Міністерство освіти і науки України

Тернопільський національний педагогічний університет

імені Володимира Гнатюка

Кафедра інформатики та

методики її викладання

ІНДЗ

Конфігурування RIP – маршрутизатора в ОС Linux

Виконала:

Студентка групи І-24

Козбур Марія Миколаївна

Керівник:

Олексюк Василь Петрович

Тернопіль 2013

ЗМІСТ

I. ВСТУП 3
II. ПОНЯТТЯ МАРШРУТИЗАЦІЇ. ДИНАМІЧНА МАРШРУТИЗАЦІЯ 4
1. Динамічна маршрутизація 5
2. Протоколи маршрутизації. 7
III. RIP - маршрутизація 9
IV. Все про Quagga. RIP – маршрутизація з Quagga 15
V. ВИСНОВКИ 17
VI. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 19
VII. ДОДАТКИ 20

I. ВСТУП

Актуальність теми і дослідження:

Протокол маршрутизації RIP (Routing Information Protocol) належить до алгоритму класу «distancevector» (алгоритм Белмана - Форда). Цей алгоритм один із перших алгоритмів маршрутизації, які було використано в інформаційно – обчислювальних мережах взагалі і у мережі Internet – зокрема. Але він досі надзвичайно поширений у обчислювальних мережах. Крім версії RIP для мереж TCP/IP, є також версія RIP для мереж IPX/SPX компанії Novell.

Цей протокол маршрутизації призначений для порівняно невеликих і відносно однорідних мереж. Протокол розроблений в університеті Каліфорнії (Берклі), виходить з розробках фірми Ксерокс і реалізують самі принципи, як і програма маршрутизації routed, яка у ОС UNIX (4BSD). Маршрут тут характеризується вектором відстані до місця призначення. Передбачається, кожен маршрутизатор є відправною точкою кількох маршрутів до мереж, із якими пов'язаний. З 1988 року RIP був повсюдно прийнято виробниками персональних комп'ютерів від використання у тому виробах передачі через мережу.

Рішення, знайдене за алгоритмом Белмана - Форда, не оптимальним, а близькими до оптимальному. Перевагою протоколу RIP є його обчислювальна простота і простота конфігурування, а недоліками – збільшення трафіку при періодичної розсилання широкомовних пакетів і не оптимальність знайденого маршруту.

У середовищі сучасних мережевих середовищах RIP – не найкраще рішення для вибору ролі протоколу маршрутизації, оскільки його можливості поступаються сучаснішим протоколів, таких як EIGRP,OSPF. Є обмеження на 15 хопів, який дає застосовувати їх у великих мережах.

Мета: навчитись застосовувати RIP-маршрутизацію за допомогою quagga на практиці.

II. ПОНЯТТЯ МАРШРУТИЗАЦІЇ. ДИНАМІЧНА МАРШРУТИЗАЦІЯ

Маршрутизація (англ. Routing) — процес визначення маршруту прямування інформації між мережами. Маршрутизатор (або роутер від англ. router) приймає рішення, що базується на IP-адресі дотримувача пакету. Для того, щоб переслати пакет далі, всі пристрої на шляху слідування використовують IP-адресу утримувача. Для прийняття правильного рішення маршрутизатор має знати напрямки і маршрути до віддалених мереж. Є два типи маршрутизації:

1. Статична маршрутизація — маршрути задаються вручну адміністратором.
2. Динамічна маршрутизація — маршрути обчислюються автоматично за допомогою протоколів динамічної маршрутизації — RIP, OSPF, EIGRP, IS-IS, BGP, HSRP та ін., які отримують інформацію про топологію і стан каналів зв'язку від інших маршрутизаторів у мережі.

Оскільки статичні маршрути конфігуруються вручну, будь-які зміни мережної топології вимагають участі адміністратора для додавання і видалення статичних маршрутів відповідно до змін. У великих мережах підтримка таблиць маршрутизації вручну може вимагати величезних витрат часу адміністратора. У невеликих мережах це робити легше. Статична маршрутизація не має можливості масштабування, яку має динамічна маршрутизація через додаткові вимоги до налаштування і втручання адміністратора. Але і у великих мережах часто конфігуруються статичні маршрути для спеціальних цілей у комбінації з протоколами динамічної маршрутизації, оскільки статична маршрутизація є стабільнішою і вимагає мінімум апаратних ресурсів маршрутизатора для обслуговування таблиці.

Динамічні маршрути виставляються іншим чином. Після того, як адміністратор активізував і налаштував динамічну маршрутизацію за одним з протоколів, інформація про маршрути оновлюється автоматично в процесі маршрутизації після кожного отримання з мережі нової інформації про

маршрути. Маршрутизатори обмінюються повідомленнями про зміни у топології мережі в процесі динамічної маршрутизації.

1. Динамічна маршрутизація

Найбільш ефективними, але і, мабуть, найскладнішими, є способи динамічної (адаптивної) маршрутизації. При динамічній маршрутизації вміст таблиць маршрутів змінюється залежно від стану і завантаження каналів передачі даних і вузлів комутації. Для адаптації до зміни навантаження кожному вузлу комутації надається певна інформація про стан мережі передачі даних і, насамперед, про її топологію, інтенсивність потоків даних і затримки (черги) у вузлах комутації. Ця інформація відстежується (збирається) спеціальними керуючими пакетами, якими обмінюються вузли комутації. Якість маршрутизації значною мірою залежить від оперативності відновлення керуючої інформації. Загалом, найбільш оптимальна маршрутизація досягається за наявності інформації про миттєвий стан мережі та її завантаження. Проте це найчастіше призводить до значного збільшення потоку керуючих пакетів у мережі передачі даних і до зниження її ефективності.

Динамічна маршрутизація є досить складним процесом, під час якого виконуються такі дії:

- формування маршрутів, яке здійснюється за допомогою алгоритмів маршрутизації шляхом упорядкування у кожному вузлі комутації таблиць маршрутів пакетів;
- реалізація маршрутів, тобто керування пакетами під час проходження їх підмережею зв’язку до місця призначення за допомогою спеціальних протоколів мережевого рівня;
- контроль стану мережі, у тому числі аналіз топології мережі, структури потоків і затримок у вузлах комутації;
- передача інформації про стан мережі, яку використовують для коригування таблиць маршрутів;
- коригування маршрутів.

Залежно від обраної стратегії коригування маршрутів розрізняють централізовану, розподілену, локальну і гібридну маршрутизації.

У разі централізованої адаптивної маршрутизації кожен вузол мережі готує і у певний момент передає менеджерові мережі інформацію про своє завантаження. Відповідно до цієї інформації менеджер складає глобальну картину стану мережі, за якою визначаються оптимальні маршрути проходження пакетів. Основним критерієм оптимальності маршруту є час затримки передачі пакетів. Обчисливши оптимальні маршрути, менеджер для кожного вузла комутації формує таблиці маршрутів, які потім розсилаються у відповідні вузли мережі передачі даних.

