (Simple Object Access Protocol), GraphQL (Graph Query Language), gRPC (Remote Procedure Call) и други. Компонентът с основни модули абстрахира детайлите за връзката между устройствата и контролера, за да опрости работата на SDN приложенията. Основните функции, които този компонент предоставя са:

- Откриване на крайни устройства – сървъри, лаптопи, десктопи, принтери и др.
- Откриване на междинни устройства – мрежови възли, които представляват инфраструктурата на мрежата като комутатори, маршрутизатори и други.
- Наблюдение на топологията на мрежата – Поддържа актуална информация за връзките между възлите както и информация за връзките с крайните устройства
- Мениджмънт на потоците – Поддържа вътрешна база данни с информация (кешира) за поточните таблици, инсталирани на всяко устройство, упражнява контрол върху тях и отговоря за синхронизиране на поточните таблици на устройствата с вътрешната база. Това позволява на SDN приложенията да четат и правят промени по съществуващи таблици, без да се налага те да се изтеглят от SDN устройството.

Всички изброени функционалности се имплементират от модулите в SDN контролера. Те трябва да поддържат бази от данни, съдържащи актуалната топология и различни статистики.

5. SDN приложения

Тези приложения се изпълняват над контролера, взаимодействайки с мрежата чрез южния му интерфейс. SDN приложенията отговарят за това какви потоци се инсталират в поточните таблици и действията, които те трябва да изпълнят. Това става през контролера. То този начин те успяват да: инсталират съответните правила и да изградят най добър маршрут между две крайни точки в мрежата, разпределят трафика до множество междинни или крайни точки. Повечето SDN приложения са конструирани да изпълняват една или няколко близки задачи като по този начин играят ролята на разпределител на натоварването, защитна стена или други подобни функции на мрежово и транспортно ниво; да реагират на промени в топологията като добавяне на устройство или възлов дефект; да наблюдават и пренасочват пакети за инспекция и автентикация. Практически тези приложения и комбинацията от тях определят поведението на мрежата. В SDN света всяка логическа единица се имплементира чрез приложение. Това включва приложения, които управляват SDN устройството да се държи като комутатор или маршрутизатор. По-сложни приложения като имплементация на известни протоколи като STP и OSPF могат да се реализират от приложения, използващи по-прости приложения, за да надградят над базовата имплементация. Основната задача на SDN приложенията е да извършват една или група от близки мрежови функции като устройство за балансиране на натоварването (разпределител), реализиране на защитни стени и др. След като контролерът стартира успешно, той започва да изпраща събития, свързани с мрежата, към приложенията, които в по-голямата част от времето ги обработват и изпълняват необходимите действия. Други събития, които представляват интерес за приложенията са външните събития като системи за наблюдения на мрежата като NetFlow, IDS (Intrusion Detection System) или BGP (Border Gateway Protocol). За да получава съобщение, че

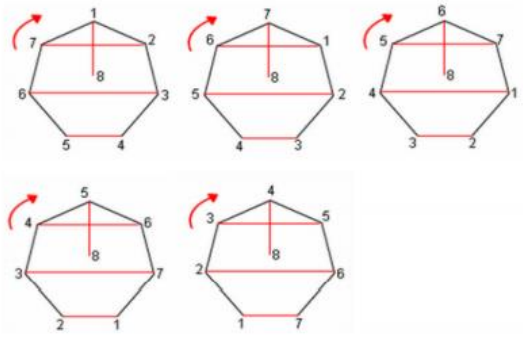
дадено събитие е настъпило, приложението първо трябва да се регистрира като слушател към това събитие като подаде метод, който да се извиква всеки пък, когато подобно събитие настъпи. С извикването на метода обикновено се подава и допълнителна информация, която това събитие представя. Примери за подобни събития са: пристигане на пакет, прикрепване на ново междинно устройство или добавяне на крайно устройство. Последните две биха носели информация като MAC (Media Access Control) адрес и други хардуерни характеристики на добавеното устройство. В случая на входящи пакети това са пакетите, които SDN устройствата изпращат към контролера, когато не знаят какво да направят с тях или имат инсталирано правило да го пренасочат към контролера. Има различни начини, по които приложението може да реагира на събитие. Един пример би бил да инсталира някакви общи правила върху ново устройство. При други събития приложението може да предприеме по-сложи действия като да се свърже с база данни или да направи заявка към сървър в друга мрежа. Всичко това зависи от събитието и информацията, която носи в себе си [3].

