Protokoły rutingu dynamicznego

Tomasz MALINOWSKI

Zakład Teleinformatyki, Instytut Automatyki i Robotyki WAT, ul Kaliskiego 2,
00-908 Warszawa.

STRESZCZENIE: W artykule omówiony został sposób implementacji i konfiguracji popularnych protokołów rutingu z wykorzystaniem oprogramowania GNU Zebra. Zamieszczone zostały wskazówki dotyczące instalacji oprogramowania w systemie Linux oraz konfigurowania najczęściej wykorzystywanych, również w ruterach CISCO, protokołów: RIP, OSPF i BGP. Zaprezentowana przykładowa konfiguracja protokołów stanowi wstęp do budowy bardziej złożonych systemów. Omawiane oprogramowanie może być podstawą budowy, niewielkim kosztem, złożonych konfiguracji ruterów, z różnymi zestawami protokołów. Zdaniem autora środowisko laboratoryjne z programowymi ruterami, zachowującymi funkcjonalność ruterów sprzętowych stanowi cenny materiał dydaktyczny. Podana charakterystyka wymienionych protokołów stanowi wstęp do studiowania zasad ich działania i wykorzystania.

Wstęp

W artykule omawiane są aspekty realizacji rutingu dynamicznego w środowisku Linux (Unix). Zaletą stosowania w roli ruterów hostów (tutaj komputerów klasy PC) jest niewielki koszt rozwiązania oraz możliwość wykorzystania hosta do spełniania dodatkowych funkcji np. serwera plików.

W przeciwieństwie do ruterów pracujących na hostach, rutery dedykowane są zoptymalizowane do przełączania pakietów IP (zarządzanie buforami w tych urządzeniach, kontrola procesów i schematy obsługi przerwań zostały opracowane z myślą o jednym zadaniu, zadaniu wymiany tablic rutingu i dokonywania wyboru drogi w sieci dla napływających do rutera pakietów). W połączeniu z brakiem konieczności obsługi wielu użytkowników, kompilatorów i systemu plików posiadamy urządzenie pracujące znacznie

wydajniej. Niewątpliwie zaletą ruterów dedykowanych jest obsługa wielu protokołów rutowania oraz dopracowane interfejsy umożliwiające ich konfigurację.

Rutery programowe pracujące w oparciu o hosty, obsługują zwykle jeden wybrany protokół rutowania dynamicznego, a sposób ich konfigurowania często pozostawia wiele do życzenia. Jednakże niezaprzeczalnym atutem ruterów programowych jest ich niska cena oraz możliwość przeprowadzenia wielu eksperymentów w warunkach laboratoryjnych, co nie zawsze jest możliwe w przypadku rutera sprzętowego wykorzystywanego najczęściej w typowej sieci.

W przypadku budowy dużej sieci konieczne jest, przed dokonaniem wyboru konkretnego urządzenia, określenie pożądanych możliwości funkcjonalnych rutera. Do podstawowych funkcji, zazwyczaj branych pod uwagę (i wymaganych), zaliczyć można [1]:

1. Obsługę przynajmniej jednego protokołu dynamicznego rutowania (RIP, OSPF, IGRP, EIGRP, BGP).
2. Możliwość przechowywania informacji o procesie rutowania w pliku jak również możliwość dynamicznego „zrzucania" tych informacji do plików.
3. Dostępność interfejsu do konfiguracji rutera i zarządzania ruterem (poprzez Telnet, protokół HTTP ze stroną WWW lub protokół SNMP).

Rutery sprzętowe są urządzeniami stosunkowo drogimi i wybór konkretnego rozwiązania musi być poprzedzony wnikliwą analizą potrzeb, stosunku zysków do kosztów, zasadności wyboru danego urządzenia z punktu widzenia np. planu modernizacji sieci w przyszłości.

Omawiane poniżej oprogramowanie o nazwie Zebra umożliwia emulowanie na komputerze klasy PC rutera CISCO, urządzenia firmy wyznaczającej od dawna standardy w zakresie wymiany informacji między ruterami jak i samej obsługi tych urządzeń. Wymienione oprogramowanie realizuje funkcje z pkt. 1-3 oraz daje możliwość sprawdzenia, w jaki sposób będzie pracowała sieć z ustaloną konfiguracją rutingu.

2. Podstawowe informacje o protokołach rutingu

W rozdziale tym przedstawione zostaną cechy charakterystyczne wybranych protokołów rutingu dynamicznego: RIP, OSPF i BGP. Oprogramowanie realizujące funkcje tych protokołów dostarczane są z pakietem Zebra.

Jedną z najważniejszych funkcji rutera, obok wyboru drogi dla napływającego strumienia pakietów, jest utrzymywanie aktualnych tablic

trasowania (rutingu). W artykule pojęcie rutingu odnosi się jedynie do procesu utrzymywania tych tablic i zarządzania nimi.

W odróżnieniu od rutingu statycznego (z niezmiennymi tablicami trasowania w pamięci rutera) protokoły rutingu dynamicznego służą do utrzymania aktualności tablic nawet wtedy, gdy następują zmiany sprzętu, uszkodzenia łączy sieciowych czy dodawane są nowe segmenty sieci. W celu ciągłego aktualizowania tablic proces trasowania wysyła oferty tras w postaci *aktualizacji tras* oraz odbiera i przetwarza uaktualnienia tras tak, aby trasy zapisywane w tablicy rutera były najbardziej efektywne.

2.1. Klasyfikacja protokołów rutingu dynamicznego

Dynamiczne protokoły rutowania można sklasyfikować na kilka sposobów. Pierwszy związany jest z tym, jaką część sieci protokół obejmuje swym zasięgiem. Wyróżnić tutaj możemy [2]:

- Protokoły ***zewnętrzne*** (*Exterior Gateway Protocols*);
- Protokoły ***wewnętrzne*** (*Interior Gateway Protocols*).

Protokół klasyfikowany jako zewnętrzny odpowiada za wymianę informacji o rutingu pomiędzy dwiema niezależnymi domenami administracyjnymi, zwanymi często systemami autonomicznymi. Najpopularniejszym obecnie protokołem rutingu, wykorzystywanym pomiędzy systemami autonomicznymi jest BGP (*Border Gateway Protocol*). Charakterystyczną cechą protokołów zewnętrznych jest ich dobra skalowalność i przystosowanie do obsługi dużych sieci. Składają się zwykle z wielu podprotokołów, a ich implementacja jest dość skomplikowana.

Wewnętrzne protokoły rutingu stosowane są wewnątrz pojedynczej domeny administracyjnej. Oznacza to, że rutery należące do różnych domen, nie komunikują się ze sobą bezpośrednio. Protokoły posiadają prostszą implementację, w mniejszym stopniu obciążają procesor i pamięć rutera. Najczęściej wykorzystywanymi reprezentantami tej klasy protokołów są: RIP (*Routing Information Protocol*), OSPF (*Open Shortest Path First*) i EIGRP (*Enhanced Interior Gateway Protocol*). RIP i OSPF są standardami otwartymi, EIGRP jest natomiast opracowany przez Cisco Systems i stosowany jest w ruterach tej firmy.

Innym kryterium klasyfikacji może być sposób w jaki wykorzystywane są przez protokoły rutingu informacje zawarte w tablicach rutingu i informacje dostarczane przez inne rutery.
Wyróżnić tutaj możemy [2]:

- protokoły typu ***dystans-wektor*** (*Distance - vector*);

- protokoły ***stanu łącza*** (*Link - state*).

Reprezentantem protokołów typu dystans-wektor jest historyczny i mocno dziś krytykowany (aczkolwiek nadal używany) protokół RIP, opisywany szczegółowo w dokumentach RFC1058 (*Routing Information Protocol. C.L. Hedrick*) i RFC2453 (*RIP Version 2. G. Malkin*).

Protokoły typu dystans-wektor regularnie rozsyłają kompletne tablice rutingu. Pojedynczy zapis w tablicy dotyczący danej drogi w sieci zwykle określa, obok adresu przeznaczenia i adresu następnego „skoku", ilość skoków do sieci docelowej (liczbę ruterów na drodze pakietu do sieci docelowej). Liczba skoków jest tutaj jedyną miarą jakości danej ścieżki.

W odróżnieniu od protokołów dystans-wektor w protokołach typu stan łącza miara jakości danej trasy może być wyznaczona z wykorzystaniem kilku wskaźników takich jak: obciążenie sieci, pasmo przepustowości danego łącza, opóźnienie na łączu oraz inne parametry opisujące ruter. Dodatkowo, administrator sieci sam może zdefiniować miarę administracyjną dla danego interfejsu rutera wykorzystywaną przez proces rutingu. Ruter z protokołem stanu łącza nie przekazuje do sąsiednich ruterów informacji o miejscach, które można osiągnąć i w jaki sposób, lecz dostarcza informacje o topologii sieci. Przekazuje listę segmentów lub łączy, do których jest przyłączony bezpośrednio oraz stan tych łączy (czy funkcjonują i jaki koszt jest z nimi związany).

2.2. Protokół RIP

Protokół RIP [5] przeznaczony jest dla środowiska o stosunkowo prostej topologii z niewielką liczbą ruterów połączonych łączami o takiej samej charakterystyce. Protokół ten, jako pierwszy protokół typu IGP (*Interior Gateway Protocol*), był powszechnie używany i darmowo dystrybuowany w systemie Unix BSD, jako proces demon o nazwie *routed*. Protokół ten funkcjonuje do dziś w najnowszych dystrybucjach systemów unixowych, aczkolwiek w nieco odświeżonej formie.

RIP jest protokołem typu dystans-wektor. Nazwa ta wywodzi się od zastosowanego tutaj algorytmu Bellmana-Forda wyznaczania najkrótszych ścieżek w grafie obrazującym topologię sieci. RIP wymaga zaangażowania wszystkich ruterów (czasem również serwerów plików, aplikacji, bazodanowych, itp.) w proces utrzymywania aktualności tras i oceny ich długości.

Zasada działania rutera z protokołem RIP jest bardzo prosta. Każdy ruter jest zaprogramowany tak, aby rozgłaszał wszystkie cele (adresy sieci docelowych, czasem również hostów), które zna oraz odległości do nich. Co określony kwant czasu wysyła w sieć pełną informację o wszystkich trasach

w znanej sobie topologii sieci. Po otrzymaniu podobnych informacji od innych ruterów zaczyna analizować wszystkie możliwe cele i obliczać (wykorzystując wspomniany algorytm Bellmana-Forda) najkrótsze ścieżki do nich. Tylko najkrótsza ścieżka dla danego celu jest zapisywana w tablicy rutingu. Tablica ta wykorzystywana jest przy wyborze drogi dla napływającego strumienia pakietów.

