УДК 004.7+004.415 **DOI:** 10.31673/2412-9070.2023.063743

В. В. ДМИТРЕНКО, ст. викладач;
О. М. МАРЧУК, ст. викладач,
Державний університет інформаційно-комунікаційних технологій, Київ

ІМІТАЦІЙНО-АНАЛІТИЧНІ МОДЕЛІ ГІБРИДНИХ ПРОГРАМНО-КОНФІГУРОВАНИХ ПРИСТРОЇВ

Проаналізовано алгоритм, на основі якого автоматично синтезується імітаційно-аналітична модель гібридного програмно-конфігурованого пристрою. Також розглянуто рішення щодо моделювання обладнання OpenFlow та створено оригінальну модель комутатора OpenFlow, що відрізняється можливістю керувати затримками за результатами роботи своєї математичної моделі. Розроблено схему роботи гібридного комутатора, що підтримує як традиційну комутацію, так і комутацію OpenFlow. Ця схема дає змогу вирішувати комплексні завдання у фізичній інфраструктурі віртуалізованих мереж. Розглянуто модуль створення реалістичної затримки й обмеження ємності та продуктивності моделі гібридного комутатора, який може працювати за даними, здобутими в результаті експериментального дослідження. Для більш детальної реалізації моделі гібридного програмно-конфігурованого обладнання запропоновано схеми додаткового оброблення трафіку. Вивчено базову модель комутатора, репрезентованого у фреймворку INET OMNeT++. Комутатор подано набором модулів, що взаємодіють через систему передавання OMNeT. Найбільш докладно реалізовано модулі інтерфейсів Ethernet, зокрема є реалізації вхідної черги типу Tail – Drop, вихідних черг типів FIFO, RED, WRED, PQ, CBFQ, а також можливу реалізацію затримки передавання на лінії. Досліджено технологію віртуалізації мережних функцій, побудовану на базі програмно-конфігурованих мереж, а найпопулярніші рішення на базі Linux використано як програмно-керований комутатор Open vSwitch. Визначено методи формування моделі такого комутатора. Open vSwitch архітектурно відрізняється як від класичного, так і від OpenFlow реального комутатора, оскільки не має виокремлених блоків передоброблення пакетів (окрім мережних карт з апаратним обробленням заголовків та FPGA), виділеної пам'яті та процесорів для комутації. Open vSwitch використовує модуль ядра (або модуль DPDK для прямого передавання без участі ядра) для передавання пакетів за інформацією з бази даних OVSDB або таблиць OpenFlow.

Ключові слова: імітаційно-аналітична модель; OpenFlow; комутація; оброблення трафіку; віртуалізовані мережі.

ВСТУП

Мета і задачі дослідження. Актуальність теми полягає в дослідженні сучасних моделей комутаторів у системі пакетного моделювання OMNeT++ та аналізу особливостей моделі комутатора OpeFlow.

Мета статті: розробити імітаційні моделі комутатора із підтриманням OpenFlow та мережного пристрою з підтриманням NFV та запропонувати алгоритм автоматизованого синтезу імітаційно-аналітичної моделі гібридного програмно-конфігурованого обладнання, на основі якого побудовано імітаційно-аналітичну модель гібридного програмно-конфігурованого пристрою.

ОСНОВНА ЧАСТИНА

Розроблення імітаційно-аналітичної моделі мережного обладнання

Дослідимо процес проходження пакету крізь гібридний програмно-конфігурований пристрій на прикладі моделі комутатора.

Розглянемо базову модель комутатора, поданого у фреймворку INET OMNeT++. Комутатор представлений набором модулів, що взаємодіють через систему передавання OMNeT [1]. Найбільш докладно виконані модулі інтерфейсів Ethernet, зокрема є розгортання вхідної черги типу Tail – Drop, вихідних черг типів FIFO, RED, WRED, PQ, CBFQ, також можлива затримка передавання на лінії. Однак у цій базовій моделі відсутня реалізація чи імітація процесів внутрішнього оброблення пакетів, а отже, всю комутацію подано модулем MacAddressTable з пошуком адрес і модулем MacRelayUnit, що зазвичай передає пакети на вихідний інтерфейс без затримок або імітує розсилання на вибрані або всі порти [2].