6. Стратегии за разпределяне на натоварването

6.1. Round Robin балансиране

При тази стратегия, както се посочва от името, сървърите се избират последователно в цикъл, както е показано на Фиг. 3. Ако имаме 7 сървъра в масив, то първият сървър ще обслужи първата заявка, вторият сървър втората, третият сървър третата и така се изрежат всички сървъри от масива. Това се води един цикъл. След приключване на един цикъл се започва следващия цикъл. Тази стратегия не взима предвид характеристиките на средата като брой връзки на сървър, време за отговор, използваемост на изчислителните възможности и редица други фактори. Това прави тази стратегия лесна за имплементиране и е заложена в почти всяко устройство

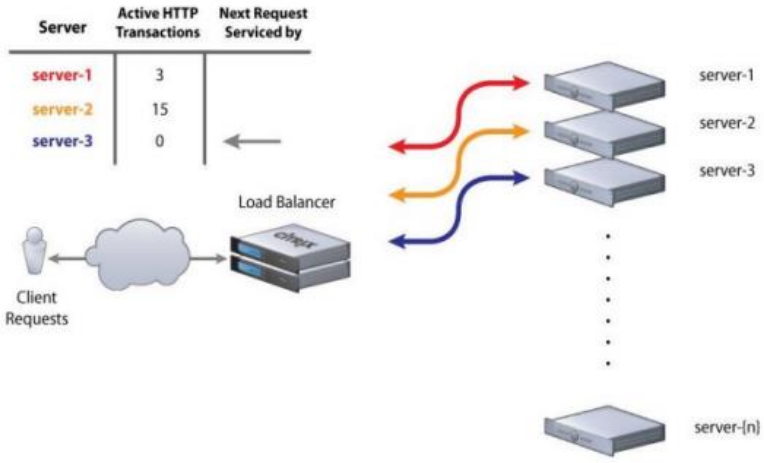
за балансиране на натоварването. В някои имплементации се добавя и допълнително условие, при което избраният сървър се проверява дали е работоспособен [5] [6].



Фиг. 3 Схема на Round Robin балансиране

6.2. Балансиране на база брой активни връзки

Тази стратегия се стреми да разпределя натоварването, базирано на броя активни връзки (least connection) на всеки един от сървърите. Когато пристигне нова заявка в балансиращото устройство, се избира сървърът с най – малък брой връзки. Това е динамична стратегия, тъй като тя взема предвид околната среда, когато взима решението към кой сървър трябва да се предаде входящата заявка. Фиг. 4 показва принципа на работа на тази стратегия. Следващите три заявки ще се пренасочат към server-3. Четвъртата ще отиде към server-1, петата към server-3, шестата към server-1 и т.н. [5] [6].

