УДК 621.391

РОЗРОБКА МЕТОДУ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ В SDN МЕРЕЖАХ НА ОСНОВІ МОДИФІКОВАНОГО ПРОТОКОЛУ STP

М. М. Климаш., М.І. Бешлей., Ю. Л. Дещинський., О.М. Панченко

Національний університет "Львівська політехніка",
вул. Степана Бандери 12, Львів, 79013, Україна

Проведено аналіз існуючої архітектури SDN, наведені особливості розвитку та взаємодії її компонентів. Мережі SDN є одним із перспективних класів мережевих архітектур, які є придатним для заміни традиційних комутованих мереж Ethernet. Програмно – конфігуровані мережі дають змогу управляти контролером поведінку комутаторів в мережі, що забезпечує високу продуктивність, керованість , надійність, масштабованість та безпеку для центрів обробки даних. Розглянуто робото STP протоколу для виявлення петель в мережі та визначено його основні недоліки при використанні в SDN мережах. Розроблено новий метод балансування трафіку в програмованих мережах на базі протоколу Openflow та модифікації STP.

Ключові слова: *Програмно-конфігурована мережа, контролер, трафік, протокол STP, протокол Openflow.*

Постановка проблеми. Бум розвитку ринку мобільних пристроїв і контенту, віртуалізація серверних додатків, і поява хмарних сервісів є деякими тенденціями розвитку мережі, які змушують мережеву індустрію переглянути традиційні підходи до розгортання мережевої архітектури. Багато традиційних мереж побудовані ієрархічно, де комутатори являють собою групи, розташовані в деревоподібній архітектурі. Така структура мережі була найбільш вигідним і домінуючим при обчисленнях типу клієнт-сервер, проте така статична архітектура неприйнятна при динамічних обчисленнях і сучасних способів зберігання даних, кампусах, і сервісних провайдерів. Нижче перераховані деякі з ключових тенденцій, які змушують мережеву індустрію переглянути традиційні підходи до мережевої архітектури.

• Консьюмерізація інформаційних технологій:

Користувачі все більше і більше використовують персональні пристрої, такі як смартфони, планшетні комп›ютери і ноутбуки для доступу до корпоративної мережі. Це чинить тиск на інформаційні технології, які повинні розмістити ці пристрої в архітектуру, при цьому зберігаючи всі необхідні вимоги, з точки зору захисту корпоративної інформації і дотримання різних умов.

• Зміна характеру трафіку:

На відміну від додатків типу клієнт-сервер, в яких велика частина трафіку проходить між одним клієнтом і одним сервером, додаткам на сьогоднішній день необхідний доступ на різні бази даних і сервера, створюючи значні

навантаження трафіку між різними машинами у всіх напрямках, перш ніж доставити необхідну інформацію клієнту з будь-якого типу обладнання, місця розташування та моменту часу.

• Зростання сервісів хмарних обчислень:

Компаній хмарних обчислень вимагають все більшої гнучкості в доступі до додатків, інфраструктури, та інших IT ресурсів на вимогу. IT планування таких хмарних сервісів повинно проводитися з урахуванням вимог щодо підвищеної безпеки, сумісності, і вимог аудиту, поряд з можливими бізнес реорганізаціями, об›єднаннями і злиттями, які можуть відбутися несподівано. Надання самостійного резервування сервісів, будь то приватні або публічні хмари, вимагає гнучкого масштабування обчислень, зберігання даних, і мережевих ресурсів, в ідеалі, з загальної точки зору і єдиного набору інструментів.

• Великий обсяг даних:

Обробка сучасних обсягів інформації вимагає паралельної обробки великого об'єму інформації тисячами серверами, кожен з яких повинен бути пов›язаний з усіма іншими. Появи величезних масивів даних підштовхує постійний попит на додаткові мережеві ємності в дата центрах. Мережеві оператори гіпермасштабних дата центрів стикаються зі складним завданням масштабування мереж до раніше немислимих розмірів, надаючи комунікації типу кожен-з-кожним, при цьому зберігаючи прибутковість.

Провайдери і підприємства прагнуть впроваджувати нові можливості та послуги, реагуючи на швидко змінюючі потреби бізнесу і вимог користувачів. Відсутність стандартних, відкритих інтерфейсів обмежує можливість мережевим операторам адаптувати мережу для індивідуальних умов. Ця невідповідність між вимогами ринку і можливостей мережі призвела телекомунікаційну галузь до переломного моменту. У відповідь індустрія створила архітектуру програмованих мереж SDN. Проте однією із проблем в програмно-конфігурованих мережах, як і у всіх сучасних комутованих мережах, є нерівномірний розподіл трафіку по мережі.