Схему проходження пакету через комутатор стандартного модуля EtherSwitch, який використовують у більшості моделей, зображено на рис. 1.

Блок «mac» реалізує функції підрівня Media Access Control (MAC) канального рівня з використанням технології IEEE 802.3 CSMA/CD. У разі посилання кадру з верхніх рівнів, кадри ставляться в чергу передавання (внутрішній submodule), крім того, за запитом верхніх рівнів може бути відправлено повідомлення PAUSE. Кадри, що надходять із мережі, обробляються так: перевіряється CRC (кадри з помилкою відкидаються), кадри PAUSE отримують відповідь, у режимі promiscuous (за замовчуванням

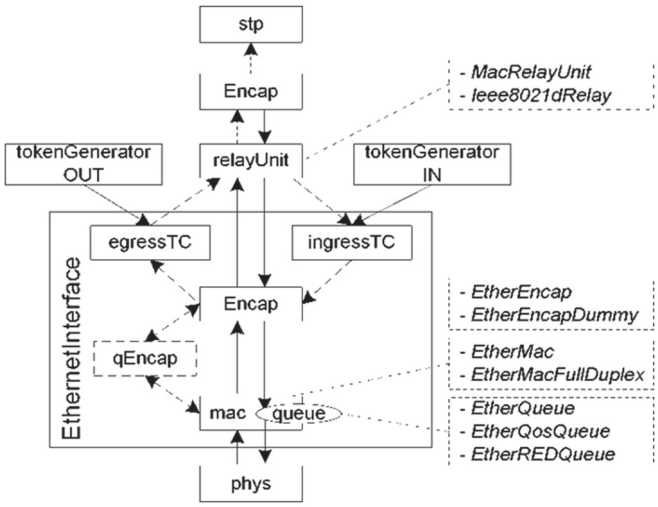


Рис. 1. Схема проходження пакету через комутатор EtherSwitch фреймворку INET

для комутатора) відправляються всі отримані кадри, в іншому разі (режим хоста) тільки кадри з MAC-адресами, що збігаються, і широкомовні кадри. Може бути встановлений напівдуплексний (EherMac) або повнодуплексний (EtherMacFullDuplex) режими роботи рівня mac [3].

Блок qEncap призначено для оброблення тегів протоколу IEEE 802.1q (VLAN) з використанням фільтра VLANID (для забезпечення роботи лише вибраних VLAN), а також правил перезапису тегів для додавання та видалення тегів у різних напрямках.

Блок Encap виконує функції підрівня LLC канального рівня. Пакети, що надходять із верхніх рівнів, інкапсулюються в кадр Ethernet і вирушають до MAC. Кадри Ethernet, що надходять від MAC, декапсулюються та відправляються на більш високі рівні. Для блока Encap можуть бути встановлені режими EtherEncap (підтримує реальні кадри, які можна передати в систему моделювання як PCAP-файли з дампом трафіку) або EtherEncapDummy (пакети з випадковим внутрішнім вмістом).

Модулі ingressTC та ***egressTC*** — вхідний та вихідний формувачі трафіку (Traffic conditioners). Блоки Traffic conditioners не є стандартними та мають бути створені розробником конкретних моделей. Вони можуть класифікувати вхідні пакети, вимірювати трафік у кожному класі, позначати або відкидати пакети залежно від результату вимірювання, формувати трафік відповідно до бажаного профілю трафіку. Для зовнішнього керування поведінкою цих блоків уведено модулі ***tokenGeneratorIN*** і ***tokenGeneratorOUT***, більш детально описані далі.

У стандартному комутаторі EtherSwitch без додаткових налаштувань поява черги можлива лише у підмодулі queue модуля mac під час проходження пакету від верхніх рівнів до нижніх.