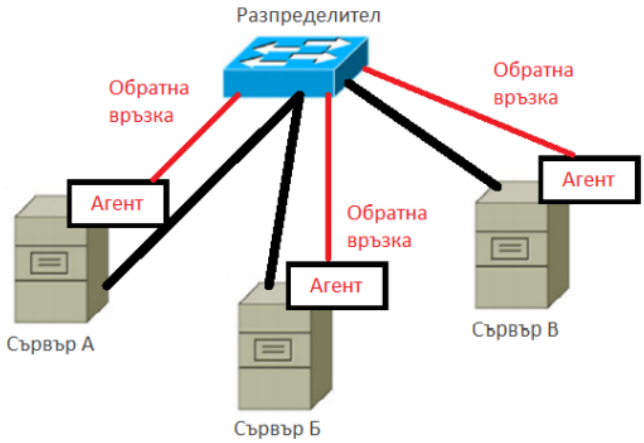


Фиг. 4 Схема на Least connection балансиране

6.3. Балансиране на натоварването чрез обратна връзка

Балансиране на натоварването с обратна връзка (dynamic feedback) е алгоритъм, който взима предвид величините в околната среда, като натовареност на сървърната машина, продължителност на обработка на заявката и закъснения по линията в реално време. Това се прави с цел да се разтоварят сървъри, които са заети със сложни операции и да се пренасочи входящия трафик към по-разтоварените сървъри. За да се постигне подобно разпределение, първо трябва да се определи метод, по който да се измерват параметрите на околната среда. Параметри като брой активни връзки към дадена сървърна машина могат, сравнително лесно да се извлекът от кешираната информация в балансиращото устройство. Други параметри като натовареност на сървърната машина не могат директно да се определят от него. Затова се налага да се сформира обратна връзка между него и сървърната машина. Съществуват много начини за постигане на тази обрана връзка. В повечето случаи това се реализира в рамките на IP мрежата, като може да се използват протоколи като SSH (Secure SHell), чрез който балансиращото устройство да достъпи сървъра и да свали статистика за използваните му ресурси. Друг метод, пак реализиран чрез мрежови връзки, е върху сървърната машина да се инсталира програма (daemon), която да следи за състоянието на машината и чрез TCP връзка или UDP пакети периодично да докладва на балансиращото устройство за текущото състояние на машината. Това може да се види на Фиг. 5 Този метод още се нарича агентно базиран, а инсталираната програма „агент“. Предимството на агентно базираната обратна връзка, пред това балансиращото устройство периодично да сваля нужната информация е, че при агента, връзката може да е само еднопосочна и второ агентът може да обяви сървъра за неактивен при частичен отказ или когато сървърът е в режим на поддръжка. Недостатък на агента, е че може да има известно закъснение на обновената

информация и балансиращото устройство ще трябва да взима решение, базирано на натоварване, което може да се различава от действителното [7].



Фиг. 5 Мониторинг на натоварване чрез агенти

Алгоритъмът, който ще използва тази информация и ще направи решението към коя сървърна машина да пренасочи заявките, може да се реализира по два начина. Първият и най-простият метод е като се избере сървърът, който е най-малко натоварен. Той е лесен за имплементиране, но може да доведе до дисбаланс при много висок поток на трафика и закъснение от агентно базирана обратна връзка. Другият подход е като се използва Round Robin балансиране с теглови коефициенти, като тегловните коефициенти на различните сървъри са динамични и се определят от тяхното натоварване, брой активни връзки както и други статични и динамични величини.

Част 3: Предложено решение

1. Въведение

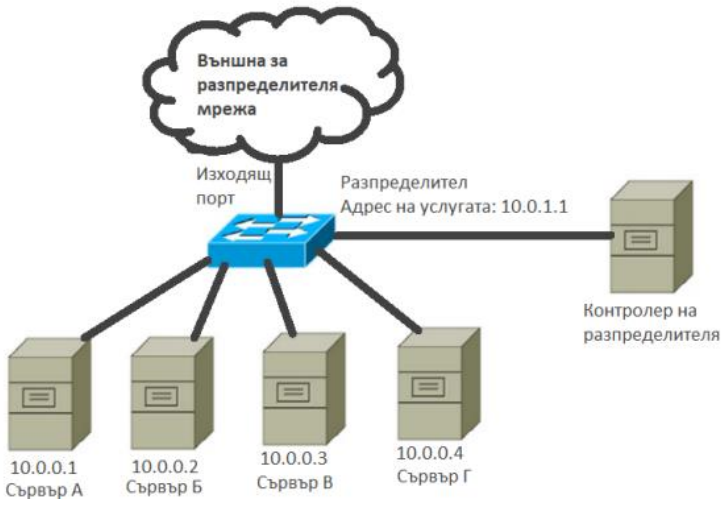
В този курсов проект е разработен прототип на устройство за балансиране на натоварването в SDN мрежа с цел да изложа преимуществата на SDN архитектурата в сферата на бизнеса и информационните технологии и как, чрез SDN, мрежата може да бъде програмирана да изпълнява всяка специфична фирмена логика. Основната цел на балансиращото устройство е да може по такъв начин да се разпредели входящия трафик, така че всички сървъри (машини) да са еднакво натоварени. Съществуват много методи за балансиране на натоварването. Могат да се характеризират в три вида. Статични като Round Robin. Статични с конфигурация като Round Robin с теглови коефициенти, при които за различните сървърни машини се задава някакъв приоритет, за да се открояят 22 изчислителните им възможности. И динамични като базирани на броя активни връзки (least connection) или с динамична обратна връзка (dynamic feedback) при които в уравнението за избиране към кой сървър да бъде насочен трафика се взима в предвид до каква степен са натоварени сървърните машини. Разбира се и при динамичните може да се добави някакъв приоритет. Текущата имплементация на разпределител на натоварването е изградена върху популярния POX контролер и тествана върху Mininet.