Kluczowym aspektem działania protokołów typu dystans-wektor jest to, że dowolny ruter zna tylko kolejny krok w sekwencji dostarczania pakietów do ich miejsca przeznaczenia.

Protokół RIP funkcjonuje do dziś, jest szeroko używany i jest jedynym protokołem trasowania „rozumianym" przez wszystkie komputery używające systemów operacyjnych typu Unix. Niestety protokół ten posiada szereg wad i ograniczeń, które dyskwalifikują go w roli protokołu rutingu dla dużych sieci.

Z założenia każdy ruter z uruchomionym protokołem RIP co 30 sekund wysyła tzw. aktualizacje RIP. Aktualizacje te to wszystkie zapisy dotyczące znanych tras z tablicy rutingu rutera. Jeśli więc tych tras jest dużo, kilka ruterów z protokołem RIP potrafi „skonsumować" znaczną część przepustowości sieci.

Informacja dotycząca pojedynczej trasy zawiera:

- adres docelowego hosta lub sieci;
- adres IP rutera (bramki) wysyłającego aktualizację;
- wartość wskazującą odległość (w sensie liczby skoków) do celu.

W momencie otrzymania aktualizacji trasy od sąsiedniego rutera, ruter porównuje przyjętą informację z zawartością własnej tablicy trasowania. W przypadku, gdy aktualizacja dotyczy nowej trasy, trasa jest dodawana do tablicy rutingu. Jeśli ruter otrzyma informację o istniejącej w tablicy trasie, z mniejszą liczbą skoków do danego celu, stary zapis o osiągalności tego celu jest zastępowany świeżą informacją. Oczywiście istnieje tutaj zabezpieczenie przed sytuacjami kiedy dana trasa przestaje funkcjonować (uszkodzenie linii, uszkodzenie rutera na drodze do sieci docelowej). Ruter RIP pamięta niestety tylko jedną, najlepszą trasę. W jego tablicy trasowania brak jest zapisów dotyczących tras alternatywnych.

Do sterowania sytuacjami awaryjnymi RIP używa czasomierzy. Obok zegara sterującego wysyłaniem aktualizacji tras (domyślnie co 30s) wykorzystywany jest zegar zliczający dla każdej z tras czas jaki upłynął od jej ostatniej aktualizacji. Jeżeli ruter nie otrzyma aktualizacji danej trasy w przeciągu 180 s trasa zaznaczana jest jako nieosiągalna. Po stwierdzeniu nieosiągalności trasy ruter wysyła specjalną wiadomość informującą sąsiadów o zaistniałej sytuacji. Po 270s nieosiągalna i zarazem nieaktualna trasa jest usuwana z tablicy rutingu rutera. Kolejne zegary noszą nazwę czasomierzy: aktualizacji (*update*), błędu (*invalid*) i usuwania (*flush*).

Podsumowując, protokół RIP posiada szereg ograniczeń i przez to nie nadaje się do wykorzystania w dużych sieciach.
Najważniejsze z nich to, że [3]:

- miarą jakości danej ścieżki RIP jest liczba skoków. RIP zakłada maksymalną liczbę skoków równą 15. Sieć, do której odległość wynosi 15 skoków jest dla protokołu RIP nieosiągalna;
- liczba skoków jest jedyną miarą jakości danej trasy, co w przypadku istnienia alternatywnych, „szybszych" tras, ale o większej liczbie skoków, przekreśla ich zastosowanie w procesie wyboru drogi dla pakietu;
- RIP w wersji 1 nie potrafi obsługiwać podsieci o różnych wielkościach (ruting klasowy);
- okresowa wymiana całych tablic rutingu wprowadza znaczne obciążenie sieci;
- konwergencja sieci z protokołem RIP jest wolniejsza niż dla protokołów typu link-state;
- sieci RIP są sieciami płaskimi. Nie zakłada się ich podziału na obszary, nie wyznacza się granic tych obszarów.

Pewne modyfikacje wprowadzone zostały w RIP2 (VLMS, autentication, multicast routing updates, "split horizon") [4]. Nadal jednak podstawową miarą rutingu pozostaje liczba skoków, nadal jej maksymalna wartość to 15 (wartość 16 odpowiada nieosiągalności miejsca przeznaczenia).

2.3. Protokół OSPF

Protokół OSPF [7] opracowany został w końcu lat 80-tych. W odróżnieniu od RIP bazuje on na przesyłaniu do swoich sąsiadów jedynie istotnych wyciągów z tablic rutingu. OSPF jest protokołem typu stanu łącza (*link-state*), więc przesyłane informacje dotyczą jedynie charakterystyk własnych interfejsów rutera. W celu utrzymania tablic rutingu protokół wykorzystuje 5 rodzajów komunikatów [2]:

1. Komunikaty '*hello*'.
2. Opisy baz danych *(database descriptions)*.
3. Żądania danych o stanie połączeń *(request link-state)*.
4. Dane aktualizujące stan połączeń (*updates link-state*).
5. Potwierdzenia stanu połączeń (*acknowledgments link-state*).

Komunikaty nr 4 są właściwymi komunikatami, na podstawie których wyznaczane i utrzymywane są tabele rutingu. Komunikaty te obejmują tzw. ogłoszenia LSA (*link-state advertisements*). Jest kilka rodzajów komunikatów

LSA i nie mogą być one mylone z wymienionymi wyżej rodzajami komunikatów.

Wymiana ogłoszeń LSA umożliwia utworzenie przez ruter bazy danych o stanie łączy (stanie własnych interfejsów i interfejsów innych ruterów w sieci). Na podstawie tej bazy danych, z wykorzystaniem dobrze znanego z teorii grafów algorytmu Dijkstry, ruter oblicza najkrótsze ścieżki w sieci.

Pierwszą operacją po włączeniu rutera z protokołem OSPF jest wysłanie komunikatu *'hello'* na wszystkie swoje interfejsy, do wszystkich ruterów OSPF w sieci. Na podstawie odpowiedzi na wysłane komunikaty *'hello'* ruter odkrywa, kto jest jego sąsiadem tzn. z którymi ruterami może komunikować się bez pośrednictwa innych ruterów. Dla całej sieci, po krótkim okresie wymiany komunikatów *'hello'*, ustalone zostaną sieciowe grupy sąsiedzkie.

Po ustaleniu swoich sąsiadów ruter próbuje zbudować tablicę wzajemnych powiązań z nimi. Określonych jest siedem poziomów wzajemnych odniesień pomiędzy ruterami OSPF: od stanu tzw. *zaniku połączenia* (*down state*) do stanu *pełnego sąsiedztwa* (*fully adjacent state*). W stanie zaniku połączenia rutery w ogóle nie mogą komunikować się ze sobą, a w stanie pełnego sąsiedztwa wymieniają wszystkie 5 rodzajów komunikatów OSPF.

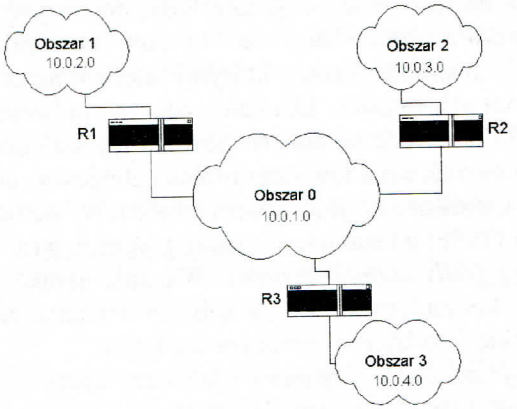
Kiedy rutery dokonają wymiany ogłoszeń LSA, na każdym z nich ukończone zostanie ustawienie takiej samej bazy danych o połączeniach (*database of links*), czyli mapy topologii sieci. Na jej podstawie ruter z algorytmem Dijkstry buduje tablice rutingu, wykorzystywaną do kierowania ruchem napływających pakietów.

Na podstawie skutków różnych zdarzeń (uszkodzenie kabla, uszkodzenie interfejsu, itd.) rutery zmieniają dane o stanie połączeń. Baza o stanie połączeń musi być aktualizowana na bieżąco, wszystkie rutery powinny posiadać identyczną bazę. W przeciwnym razie obraz sieci byłby niespójny. W konsekwencji tego założenia, jeśli ruter nie otrzymuje od swego sąsiada przez pewien czas (tzw. przedział martwy – *dead interval*) żadnej wiadomości, uzna, że łącze utraciło stan gotowości do działania. Stan ten jest określany mianem 'połączenie przejściowe' (*link transition*). Informacja o stanie przejściowym musi być natychmiast wymieniona ze wszystkimi ruterami OSPF, by mogły przeprowadzić rekalkulację swych tablic rutingu. Proces synchronizowania baz danych przez rutery nazywany jest konwergencją (*convergence*). Zwykle zależy nam na jak najkrótszym czasie konwergencji, po którym obraz sieci staje się spójny. W typowej sieci konwergencja dokonuje się w przeciągu milisekund.

Obszary OSPF

Kiedy grupa sieci i ruterów staje się zbyt duża, może być podzielona na obszary (strukturę hierarchiczną, określoną ponad strukturą odpowiadającą adresacji IP).

Protokół OSPF do prawidłowego funkcjonowania wymaga zdefiniowania tzw. obszarów. Wymagane jest zdefiniowanie przynajmniej jednego obszaru, nazywanego *obszarem szkieletu*, do którego będą należały wszystkie tzw. rutery brzegowe (graniczne) OSPF. Rysunek 1 przedstawia przykładową strukturę sieci z umiejscowionym centralnie obszarem szkieletu (Obszar 0) i dołączonymi do szkieletu obszarami o numerach 1, 2 i 3.



Rys.1. Przykładowa struktura sieci OSPF.

Każdy z obszarów może obejmować kilka podsieci, zgrupowanych z wykorzystaniem techniki grupowania, omówionej w dalszej kolejności.