У маршрутизаторах модуль MAC використовує модуль зовнішньої черги для моделювання кінцевого буфера, реалізації QoS і/або RED та запиту пакетів із цієї зовнішньої черги. У хостах така черга не використовується, тому MAC містить внутрішню чергу для зберігання пакетів, що очікують на передавання. Концептуально черга нескінченного розміру, але для кращої діагностики в параметрі packetCapacity можна зазначити жорстку межу, у разі перевищення якої симуляція зупиняється з помилкою.

За замовчуванням черга має тип EtherQueue. Схематично цю чергу наведено на рис. 2.

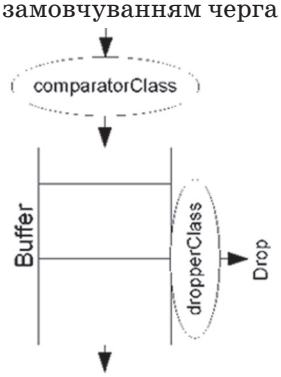
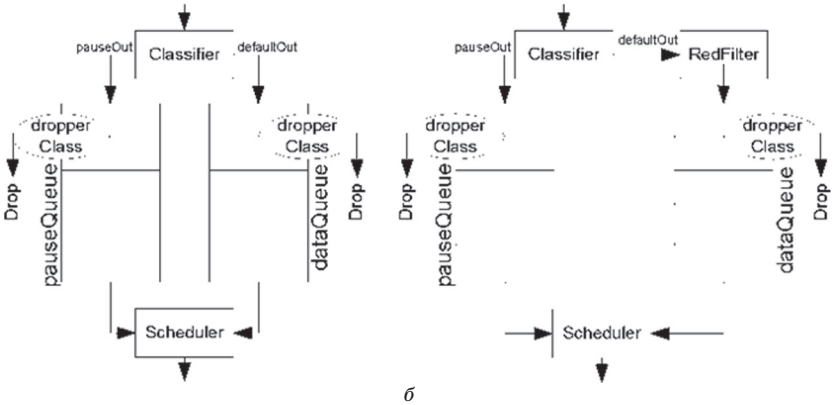


Рис. 2. Черга EtherQueue

Клас comporatorClass визначає порядок пакетів у черзі та порядок вставлення їх у чергу. Для черги EtherQueue він має значення «inet::EthernetFrameTypeComparator», що забезпечує пріоритет кадрам PAUSE Ethernet. Параметр dropperClass визначає, які пакети відкидаються під час переповнення черги спочатку.

Окрім черги EtherQueue можуть бути встановлені черги EtherQosQueue (рис. 3, *а*) та EtherQosRedQueue (рис. 3, *б*). Модуль черги EtherQosQueue дає кадрам PAUSE вищий пріоритет. Для обслуговування кадрів даних він може налаштуватись за допомогою інтерфейсу IPacketQueue. Модуль EtherQosQueue має у своєму складі класифікатор, дві черги та планувальник [4]. Класифікатор пакетів — це модуль, який має один пасивний вхід та кілька активних виходів. Пакети, надіслані на пасивний вхід, передаються одним з активних виходів без будь-якої затримки і перевпорядкування. За замовчуванням модуль EtherQosQueue містить класифікатор, який перенаправляє кадри Ethernet PAUSE в чергу pauseOut, а інші кадри — у чергу defaultOut. Обидві ці черги є чергами типу

DropTailQueue, тобто, являють собою обмежену чергу пакетів, яка відкидає пакети у хвіст черги. Планувальник пакетів — це модуль, який має кілька активних входів та один пасивний вихід. Пакети, витягнуті з пасивного виходу, надаються одним із входів без будь-якої затримки та перевпорядкування. За замовчуванням у модулі EtherQosQueue увімкнений планувальник, який виштовхує пакети з першого непустого джерела пакетів.



а *б*

Рис. 3. Проходження пакету через черги EtherQosQueue (*а*) та EtherQosRedQueue (*б*)

Модуль EtherQosRedQueue відрізняється від EtherQosQueue використанням алгоритму випадкового раннього виявлення кадрів даних [5]. Пакети, що надійшли в *i*-й вхідний шлюз, пересилаються в *i*-й вихідний шлюз або відкидаються. Модуль підсумовує використовуваний буферний простір черг, прикріплених до вихідних шлюзів. Якщо простір нижчий за мінімальний поріг, то пакет не буде відкинуто, якщо вищий за максимальний поріг, його буде відкинуто, а якщо простір міститься між мінімальним і максимальним порогом, його буде відкинуто з певною ймовірністю.