2. Приложение на решението

Предложеното в курсовият проект решение е предвидено да се използва като разпределител на натоварване. Като разпределителя може да се състои от минимум две устройства. SDN контролер и поне едно SDN устройство, което да играе ролята на разпределител в равнината за данни. Всички сървърни машини са свързани директно или индиректно, през други устройства към разпределителя. Разпределителят от своя страна може да има един или повече портове, свързани към различни мрежи или към други

сегменти на същата мрежа. Основните параметри, които приема разпределителят са:

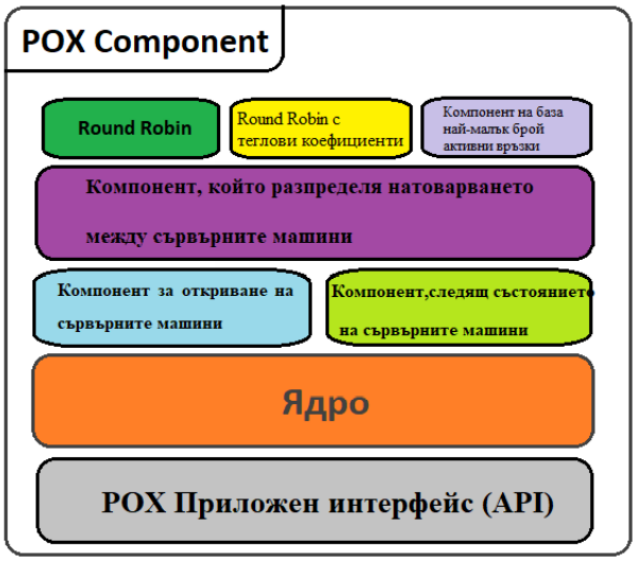
- Адрес на услугата – това е адресът, който клиентите ще ползват да използват услугата. От тяхна страна няма да има разлика кой сървър обслужва дадената заявка. Разпределителят също имплементира функционалност да отговаря на ARP запитвания за IP адреса на услугата, при което той изпраща собствения си MAC адрес.
- Изходящи портове – това са портовете, които водят към други мрежи или сегменти на същата мрежа. Дефинирането на тези портове спомага за по-оптималната работа на разпределителя.



Фиг. 6 Примерна топология за реализация на решението

3. Архитектура на решението

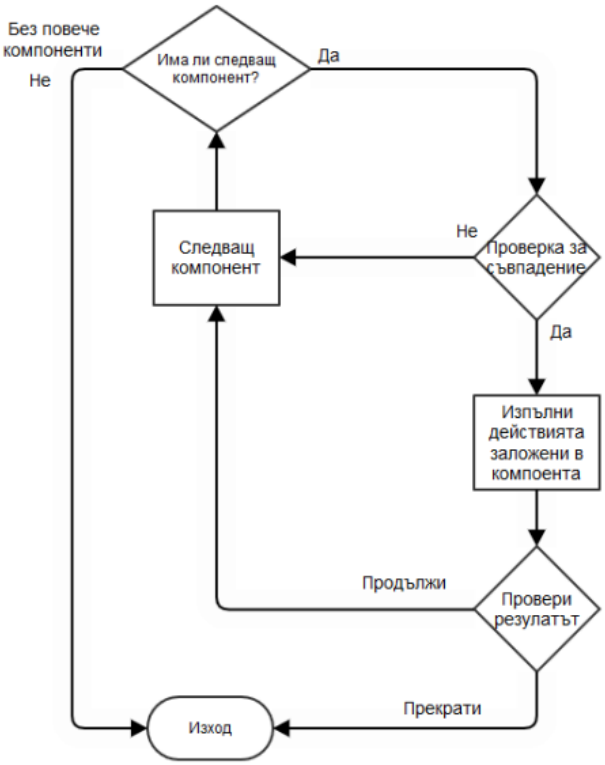
По аналогия на POX контролера, както и на обектно-ориентираните принципи текущото решение също може да се раздели на отделни компоненти Фиг. 7.



Фиг. 7 Структурна схема на архитектурния стек

3.1. Ядро

Ядрото в своята същност представлява абстракция на събитието за входящ пакет на POX приложния интерфейс и дефинира общ модел, с които по-горните компоненти предприемат действия срещу пакетите идващи от устройствата. Обработката на едно събитие (входящ пакет) може да се представи чрез блок схемата на Фиг. 8.