За способом збирання інформації про стан мережі і розсилання керуючих директив процес маршрутизації може бути синхронним або асинхронним. У першому випадку збирається інформація і надсилаються керуючі директиви через регулярні проміжки часу, а у другому — ця процедура здійснюється тільки при істотних змінах мережі передачі даних. Здебільшого за синхронного режиму обмін службовою інформацією є інтенсивнішим, а за асинхронного — необхідний постійний контроль за станом мережі. У будь-якому разі на менеджері мережі лежить основне навантаження щодо формування маршрутів, яке різко зростає зі збільшенням кількості вузлів мережі передачі даних. Як уже зазначалося, загальним недоліком централізованих методів маршрутизації є цілковита залежність функціонування мережі від її менеджера, що стає небезпечним зі збільшенням навантаження на нього. Крім цього, затримки, зумовлені обміном і обробкою великого обсягу керуючої інформації, знижують ефективність керування мережею, особливо у разі швидкої зміни потоків даних.

Цих недоліків позбавлені методи розподіленого керування маршрутизацією, які застосовуються у сучасних глобальних комп’ютерних мережах.

При розподіленій адаптивній маршрутизації кожен вузол комутації сам формує свою таблицю маршрутів, використовуючи для цього інформацію, яку він отримує від вузлів, що розміщуються на можливих шляхах до одержувача. Вузли обмінюються інформацією про свій стан, часові затримки і черги пакетів. Під час вибору маршрутів враховується час, потрібний для одержання

позитивних підтверджень на попередні пакети. Отже, будь-яке істотне відхилення від стабільного стану відразу ж передається суміжним вузлам для коригування їхніх таблиць маршрутів.

Одним із найпростіших варіантів розподіленої динамічної маршрутизації є локальна адаптивна маршрутизація, при якій вузол комутації практично сам вибирає маршрути передачі пакетів, не одержуючи інформації від інших вузлів. Таблиці маршрутів завантажуються централізованим способом заздалегідь. Надалі маршрут вибирається згідно з відомостями про довжину вихідних черг топологією мережі передачі даних. Пакет направляється найкоротшим шляхом з урахуванням мінімальної довжини вихідної черги.

Локальна адаптивна маршрутизація забезпечує високу гнучкість роботи мережі передачі даних, швидкий і ефективний спосіб вирішення проблеми обходу несправних або перевантажених вузлів. Водночас вона характеризується складністю програми формування та обробки таблиць маршрутів, можливістю «автоколивання» і втрати пакетів під час зміни таблиць маршрутів.

Ефективнішим методом маршрутизації можна вважати гібридну маршрутизацію, яка поєднує позитивні властивості локальної і централізованої маршрутизацій. Прикладом є «дельта-маршрутизація», за якої менеджер з деяким запізненням стежить за глобальною ситуацією у мережі, тоді як вузлам надана певна свобода дій з тим, щоб вони могли швидко реагувати на локальні коливання навантаження мережі та зміни стану її окремих компонентів.

Різноманітність у способах маршрутизації пояснюється відсутністю універсального способу, оптимального для різних прикладних програм і характеристик мережі (рівня потоку даних, надійності передачі, часу встановлення наскрізного (через мережу) з’єднання, швидкості передачі блоків даних тощо).

2. Протоколи маршрутизації.

Протокол маршрутизації — це мережевий протокол, використовуваний маршрутизаторами для визначення можливих маршрутів дотримання даних в

складеній комп'ютерній мережі. Вживання протоколу маршрутизації дозволяє уникнути ручного введення всіх допустимих маршрутів, що, у свою чергу, знижує кількість помилок забезпечує узгодженість дій всіх маршрутизаторів в мережі і полегшує працю адміністраторів.

III. RIP - маршрутизація

Алгоритм маршрутизації протоколу RIP (Roster Image Processor, Процесор растрових образів) належить до класу дистанційно-векторних алгоритмів. Цей клас алгоритмів також відомий на ім'я автора алгоритму Форда - Фулкерсона (Ford-Fulkerson). Крім того, для цього класу також використовується назва "алгоритми Беллмана - Форда" (Bellman-Ford), яке з'явилося після остаточної формалізації алгоритму, яка була зроблена на основі основного рівняння динамічного програмування Беллмана.

Протокол RIP був спочатку розроблений для Універсального протоколу PARC Xerox (де він називався GW1NFO) і використовувався в комплекті протоколів XNS (Xerox Network Systems). У 1982 г. RIP почали зв'язувати як з UNIX, так і з TCP/IP. У тому ж році версію UNIX, звану Berkeley Standard Distribution (BSD), почали постачати з однією з реалізації RIP. З 1988 року RIP був повсюдно прийнятий виробниками персональних комп'ютерів для використання в їх виробах передачі даних по мережі.

У такій гігантській міжнародній мережі як Internet, яка являє собою об'єднання багатьох автономних систем, кожна з яких адмініструється і справляється самостійно, неможливо використати для управління маршрутизацією по всій мережі який-небудь один протокол. Кожна автономна система використовує для маршрутизації всередині своєї області який-небудь один протокол, який і називається (або краще сказати відноситься до класу) протоколом IGP Interior Gateway Protocol. RIP спроектований і використовується саме як протокол цієї категорії.

Дистанційно-векторний алгоритм маршрутизації R1P, як вже відмічалося вище, побудований на основі механізму обміну невеликими блоками інформації таблиць маршрутизації між сусідніми маршрутизаторами мережі. Таким чином, кожний шлюз або хост, що бере участь в роботі протоколу маршрутизації, зберігає у себе інформацію про всіх членів мережі у вигляді бази даних маршрутизації (таблиці).

Кожний запис в базі даних маршрутизації складається з:

1. Адреси шлюзу, куди повинна прямувати дейтаграмма для даного одержувача
2. Поля "metric", вказуючого відстань до одержувача (воно може виражатися, наприклад, у вартості лінії зв'язку, часу передачі інформації по лінії до
3. Адреси мережі або хоста відправника
4. Адреси інтерфейсу фізичного з'єднання
5. Різних тимчасових міток та мітки останньої зміни даного запису

Дистанційно-векторні алгоритми побудовані на основі теоретично доведеного положення, що завжди можливо обчислити оптимальний шлях передачі дейтаграми тільки на основі інформації, що міститься а таблицях маршрутизації.

У цьому випадку, оптимальний шлях це шлях з найменшою "довжиною" ("metric"), яка може визначатися, виходячи з вимог алгоритму. У невеликих мережах "metric" визначають або як кількість шлюзів (переправ), які необхідно подолати по шляху до одержувача, або як сумарний час затримки пересилки дейтаграми, вартість даного каналу зв'язку і т. п. В більш складних мережах це може бути комбінацією декількох параметрів з різними коефіцієнтам у. В будь-якому випадку, оптимальний шлях це один з шляхів з мінімальним параметром "metric1".

Далі представлена схема роботи найпростішого дистанційно-векторного алгоритму. Дана процедура повинна виконуватися на всіх об'єктах (як на всіх шлюзах, так і на хостах), працюючих з протоколом маршрутизації.

Отже, маршрутизатор повинен:

1. Зберігати таблицю маршрутизації із записами кожного потенційного одержувача дейтаграмми в системі. Запис повинен містити відстань до об'єкта (D) і адресу першого шлюзу (G) на шляху до цього об'єкта.
2. Періодично відправляти інформаційні повідомлення, що містять всю інформацію своєї таблиці, кожному з своїх сусідів по мережі (об'єктам, що знаходяться в області прямої видимості).

3. При отриманні інформаційного повідомлення від сусіда G (яке містить його таблицю маршрутизації) вважати з отриманого повідомлення метрики об'єктів мережі і додати до них метрику мережі до сусіда G (по цій мережі поступило дане повідомлення). Порівняти результати з результатами власної таблиці маршрутизації. Якщо яка-небудь метрика D до об'єкта N менше існуючої метрики до об'єкта N у власній таблиці, змінити запис у власній базі даних маршрутизації для цього об'єкта (метрику на D, а шлюз на G).