Розроблення додаткових модулів моделі гібридного програмно-конфігурованого обладнання для реалізації внутрішніх процесів оброблення трафіку мережного пристрою

Для більш детальної реалізації моделі гібридного програмно-конфігурованого обладнання розроблено схеми додаткового оброблення трафіку. На комутаторі опціонально можуть бути налаштовані такі функції, як фільтрація пакетів за допомогою списків ACL, профільники policing та shaping, класифікація та модифікація пакетів (рис. 4). Усі ці функції викликаються під час проходження пакету через пристрій послідовно, вносячи затримку. Оскільки в реальному комутаторі складно чи неможливо оцінити затримку кожного елемента на різних навантаженнях, прийнято рішення додати модуль внесення агрегованої затримки RealDelay.

Фреймворк INET дає змогу реалізувати зазначені функції за допомогою таких модулів:

• *contentBaseFilter* є модулем, котрий відкидає пакети на основі даних, які вони містять, аналог Access Control List;

• *leakyBucket* — універсальний профільник із переповненням і швидкістю виведення, що настроюється, реалізує параметризований алгоритм із витоком, аналог Traffic Shaper;

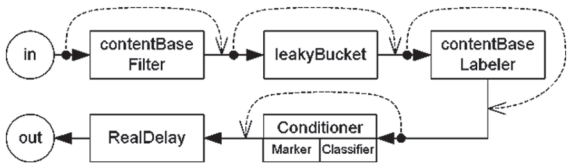


Рис. 4. Черга EtherQueue

• *contentBaseLabeler* прикріплює позначки до пакетів на основі даних, які вони містять, аналог Class - Map;

• як модифікатор може бути використаний модуль *Conditioner* (обробник за умовою), що складається з двох частин: Classifier і Marker. *Classifier* — класифікатор, що містить список фільтрів (модуль для перезапису полів протоколів пакету, наприклад DSCP), який ідентифікує потоки, визначає їх класи, робить заміну полів класів. Кожен фільтр може збігатися з адресою джерела та призначення, номером протоколу IP, портами джерела та призначення або ToS/CoS/DSCP. Перший відповідний фільтр визначає індекс виходу. Якщо відповідний фільтр не знайдено, пакет буде надіслано через шлюз defaultOut. *Marker* — модуль, що встановлює в полі DSCP (молодші шість бітів Tos/TrafficClass) IP-дейтаграм значення, зазначене параметром dscps;

• *RealDelay* — розроблений модуль для імітації затримки, що вноситься під час проходження пакетом описаних блоків. Схему модуля зображено на рис. 5.

Розмір затримки може бути задано як функція ймовірності, наприклад, exponential (lambda), або встановлено програмно. Черга емулює роботу вхідного буфера оброблення механізмом Tail drop (FIFO)

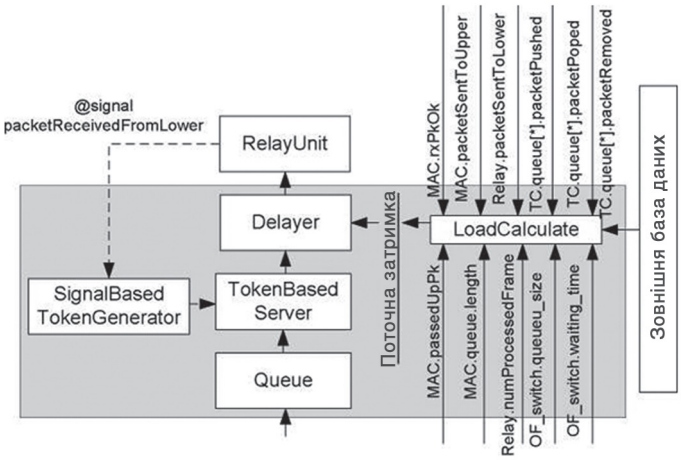


Рис. 5. Модуль імітації затримки та обмеження ємності та продуктивності RealDelay