W protokole OSPF odpowiedzialność za funkcjonowanie sieci spada na obszar szkieletu. Poszczególne obszary nie współdziałają ze sobą w ramach OSPF. Konstruowanie obszarów pozwala zredukować rozmiar bazy danych o połączeniach w każdym ruterze (wymóg posiadania identycznych baz danych zostaje ograniczony do obszaru).

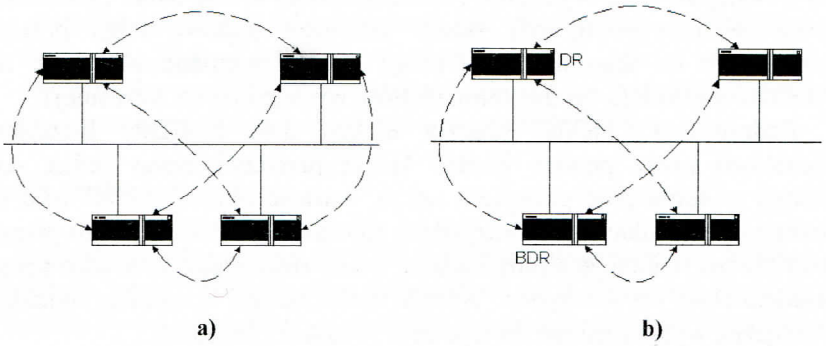
Rutery umiejscowione na granicy obszarów (posiadające interfejs łączący je ze szkieletem sieci i przynajmniej jeden interfejs do obszaru nie zerowego nazywane są ruterami brzegowymi obszarów tzw. ABR-ami (*Area Border Router*). Jedną z funkcji rutera ABR jest zapewnienie obszarowi OSPF dostępu do szkieletu sieci. Przy wyborze obszarów należy zachować daleko posuniętą ostrożność, chociażby z tego powodu, że cały ruch pakietów będzie odbywał się przez obszar zerowy nawet w przypadku, gdy pomiędzy obszarami istnieją alternatywne, być może szybsze łącza.

Obszary OSPF mają przydzielone identyfikatory ID obszaru, które mają taką samą notację dziesiętną jak adresy IP (adresy te nie powinny być jednak ze sobą mylone). W praktyce obszary OSPF są często określane skrótowo

pojedynczą liczbą (1, 2,... itd. które są odpowiednikiem numerów ID: 0.0.0.1, 0.0.0.2, itd.).

Rutery desygnowane

Obok pomysłu dekompozycji sieci na obszary, w protokole OSPF wprowadzona została koncepcja wyróżnienia niektórych ruterów do spełniania specjalnych funkcji. Rutery OSPF dokonują wyboru spośród siebie reprezentanta, który przejmuje obsługę swoich sąsiadów przy kontaktach z innymi ruterami w sieci, tzw. rutera desygnowanego DR (*designated router*).



Rys.2. Schemat połączeń (sesji) między ruterami OSPF dla przypadku: a) bez wyboru ruterów desygnowanych, b) z wybranym DR i BDR.

W tym samym czasie wybierany jest również zapasowy ruter desygnowany BDR (*backup designated router*). Funkcja rutera desygnowanego sprowadza się do czuwania nad spójnością obrazu sieci w każdym z ruterów. Obecność DR-a i BDR-a w sieciach z wielodostępem rozgłoszeniowym (przykładowo sieć Ethernet) redukuje liczbę tras komunikacyjnych (powiązań sąsiedzkich pomiędzy ruterami). W efekcie zmniejsza się ruch multicastowy w sieci. Na rysunku 2 przedstawiony jest schemat powiązań między ruterami w sieci bez ruterów desygnowanych i z wybranymi DR i BDR.

Liczba wzajemnych połączeń (sesji), przy N-ruterach w sieci, dla przypadku bez DR/BDR jest określona przez N*(N-1)/2. Dla przypadku z wybranymi ruterami desygnowanymi (N-1)+(N-2). W podanym przykładzie zysk jest niewielki (liczba sesji zredukowana zostaje z 6 do 5). Stosownie DR-a i BDR-a nabiera większego znaczenia przy większej liczbie ruterów. Wybór rutera desygnowanego i zapasowego rutera desygnowanego dokonywany jest z wykorzystaniem pakietów *„hello"*. Pakiet ten zawiera pole przeznaczone na priorytet rutera (0-255). Wartość priorytetu rutera ustalona może być podczas

jego konfiguracji. W procesie elekcji rutery wymieniają pakiety *„hello"* i wybierają spośród siebie dwa rutery o najwyższych priorytetach. Proces reelekcji nie dokonuje się automatycznie.

Podprotokoły OSPF

Protokół OSPF jest dość skomplikowanym protokołem. W jego skład wchodzą trzy podprotokoły: „Hello", „Exchange" i „Flooding". Podprotokoły OSPF operują w różnych stadiach wzajemnych kontaktów pomiędzy ruterami OSPF. Podprotokół „Hello" działa podczas wzajemnego rozpoznawania się ruterów (poszukiwania sąsiadów). „Exchange" pojawia się wtedy, gdy dwa rutery uznają, że chcą wymienić (uszczegółowić) swoje dane. „Flooding" jest używany w przypadku, gdy rutery zakończą budowę tabel przyległości wskazując, że od tego momentu mogą być wymieniane wszystkie rodzaje komunikatów OSPF (5 typów komunikatów wymienionych wcześniej).

Podprotokół „Hello" ułatwia elekcję DR i BDR. Rozsyłane są multicastowo przez pewien bardzo krótki przedział czasu (kilka sekund). Zawierają w sobie pole priorytetu rutera, wartość okresu „hello" oraz okresu martwego i identyfikatory ID wszystkich ruterów sąsiedzkich. Jeżeli przez okres martwy nie ma reakcji ze strony sąsiada, wtedy ruter ogłasza, że jego połączenie z sąsiadem stało się nieczynne. Wysyła multicastowo komunikat aktualizujący stan połączeń wykorzystując do tego celu protokół „Flooding".

Podprotokół „Exchange" uruchamiany jest po tym jak rutery określą zakres wzajemnych powiązań sąsiedzkich (korzystając z „Hello"). Rozpoczynają wtedy synchronizowania baz danych o stanie połączeń. Za pomocą protokołu „Exchange" dokonywana jest wymiana informacji o stanie połączeń. Proces wymiany odbywa się stopniowo, fragmentami. Takie fragmenty baz danych zawierają komunikaty opisu baz danych (*database description* – komunikaty typu 2). Idea wymiany częściowych opisów polega na dostarczeniu ruterom tylko takiej informacji (takiej części), która pozwoli im określić, czy są oni zainteresowani poszczególnymi rekordami o stanie połączeń. Z punktu widzenia rutera interesujące dla niego są tylko takie rekordy, które różnią się od posiadanych we własnej bazie lub po prostu nie istnieją. Kiedy ruter stwierdzi, że jest zainteresowany pewnymi rekordami, zgłasza na nie zapotrzebowanie z pomocą komunikatu OSPF nr 3. Pełne rekordy zwracane są za pomocą komunikatu OSPF typu 4. „Exchange" jest najbardziej skomplikowanym z podprotokołów OSPF, ale wykorzystywany jest najrzadziej. W stabilnej sieci ruch generowany przez OSPF w dominującej części związany będzie z komunikatami „Hello" i od czasu do czasu pojawiającym się wylewem informacji o stanie połączeń (protokół „Flooding").

Podprotokół „Flooding" służy do zsynchronizowania baz danych o stanie połączeń między ruterami po zbudowaniu przez nie pełnych powiązań.

Wykorzystywane są tutaj komunikaty typu 4 (dane aktualizujące stan łącza) oraz typu 5 (potwierdzenia stanu łącza).

„Flooding" stosowany jest przez rutery w trzech sytuacjach:

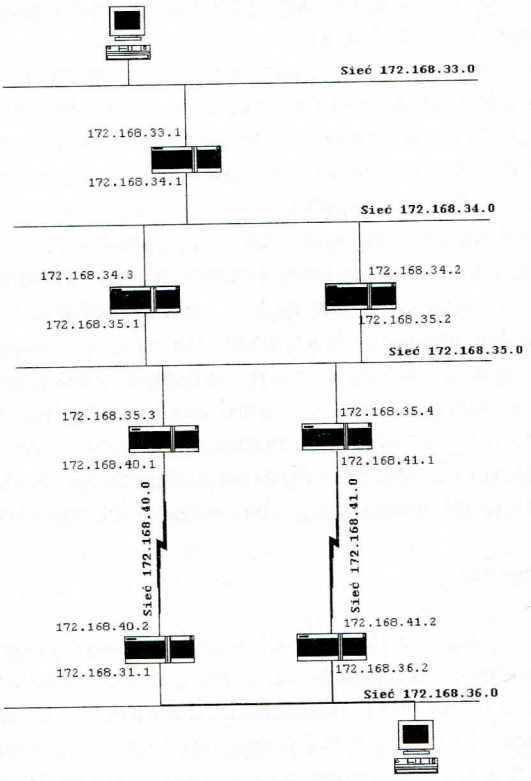
a) do powiadomienia swoich sąsiadów o stanie przejściowym łącza – zawsze po tym, jak ruter stanie się świadom jakiegokolwiek problemu na którymś ze swoich interfejsów (np. w momencie zaniku napływu pakietów *'hello'* od sąsiada);
b) przy cyklicznie rozsyłanych w ustalonych okresach danych aktualizujących o stanie łączy (zazwyczaj co 30 minut);
c) przy wysyłaniu danych aktualizujących o rekordach zdezaktualizowanych. Wszystkie rekordy w bazie posiadają swoje liczniki czasu. Kiedy licznik osiągnie maksymalną dopuszczalną wartość (zwykle 1 godz.) rekord taki nie będzie uwzględniany przy wyborze tras i zostaje zaznaczony jako rekord do usunięcia. W konsekwencji zdezaktualizowane rekordy są „wylewane", aby mogły być usunięte jak najszybciej z baz wszystkich ruterów.

Technika grupowania

Technika tworzenia grup w OSPF (*summarization*) pozwala ruterom ABR na zgłaszanie mniejszej liczby tras dotyczących tych obszarów, które są przez nie włączone do szkieletu sieci. Warunkiem stosowania techniki grupowania jest to, aby numery sieci, które mają być połączone były w bliskim sąsiedztwie.

Na rysunku 3 przedstawione zostało przykładowe środowisko rutingu, w którym zastosowane będzie grupowanie sieci.