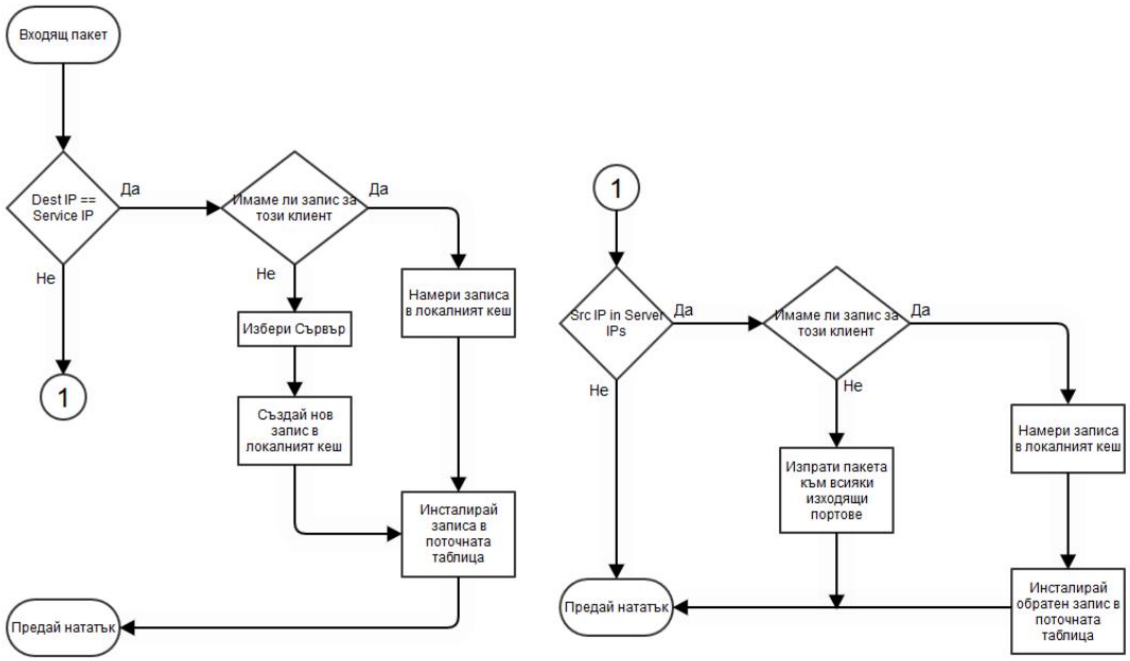
Фиг. 8 Блокова схема на ядрото

За дадено входящо събитие се започва цикъл, който преминава през всеки регистриран компонент. Всеки от компонентите имплементира метод за проверка, с който той може да потвърди, че той се интересува от това събитие. След изпълнение на метода за потвърждаване се изпълнява събитието, изложено в компонента. Това събитие може да върне резултат, който се използва само и единствено за прекратяване на цикъла. С други думи, ако методът за действие не върне никакъв резултат (в Python методите, които не връщат резултат практически връщат None) се взима предвид, че компонентът не иска да се прекрати обработката на събитието и цикълът продължава. Ако компонентът върне резултат, се подразбира, че цикълът трябва да се прекрати и обработката на събитието да спре.

3.2. Компонент, който разпределя натоварването между сървърните машини

Разработен е и компонент, който разпределя натоварването между сървърните машини. Целта на този компонент е да следи входящия трафик

в балансиращото устройство и да пренасочва входящите заявки към съответната сървърна машина. Този компонент не имплементира логика за избиране на сървър при пренасочване, а оставя тази функционалност на други обекти. Като входен параметър на този компонент се подава IP адреса на услугата, която се балансира.