з обмеженою ємністю. TokenBasedServer пропускає пакети за сигналом SignalBasedTokenGenerator, який зі свого боку формує сигнал під час отримання пакету комутаційним модулем relayUnit. Для формування затримки за зовнішніми даними, зокрема за даними реального комутатора і даними від модельних модулів, розроблено модуль LoadCalculate. На основі симплексної інтерполяції модуль здійснює розрахунок затримки за всіма вхідними даними. Розроблені модулі сумісні з наявними моделями та можуть бути використані спільно в будь-яких схемах із комутаторами.

Розроблення моделі комутатора з підтриманням OpenFlow

Різні рішення щодо моделювання обладнання OpenFlow мають за джерело одного попередника [6], який працює на нових версіях системи. Для забезпечення працездатності на версії INET 4.3 довелося внести більш як 100 змін у вихідний код C ++, а також файли опису елементів. Змінений дистрибутив наведено в [7]. Було додано MessageDispatcher модулі для зв'язку між рівнями з багатьма входами, також було приведено до сучасних класів увесь вихідний код, логіку перероблено для роботи з новими сутностями пакетів та тегами, які тепер містять метадані мережного пакету. Версія підтримуваного протоколу OpenFlow — 1.5.1.

Модуль OpenFlow Swich структурно не відрізняється від звичайного комутатора з боку фізичного з'єднання з мережею, проте має друге виокремлене з'єднання для роботи з контролером. Сам модуль комутації OF_Switch за логікою роботи схожий на модуль RelayUnit, який є основою традиційного комутатора. Модуль OF_Switch містить механізм формування реалістичних затримок оброблення з параметром serviceTime (використовується в режимі безумовної затримки будь-якого пакету, що прийшов на вхід, допускається застосування генераторів для завдання випадкового часу заданим законам розподілу). Також на відміну від звичайного комутатора в OF Switch є можливість задання кінцевої черги через параметр bufferCapacity, причому внутрішній елемент buffer при цьому використовується лише для спілкування з контролером, а повідомлення просто зберігаються в списку типу std :: list час serviceTime. Як випливає з рис. 6, усі комунікації controlPlane для керування комутатором здійснюються виділеними каналами, що завжди відповідає реальному комутатору [8].

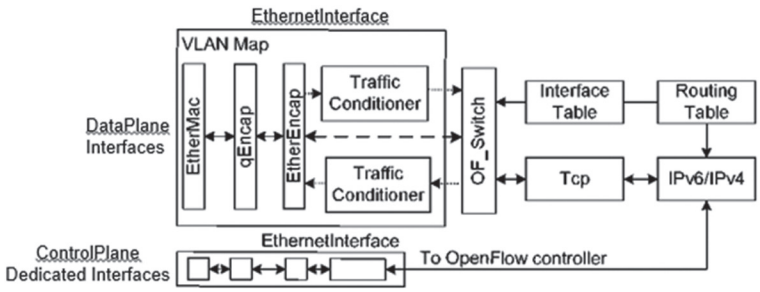


Рис. 6. Структура комутатора OpenFlow

Крім того, модуль OpenFlow Switch не зважає на те, що здебільшого комутатори поєднують як режим класичної комутації, так і різні режими OpenFlow-оброблення пакетів [9].

Схему гібридної комутації зображено на рис. 7.