Відповідно основним недоліком в мережах SDN є неефективне використання всіх існуючих з›єднань при використанні протоколу STP (Spanning Tree Protocol), який призначений тільки для запобігання петель в мережі, в його найпростішому варіанті 802.1D та без можливості передавати потоки по різних шляхах. Дана ситуація стосується топології, в якій закладена можливість резервування. Ця проблема частково була вирішена шляхом поділу топології щодо належності трафіку VLAN (протоколи MSTP і PVRST +). PVRST + створює окрему топологію для кожного VLAN, коли як MSTP групує VLAN за належністю до однієї топології. Таким чином, з'являється можливість пересилання трафіку по різних шляхах прив›язуючи VLAN до різних топологій. Хоча протоколи MSTP і PVRST + і оптимізують пересилку трафіку, використовуючи різні шляхи для різних VLAN, ці протоколи є досить статичними для конфігурування. Прив›язка до топології проводиться в ручному режимі.

Дослідженням задач стандартизації, управління інформаційними потоками та балансуванням навантаження в SDN мережах активно займаються фахівці в Україні та закордоном. Зокрема, Орлов Є. В. аналізує стан міжнародної стандартизації програмно-конфігурованих мереж (Software-Defined Network, SDN), архітектура яких, на відмінність від традиційної, передбачає відділення площини управління від площини передачі даних, а також використання спеціалізованого протоколу для обміну управляючою інформацією між двома площинами, наприклад протоколу OpenFlow. Фещенко О.А., Лемешко О.В, Глоба Л.С. які досліджують якість обслуговування послуг в програмно-конфігурованих мережах. Серед іноземних дослідників слід відзначити роботи з питань забезпечення QoS з кінця-в-кінець, розроблення ефективних методів маршрутизації та балансування навантаження в SDN мережах Widhi Yahya, Achmad Basuki, Jehn-Ruey Jiang.

Мета статті — дослідити складові архітектури Openflow, у тому числі моделі комутованої мережі з використанням протоколу Spanning tree, а також розробка альтернативного механізму балансування, для розширення можливостей передавання трафіку в комутованій SDN мережі.

Виклад основного мматеріалу дослідження. На сьогоднішній день парадигма SDN набуває все більшої популярності. Безліч IT організацій і мережевих провайдерів застосовує її, в першу чергу, з метою зниження вартості мережевої інфраструктури та витрат на її утримання, а також забезпечення високого рівня керованості, захищеності та надійності мережі [1]. Основною ідеєю SDN є поділ рівня управління мережею (control plane) і рівня передачі трафіку (forwarding plane). На рисунку 1 показана логічна архітектура SDN.

Логіка мережі централізована в програмних контролерах SDN, які підтримують загальне уявлення про мережі. В результаті, додатки представляють мережу як єдиний логічний комутатор. SDN також значно спрощує самі мережеві пристрої, так як вони більше не повинні розуміти і обробляти тисячі протоколів, а просто отримувати вказівки від контролерів SDN. Крім того, використовуючи централізовану логіку SDN контролера, IT адміністратори можуть змінити поведінку мережі в режимі реального часу і розгортати нові додатки та мережеві сервіси протягом декількох годин або днів, а не тижнів або місяців необхідних на сьогоднішній день. За рахунок централізації стану мережі на рівні управління, SDN дає мережевим адміністраторам гнучкість в налаштуванні, управлінні, захисту та оптимізації мережевих ресурсів за допомогою динамічних, автоматизованих програм SDN.

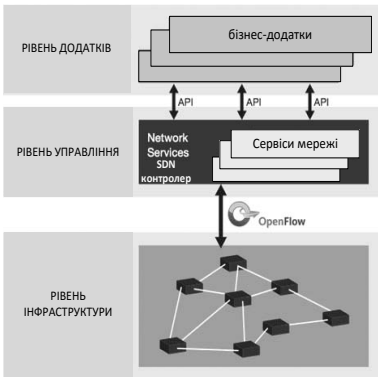

Рис.1. Архітектура SDN

Крім того, вони можуть написати ці програми самостійно, замість очікування впровадження певної функції в закриті програмні

середовища мережі. На додаток до абстрагування мережі, SDN архітектури підтримують набір API-інтерфейсів, які дозволяють здійснювати загальне обслуговування мережі, включаючи маршрутизацію, мультикастовий трафік, безпека, контроль доступу, управління смугою пропускання, управління трафіком, якість обслуговування, оптимізацію процесора і зберігання даних, використання енергії, і всі форми управління політиками, виготовлені за індивідуальним замовленням для задоволення бізнес-цілей.