Фиг. 9 Блокова схема на разпределителя на натоварването

Този компонент работи като следи входящите пакети и инсталира поточни записи в балансиращото устройство. Освен инсталацията на записите в устройството, този компонент ги записва и локално, в паметта на приложението. Записите и на двете места се следят от таймер и, ако пресрочат зададеното време, те се изтриват. Като времето за пребиваване на записа в устройството е 10 пъти по-малко от времето му за пребиваване в паметта на приложението. Това се прави, за да може приложението по-бързо да реагира, ако се случи някаква промяна в сървъра като да спре сървърът, който поема трафика на клиента. В такива случаи приложението може да избере нов сървър, да обнови записа за този клиент и да му прехвърли пакетите към новия сървър без загуби. Когато постъпи входящ

пакет, първо се проверява дали този пакет принадлежи на IP адреса на услугата. Ако това е изпълнено, то се проверява дали вече има направен запис за този клиент в паметта на контролера. Ако има, това означава, че записът в устройството е изтекъл и се създава заявка за инсталирането на записа отново. Ако не е открит запис за този клиент, се извиква обекта за избиране на сървър и му се поддава текущата информация за средата. Обектът прави изчисления и връща сървърът, който трябва да поеме трафика. От този сървър се създава запис в паметта на приложението и същият запис се изпраща за инсталация в устройството.

При обратния случай, когато IP адресът на източника е на някой от сървърите, се търси дали съществува запис с този сървър и клиента, за който е предназначен пакета. Ако такъв запис се намери, се инсталира така наречения обратен запис в устройството, който показва по какъв начин да се маршрутизират пакетите от съответния сървър до клиента. В противен случай, ако такъв запис не е намерен, се изпраща заявка на устройството да изпрати този пакет през всички изходни точки с преценка, че устройствата, директно свързани към разпределителя, ще знаят как да го доставят или чрез MAC адреса или чрез IP адрес на клиентското устройство.

3.3. Компонент следящ натоварването по сървърите

Този компонент има за цел да следи до каква степен са натоварени сървърните машини. Това става чрез специален агент, инсталиран на всяка една сървърна машина седяща зад разпределителя. Този агент създава обратна връзка с разпределителя и на всеки N на брой секунди докладва какво е състоянието на сървъра. Тъй като реализацията се изпълнява в симулирана среда този агент е интегриран с тестовият HTTP сървър, стартиран на всяка една от машините. Натоварването на сървъра се определя само от един коефициент, който е заложен като константа към всяка текущо изпълняваща се заявка или процес. Това значи, че ако клиентите на услугата изпратят 3 заявки, които натоварват сървъра с

коефициент 5 и те попаднат върху една сървърна машина, то агентът на този сървър ще изпрати, че машината е натоварена с коефициент 15. Тези коефициенти са абстрактна величина, представяща натоварването на сървърна машина в стимулационна среда като Mininet. В реални условия изпращаната информация може да е натовареност на процесора, каква част от оперативната памет е заета, до колко е натоварен хард диска и други подобни параметри. Разбира се, това налага и, че алгоритъмът за анализ на тези параметри ще е по-сложен, било то от страна на агента или на разпределителя. Агентът може да се настрои на колко секунди да изпраща съобщения за натоварването към разпределителя. Съобщенията се изпращат чрез UDP протокола към IP адреса на услугата и настроен порт, на който слуша компонента за следене на натоварването. На разпределителя има регистрирано правило, което да прихваща ARP request пакети с целеви IP адрес на услугата, след което те се подават към контролера и той генерира ARP reply с MAC адреса на разпределителя и през него го изпраща на питащата машина. Чрез това сървърната машина знае, че трябва да изпрати пакетите за IP адреса на услугата към разпределителя. На разпределителя също се инсталира правило да прихваща и изпраща към контролера UDP пакетите с IP адреса на услугата и порт, който е конфигуриран за компонента за следене на натоварването. По този начин разпределителят научава за натоварването на сървърните машини. Компонентът кешира в оперативната памет текущите коефициенти на натовареност на всяка една сървърна машина, заедно с IP адреса, от който го е изпратила. Така разпределителят може да определи статистически данни като средна натовареност на сървърните машини, с колко се очаква да падне или дали сериозно се е покачила. Тези характеристики се използват после в алгоритми и за определяне на полезната работа на разпределителя в симулациите.