Załóżmy, że wszystkie rutery na rysunku, należą do jednego obszaru i że tylko jeden z nich (w przykładzie nie jest istotne który) posiada dodatkowy interfejs dołączający go do obszaru zerowego (szkieletu) sieci. Ruter ten będzie komunikował się w imieniu pozostałych ruterów z innymi ruterami szkieletowymi zgłaszając im, jak wygląda topologia obszaru niezerowego, do którego sam należy. Załóżmy, że administrator sieci chce zamaskować dla innych ruterów szczegóły podanej konfiguracji sieci należących do obszaru o identyfikatorze ID=1. Najprościej pogrupować sieci w ramach nowej podsieci i skonfigurować ruter tak, aby do szkieletu zgłaszał tylko jedną trasę dotyczącą sieci sumarycznej.



Rys.3. Przykładowe środowisko rutingu

Niech sieci o numerach:

- 172.168.33.0
- 172.168.34.0
- 172.168.35.0
- 172.168.36.0

będą sieciami typu ethernet, natomiast sieci o adresach 172.168.40.0 i 172.168.41.0 będą sieciami typu punkt-punkt.

Zwróćmy uwagę, że pierwsze dwa oktety w adresach wszystkich sieci są identyczne. Można więc dokonać prostego grupowania podsieci w jedną sieć o nowym adresie 172.168.0.0 (stanowiącym część wspólną wszystkich adresów) i masce 255.255.0.0. Łatwo zauważyć, że w tak wyznaczonej sieci grupowej istnieje miejsce na 254 podsieci ethernet lub punkt-punkt o adresach z zakresu 172.168.**1**.0 – 172.168.**254**.0. Z punktu widzenia administratora sieci, taki sposób grupowania może oznaczać utratę dużej puli adresów IP (nie wykorzystane w przedstawionej konfiguracji adresy). Dokonując konwersji

adresów IP z notacji dziesiętnej na binarną zauważamy, że wspólna część wszystkich adresów to 20 bitów. Sieci grupowej można więc przypisać adres 172.168.32.0. Długość grupowej maski odpowiadająca wspólnej części adresów wszystkich sieci wynosi więc 20 bitów, co odpowiada masce 255.255.240.0. W rozważanym przykładzie zakres adresów sieci, które mieściłyby się w ramach grupowego zgłoszenia zawiera się w przedziale 172.168.**32**.0 – 172.168.**47**.0.

Zamiast jednego grupowego zgłoszenia można posługiwać się kilkoma, np. jednym dla wszystkich sieci ethernet i drugim dla sieci punkt-punkt. Łatwo zauważyć, że dla sieci typu ethernet wspólna część adresowa to 21-bitów, dla sieci typu punkt-punkt 23 bity. Sieć grupująca sieci ethernet miłaby więc adres 172.168.32.0 z maską 255.255.248.0 (możliwość zgrupowania sieci o adresach z przedziału 172.168.**32**.0-172.168.**39**.0), zaś adres drugiej sieci grupowej wynosiłby 172.168.40.0 z maską 255.255.254.0 (co daje możliwość zgrupowania sieci z adresami z zakresu 172.168.**40**.0-172.168.**41**.0).

Przy konfigurowaniu protokołu OSPF w złożonych sieciach, należy zwrócić uwagę na następujące wskazówki implementacyjne [2]:

- jeśli jest to możliwe należy dzielić złożoną sieć na mniejsze części;
- należy unikać sytuacji, w których jakiś ruter pełni rolę rutera desygnowanego dla kilku obszarów OSPF;
- należy unikać stosowania rutera ABR w roli rutera desygnowanego, ponieważ obsługiwałby on dwie kopie baz danych o stanie połączeń: jedną dla obszaru szkieletowego, drugą dla obszaru, który jest dołączony do sieci szkieletowej;
- należy korzystać z techniki grupowania sieci, dla celów zbiorczego zgłaszania ich do obszaru szkieletowego i innych obszarów. Pozwala to na redukcję zajętości pasma sieciowego (mniejsza liczba LSA do przetworzenia oraz mniejszy roamiar tabel rutingu).

2.4. Protokół BGP

Protokół BGP [8] umożliwia przekazywanie pakietów z danymi użytkowymi pomiędzy odległymi lokalizacjami w Internecie, tzn. pomiędzy dużymi systemami zarządzanymi przez operatorów sieci, zwanymi systemami autonomicznymi (są to systemy typu „transit", „multihomed" i „stub"). Każdy system autonomiczny jest rejestrowany przez organizację InterNIC, gdzie przydzielany jest mu odpowiedni numer. Numery te są wykorzystywane podczas konfigurowania ruterów dla BGP.

W systemie autonomicznym jeden lub więcej ruterów, które zostają skonfigurowane do obsługi BGP, stają się *speakerami* BGP (reprezentantami

pozostałych ruterów i sieci tworzących system autonomiczny). *Speakery* wiedzą wszystko o konfiguracji sieci w ramach AS. Odbierają także komunikaty aktualizujące ruting z danymi o sieciach z innych systemów autonomicznych. Przekazują swoim partnerom aktualizacje rutingu zarówno sieci wewnętrznych jak i sieci z innych AS.

Dowolne rutery skonfigurowane do obsługi BGP wymieniające aktualizacje rutingu nazywane są *BGP peers* (partnerzy BGP). Ruter BGP pozyskuje dane o swoich sąsiadach za pomocą protokołu TCP korzystając z powszechnie znanego portu o numerze 179. Adresy IP oraz numery systemów autonomicznych ze wszystkich partnerskich ruterów BGP, które konfigurowany ruter będzie dostrzegał, muszą być wyspecyfikowane za pomocą polecenia konfiguracyjnego *„neighbor"*.

Po nawiązaniu połączenia TCP pomiędzy partnerami BGP do wzajemnego komunikowania wykorzystywane są 4 rodzaje komunikatów: *open, update, keepalive* i *notification.* Najbardziej użytecznym jest *update*, który służy do przekazywania informacji o rutingu. Komunikaty *open* i *keepalive* wykorzystywane są do nawiązania i utrzymywania sesji BGP. Komunikat *notification* służy do powiadamiania o błędnych warunkach pomiędzy partnerskimi BGP.

Partnerzy BGP mogą być zarówno wewnętrzni jak i zewnętrzni. Wewnętrzni partnerzy budują (nawiązują) sesję w ramach jednego systemu autonomicznego, a partnerzy zewnętrzni to rutery należące do różnych systemów autonomicznych.

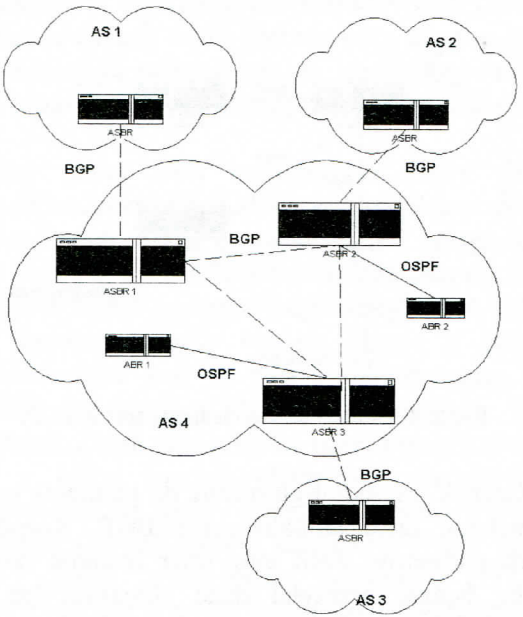
Protokół BGP posiada możliwość realizacji następujących funkcji:

- agregacji tras;
- egzekwowania reguł różnej polityki rutingu dla różnych systemów autonomicznych;
- poprawy skalowalności poprzez wykorzystanie dla tras reflektorów i konfederacji;
- współdziałanie z protokołami IGP poprzez redystrybucję i synchronizację.

Agregacja tras jest realizowana podobnie jak dla protokołu OSPF, z tą różnicą, że dla BGP zagregowane adresy sieci i ich maski wyznaczane są na poziomie systemu autonomicznego, a nie obszaru. Dzięki funkcji agregacji tras można zredukować liczbę tras (sieci) w ramach systemów autonomicznych, które rutery BGP będą ogłaszać dla swoich zewnętrznych partnerów. Proces BGP pracujący na ruterze musi być specjalnie poinformowany poprzez odpowiednie polecenia konfiguracyjne, że ma obsługiwać agregację tras. Koncepcja agregacji tras jest bardzo prosta, aczkolwiek szczegóły implementacji i opcje konfiguracyjne są bardzo złożone [3].

Pomiędzy procesami realizującymi różne protokoły rutingu możliwa jest redystrybucja tras. Można tutaj dystrybuować trasy statyczne zapisane w tabelach tras rutera (redystrybucja statyczna) jak i trasy otrzymywane od innych ruterów i różnych protokołów rutingu (redystrybucja dynamiczna). Redystrybucja dynamiczna polega na zleceniu procesowi BGP akceptacji tras zgłaszanych przez protokół typu IGP (RIP, OSPF, IGRP...). Wymaga to jednak takiego skonfigurowania procesu IGP, aby dystrybuował on swoje trasy dla BGP. Możliwa jest wzajemna redystrybucja tras, zarówno statyczna jak i dynamiczna, pomiędzy protokołami BGP a IGP. Na rysunku 4 przedstawiony został przykład środowiska, w którym możliwe jest zastosowanie redystrybucji statycznej i dynamicznej między protokołami OSPF i BGP. Liniami ciągłymi zaznaczone zostały sesje BGP, liniami przerywanymi sesje OSPF. Jak widać, na ruterach ASBR (*Autonomous System Border Router*) powinny być uruchomione dwa procesy: OSPF i BGP.

Podstawowy przepływ danych o osiągalnych sieciach dokonuje się z IGP (tutaj procesu OSPF) do BGP. Proces BGP uruchomiony na ASBR dowiaduje się o sieciach w systemie AS za pomocą redystrybucji statycznej, dynamicznej lub za pomocą odpowiedniego polecenia konfiguracyjnego (np. *network*, udostępnianego przez system sieciowy IOS Cisco).