Описаний вище алгоритм має на увазі, що топологія мережі не міняється. Однак в реальних мережах шлюзи і лінії передачі дуже часто як виходять з ладу, так і відновлюються, т. с. відбувається зміна топології мережі. Тоді, якщо, наприклад, один з маршрутизаторів, що здійснює оптимальну маршрутизацію, вийшов з ладу, приведений вище алгоритм буде нс спроможний змінити шлях маршрутизації дейтаграм на іншій. Дня розв'язання цієї проблеми протоколи дистанційно-векторної маршрутизації повинні вжити заходів по синхронізації встановлених маршрутів.

Наприклад, працюючи з RIP-протоколом, кожний шлюз, що бере участь в маршрутизації, відправляє свою таблицю маршрутизації всім своїм сусідам кожних 30 секунд. Передбачимо тепер, що маршрут в мережу N лежить через шлюз G. Тоді, якщо ми не отримуємо від шлюзу G ніяких пакетів протягом 180 секунд, ми вважаємо, що шлюз або мережа вийшли з ладу і позначаємо даний запис в таблиці маршрутизації як невірну. Тепер, допустимо, що ми отримуємо від іншого сусіда по мережі інформацію про маршрут до мережі N, тоді ми замінюємо старий невірний запис на нову.

Описаний алгоритм дозволяє хостом і маршрутизаторам обчислювати правильний шлях. Однак цього не досить в реальних умовах роботи мережі. Описаний алгоритм дозволяє Зупинити функціональність системи за певний час. Однак цей алгоритм не гарантує, що проміжок часу в реальних умовах робота мережі буде досить малий, що може привести (і приводило б - при відсутності додаткових механізмів забезпечення надійності) до петель або порушень роботи всього механізму.

Перш ніж детальніше розбирати механізми протоколу RIP, які дозволяють йому надійно функціонувати в умовах реальних мереж з топологією, що швидко міняється, необхідно зазначити, що протокол RIP має характеристики, які, з одного боку, все ж обмежують область його застосування, а з іншою, роблять його більш детермінований і більш стійким протоколом:

1. Протокол не може використовуватися в мережах, де кількість пересилок (проміжних шлюзів на шляху Дейтаграмми) перевищує 15. Розробники протоколу вважають, що в мережах такого масштабу RIP буде працювати найбільш ефективно. Якщо ваша мережа настільки розрослася, що їй необхідні маршрути з великою кількістю пересилок, вам потрібно вибрати більш могутній протокол маршрутизації або розбити мережу на більш дрібні самостійні сегменти.
2. RIP-протокол має обмеження на час відновлення шляхів маршрутизації. Тому, якщо ваша мережа складається з декількох сотень підмереж і цикл оновлення маршрутів торкається всіх цих підмереж, то вам необхідно або збільшити пропускну спроможність каналів, або збільшити обмеження на час відновлення шляхів маршрутизації. Інакше у вас можуть виникнути петлі або зациклення маршрутів (це пов'язано з феноменом, званим "рахунком до нескінченності"
3. Rip-протокол використовує статичні "метрики" для порівняння різних маршрутів. Це незручне в тих випадках, коли маршрут необхідно вибрати на основі тимчасових характеристик, наприклад, завантаження каналу, надійності, затримки. Якщо вам необхідно оцінювати маршрути саме за цими параметрах, то RIP стає непридатним для роботи у вашій мережі.

RIP — так званий дистанційно-векторний протокол, який оперує хопами як метрикою маршрутизації. Кожен RIP-маршрутизатор за замовчуванням сповіщає в мережу свою повну таблицю маршрутизації раз на 30 секунд, генеруючи досить багато трафіку на низько-швидкісних лініях зв'язку. RIP працює на прикладному рівні сітки TCP/IP, використовуючи UDP порт 520.

У сучасних мережних середовищах RIP — **не найкраще рішення** для вибору в якості протоколу маршрутизації, тому що його можливості поступаються сучаснішим протоколам, таким як EIGRP, OSPF. Обмеження в 15 хопів не дає застосовувати його у великих мережах. Перевага цього протоколу — простота конфігурування. Внаслідок простоти його підтримують практично всі маршрутизатори початкового рівня.

Для порівняння маршрутів протокол RIP використовує досить просту В «МетрикуВ» - число переходів. Однак використання даного критерію в цілому ряді випадків не може забезпечити оптимальний вибір маршруту.

Специфікації

В· RFC -1388. Протокол RIP-2 (1993 рік) є новою версією RIP, яка на додаток до широкомовний режим підтримує мультикастинг.

В· RFC -1582. Розширення до RIP за вимогами до хостом до підтримки певних параметрів.

В· RFC -1721. Аналіз протоколу RIP версії 2.

В· RFC -1722 (STD 0057). Протокол RIP версії 2, припис до застосуванню.

В· RFC -1724. Протокол RIP версії 2, розширення по MIB (база керуючої інформації - management information base).

В· RFC -2080. Специфікації протоколу RIP для IPv6.

В· RFC -2082. Протокол RIP версії 2, проблеми аутентифікації з використанням MD5 (Message Digest 5) - 128-бітний алгоритмом хешування, розроблений в 1991 році. MD5 призначений для створення В «ВідбитківВ» або В «дайджестівВ» повідомлень довільної довжини.

В· RFC -2092. Специфікація для автоматично запускається протоколу RIP (triggered RIP).

В· RFC -2453 (STD 0056). Загальний опис протоколу другої версії.

Алгоритми вектору відстані основані на обміні малої кількості інформації . Кожний об'єкт (шлюз або хост), що приймає доля в маршрутизації, має тримати інформацію про ВСІ комп'ютери Системі . Кожний запис у таблиці маршрутизації включає наступний шлюз, на який дані, напрямлені до об'єкту, мають бути відправлені. Відстань - це узагальнена характеристика, що може

відображати швидкість передачі даних, копійчаний вартість передачі тощо. Алгоритми вектору відстані дістали свою назву від того, що можуть обчислити оптимальний маршрут коли змінюється список відстаней. Крім того, має місце обмін маршрутизаційною інформацією між безпосередньо зв'язаними об'єктами, тобто елементами спільної мережі . Записи в таблиці маршрутизації має місце так інформація про комп'ютер - отримувач :

" Адреси : в IP реалізації це має буті IP адреси хост або мережі ;

" Шлюз : перший шлюз на цьому маршруті ;

" Інтерфейс : інтерфейс, що має бути використати, щоб досягти першого шлюзу;

IV. Все про Quagga. RIP – маршрутизація з Quagga

Quagga - розширений пакет програм маршрутизації, який забезпечує реалізацію протоколів маршрутизації, заснованих на TCP / IP. Цей документ - посібник для quagga-0.96. Quagga - otvetvlenie GNU Zebra.

Quagga - пакет програм, що реалізують протоколи маршрутизації, заснованих на TCP / IP і підтримує такі протоколи як RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, BGP-4 і BGP-4. Quagga також підтримує BGP Route Reflector і Route Server поведінки. На додаток до протоколу IPv4 tradicionnomu Quagga також підтримує протоколи маршрутизації для IPv6. Спільно з демоном SNMP, який підтримує SMUX протокол, Quagga забезпечує протокол маршрутизації MIBs Quagga використовує розширену програмну архітектуру для того, щоб надати Вам якісний механізм маршрутизації. Quagga має інтерактивний інтерфейс користувача для кожного протоколу маршрутизації і підтримує загальні клієнтські команди. Ви можете використовувати бібліотеку Quagga для написання власних програм. Zebra підпадає під Public License GNU General.