Сама система SDN складається з двох компонентів: комутатор з підтримкою протоколу Openflow, і контролер. Openflow комутатор, у свою чергу, складається з трьох частин: (1) Таблиця потоків (Flow Table), яка визначає дії комутатора для кожного потоку (2) Безпечний Канал (Secure Channel), що забезпечує передачу службової інформації між контролером і комутатором (3) Протокол Openflow, що надає відкритий і стандартизований метод комунікації комутатора з самим контролером. Таким чином, надається стандартний метод для програмування комутаторів без необхідності настройки кожного комутатора окремо. Комутатор Openflow являє собою простий комутуючий елемент, який пересилає пакети між портами. Самі правила для комутації визначаються безпосередньо віддаленим контролером. Контролер може являти собою будь-яку x86 машину з блоком коду з головним модулем, що з›єднує додаткові модулі, кожен відповідальний за певні функції. Функції включають в себе визначення активних комутаторів в мережі, визначення активних портів на комутаторах, зв›язок з комутаторами по протоколу Openflow, і опису логіки комутації і маршрутизації пакетів по всій мережі, для заповнення таблиці потоків. Однією з цілей розробки Openflow є впровадження загальної системи, над якою може працювати будь-який бажаючий. Тому Openflow є платформою, незалежної від вендорів і мов програмування. Openflow також є проектом «open source», тобто доступ до вихідного коду є у всіх бажаючих.

На сьогоднішній час контролери розробляються на різних мовах, найуспішніші з яких є NOX (C ++, Python), POX (Python), Floodlight, Trema, Beacon (Java). Метою даної роботи була розробка додаткових модулів та модифікація вже існуючих модулів для контролера Openflow. В якості базового компонента вибрано контролер POX. Даний контролер був обраний з кількох причин. Контролер POX є наступником NOX, першим з успішних контролерів для програмованих мереж. Відповідно, розробники змогли усунути недоліки і недоробки NOX, які були усуненні вже в існуючому коді із за його дизайну. Також, POX написаний на мові Python, що робить його більш гнучким і легким для компілювання і використання, ніж контролери написані на C ++. За своєю суттю, це платформа для швидкої розробки програмного забезпечення для мережевого управління, розроблена на мові Python. Перевагами POX є стабільність платформи, швидкість роботи, а також спрощений дизайн мови Python.За обмеженої на даній стадії підтримки обладнання контролерами Openflow, було прийнято рішення проводити всі тести на симуляторі Mininet. Mininet є інструментом для швидкого створення прототипів програмованих

мереж (SDN). Mininet створює близькі до реальності віртуальні мережі з реальними робочими компонентами, яке працює на одній машині для зручності тестування. Mininet дозволяє створювати користувальницькі машини, комутатори (Openvswitch), а також контролери, забезпечуючи при цьому зв›язність цих компонентів в мережі. Openvswitch представляє багаторівневий програмний комутатор на відкритій ліцензії Apache2. Метою розробки даного програмного забезпечення є отримання платформи для комутатора в якості експлуатованого у виробництві обладнання, який підтримує всі стандартизовані протоколи, а також відкриває можливості пересилання даних у вигляді програмованих засобів управління. Openvswitch добре підходить для роботи в якості віртуального комутатора в середовищі VM. На додаток до відкриття стандартних засобів контролю та моніторингу, він розрахований підтримувати дистрибуцію між декількома фізичними серверами. Openvswitch підтримує численні технології віртуалізації, в тому числі і Xen / XenServer, KVM, і VirtualBox. Основна частина коду написана на незалежному від платформ мові C і легко переноситься на інші середовища.

На даний момент йде активна робота над створенням додаткових модулів до SDN системи. Одним з таких модулів є модуль для обчислення дерева «spanning-tree», аналогом протоколів STP (IEEE 802.1D) в традиційних мережевих технологіях. Метою даного модуля є обчислення дерева для поточної топології мережі, в якому не існує резервних шляхів від одного комутатора до другого.

Існуючий модуль «spanning-tree» працює за наступним принципом:

1) Виділення dpid;

Кожному комутатора виділяється унікальний ідентифікатор (в рамках протоколу Openflow), dpid.