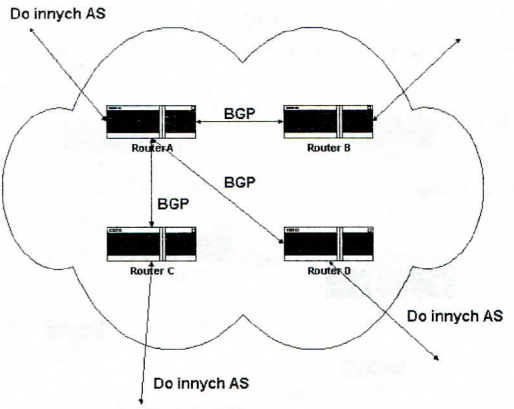
Rys.4. Ruting OSPF i BGP wewnątrz systemu autonomicznego.

W przypadku jakiejkolwiek niestabilności w pracy sieci obsługiwanych np. przez protokół OSPF dane o trasach redystrybuowane są w sposób dynamiczny. Jakakolwiek zmiana w trasach OSPF powoduje wygenerowanie danych do aktualizacji dla BGP. Poziom ruchu w sieci staje się w tym momencie znaczny. Scenariusz ten nosi nazwę „trzepotania tras". Jedynym sposobem uniknięcia trzepotania tras jest redystrybucja statyczna.

Jeśli wewnątrz AS znajduje się bardzo dużo sieci OSPF dostarczanie informacji o każdej z nich do zewnętrznego rutera BGP jest niepożądane. Sytuacja ta kwalifikuje się do przedstawienia tych sieci za pomocą metody agregacji tras.

Aby rutery BGP mogły komunikować się ze sobą muszą być skonfigurowane jako sąsiedzkie. Liczba sesji TCP w ruterach przy konfiguracji połączeń typu każdy z każdym (tzw. „full-meshed") może być znaczna. Aby zminimalizować liczbę sesji stosuje się metodę odzwierciedlania tras i metodę konfederowania [9].

Odzwierciedlanie tras jest techniką podobną do stosowania rutera desygnowanego. Rysunek 5 przedstawia powiązania między ruterami BGP przy zastosowaniu odzwierciedlania tras.



Rys.5. Technika odzwierciedlania tras w BGP.

Zakłada się, że Ruter A jest skonfigurowany do pełnienia roli zwierciadła tras i przyjmuje zewnętrzne dane aktualizacyjne BGP. Rozpowszechnia je do wszystkich swoich partnerów. Jeśli inny ruter zostanie skonfigurowany jako klient zwierciadła, będzie przesyłał dane aktualizacyjne tylko do rutera odzwierciedlającego. Ten zaś odbije (ponownie ogłosi) je do swoich partnerów. Redukowana w ten sposób jest liczba sesji BGP. Grupa ruterów skonfigurowana do uczestnictwa w odzwierciedlaniu tras nazywana jest „klastrem".

Inną metodą redukowania wzajemnych powiązań wewnętrznych BGP w ramach systemu AS zawierającego dużą liczbę speakerów są „konfederacje". W metodzie tej poszczególne speakery grupuje się w mini-ASy. Konfederacje BGP definiowane są za pomocą poleceń konfiguracyjnych ruterów.

BGP jest protokołem typu *path-vector*. Ścieżka odnosi się do serii kroków, które muszą być podjęte pomiędzy punktem początkowym, a docelowym. Ścieżka BGP jest więc serią liczb z numerami AS, przez które będą przechodzić pakiety, aby dotrzeć do miejsca przeznaczenia. Kompletny opis ścieżki dokonywany jest poprzez zestawienie jej atrybutów, co nie będzie tutaj omawiane. Atrybuty ścieżek przenoszone są w komunikatach aktualizacyjnych (update messages), wymienianych między speakerami BGP.

3. Pakiet GNU Zebra

Pakiet o nazwie Zebra [10] jest niekomercyjnym zbiorem oprogramowania realizującego wybrane protokoły rutingu na dedykowanym do tego celu komputerze, pracującym pod kontrolą unixowego systemu operacyjnego. Umożliwia wymianę tablic routingu zgodnie z zasadami wymiany komunikatów dla protokołów RIP, BGP, OSPF. Wymieniane tablice rutingu są używane do uaktualnienia systemowych (kernelowych) tablic rutingu. Pakiet umożliwia dynamiczną zmianę tablic rutingu oraz jej przeglądanie z poziomu terminala Zebry. System z zainstalowanym oprogramowaniem pracuje jako dedykowany ruter programowo realizujący wybrane funkcje rutera firmy Cisco Systems.

W przypadku małej sieci konfiguracja Zebry sprowadza się do skonfigurowania kart sieciowych i dodaniu kilku poleceń dotyczących rutingu statycznego i domyślnych bram (*gateways*). W przypadku sieci bardziej rozbudowanej lub dla sieci niestabilnej (ze zmieniającą się strukturą) możliwe jest wykorzystanie protokołów rutingu dynamicznego (RIP, OSPF,BGP).

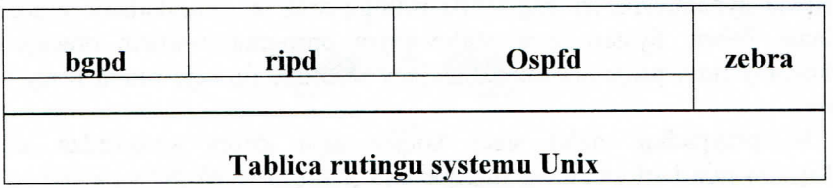
Tradycyjne oprogramowanie rutingu (znane z różnych dystrybucji systemu Unix) dostarczane jest w postaci pojedynczego procesu-demona. Większość poleceń konfigurujących ruting dostępna jest jedynie dla administratora systemu. Zebra opiera się na rozwiązaniu innym. Jest produktem składającym się z kilku procesów-demonów pracujących równocześnie i operujących na pojedynczej tablicy rutingu. Demon o nazwie *zebra* stanowi jądro systemu rutingu (*kernel routing manager*). Nieco inny niż w ruterze sprzętowym jest sposób zarządzania rutingiem [11]. Możliwe jest komunikowanie się z ruterem w trybie zwykłego użytkownika (***user mode***) oraz w trybie uprzywilejowanym (***enable mode user*** – odpowiadający trybowi ***privilage*** rutera firmy Cisco Systems).

Omawiane oprogramowanie jest dostępne pod adresem ftp://ftp.zebra.org/pub/zebra.

Obsługiwane przez Zebrę protokoły routingu to:
1. Border Gateway Protocol 4 (BGP-4) opisywany przez RFC1771.
2. OSPF v.2 – RFC2328.
3. RIP v.2 – RFC2453.
4. RIPng for IPv6 – RFC2080.

3.1. Architektura oprogramowania

Za realizacje funkcji poszczególnych protokołów rutingu (omawianych w niniejszym opracowaniu) odpowiedzialne są wyizolowane procesy, odpowiednio: **ripd** (protokół RIP), **ospfd** (protokół OSPF) i **bgpd** (protokół BGP). Za zmianę systemowych tablic rutingu (kernel routing table) oraz redystrybucję tras pomiędzy procesami rutingu odpowiedzialny jest demon **zebra**. Modułowa architektura systemu (rysunek 6) umożliwia dodanie kolejnych procesów-demonów, realizujących inne protokoły rutingu, bez szczególnej ingerencji w zainstalowane oprogramowanie. Oczywiście możliwe jest równoległe uruchomienie kilku procesów realizujących ten sam protokół.



Rys.6. Architektura systemu Zebra

W trakcie opracowywania artykułu Zebra została przetestowana na następujących platformach systemowych:
- GNU/Linux 2.0.37,
- GNU/Linux 2.2.x,
- GNU/Linux 2.3.x,
- FreeBSD 2.2.8,
- FreeBSD 3.1,
- FreeBSD 4.x.

Prowadzone są prace nad oprogramowaniem dla systemów:
- GNU/Hurd 0.2,
- Solaris 7.

3.2. Pliki konfiguracyjne zebry i pozostałych demonów

Instalacja pakietu zebra (opisywana szczegółowo w dokumentacji [11]) kończy się skopiowaniem skompilowanych programów i plików konfiguracyjnych do standardowych lokalizacji dla systemu Linux (/usr/local/bin i /usr/local/etc).

Demony wchodzące w skład pakietu dostępne są poprzez własne, dedykowane interfejsy. Po instalacji oprogramowania (tutaj w wersji 0.86) należy zadbać o właściwe przyporządkowanie im numerów portów. Niezbędne jest dodanie następujących zapisów do pliku /etc/services:

```
zebrasrv    2600/tcp    # ZEBRA service
zebra       2601/tcp    # ZEBRA vty
ripd        2602/tcp    # RIPd vty
ripngd      2603/tcp    # RIPngd vty
ospfd       2604/tcp    # OSPFd vty
bgpd        2605/tcp    #BGPd vty
ospf6d 2606/tcp         # OSPF6d vty
```

gdzie **ripngd** oraz **ospf6d** są procesami działającymi w oparciu o protokół IPv6. Virtual Terminal Interface (VTY) - interfejs terminalowy - umożliwia połączenie z demonem przez protokół telnet.

Każdy demon posiada swój własny plik konfiguracyjny (odpowiednio: zebra.conf, ripd.conf, ospfd.conf i bgpd.conf). Przykładowa zawartość plików omówiona zostanie w dalszej kolejności.

Odpowiednio skonfigurowane procesy mogą realizować następujące funkcje:

W przypadku protokołu RIP:

- posługiwanie się sposobem adresacji z wersji 1 (maska określana jest na podstawie klasy adresu IP) jak również posługiwanie się VLSM (RIP v.2 jest domyślnym protokołem);
- zablokowanie / odblokowanie procesu RIP;
- ustalenie wykorzystywanej wersji RIP (v.1, v.2);
- zablokowanie / odblokowanie danego interfejsu sieciowego (przekazywania pakietów RIP do/z wyspecyfikowanej sieci);
- specyfikacja sąsiadów dla RIP (w przypadku gdy sąsiednie rutery RIP nie obsługują trybu multicast – zdefiniowanie ścieżki między ruterami tzw. direct link);
- odblokowanie / zablokowanie redystrybucji statycznych tablic rutingu;
- zdefiniowanie zegarów : *update*, *timeout*, *garbage*, odpowiadających wymienionym wcześniej czasomierzom: *update*, *invalid* i *flush*;

- ustalenie tzw. *rip authentication string*, czyli klucza autentykacji, jakim będzie posługiwał się ruter przy przesyłaniu uaktualnień przez dany interfejs lub do określonego sąsiada;
- zablokowanie / odblokowanie odbierania pakietów RIP konkretnej wersji protokołu;
- filtrowanie pakietów zgodnie z listą dostępu, tzw. *access list*;
- śledzenie na bieżąco wymienianych komunikatów.