Сьогодні мережі TCP / IP повсюдно поширені. Internet проник в країни, компанії і вдома. Коли Ви підключаєтеся до Internet, то інформація проходить через незліченні натовпи безвісних маршрутизаторів. Quagga дозволяє здійснити функції маршрутизації на окремо взятому комп'ютері, обмінюючись маршрутною інформацією з сусідами і відповідним чином змінюючи таблицю маршрутів ядра. Ви маєте можливість динамічно змінювати конфігурацію Quagga і переглядати інформацію про поточну конфігурації. На додаток до реалізації протоколів маршрутизації, Quagga може додавати статичні маршрути, адреси інтерфейсів і специфічні прапори інтерфейсів. Конфігурування Quagga у разі малої мережі, мережі xDSL мережі не відрізняється складністю. Єдина річ, яку Ви повинні зробити, це налаштувати адреси мережевих інтерфейсів, вказати кілька статичних маршрутів і налаштувати маршрут по замовчуванням. Якщо мережа є досить великий, або якщо мережева структура досить часто змінюється, можливо Ви

захочете використовувати протоколи динамічної маршрутизації типу RIP, OSPF або BGP.

Традиційно настройка маршрутизації в UNIX здійснюється за допомогою команд ifconfig та route. Стан таблиці маршрутизації відображається утилітою netstat. У більшості випадків використання цих утиліт вимагає повноважень користувача root. Quagga має зовсім інший принцип управління. Існують два різних режими - normal mode і enable mode. У режимі normal можливий лише перегляд поточної конфігурації, в режимі enable можлива зміна конфігурації. Ці режими, не залежні від облікових записів UNIX, будуть великою підмогою адміністраторам в плані збільшення безпеки. В даний час Quagga підтримує загальні unicast протоколи Маршрутизації. Протоколи багато - адресної передачі , типу BGMP, PIM - SM, PIM - DM CAN бути будуть підтримуватися в Quagga 2.0. Наступає підтримка MPLS, фільтрація TCP / IP, підтримка QoS. Традиційне програмне забезпечення маршрутизації зроблено як одна програма, яка забезпечує всі функціональні можливості протоколу маршрутизації. Quagga використовує дещо інший підхід. Кожен протокол маршрутизації обслуговується окремим демоном, з подальшим формуванням таблиць маршрутизації. Одночасно працювати можуть дещо різні демони в співтоваристві з керуючим демоном зебри. ripd демон обробляє протокол RIP, в той час як ospfd - демон, який підтримує OSPF v 2. Bgpd підтримує протокол BGP - 4. Демон Zebra служить для формування таблиці маршрутизації і перерозподілу маршрутів між різними протоколами. Зовсім необов'язково, щоб демони виконувалися на одній і машині. Ви можете навіть запустити кілька однакових демонів на одній машині, що відкриває великі перспективи.

Zebra - це програма управління процесом маршрутизації. Вона забезпечує оновлення таблиці маршрутизації ядра, пошук інтерфейсу і розподіл маршрутів між різними демонами маршрутизації.

V. ВИСНОВКИ

Використання динамічної маршрутизації доречно в середніх і великих мережах з розгалуженою і неоднорідною топологією. Певну підтримку тут надають стандартні демони або спеціально розроблені утиліти, за допомогою яких можливо створювати досить складні конфігурації маршрутів. При цьому залежно від поставленого завдання необхідно враховувати переваги і недоліки кожного з протоколів динамічної маршрутизації, обраних для найбільш оптимального рішення.

GNU Quagga - це вільне ПЗ , керуюче протоколами маршрутизації на основі TCP / IP. Підтримуються BGP4 , BGP4 + , OSPFv2 , OSPFv3 , IS -IS , RIPv1 , RIPv2 і RIPng , а також їх різновиди для IPv6.

Quagga - це вдосконалена версія GNU Zebra , комп'ютерної програми , розвиток якої зупинилося в 2005 році. Після того як Zebra був покинутий , колишні учасники списку розсилки Zebra продовжили розробку проекту , раніше відомого як Zebra - pj , і заснували проект Quagga .

Quagga використовує багато-поточність , якщо вона підтримується ядром , але може працювати і з ядрами , які не підтримують багато-поточність . Кожен протокол обслуговується власним процесом .

Quagga - щось значно більше , ніж заміна routed . Воно може використовуватися як Route Server і Route Reflector (сервера і відбивача маршрутів) .

Комп'ютер з встановленим та сконфігурованим пакетом Quagga стає здатний використовувати будь-які з нижченаведених протоколів динамічної маршрутизації :

* Routing Information Protocol (RIP) : v1 , v2 , v3 ;

* Open Shortest Path First (OSPF) : v2 , v3 ;

* Border Gateway Protocol (BGP) : v4.

Пакет Quagga може бути встановлений на UNIX- подібні операційні системи. Quagga складається з базового ядра (core daemon) Zebra , що виконує

роль проміжного рівня абстракції (abstraction layer) ядра ОС , і надає Zserv API клієнтам по протоколу TCP . Клієнтами Zserv виступають служби (демони) :

* Ospfd (протокол OSPFv2) ;

* Ripd (протокол RIP v1 , V2) ;

* Ospf6d (протокол OSPFv3 IPv6) ;

* Ripngd (протокол RIP ng IPv6) ;

* Bgpd (протокол BGPv4 + , включаючи підтримку multicast і IPv6)) .

Бібліотека Quagga істотно полегшує розробку додаткових модулів , дозволяючи всім її службам використовувати уніфікований спосіб конфігурації і управління.

VI. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Маршрутизация с Quagga : [Електронний ресурс]. – Режим доступу до документа: http://www.tux.in.ua/articles/2182
2. root@opentodo#: Configuring routing protocols with Quagga: [Електронний ресурс]. – Режим доступу до документа: http://opentodo.net/2012/08/configuring-routing-protocols-with-quagga/
3. Внутренний протокол маршрутизации RIP: [Електронний ресурс]. – Режим доступу до документа: http://book.itep.ru/4/44/rip44111.htm
4. IP-маршрутизация: RIP и OSPF : [Електронний ресурс]. – Режим доступу до документа: http://system-administrators.info/?p=288
5. Quagga Routing Suite: [Електронний ресурс]. – Режим доступу до документа: http://www.nongnu.org/quagga/
6. Динамическая маршрутизация в Linux: [Електронний ресурс]. – Режим доступу до документа: http://www.ibm.com/developerworks/ru/library/l-dynamic_routing/

VII. ДОДАТКИ. ПРАКТИЧНА ЧАСТИНА

Перевіряємо скільки мережевих карт на комп'ютері. Якщо 2 то це зовнішній, якщо одна то це внутрішній. Отже цей в мене зовнішній:

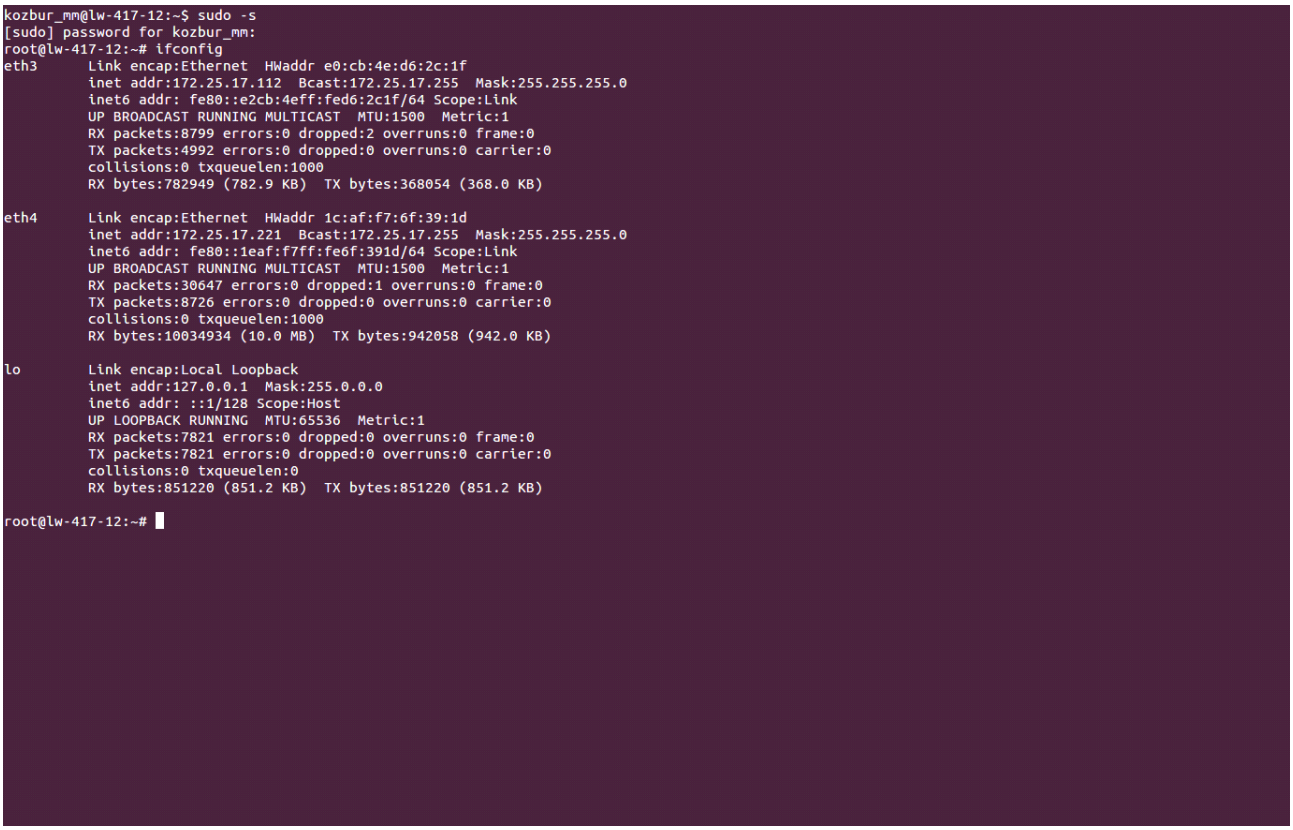

Рис. 1

Далі перевіряємо чи є з'єднання з комп'ютером підєднаним через мережевий кабель.

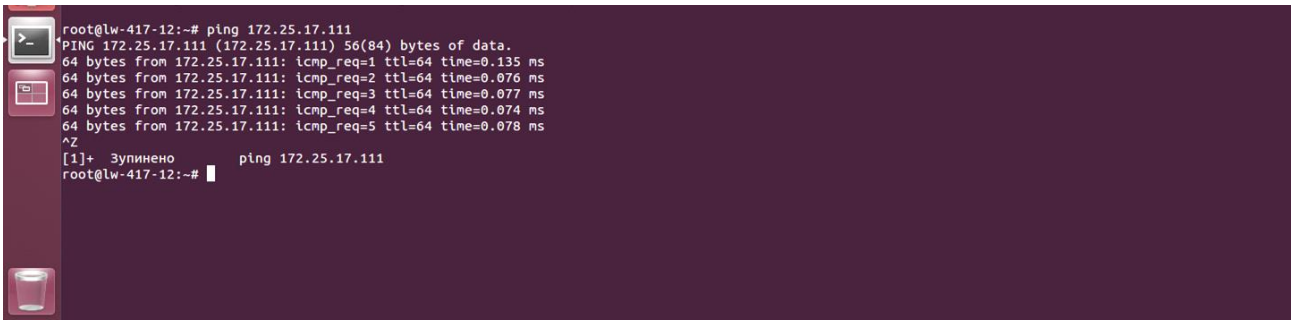

Рис. 2

Щоб встановити Quagga в Ubuntu, достатньо ввести: **sudo apt-get install** quagga quagga-doc

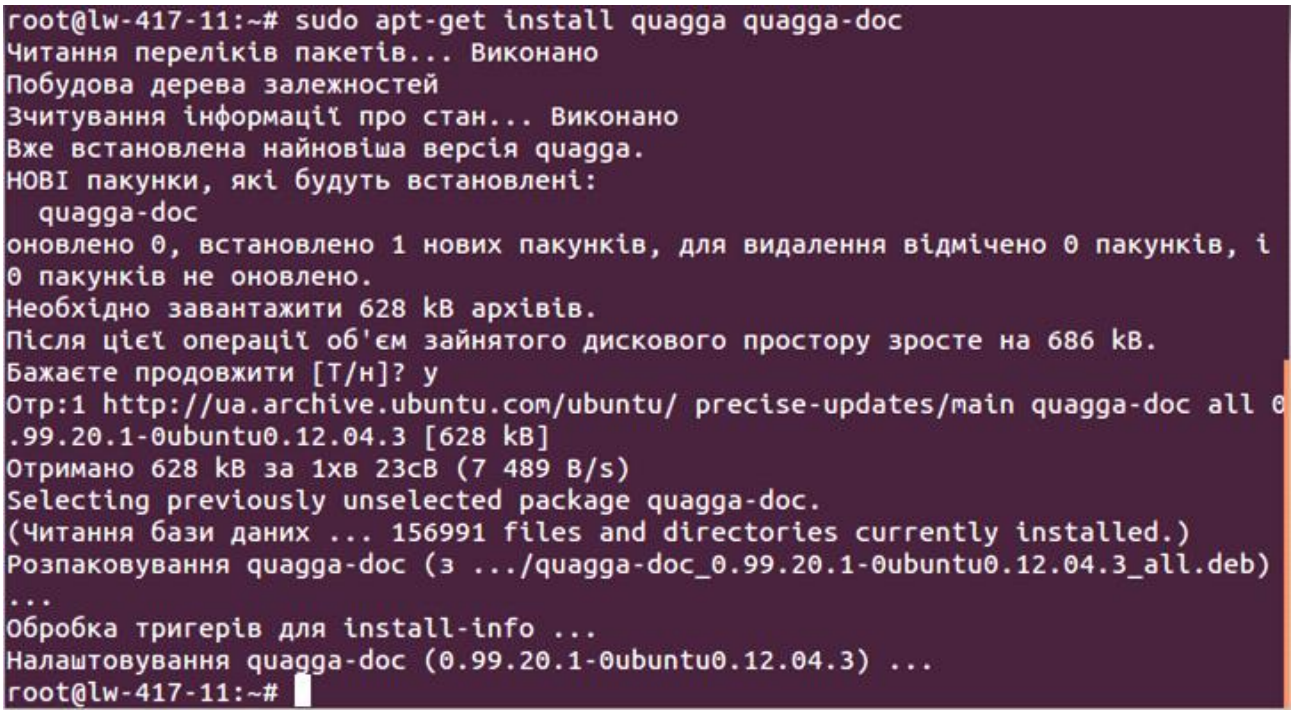

Рис. 3

Після виконання команди в каталозі / etc / quagga з'явиться два конфігураційних файлу. Тут потрібно відзначити , що особливістю Quagga є запуск окремого демона на кожен підтримуваний протокол , базове управління здійснює демон zebra (core daemon) , який представляє необхідні API іншим демонам і перебудовує таблицю маршрутизації. Статичні маршрути встановлюються також за допомогою демона zebra . Така схема дозволяє при необхідності легко додати підтримку іншого протоколу , але наявність декількох процесів і конфігураційних файлів робить її менш зручною для управління.