2) Сортування по dpid;

Проводиться сортування серед комутаторів за значенням ідентифікатора dpid. Результат записується в список Nodes.

3) Створення списків Nodes і Done;

Вибирається найменше значення dpid, комутатор з даними dpid вибирається як корінь дерева. Даний комутатор видаляється зі списку Nodes, і додається в список Done (список комутаторів, які вже в дереві).

4) Визначення сусідства;

Для обраного кореня виявляються всі сусідні комутатори і відповідні з›єднання. Для цього використовується модуль Discovery, який періодично опитує всі комутатори для підтримки достовірної інформації про топологію мережі.

5) Додавання комутаторів в дерево;

Сусідні з коренем комутатори включаються в дерево, також як і з›єднання з коренем. З›єднання стають гілками дерева, а самі комутатори - вузлами дерева. При цьому ці комутатори додаються в список Done, і відсортовуванні додаються до початку списку Nodes.

6) Виняток зайвих з'єднань;

Зі списку Nodes береться наступний комутатор, і аналогічна операція

повторюється, проте вже з урахуванням списку Done - якщо сусід вже знаходиться в дереві, даний сусід і відповідне з›єднання не враховується.

7) Завершення розрахунків.

Обчислення відбувається до тих пір, поки список Done не включатиме в себе всі комутатори в мережі.

У даній версії spanning tree для обчислення топології дерева використовується алгоритм Breadth First Traversal. Логіка алгоритму полягає в тому, що вершини графа обробляються на рівні і перед рівнем і + 1(рис.2).

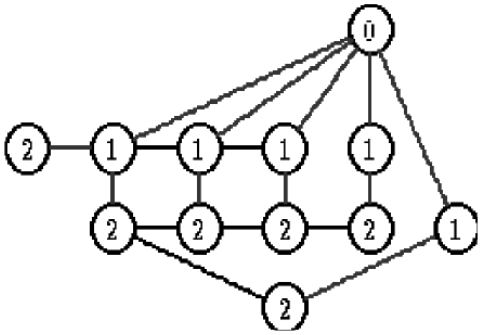

Рис.2 Приклад розрахунку алгоритму Breadth First Traversal

Такий механізм призначений тільки для запобігання петель в мережі, не передбачаючи можливість використання додаткових резервних шляхів для балансування трафіку. Враховуючи даний недолік, розроблено метод, який враховує додаткові шляхи для балансування трафіку.

Доповнення полягає в тому, що дерево «spanning-tree» розраховується не для одного варіанта дерева, де коренем вибирається один комутатор, в даному випадку комутатор з найменшим dpid, а для N дерев, де N - кількість комутаторів в мережі. Для кожного з цих дерев в ролі кореня вибирається один з комутаторів мережі, для якого потім розраховується дерево, дотримуючись принципу, описаному раніше. Версії дерев розраховуються, коли контролер підключається до мережі, і оновлюються кожного разу, коли в мережі відбувається зміна. Після того, як відбувається розрахунок дерев, дані передаються модулю l2 learning, який також був модифікований під роботу нового алгоритму. У цій модифікації, замість комутації пакетів на основі MAC адрес, контролер збирає інформацію про MAC адреси, IP адреси, і портах TCP і UDP джерела і призначення, а комутатор комутує пакети відповідно цим правилам. При цьому кожен потік класифікується відповідно цими параметрам. Наприклад, два потоки з різними MAC адресами джерела або різними портами TCP / UDP призначення будуть вважатися як різні потоки. Балансування проводиться на основі класифікації потоків. Кожен раз, коли в комутатор надходить пакет, з параметрами раніше не описаних у таблиці комутації, для цього пакета визначається дерево spanning-tree, відповідно якому і відбувається передача пакетів в даному потоці. Таким

чином, оскільки дерева використовують різні шляхи досягнення від одного вузла до іншого (враховуючи що існують резервні шляху), трафік буде розподілятися по всій мережі з певним ступенем рівномірності. Ступінь ефективності оптимізованих модулів залежить від наявності резервних шляхів між точками пересилання трафіку. Так, для повнозв›язкову графа G з N комутаторів, кількість шляхів P між двома точками буде дорівнювати:

$$P(G_N) = \mathrm{N} - 1. \tag{1}$$

Дану залежність досить легко довести, так як між будь-якою парою точок A і B є пряма зв'язність, а також по одному резервному шляху для кожної з точок C, проходить через $A - C - B$, причому ці шляхи не перетинаються між собою. Кількість подібних точок у графі одно $N - 2$. Відповідно, враховуючи прямий стик, кількість шляхів між двома точками буде дорівнювати $N - 1$. Модифікований в даній роботі модуль використовує кожен з цих шляхів, так як в повнозв'язному графі шлях між точками A і B, при обраному корені C, буде проходити через $A - C - B$.