Основна складність реалізації такого комутатора в OMNET ++ — це автоматичне заповнення списку інтерфейсів модулем InterfaceTable, який бачитиме всі інтерфейси комутатора. Для розв'язання цієї

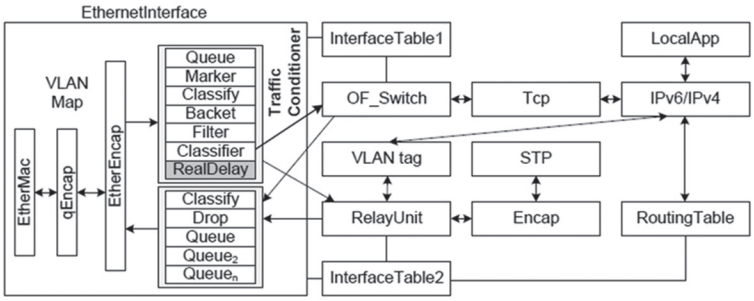


Рис. 7. Схема гібридної комутації

проблеми додано дві таблиці інтерфейсів, одна з яких використовується лише для OpenFlow. Інтерфейси до цієї таблиці додаються за фільтром VLAN.

Поділ вхідного трафіку для оброблення OpenFlow також відбувається за VLANID через ContentBasedClassifier, водночас зберігаючи можливість роботи комутатора за класичною схемою. Вихідні пакети обробляються однаково і проходять через черги, налаштовані на інтерфейсах. Один інтерфейс може приймати пакети як від OpenFlow DataPlane, OpenFlow ControlPlane, так і від комутації.

Керування комутацією OpenFlow відбувається через модуль зв'язку з контролером на основі модуля NetworkLayer, який за таблицею маршрутизації вирішує як надіслати пакет, формує ARP-запити, приймає та фільтрує всі IP-пакети, що прийшли в комутатор, обробляє призначені для встановленої IP-адреси пакети.

Розроблення моделі мережного пристрою NFV

Технологію віртуалізації мережних функцій побудовано на базі програмно-конфігурованих мереж, а найбільш популярні вирішення на базі Linux використовують як програмно-керований комутатор Open vSwitch. Розглянемо методи формування моделі такого комутатора.

Open vSwitch архітектурно відрізняється як від класичного, так і від OpenFlow реального комутатора. Оскільки немає виокремлених блоків передоброблення пакетів (окрім мережних карт з апаратним обробленням заголовків та FPGA), виокремленої пам'яті та процесорів для комутації, Open vSwitch використовує модуль ядра (або модуль DPDK для прямого передавання без участі ядра) для передавання пакетів з інформацією з бази даних OVSDB або таблиць OpenFlow. Оскільки кількість зовнішніх інтерфейсів сервера істотно менша за такі в комутатора, вхідний трафік не фільтрується автоматично за VLAN, не обробляється (крім декапсуляції Ethernet) до надходження в модуль комутації. Усе оброблення трафіку здійснюється тим самим процесором, на якому виконуються віртуальні машини NFV, що є істотним фактором внесення випадкових затримок в оброблення пакетів. Багато стандартних процедур в OVS виконуються засобами відповідних Linux програм, наприклад, маршрутизація, маркування пакетів, фільтрація. В останніх версіях з'явилося маркування пакетів засобами OVSDB, але більшість NFV-середовищ використовує OpenFlow для роботи, керування, забезпечення QoS та обмеження смуги пропускання, а також для призначення відповідності заздалегідь створеним чергам класів пакетів. Схематично структуру NFV-комутації наведено на рис. 8.

Моделювання Open vSwitch у складі NFV ускладнено тим, що безліч процесів, які відбуваються, приховано за контролером. Наприклад, для нестаціонарних мереж 5G VANET, MANET основним для роботи сервісів може бути Virtual Path Computation на основі даних про розміщення NFV та топології мережі, крім того, сам контролер також може бути одним з елементів NFV, зокрема розподіленим на кілька серверів.

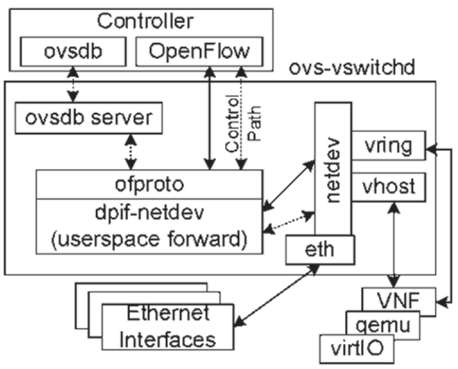


Рис. 8. Структура NFV-комутації

Оскільки Open vSwitch використовує загальну фабрику для оброблення всіх правил (рис. 9), то схема роботи такого комутатора більш універсальна. Для створення моделі такого комутатора в системі OMNeT ++ потрібно поєднати швидку комутацію, стандартну L 2/ L 3/ L 4 комутацію та запам'ятовування даних, а також підхід OpenFlow.