У файлі / etc / quagga / daemons якраз вказується, які демони будуть завантажуватися , тобто протоколи буде підтримувати Quagga . За замовчуванням всі параметри встановлені в no , і після установки нічого працювати не буде.

Рис. 4

Далі активуємо підтримку BGP і OSPF: **sudo nano** /etc/quagga/daemons

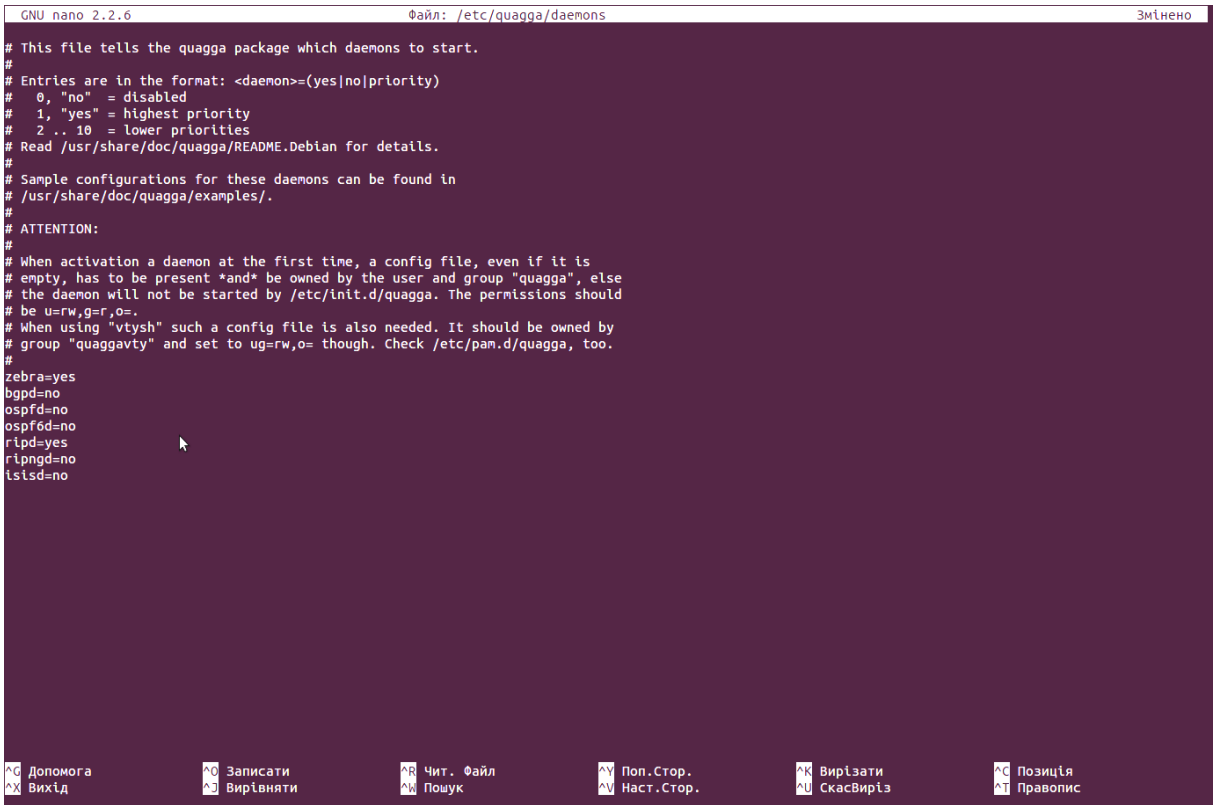

Рис. 5

Список TCP портів, на яких слухають підключення демони, можна дізнатися, переглянувши / etc / services: **grep** zebra /etc/services