Dla procesu OSPF:

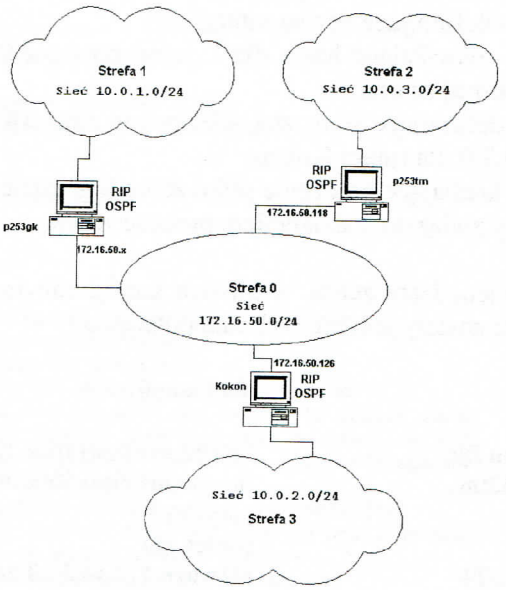
- realizacja założeń protokołu ver2 (RFC2178);
- odblokowanie / zablokowanie procesu OSPF'
- odblokowanie / zablokowanie konkretnego interfejsu;
- definiowanie obszarów OSPF;
- grupowanie tras;
- definiowanie wirtualnych ścieżek dla obszarów nie połączonych bezpośrednio ze szkieletem sieci;
- redystrybucja tras (statyczna, dynamiczna);
- funkcja autentykacji podobnie jak dla protokołu RIP;
- śledzenie wymienianych komunikatów;
- przeglądanie stanu interfejsów, sąsiadów, baz danych rutera;
- tworzenie list dystrybucyjnych.

Dla procesu BGP:

- definiowanie systemów autonomicznych;
- blokowanie / uruchamianie procesów BGP;
- ustalanie sąsiednich *speakerów*;
- ustalanie jakie trasy będą redystrybuowane i do jakich procesów;
- filtrowanie ruchu zgodnie z określoną listą;
- przeglądanie ruchu i statystyk związanych z procesem BGP;
- definiowanie klastrów w celu realizacji techniki odzwierciedlania tras.

4. Przykładowa konfiguracja środowiska rutingu

Poniżej przedstawiona została przykładowa konfiguracja środowiska, w którym uruchomione i przetestowane zostały procesy RIP, OSPF i BGP.



Rys.7. Konfiguracja środowiska testowego.

Sieć testowa zawiera trzy rutery o nazwach: p253gk, p253tm i Kokon oraz cztery obszary zdefiniowane na potrzeby protokołu OSPF. Na rysunku zaznaczone są adresy portów ruterów (kart sieciowych zainstalowanych w środowisku Linux) oraz adresy IP sieci tworzących obszary OSPF. Poniżej omówiona zostanie konfiguracja ruterów p253tm i Kokon, czytelnikowi pozostawione jest (jako ćwiczenie) napisanie analogicznej konfiguracji dla rutera p253gk.

Zawartość plików konfiguracyjnych zebra.conf dla ruterów p253tm i Kokon jest następująca:

dla p253tm:

```
!zebra configuration file

hostname zebra-p253nt
password z
ip route 10.0.10.0/24 10.0.3.1
log file /var/log/zebra/zebra.log
```

dla Kokon:

```
! Zebra configuration saved from vty
! 2000/11/23 14:32:59
hostname zebra-Kokon
password z
ip route 10.0.7.0/24 10.0.2.1
log file /var/log/zebra/zebra.log
```

W plikach tych zamieszczone zostały podstawowe polecenia:

- hostname – definiujące nazwę rutera;
- password – określające hasło dla logującego się z terminala operatora (administratora) rutera;
- ip route – definiujące statyczną ścieżkę (do sieci 10.0.10.0 dla p253tm i sieci 10.0.7.0 dla rutera Kokon;
- log file – określające położenie pliku zebra.log, gdzie umieszczane będą komunikaty związane z działaniem procesu zebra.

Podobnie jak dla menadżera zebra, w plikach konfiguracyjnych protokołu RIP (ripd.conf), zawarte zostały jedynie wybrane polecenia.

dla p253tm:

```
! RIPd configuration file
!hostname ripd-p253tm
password z
router rip
network 172.16.50.0/24
network 10.0.3.0/24
neighbor 172.16.50.126
redistribute static
route 10.0.5.0/24
timers basic 30 150 120
log file /var/log/zebra/ripd.log
```

dla Kokon:

```
! RIPd configuration file
!hostname ripd-Kokon
password z
router rip
network 172.16.50.0/24
network 10.0.2.0/24
neighbor 172.16.50.118
redistribute static
route 10.0.1.0/24
timers basic 30 150 120
log file /var/log/zebra/ripd.log
```

gdzie:

- **router rip** – odblokowuje proces RIP na ruterze;
- **network** – odblokowuje przyjmowanie komunikatów RIP z wyszczególnionej w poleceniu sieci;
- **neighbor** – określa sąsiada, z którym wymieniane będą tablice rutingu;
- **redistribute static** – wymusza przekazywanie do sąsiada informacji o trasie statycznej (zadeklarowanej w zebra.conf);
- **route** – określa sieć docelową, do której wiedzie droga przez dany ruter;
- **timers basic** - ustala wartości krytyczne czasomierzy protokołu RIP.

Po wykonaniu czynności konfiguracyjnych należy uruchomić proces zebra i ripd.

Pierwszym krokiem na drodze do zarządzania rutingiem z poziomu terminala jest zalogowanie się na wybranym ruterze. Poniżej podany jest sposób zalogowania się do rutera uruchomionego na lokalnym hoście (ruter Kokon).

```
[root@Kokon init.d]# telnet localhost 2601
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is zebra (version 0.88)
Copyright 1996-2000 Kunihiro Ishiguro


User Access Verification

Password:
zebra-Kokon> en
zebra-Kokon#
  configure  Configuration from vty interface
  copy       Copy configuration
  debug      Debugging functions
  disable    Turn off privileged mode command
  end        End current mode and change to enable mode.
  exit       Exit current mode and down to previous mode
  help       Description of the interactive help system
  list       Print command list
  no         Negate a command or set its defaults
  quit       Exit current mode and down to previous mode
  service    Set up miscellaneous service
  show       Show running system information
  terminal   Terminal configuration setup
  who        Display who is on vty
  write      Write running configuration to memory, network, or terminal
zebra-Kokon#
```

Wejście w tryb uprzywilejowany poprzedzone jest wydaniem polecenia „enable" (na widocznym powyżej ekranie polecenie **en**) wraz z odpowiednim hasłem dla administratora (jeśli takie było ustawione w pliku konfiguracyjnym), w tym przypadku dla demona zebra. Wylistowana tutaj została lista komend, w dużej mierze pokrywających się z komendami systemu IOS Cisco. Podobnie jak w systemie IOS możliwe jest wydawanie poleceń w formie skróconej (np. **enable** jest równoważne poleceniu **en**).

Na wydruku poniżej przedstawione są komendy dostępne z poziomu terminala po zalogowaniu się do demona ripd. Jak pamiętamy prezentowany system jest zdekomponowany na kilka procesów dostępnych oddzielnie przez wirtualne terminale i odpowiednie porty.

```
ripd-Kokon#
  clear      Reset functions
  configure  Configuration from vty interface
  copy       Copy configuration
  debug      Debugging functions
  disable    Turn off privileged mode command
  end        End current mode and change to enable mode.
  exit       Exit current mode and down to previous mode
  help       Description of the interactive help system
  list       Print command list
  no         Negate a command or set its defaults
  quit       Exit current mode and down to previous mode
  service    Set up miscellaneous service
  show       Show running system information
  terminal   Terminal configuration setup
  who        Display who is on vty
  write      Write running configuration to memory, network, or terminal
ripd-Kokon#
```

Administrator systemu może skonfigurować system edytując odpowiednie pliki konfiguracyjne lub też wykorzystując opcję „configure", która daje możliwość konfigurowania wprost z linii poleceń terminala. Zasada ta dotyczy wszystkich omawianych tutaj procesów, jak również menadżera o nazwie zebra.

Podany ekran pokazuje konfigurację protokołu RIP dla rutera o nazwie Kokon po wydaniu polecenia „show ip protocol" z poziomu terminala demona ripd.

```
ripd-Kokon# sh ip pro
Routing Protocol is "rip"
  Sending updates every 30 seconds with +/-50%, next due in 7 seconds
  Timeout after 150 seconds, garbage collect after 120 seconds
  Outgoing update filter list for all interface is not set
  Incoming update filter list for all interface is not set
  Default redistribution metric is 1
  Redistributing: static
  Default version control: send version 2, receive version 2
    Interface        Send  Recv   Key-chain
    eth0             2     2
  Routing for Networks:
    10.0.2.0/24
    172.16.50.0/24
    172.16.50.118
  Routing Information Sources:
    Gateway          BadPackets BadRoutes  Distance Last Update
    172.16.50.118             0         0       120   00:00:23
  Distance: (default is 120)
ripd-Kokon#
```

Jak widać uaktualnienia RIP wysyłane są co 30 sekund (następne uaktualnienie zostanie wysłane za 7s), co określone jest przez wartość czasomierza aktualizacji. Domyślna wartość czasomierza błędu zmieniona została ze 180s na 150s (zapis Timeout after 150 seconds), czasomierz usuwania

z domyślnej wartości 270s został ustawiony na 120s (garbage collect after 120seconds), co jest zgodne z konfiguracją w pliku ripd.conf.

Przy konfiguracji protokołu nie zastosowano żadnych reguł filtrowania dla pakietów z trasami napływających do rutera i wychodzących z niego. Zastosowana została (domyślnie) wersja RIP 2 protokołu. Podane zostały również adresy IP obsługiwanych przez protokół podsieci. Przypisana protokołowi odległość administracyjna to 120. Wartość redistribution metric równa 1 odpowiadająca kosztowi pojedynczego skoku (jeden ruter pośredni) na drodze do sieci docelowej. Ostatnia otrzymana aktualizacja od sąsiada o adresie 172.16.50.118 miała miejsce 23s przed wylistowaniem informacji na temat konfiguracji protokołu.