Робота Datapath Classifier та OpenFlow дуже схожа, пошук ведеться за однаковими полями, різняться лише події над пакетами, які специфіковані кожним типом таблиць. Схему Open vSwitch комутатора, де модуль ovs-controller відповідає за швидку комутацію з вимкненим режимом OpenFlow, унаочнює рис. 10.

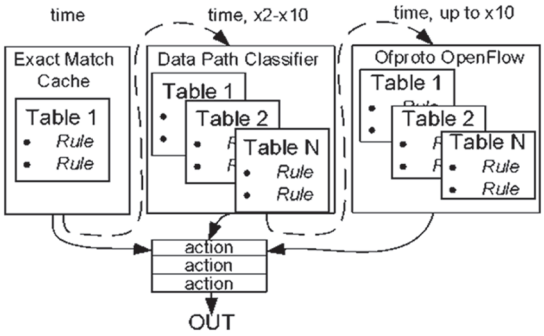


Рис. 9. Пошук правил у Open vSwitch

Таблиці Datapath Classifier емулюються режимом роботи OpenFlow з локальним контролером, за наявності правил LOCAL звернення до зовнішнього контролера немає. Якщо правило вказує на здійснення дії, пакет проходить крізь вибраний модуль L 2 або L 3, модуль Ovs-App приймає та генерує пакети для специфічних протоколів. У разі ввімкнення режиму OpenFlow для пошуку модуль OpenFlow під час надходження пакету передає локальні таблиці та дані таблиць OpenFlow.

Такий режим роботи OpenFlow дає змогу емулювати весь функціонал Open vSwitch крім черг, для них залишено класичний режим роботи, хоча Open vSwitch може організовувати власні черги і сам комутувати пакети.

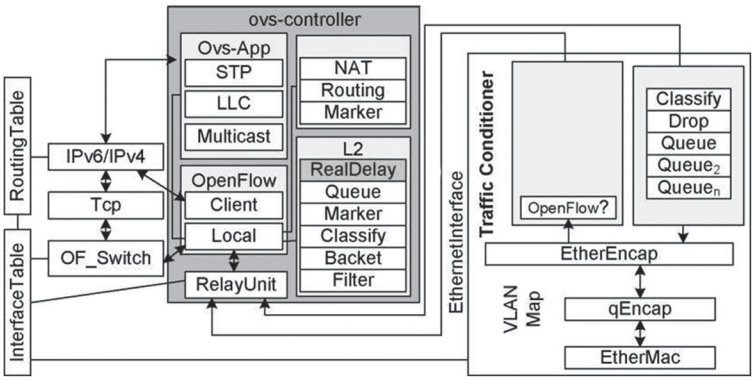


Рис. 10. Схема Open vSwitch комутатора

Висновки

У статті розглянуто рішення щодо моделювання обладнання OpenFlow та розроблено оригінальну модель комутатора OpenFlow, що відрізняється можливістю керувати затримками за результатами роботи своєї математичної моделі. Також запропоновано схему роботи гібридного комутатора, що підтримує як традиційну комутацію, так і комутацію OpenFlow. Ця схема дає змогу вирішувати комплексні завдання у фізичній інфраструктурі віртуалізованих мереж. Розроблено модуль створення реалістичної затримки та обмеження ємності та продуктивності моделі гібридного комутатора, який може працювати за даними, отриманими в результаті експериментального дослідження. Запропоновано алгоритм, на основі якого автоматично синтезується імітаційно-аналітична модель гібридного програмно-конфігурованого пристрою.