Виходячи з вищеописаних обчислень, для повнозв'язної мережі з N комутаторів, передбачуване збільшення корисного трафіку I в модифікованих модулях, у порівнянні з існуючим модулем, буде:

$$I(G_N) = ((\mathrm{N} - 1) - 1) \cdot 100\% = (\mathrm{N} - 2) \cdot 100\%. \tag{2}$$

Для тестування ефективності доданих алгоритмів, проведено наступні експерименти:

Експеримент 1:

Побудована мережа з трьох комутаторів, де між кожним комутатором існує два альтернативні способи доставки пакетів. Мережа відображена на рисунку 3. Для пари віртуальних машин h1-h2 запускається утиліта iperf для передачі TCP трафіку. Запускаються три різні сесії TCP з різними портами TCP джерела, так як IP / MAC адреси залишаються незмінними.

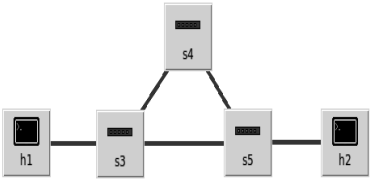

Рис.3 Експериментальна мережа з 3-х комутаторів

У вищеописаної мережі протестовані алгоритми spanning-tree до внесення додаткових змін, і після. Підрахований загальний корисний трафік для кожного з варіантів. Проведено 30 вимірювань для кожного випадку, де кожна зміна проводилося протягом 10 секунд. Результати вимірювань наведені на рисунку 4.

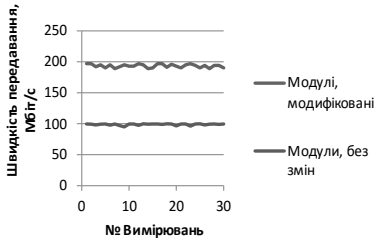

Рис.4 Результати порівняння модулів spanning-tree в мережі з 3 комутаторів

Експеримент 2:

Експеримент проведений на мережі з 6 комутаторами. На рисунку 5 відображена мережу, на якій проводився експеримент.

Рис.5 Експериментальна мережа з 6-х комутаторів

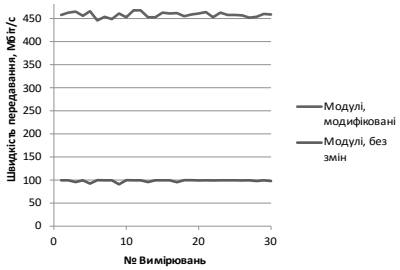

Рис.6 Результати порівняння модулів spanning-tree в мережі з 6 комутаторів

Для оцінки середнього значення пропускної здатності і відхилень у вимірах, використані наступні формули

$$m_x = \frac{\sum_{i=1}^{n} x_i}{n} \tag{3}$$

$$\sigma_n = \sqrt{\frac{\sum_{k=1}^{n} (x_k - m_x)^2}{n}}, \tag{4}$$

де n- кількість проведених замірів,

x_i- результат і-го розміру,

m_x - середнє значення отриманих результатів,

σ_n - середньоквадратичне відхилення.

Результати проведених експериментів показують, що в результаті балансування в мережі з 3 комутаторів, математичне очікування об'єму корисного трафіку становить:

Для не модифікованого модуля: використовуючи (5) і (6), отримуємо $m_{x(без.мод.)}$= 98,6 Мбіт/с при середньоквадратичному відхиленні σ_n=1,1 Мбіт/с; Для оптимізованого модуля: $m_{x(із.мод.)}$= = 194,5 Мбіт/с при середньоквадратичному відхиленні σ_n = 2,6 Мбіт/с; Збільшення I математичного очікування об'єму корисного трафіку становить 96,9%, при очікуваному I = (3-2) * 100% = 100%.

У мережі з 6 комутаторів: $m_{x(без.мод.)}$=97,4 Мбіт/с при середньоквадратичному відхиленні 2,4 Мбіт/с;$m_{x(із.мод.)}$=459 Мбіт/с при середньоквадратичному відхиленні 5,5 Мбіт/с;

Збільшення математичного очікування об'єму корисного трафіку становить 365%, при очікуваному I = (6-2) * 100% = 400%.