„Show ip rip" jest poleceniem umożliwiającym wylistowanie tras wymienianych przez proces RIP. Obok sieci docelowych podane są tutaj adresy następnego skoku, liczba skoków, źródło informacji o danej trasie i czas ostatniej aktualizacji (jak dawno zapis był aktualizowany). Poniżej przedstawione są informacje na temat ścieżek w sieci skompletowane przez rutery Kokon i p253tm po pewnym czasie działania procesów RIP.

```
ripd-Kokon# sh ip ri
Codes: R - RIP, C - connected, O - OSPF, B - BGP

   Network          Next Hop        Metric From            Time
R  10.0.1.0/24                          1
R  10.0.5.0/24      172.16.50.118       2 172.16.50.118    02:20
S  10.0.7.0/24      10.0.2.1            1
R  10.0.10.0/24     172.16.50.118       2 172.16.50.118    02:20
ripd-Kokon#
```

```
ripd-p253nt# sh ip rip
Codes: R - RIP, C - connected, O - OSPF, B - BGP

   Network          Next Hop        Metric From            Time
R  10.0.1.0/24      172.16.50.126       2 172.16.50.126    02:16
R  10.0.5.0/24                          1
R  10.0.7.0/24      172.16.50.126       2 172.16.50.126    02:16
S  10.0.10.0/24     10.0.3.1            1
ripd-p253nt#
```

W każdej chwili dostępna jest funkcja śledzenia wymienianych pakietów. Operator rutera może obejrzeć wysyłane w sieć pakiety RIP sięgając do pliku ripd.log lub oglądać je na ekranie (dokonując odpowiedniego przekierowania strumienia komunikatów). Wymieniane w trakcie działania

protokołu RIP komunikaty między ruterami p253tm i Kokon przedstawia poniższy ekran.

```
2001/01/08 14:14:28 RIP: SEND UPDATE to eth0 ifindex 2
2001/01/08 14:14:28 RIP: multicast announce on eth0
2001/01/08 14:14:28 RIP: update routes on interface eth0 ifindex 2
2001/01/08 14:14:28 RIP: SEND to socket 11 port 520 addr 224.0.0.9
2001/01/08 14:14:28 RIP: SEND RESPONSE version 2 packet size 44
2001/01/08 14:14:28 RIP:   10.0.5.0/24 -> 0.0.0.0 family 2 tag 0 metric 1
2001/01/08 14:14:28 RIP:   10.0.10.0/24 -> 0.0.0.0 family 2 tag 0 metric 1
2001/01/08 14:14:28 RIP: update routes to neighbor 172.16.50.126
2001/01/08 14:14:28 RIP: SEND to socket 5 port 520 addr 172.16.50.126
2001/01/08 14:14:28 RIP: SEND RESPONSE version 2 packet size 44
2001/01/08 14:14:28 RIP:   10.0.5.0/24 -> 0.0.0.0 family 2 tag 0 metric 1
2001/01/08 14:14:28 RIP:   10.0.10.0/24 -> 0.0.0.0 family 2 tag 0 metric 1
2001/01/08 14:14:28 RIP: ignore packet comes from myself
2001/01/08 14:14:53 RIP: update timer fire!
2001/01/08 14:14:53 RIP: SEND UPDATE to eth0 ifindex 2
2001/01/08 14:14:53 RIP: multicast announce on eth0
2001/01/08 14:14:53 RIP: update routes on interface eth0 ifindex 2
2001/01/08 14:14:53 RIP: SEND to socket 11 port 520 addr 224.0.0.9
2001/01/08 14:14:53 RIP: SEND RESPONSE version 2 packet size 44
2001/01/08 14:14:53 RIP:   10.0.5.0/24 -> 0.0.0.0 family 2 tag 0 metric 1
2001/01/08 14:14:53 RIP:   10.0.10.0/24 -> 0.0.0.0 family 2 tag 0 metric 1
2001/01/08 14:14:53 RIP: update routes to neighbor 172.16.50.126
2001/01/08 14:14:53 RIP: SEND to socket 5 port 520 addr 172.16.50.126
2001/01/08 14:14:53 RIP: SEND RESPONSE version 2 packet size 44
2001/01/08 14:14:53 RIP:   10.0.5.0/24 -> 0.0.0.0 family 2 tag 0 metric 1
2001/01/08 14:14:53 RIP:   10.0.10.0/24 -> 0.0.0.0 family 2 tag 0 metric 1
2001/01/08 14:14:53 RIP: ignore packet comes from myself
```

Oczywiście komunikaty te mogą być filtrowane i zbierane w pliku logów (ripd.log). Zawartość takiego pliku jest bardzo cenna z punktu widzenia diagnozowania problemów w sieci wynikających z niewłaściwego skonfigurowania protokołu rutingu.

Kolejne pliki konfiguracyjne – ospfd.conf dotyczą protokołu OSPF uruchomionego na ruterach p253tm i Kokon obok działających na nich procesach RIP.

dla p253tm:

```
!OSPFd configuration file
!hostname ospfd-p253tm
password z
router ospf
network 172.16.50.0/24 area 0
network 10.0.3.0/24 area 2
redistribute kernel
redistribute static
redistribute connected
log file /var/log/zebra/ospfd.log
```

dla Kokon:

```
! OSPFd configuration file
hostname ospfd-Kokon
password zebra
router ospf
network 172.16.50.0/24 area 0
network 10.0.2.0/24 area 3
redistribute kernel
redistribute static
redistribute connected
log file /var/log/zebra/ospfd.log
```

Najważniejsze z poleceń to:

- **router ospf** – uruchamiające (odblokowujące proces OSPF) na danym ruterze;
- **network** – określenie adresów: dla obszaru zerowego i obszaru włączanego przez ruter do szkieletu;
- **redistribute** – wyspecyfikowanie rodzaju redystrybuowanych przez ruter tras (statycznych, kernelowych, adresów osiągalnych bezpośrednio przez interfejs rutera).

Po skonfigurowaniu w zakresie podstawowym ruterów i uruchomieniu procesu **ospfd** rozpoczyna się etap wymiany komunikatów „odkrywających" obecność sąsiadów i elekcji rutera desygnowanego (DR-a) oraz zapasowego rutera desygnowanego (BDR-a). Komunikaty te możemy podglądać ustawiając śledzenie pakietów protokołu OSPF. Wynik śledzenia uruchomionego na ruterze Kokon przedstawiony jest poniżej.

```
2001/01/08 14:27:41 OSPF: Hello received from [172.16.50.118] via [eth0]
2001/01/08 14:27:41 OSPF: Packet 172.16.50.118 [Hello:RECV]: Options *|*|-|-|-|-|E|*
2001/01/08 14:27:48 OSPF: Z: make_hello: options: 2, int: eth0
2001/01/08 14:27:48 OSPF: Hello sent to [224.0.0.5] via [eth0].
2001/01/08 14:27:51 OSPF: Hello received from [172.16.50.118] via [eth0]
2001/01/08 14:27:51 OSPF: Packet 172.16.50.118 [Hello:RECV]: Options *|*|-|-|-|-|E|*
2001/01/08 14:27:58 OSPF: Z: make_hello: options: 2, int: eth0
2001/01/08 14:27:58 OSPF: Hello sent to [224.0.0.5] via [eth0].
2001/01/08 14:28:01 OSPF: Hello received from [172.16.50.118] via [eth0]
2001/01/08 14:28:01 OSPF: Packet 172.16.50.118 [Hello:RECV]: Options *|*|-|-|-|-|E|*
2001/01/08 14:28:08 OSPF: Z: make_hello: options: 2, int: eth0
2001/01/08 14:28:08 OSPF: Hello sent to [224.0.0.5] via [eth0].
2001/01/08 14:28:11 OSPF: Hello received from [172.16.50.118] via [eth0]
2001/01/08 14:28:11 OSPF: Packet 172.16.50.118 [Hello:RECV]: Options *|*|-|-|-|-|E|*
2001/01/08 14:28:18 OSPF: Z: make_hello: options: 2, int: eth0
2001/01/08 14:28:18 OSPF: Hello sent to [224.0.0.5] via [eth0].
2001/01/08 14:28:21 OSPF: Hello received from [172.16.50.118] via [eth0]
2001/01/08 14:28:21 OSPF: Packet 172.16.50.118 [Hello:RECV]: Options *|*|-|-|-|-|E|*
2001/01/08 14:28:28 OSPF: Z: make_hello: options: 2, int: eth0
2001/01/08 14:28:28 OSPF: Hello sent to [224.0.0.5] via [eth0].
2001/01/08 14:28:31 OSPF: Hello received from [172.16.50.118] via [eth0]
2001/01/08 14:28:31 OSPF: Packet 172.16.50.118 [Hello:RECV]: Options *|*|-|-|-|-|E|*
2001/01/08 14:28:38 OSPF: Z: make_hello: options: 2, int: eth0
2001/01/08 14:28:38 OSPF: Hello sent to [224.0.0.5] via [eth0].
```

Polecenie „show ip ospf interface" umożliwia wgląd w status interfejsów rutera, jak również obejrzenie stanu jaki ustalił się po okresie wymiany pakietów *'hello'*.

```
ospfd-Kokon# sh ip os in
lo is up, line protocol is up
  OSPF not enabled on this interface
eth0 is up, line protocol is up
  Internet Address 172.16.50.126/24, Area 0.0.0.0
  Router ID 172.16.50.126, Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State Backup, Priority 1
  Designated Router (ID) 172.16.50.118, Interface Address 172.16.50.118
  Backup Designated Router (ID) 172.16.50.126, Interface Address 172.16.50.126
  Timer intarvals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:08
  Neighbor Count is 1, Adjacent neighbor count is 1
eth1 is up, line protocol is up
  OSPF not enabled on this interface
ospfd-Kokon# sh ip os in
```

W naszym przykładzie ruterem desygnowanym wybrany został ruter o adresie interfejsu 172.16.50.118 (p253-tm), a zapasowym ruter 172.16.50.126 (Kokon). Jak widać dla rutera Kokon protokół OSPF działa jedynie na interfejsie łączącym go ze szkieletem sieci. Analogicznie przedstawia się stan rutera p253-tm. Sieci 10.0.3.0 i 10.0.2.0, przedstawione na rysunku 7 są sieciami „martwymi" bez aktywnych wewnętrznych ruterów.