Список використаної літератури

*1. **Veselý V., Matousek P., Svéda M.** Multicast simulation and modeling in OMNeT++ // SimuTools. 2013. С. 142–145.*

*2. **Research** of Productivity of Software Configurable Infrastructure in Vanet Networks on the Basis of Models of Hybrid Data Transmission Devices / M. Ushakova, Y. Ushakov, I. Bolodurina [et al.] // International Scientific and Technical Conference Modern Computer Network Technologies (MoNeTeC). 2020. P. 1–9.*

*3. **RedDropper:** INET Framework 4.2 documentation URL:*
https://doc.OMNeTpp.org/inet/api-current/neddoc/ inet.queueing.filter.RedDropper.html

*4. **Исследование** производительности программно-конфигурируемой инфраструктуры в сетях VANET на базе моделей гибридных устройств передачи данных / М. В. Ушакова, Ю. А. Ушаков, И. П. Болодурина, А. Л. Коннов // Современные информационные технологии и ИТ-образование. 2020. Т. 16, № 3. С. 630–640.*

*5. **Развитие** сетевой операционной системы NOX для создания распределенного центра обработки данных с программно-конфигурируемыми сетями его сегментов / П. Н. Полежаев, Ю. А. Ушаков, М. В. Ушакова, А. Е. Шухман // Информационные технологии моделирования и управления, 2013. №2. С. 177–185.*

*6. **Ушаков Ю. А., Ушакова М. В., Коннов А. Л.** Маршрутизация в распределенной программно-конфигурируемой сети на основе сервис-ориентированного подхода // Научно-технический вестник Поволжья, 2020. № 1. С. 91–96.*

*7. **Service-oriented** routing in distributed self-organizing software-defined networks / Y. A. Ushakov, M. V. Ushakova, A. E. Shukhman [et al.] // IEEE 12th International Conference on Application of Information and Communication Technologies (AICT). 2018. С. 1–5.*

*8. **Wong M. D., Varma A., Sivaraman A.** Testing Compilers for Programmable Switches Through Switch Hardware Simulation // arXiv preprint arXiv:2005.02310. 2020.*

*9. **Implementation** of network traffic monitor system with SDN / Y. Y. Yang, C. T. Yang, S. T. Chen [et al.] // IEEE 39th annual computer software and applications conference. 2015. Т. 3. С. 631–634.*

V. V. Dmytrenko, O. M. Marchuk

SIMULATION AND ANALYSIS MODELS OF HYBRID SOFTWARE-CONFIGURED DEVICES

The analysis of the algorithm, based on which the simulation and analytical model of the hybrid software-configurable device is automatically synthesized, was carried out. The solution for modeling OpenFlow equipment was also considered, and an original OpenFlow switch model was developed, which is distinguished by the ability to manage delays based on the results of its mathematical model. The operation scheme of a hybrid switch that supports both traditional switching and OpenFlow switching has been developed. This scheme allows solving complex tasks in the physical infrastructure of virtualized networks. A module has been developed to create a realistic delay and limit the capacity and performance of the hybrid switch model, which can work according to the data obtained as a result of the experimental study. Additional traffic processing schemes have been developed for a more detailed implementation of the hybrid software-configurable equipment model. The basic model of the switch presented in the INET OMNeT++ framework is considered. The switch is represented by a set of modules interacting through the OMNeT transmission system. The most detailed modules of Ethernet interfaces are implemented, in particular, there are implementations of the input queue of the Tail-Drop type, output queues of the FIFO, RED, WRED, PQ, CBFQ types, as well as the implementation of the transmission delay on the line. The virtualization technology of network functions, which is built on the basis of software-configured networks, has been studied, and the most popular Linux-based solutions are used as a software-controlled Open vSwitch switch. The considered methods of forming a model of such a switch. Open vSwitch is architecturally different from both classical and OpenFlow real switches, as there are no dedicated packet refactoring units (except for network cards with hardware header processing and FPGAs), dedicated memory and processors for switching, Open vSwitch uses a kernel module (or module DPDK for direct transmission without the participation of the kernel) to transmit packets based on information from the OVSDB database or OpenFlow tables.

Keywords: simulation and analytical model; OpenFlow; switching; traffic processing; virtualized networks.