Част 4: Верификация на разработеното решение

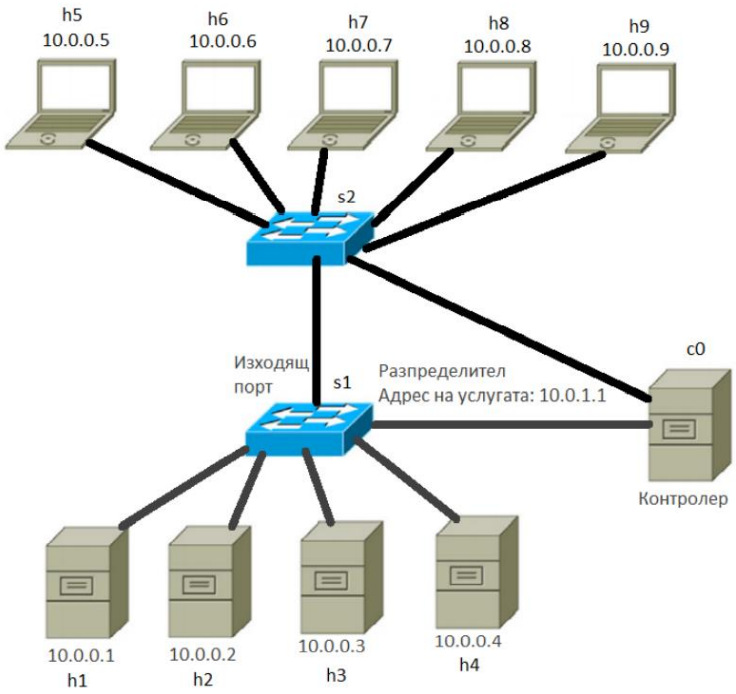
1. Mininet

Концепцията за SDN мрежите е сравнително нова. Когато се наложи да се експериментира с нови идеи в SDN мрежа, е трудно да се намерят евтини устройства или контролери, които да имплементират SDN стандарта. Освен това, когато се налага да се симулират големи мрежи с много на брой устройства участващи в тях, физическата реализация става трудоемка, понеже всяко устройство трябва да се конфигурира индивидуално. Едно решение на този проблем е симулирането на подобна мрежа. Mininet е софтуер, който позволява за бързото прототипиране на големи мрежи. Чрез него могат да се виртуализират множество SDN устройства и крайни устройства, използвайки методи за виртуализация, които не изискват значителни изчислителни ресурси. Това позволява на учени и разработчици да симулират големи мрежи със сложни топологии директно върху персоналния си компютър.

2. Сравнение между Round Robin и Least Connection алгоритмите

2.1. Опитна постановка

За изследване на предложеното решение ще се използва платформата за мрежови симулации Mininet. Схемата за изследването е показана на Фиг. 10.



Фиг. 10 Топология на опитната постановка

Представени са:

- четири сървърни машини: h1, h2, h3, h4
- пет клиентски машини: h5, h6, h7, h8, h9
- SDN устройство s1, което играе ролята на разпределителя на трафика
- SDN устройство s2, което играе ролята на комутатор на каналния слой в традиционната Ethernet мрежа.
- един SDN контролер, върху който са инсталирани приложенията за разпределителя и за Ethernet комутатора. Разпределителят представя услуга с IP адрес 10.0.1.1.

Върху всяка една сървърна машина има инсталиран прост HTTP сървър, който ще играе ролята на доставчик на услуга и ще събира статистика за броя връзки към конкретната сървърна машина. Клиентите с помощта на програмата cURL ще изпращат последователно заявки към IP адреса на услугата. Времето за обработка на една заявка във всеки сървър може да се зададе по три варианта. При първия вариант не се задава време за обработка и сървърът връща отговор към клиента на момента. При

втория вариант се задава конкретен брой секунди, за които да се обработва една заявка на сървъра. При третия вариант може да се зададе, че всяка заявка постъпваща в сървъра може да отнеме произволно време в зададен интервал от секунди, за да се обработи.

2.2. Опит едно

Стратегия за разпределението: Round Robin

Време за обработка на заявка: постоянно, 30 сек на заявка.

Начин на работа: всеки клиент последователно ще изпрати по 200 заявки към услугата.

След изпълнение на симулацията са налице резултатите, показани на Таблица. 1.

Таблица. 1

	h1	h2	h3	h4
Брой заявки	250	250	250	250
Среден брой активни TCP връзки	78.75	78.75	78.75	78.75
Максимален брой активни TCP връзки	103	103	104	103

Както се виждат от графиките. При заявки с константно време на обработка и сървърни машини с еднакви изчислителни възможности, като пренебрегнем закъснения по линията, каквито има в реални условия, Round Robin методът за разпределяне постига по-равномерно разпределение на товара.

2.3. Опит две

Стратегия за разпределението: Round Robin

Време за обработка на заявка: произволно, 0-30 сек на заявка.