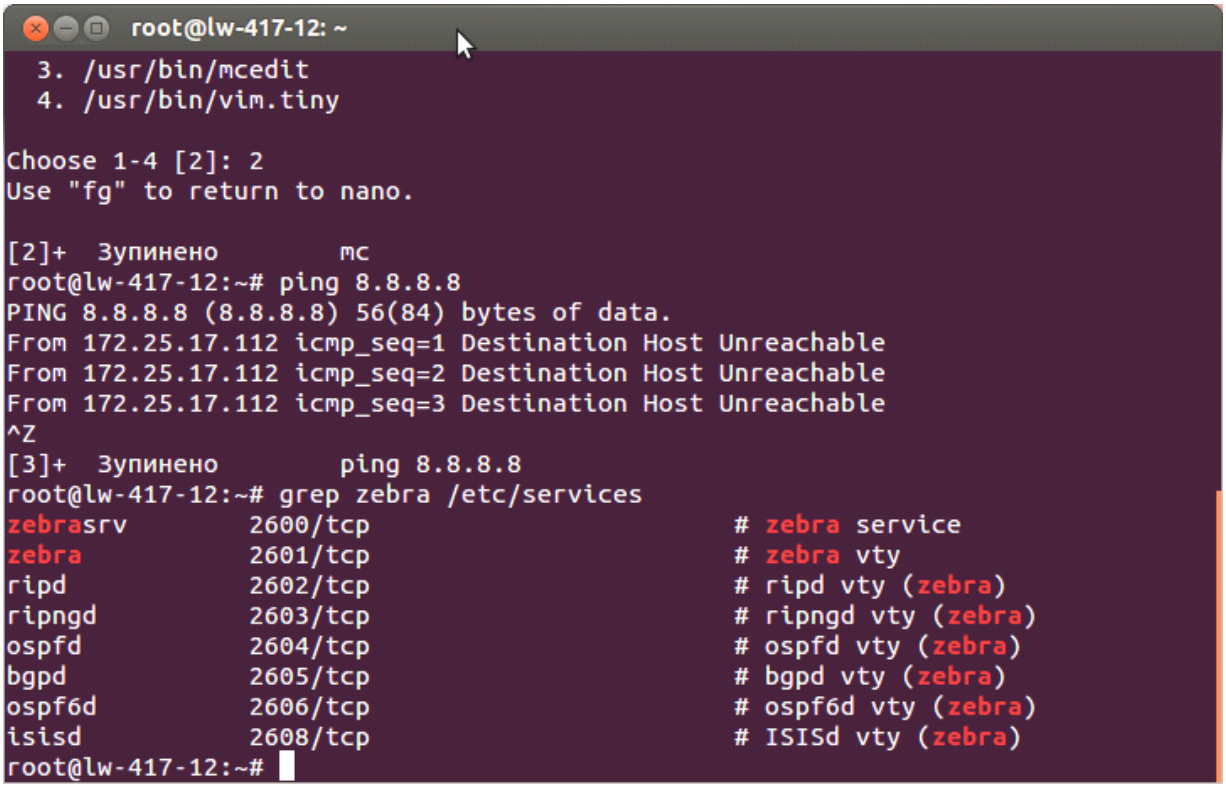

Рис. 6.1

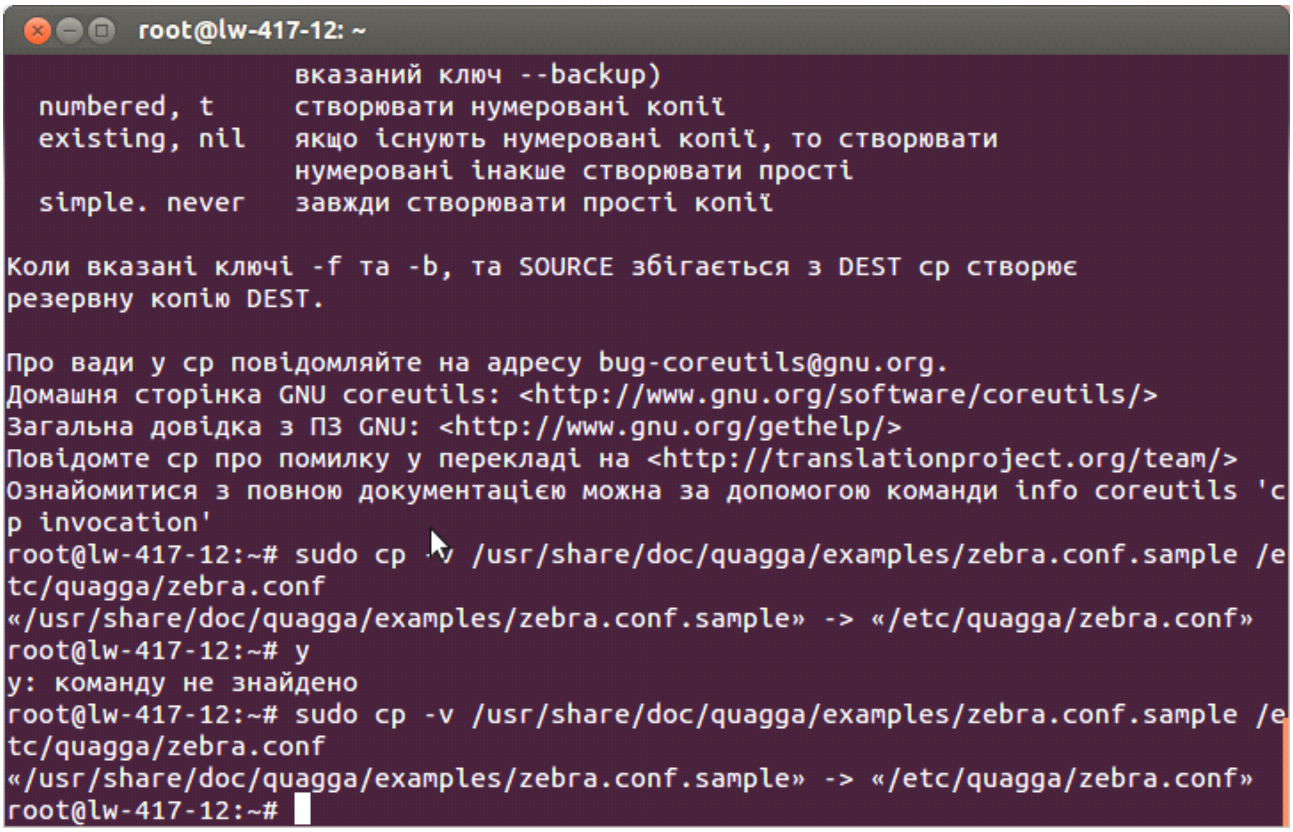

Рис. 6.2

У Quagga реалізовано два способи подачі робочих параметрів демонам - за допомогою конфігураційних файлів і безпосередньо в термінальному режимі (terminal mode) , підключившись до потрібного порту . Зазвичай використовують обидва варіанти , завантажуючи початкові параметри за допомогою файлів , а при необхідності налаштування коригуються на льоту . При цьому установки в terminal mode також зберігаються в конфігураційні файли , тому при перезавантаженні таблиця маршрутизації повністю відновлюється. Шаблони файлів налаштувань демонів знаходяться в каталозі / usr / share / doc / quagga / examples , перейменовуємо і копіюємо їх в / etc / quagga :

```
sudo cp -v
/usr/share/doc/quagga/examples/zebra.conf.sample
/etc/quagga/zebra.conf

sudo cp -v
/usr/share/doc/quagga/examples/bgpd.conf.sample
/etc/quagga/bgpd.conf
```

```
sudo cp -v
/usr/share/doc/quagga/examples/ospfd.conf.sample
/etc/quagga/ospfd.conf
sudo cp -v
/usr/share/doc/quagga/examples/vtysh.conf.sample
/etc/quagga/vtysh.conf
```

```
root@lw-417-12:~# sudo cp -v /usr/share/doc/quagga/examples/zebra.conf.sample /e
tc/quagga/zebra.conf
«/usr/share/doc/quagga/examples/zebra.conf.sample» -> «/etc/quagga/zebra.conf»
root@lw-417-12:~# y
y: команду не знайдено
root@lw-417-12:~# sudo cp -v /usr/share/doc/quagga/examples/zebra.conf.sample /e
tc/quagga/zebra.conf
«/usr/share/doc/quagga/examples/zebra.conf.sample» -> «/etc/quagga/zebra.conf»
root@lw-417-12:~# sudo cp -v /usr/share/doc/quagga/examples/zebra.conf.sample /e
tc/quagga/bgpd.conf
«/usr/share/doc/quagga/examples/zebra.conf.sample» -> «/etc/quagga/bgpd.conf»
root@lw-417-12:~# sudo cp -v /usr/share/doc/quagga/examples/zebra.conf.sample /e
tc/quagga/ospfd.conf
«/usr/share/doc/quagga/examples/zebra.conf.sample» -> «/etc/quagga/ospfd.conf»
root@lw-417-12:~# sudo cp -v /usr/share/doc/quagga/examples/zebra.conf.sample /e
tc/quagga/vtysh.conf
«/usr/share/doc/quagga/examples/zebra.conf.sample» -> «/etc/quagga/vtysh.conf»
root@lw-417-12:~#
```

Рис. 7

Також під час завантаження скрипту / etc / init.d / quagga встановлює налаштування роботи демонів з файлу / etc / quagga / debian.conf. Типово для кожного демона вказано режим роботи "-daemon" і IP-адресу на якому він бере підключення "-A 127.0.0.1". Додатково пропишемо параметри "-keep_kernel" і "-retain", що дозволить автоматично зберігати старі і нові маршрути, плюс вкажемо обліковий запис quagga, від імені якої будуть виконуватися демони:

```
root@lw-417-12:~# vtysh_enable=yes
root@lw-417-12:~# zebra_options=" --deamon -A 127.0.0.1 -u quagga --keep_kernel
--retain"
root@lw-417-12:~# bgpd_options=" --deamon -A 127.0.0.1 -u quagga --retain"
root@lw-417-12:~# ospfd_options=" --deamon -A 127.0.0.1 -u quagga"
root@lw-417-12:~#
```