Отримані результати за проведеними експериментам показують, що фактичне збільшення пропускної здатності є прогнозованим. Похибки, отримані в результаті, пояснюються обмеженням продуктивності комутаторів і самих термінальних пристроїв, які фактично є віртуальними машинами, що використовують один процесорний ресурс - CPU локальної машини.

Висновки. Майбутнє мереж буде більше і більше покладатися на програмне забезпечення, яке дасть змогу прискорити темпи інновацій для існуючих вимог до побудови телекомунікаційних систем. SDN може перетворити сучасні статичні мережі в гнучкі, програмовані платформ з умінням динамічно розподіляти ресурси, масштабами, здатними підтримати величезні центри обробки даних і віртуалізацію, необхідну для підтримки динамічної, автоматизованої і безпечної Cloud платформи.

У даній роботі ми описали важливість нової архітектури програмованих мереж в телекомунікаціях. Розглянута архітектура протокола Openflow в цілому, а також складові архітектури Openflow, у тому числі комутатор Openvswitch та контролер SDN. Досліджено і протестовано різні модулі контролера Openflow на основі мови Python. Створено модель комутованої мережі з використанням протоколу Spanning tree в програмному середовищі Mininet. Розглянута реалізація протоколу Spanning tree в архітектурі Openflow, у тому числі взаємодія модуля Spanning tree з модулями Discovery, l2-learning та виявлено недоліки моделі комутованої мережі з використанням протоколу Spanning tree, а також недоліки реалізації протоколу Spanning tree в архітектурі Openflow. На основі виявлених недоліків, розроблені модифіковані версії модулів Spanning tree і l2- learning. Проведено оцінку очікуваного приросту трафіку в розглянутих моделях мереж при тестуванні розроблених розширених версій модулів Spanning tree і l2-learning, в ході якого отримано позитивний результат з погляду поставленої в даній роботі мети.

Список використаних джерел

1. Architecture SDN Open networking foundation», av ailable at.: https://www.opennetworking.org. (accessent at 2014)

2. «Mininet», av ailable at : http://www.mininet.org/.
3. Yuanhao Zhou, Li Ruan, Limin Xiao, Rui Liu. (2014), A Method for Load Balancing based on Software Defined Network. School of Computer Science and Engineering Beihang University, Beijing, China Vol. 45, pp.43-48.
4. Beshley M., Klymash M., Strykhalyuk B., Shpur O., Bugil B., Kagalo I. (2015), «SOA quality management subsystem on the basis of load balancing method using fuzzy sets», International Journal of Computer Science and Software Engineering (IJCSSE), Vol. 4, pp.10-21.

References

1. «Architecture SDN Open networking foundation», av ailable at.: https://www.opennetworking.org. (accessent at 2014)
2. «Mininet», av ailable at : http://www.mininet.org/.
3. Yuanhao Zhou, Li Ruan, Limin Xiao, Rui Liu. (2014), A Method for Load Balancing based on Software Defined Network. School of Computer Science and Engineering Beihang University, Beijing, China Vol. 45, pp.43-48.
4. Beshley M., Klymash M., Strykhalyuk B., Shpur O., Bugil B., Kagalo I. (2015), «SOA quality management subsystem on the basis of load balancing method using fuzzy sets», International Journal of Computer Science and Software Engineering (IJCSSE), Vol. 4, pp.10-21.

THE DEVELOPMENT OF LOAD BALANCING METHOD IN SDN NETWORKS BASED ON MODIFIED PROTOCOL STP

M. M. Klymash., M.I Beshley., Y.L Deschynskyy., O.M. Panchenko

Lviv Polytechnic National University, S. Bandery Str., 12, Lviv, 79013, Ukraine
mklimash@lp.edu.ua

The existing architecture SDN, especially the development and interaction of its components have been analized and given. Software-defined networking (SDN) is one promising class of network architectures that are suitable substitutes for traditional switched Ethernet. SDN lets an external controller manage the behavior of switches in a network. The global network view and control enabled by SDN allows for the fine-grained security and high performance required from a datacenter network providing high performance, manageability, reliability, scalability and security for data centers. Considered job of STP protocol to detect loops in the network and its main disadvantages when used in SDN networks has been defined. A new method for balancing traffic based on programmable network protocol Openflow and modification of STP has been created.

Keywords: *Software-configured network, controller, traffic, protocol STP, protocol Openflow.*

Стаття надійшла до редакції 10.02.2015.
Received 10.02.2015.