Начин на работа: всеки клиент последователно ще изпрати по 200 заявки към услугата.

След изпълнение на симулацията са налице резултатите, показани на Таблица. 2.

Таблица. 2

	h1	h2	h3	h4
Брой заявки	250	250	250	250
Среден брой активни TCP връзки	46.69	45.8	48.16	46.71
Максимален брой активни TCP връзки	66	66	69	66

За разлика от предните два опита, тук се виждат значително по-малки стойности на активните TCP връзки за единица време на Таблица. 2. Това се дължи на променливото време за обработка на заявките, при което някои заявки се обработват мигновено, докато други отнемат към 30 сек. Тук може да се забележи и слабостта на този метод. На Таблица. 2 ясно се вижда, че сървър h3 има повече връзки от другите, понеже при него са се натрупали заявки с по-голямо време за обработка.

2.4. Опит три

Стратегия за разпределението: Least connection

Време за обработка на заявка: произволно, 0-30 сек на заявка.

Начин на работа: всеки клиент последователно ще изпрати по 200 заявки към услугата.

След изпълнение на симулацията са налице резултатите, показани на Таблица. 3.

Таблица. 3

	h1	h2	h3	h4
Брой заявки	255	242	252	251
Среден брой активни TCP връзки	63.34	62.98	62.74	63.15
Максимален брой активни TCP връзки	75	75	74	75

Тук за разлика от опит две, всеки сървър има почти равен брой активни връзки. Друга разлика от предния опит е, че сървърите не са обработили еднакъв брой заявки. Това показва, че Least connection, въпреки че е по-сложен, се изпълнява приложената задача по качествено от Round Robin в

динамични условия, където TCP връзките могат да приключат по различно време.

3. Сравнение между Round Robin и Dynamic Feedback алгоритмите

Опитната постановка е същата като предишното сравнение, но с известни промени. Към HTTP сървъра на сървърните машини се добавя UDP агент следящ натоварването на сървъра. Реализират се REST заявки, подаващи за изпълнение асинхронни задачи, които фиктивно да натоварват сървъра с даден коефициент на натоварване за определено време.

3.1. Опит едно

Стратегия за разпределението: Round Robin

Начин на работа: всеки клиент последователно ще изпрати по 10 заявки към услугата. Един от всички клиенти еднократно ще изпрати заявка с коефициент на натоварване 500, с продължителност 120 сек. След изпълнение на симулацията са налице резултатите, показани на Таблица. 4.

Таблица. 4

	h1	h2	h3	h4
Брой заявки	25	25	25	25
Коефициент на натовареност	113	392	112	111

3.2. Опит две

Стратегия за разпределението: Dynamic Feedback Начин на работа: като предишния опит (3.1) Получените резултати след изпълнение на симулацията са представени на Таблица. 5.

Таблица. 5

	h1	h2	h3	h4
Брой заявки	25	25	25	25
Коефициент на натовареност	113	392	112	111

От двата опита може да се заключи, че за разлика от Round Robin алгоритъма, Dynamic Feedback работи по-качествено, когато се очаква голямо натоварване на сървърните машини като изпълнение на сложни асинхронни задачи.

Част 5: Изводи

- Чрез SDN технологията, лесно може да се въведат специфични правила в мрежата.
- SDN улеснява контрола над мрежата като я централизира в едно (или няколко) устройство наречено контролер.
- SDN използва съществуващ хардуер и похвати от компютърното програмиране и разработка на приложения.
- Различните стратегии в балансирането на трафика към сървъри спомага за приложимостта му в по-специфични сървърни приложения.

Цитирани източници

[1] S. Mishra, "Software Defined Networking: Research Issues, Challenges and," 2017.

[2] K. Benzekki, "Software-defined networking (SDN): A survey," 2017.

[3] P. Goransson, "Software Defined Networks: A Comprehensive," *978-0-12-804555-8*, pp. (стр. 91, 94, 45), 2017.

[4] Y. Li, "SDN-Based Switch Implementation on Network Processors," 2013.

[5] M. Rahman, "Load-Balancer-as-a-Service-in-Cloud-Computing-v7," 2015.

[6] A. Ghosh, "A study on load balancing techniques in SDN," 2018.

[7] X. Yang, "Load balancing scheduling algorithm for storage system based on," August 2016.