Рис. 8

При установці за допомогою пакетів в системі вже створюється потрібна обліковий запис:

```
root@lw-417-12:~# grep quagga /etc/passwd
quagga:x:124:131:Quagga routing suite,,,:/var/run/quagga/:/bin/false
root@lw-417-12:~#
```

Рис. 9

Але я створюю вручну. Обов'язково змінюючи власників конфігураційних файлів:

```
sudo chown quagga:quagga /etc/quagga/zebra.conf
sudo chown quagga:quagga /etc/quagga/ospfd.conf
sudo chown quagga:quagga /etc/quagga/bgpd.conf
```

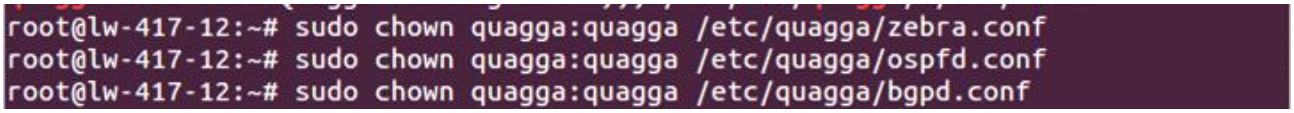

Рис. 10

Запускаю Quagga. `sudo /etc/init.d/quagga restart`
`Starting Quagga daemons (prio:10): zebra bgpd ospfd.`

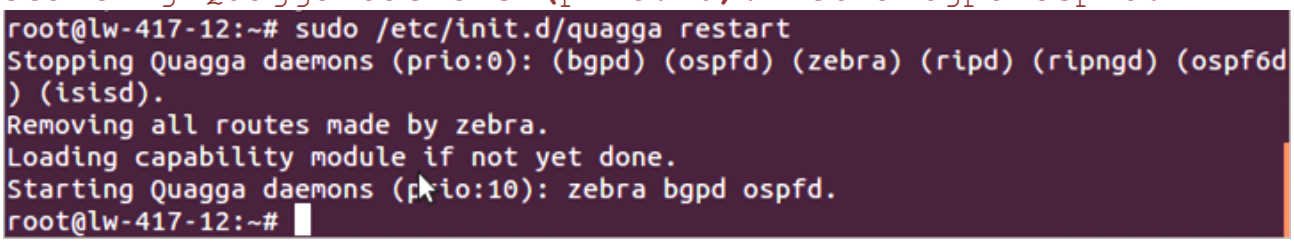

Рис. 11

Конфігураційні файли Quagga містять набір команд, які виконує демон після запуску. У більшій частині вони з синтаксису збігаються з вводиться вручну в terminal mode. Коментарі вказуються за допомогою оклику "!" Знака або решітки "#". Для початку подивимося zebra.conf:

```
sudo nano /etc/quagga/zebra.conf
```

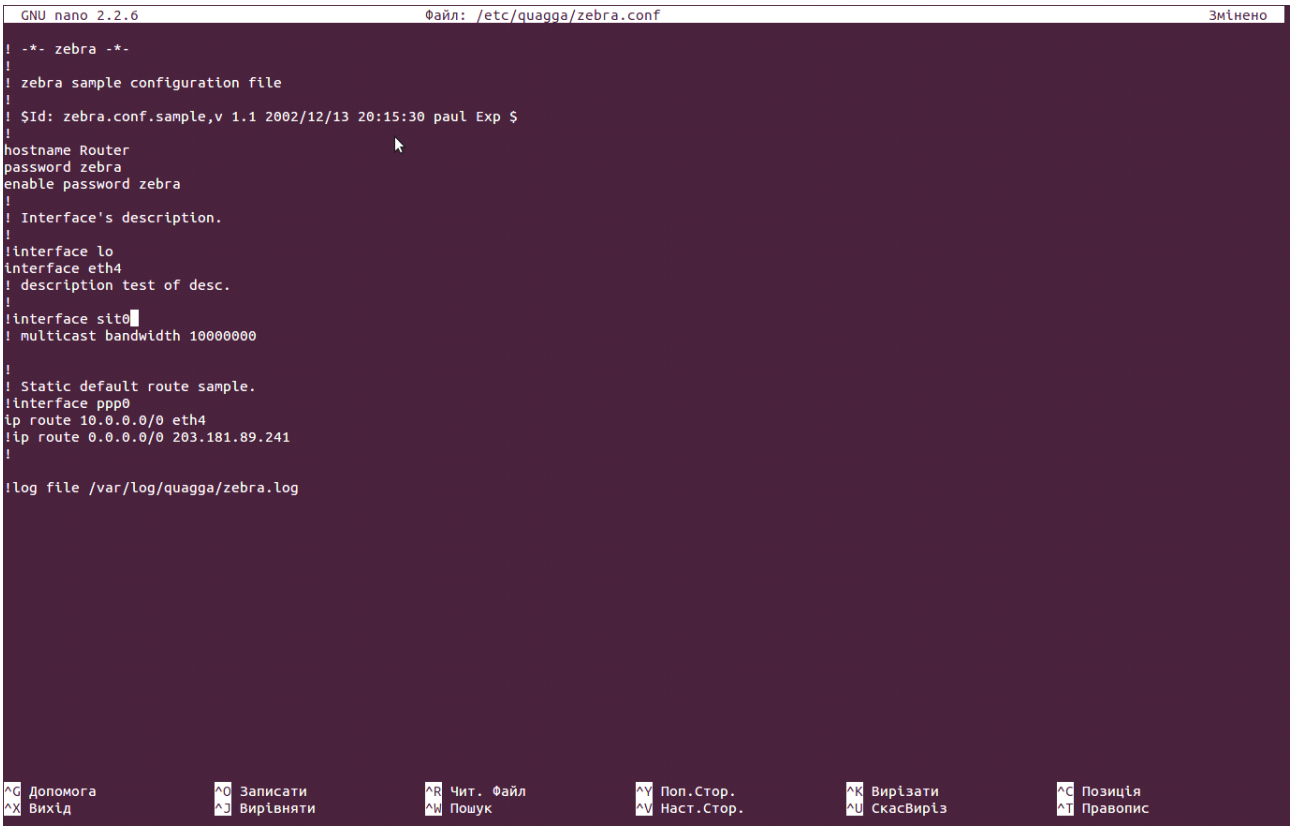

Рис. 12

Перевіряємо наявність каталогу / var / log / quagga і права, його власником повинні бути користувач і група - quagga: quagga. Типово у файлах bgpd.conf і ospfd.conf знятий коментар лише зі рядків установки пароля.

Демони Quagga підтримують управління за допомогою Cisco подібного синтаксису. Всі команди, введені в консолі, зберігаються в конфігураційний файл (автоматично і по команді). Підключитися до робочого демону можна за допомогою telnet або vtysh. При використанні консолі vtysh відразу потрапляємо в режим управління. **sudo** vtysh

Далі я оголошую сусідній комп'ютер, вказавши IP і номер AS

```
(config-router)# neighbor 10.0.0.5 remote-as 64513
(config-router)# neighbor 172.16.103.1 remote-as 65535
(config-router)# log file /var/log/bgp.log
```

Виходимо і записуємо інформацію:
(config-router)*# end*
 # write file
Building Configuration...
Integrated configuration saved to /etc/quagga/Quagga.conf

Так же і для інших двох.