Uzupełnieniem tych informacji są dane na temat aktywności interfejsów rutera dla kolejnych obszarów (polecenie „sh ip ospf")

```
ospfd-Kokon# show ip ospf
 OSPF Routing Process, Router ID: 172.16.50.126
 Supports only single TOS (TOS0) routes
 This implementation conforms to RFC2328
 RFC1583Compatibility flag is disabled
 SPF schedule delay 5 secs, Hold time between two SPFs 10 secs
 Refresh paramters:
   Group Limit 10, Refresh Per Slice 25, Age difference 3
 This router is an ASBR (injecting external routing information)
 Number of external LSA 3
 Number of areas attached to this router: 2

 Area ID: 0.0.0.0 (Backbone)
   Number of interfaces in this area: Total: 1, Active: 1
   Number of fully adjacent neighbors in this area: 1
   Area has no authentication
   SPF algorithm executed 5 times
   Number of LSA 3

 Area ID: 0.0.0.3
   Shortcutting mode: Default, S-bit consensus: ok
   Number of interfaces in this area: Total: 0, Active: 0
   Number of fully adjacent neighbors in this area: 0
   Area has no authentication
   Number of full virtual adjacencies going through this area: 0
   SPF algorithm executed 5 times
   Number of LSA 1

ospfd-Kokon# ▌
```

oraz „show ip ospf database", przedstawiający pełen obraz na temat wymienianych przez ruter tras.

```
ospfd-Kokon# sh ip os dat

       OSPF Router with ID (172.16.50.126)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
172.16.50.118   172.16.50.118   1703 0x8000000e 0xc0bd 2
172.16.50.126   172.16.50.126   1697 0x8000000c 0x9fc9 2

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
172.16.50.118   172.16.50.118   1703 0x80000002 0xb902

                Router Link States (Area 0.0.0.3)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
172.16.50.126   172.16.50.126   1746 0x80000002 0xfb89 0

                Type-5 AS External Link States

Link ID         ADV Router      Age  Seq#       CkSum  Route
10.0.1.0        172.16.50.126   1693 0x80000006 0x2b16 E2 10.0.1.0/24 [0x0]
10.0.7.0        172.16.50.126   1697 0x80000006 0x7fae E2 10.0.7.0/24 [0x0]
10.0.10.0       172.16.50.118    909 0x80000005 0x9d95 E2 10.0.10.0/24 [0x0]
```

Wylistowane powyżej dane zawierają pełną informację o trasach wiodących do każdego obszaru (Router Links, Net Links) oraz trasach zgłaszanych do szkieletu wiodących do innych sieci (External Links), które nie wchodzą w skład wymienianych tutaj obszarów OSPF tzw. trasach zewnętrznych osiągalnych przez dany ruter. Każda trasa opisywana jest przez jej adres IP (Link ID), ruter rozgłaszający tę trasę (ADV Router), wiek trasy (Age), numer sekwencyjny (Seq#), sumę kontrolną (CkSum), liczbę interfejsów, na których uruchomiono OSPF w ruterze (Link count). Numer sekwencyjny używany jest tutaj do wykrywania łączy starych, zdublowanych lub spoza sekwencji ogłoszeń. Suma kontrolna, w zasadzie najważniejszy element opisujący daną trasę, w pełni ją identyfikuje i jest wykorzystywana w procesie konfrontowania posiadanych danych z danymi rutera desygnowanego (zapasowego rutera desygnowanego). Jest to sposób na stwierdzenie, czy posiadane przez ruter dane są aktualne.

Ostatnim konfigurowanym protokołem był protokół BGP. Protokół BGP przeznaczony jest do wymiany tablic rutingu między ruterami stojącymi na granicy dużych (zawierających powyżej 50 ruterów) systemów autonomicznych. Uruchamianie go w tak małym, realnym środowisku sieciowym byłoby posunięciem przesadnym.
Pliki konfiguracyjne bgpd.conf ruterów p253-tm i Kokon przedstawione są poniżej.

dla p253tm:

```
! BGPd configuration file
!hostname bgpd-p253tm
password z
router bgp 3
neighbor 172.16.50.126 remote-as 2
redistribute static
redistribute rip
log file /var/log/zebra/bgpd.log
```

dla Kokon:

```
! BGPd configuration file
hostname bgpd-Kokon
password z
router bgp 2
neighbor 172.16.50.118 remote-as 3
redistribute static
redistribute rip
log file /var/log/zebra/bgpd.log
```

Niezbędne minimum to:

- odblokowanie procesu bgp na ruterze – polecenie **ruter bgp**;
- określenie sąsiada – polecenie **neighbor**;
- określenie rodzajów rozgłaszanych tras – polecenie **redistribute**.

Podstawowym testem działania procesu BGP jest wydanie polecenia „show ip bgp neighbors" po zalogowaniu się do systemu z poziomu terminala dla demona bgpd (telnet Kokon 2605). Efekt wydania polecenia przedstawiony jest na ekranie poniżej.

```
bgpd-Kokon# show ip bgp neighbors
172.16.50.118
  BGP version: 4
  Remote AS: 3, Local AS: 2, Link type: EBGP
  Remote router ID: 172.16.50.118, Local router ID: 172.16.50.126
  Remote address: 172.16.50.118
  Local address: 172.16.50.126
  Nexthop: 172.16.50.126
  Nexthop global: ::  Nexthop local: ::
  BGP connection: non shared network
  BGP Status: Established, Old status: OpenConfirm
  Received packets: 80 Send packets: 71
  Keepalive: 60 Holdtime: 180
  Elapsed time after last connection: 00:57:14
  Read thread: on  Write thread: off
  IPv4 prefix count unicast/multicast: 1/0
  IPv6 prefix count unicast/multicast: 0/0
  Neighbor capabilities:
    Route refresh: advertised and received
    Address family IPv4 unicast
    Negotiated for IPv4 unicast
bgpd-Kokon# ▌
```

Ostatnim testem na poprawność konfiguracji i działania ruterów jest obejrzenie tablic rutingu i skonfrontowanie ich z własnymi oczekiwaniami. Po zalogowaniu się na ruterze z terminala zebry (telnet p253-tm 2601) systemowa tablica rutingu może być wylistowana poleceniem „show ip route". Kolejne dwa zrzuty ekranu

przedstawiają tablice rutingu ruterów p253tm i Kokon skompletowane po pewnym czasie działania na nich trzech protokołów: RIP, OSPF i BGP. Odniesienie otrzymanych wyników do konfiguracji sieci testowej (rysunek 7) pozostawia się czytelnikowi.

```
zebra-p253nt# sh ip ro
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       B - BGP, * - FIB route.

K* 0.0.0.0/0  sh ip ro          eth0 (2) 172.16.50.254
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       B - BGP, * - FIB route.

K* 0.0.0.0/0                    eth0 (2) 172.16.50.254
O* 10.0.1.0/24 [110/10]         eth0 (2) 172.16.50.126
R  10.0.1.0/24 [120/2]          eth0 (2) 172.16.50.126
O* 10.0.2.0/24 [110/20]         eth0 (2) 172.16.50.126
C* 10.0.3.0/24                  eth0 (2) direct
B* 10.0.5.0/24 [20/0]           eth0 (2) 172.16.50.126
B* 10.0.7.0/24 [20/0]           eth0 (2) 172.16.50.126
O  10.0.7.0/24 [110/20]         eth0 (2) 172.16.50.126
R  10.0.7.0/24 [120/2]          eth0 (2) 172.16.50.126
S* 10.0.10.0/24 [1/0]           eth0 (2) 10.0.3.1
C* 127.0.0.0/8                    lo (1) direct
C* 172.16.50.0/24                eth0 (2) direct
zebra-p253nt# ▌


zebra-Kokon# sh ip ro
  bgp         Border Gateway Protocol (BGP)
  connected   Connected
  kernel      Kernel
  ospf        Open Shortest Path First (OSPF)
  rip         Routing Information Protocol (RIP)
  static      Static routes
  A.B.C.D     Network in the IP routing table to display
  A.B.C.D/M   IP prefix <network>/<length>, e.g., 35.0.0.0/8
  <cr>
zebra-Kokon# sh ip ro
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       B - BGP, * - FIB route.

K* 0.0.0.0/0                    eth0 (2) 172.16.50.254
C* 10.0.1.0/24                  eth1 (3) direct
C* 10.0.2.0/24                  eth0 (2) direct
O* 10.0.3.0/24 [110/20]         eth0 (2) 172.16.50.118
R* 10.0.5.0/24 [120/2]          eth0 (2) 172.16.50.118
S* 10.0.7.0/24 [1/0]            eth0 (2) 10.0.2.1
B* 10.0.10.0/24 [20/0]          eth0 (2) 172.16.50.118
O  10.0.10.0/24 [110/20]        eth0 (2) 172.16.50.118
R  10.0.10.0/24 [120/2]         eth0 (2) 172.16.50.118
C* 127.0.0.0/8                    lo (1) direct
C* 172.16.50.0/24                eth0 (2) direct
zebra-Kokon# ▌
```

5. Podsumowanie

Opracowanie stanowi wstęp do badania najpopularniejszych obecnie protokołów rutingu, wykorzystywanych zarówno w małych (z niewielką liczbą ruterów) sieciach LAN, jak i w dużych systemach autonomicznych sieci Internet. Wybór protokołu rutingu i jego właściwe skonfigurowanie jest ciekawym zadaniem projektowym. Przemyślana konfiguracja sieci owocuje wysoką wydajnością, a przede wszystkim jej niezawodnością działania (ciągłą dostępnością najlepszych ścieżek dla trasowanych pakietów). Znajomość tajników działania protokołów trasowania jest również kluczem do zabezpieczenia sieci przed nadmiernym ruchem o charakterze organizacyjnym i ruchem związanym z wszelkimi próbami niepowołanego dostępu z zewnątrz. Omawiane w pracy rozwiązanie ma duże walory poznawcze. Umożliwia zapoznanie się z oprogramowaniem odpowiadającym na poziomie funkcjonalnym oprogramowaniu ruterów Cisco, firmy nazywanej „szarą eminencją Internetu", której wysokiej klasy produkty dominują ostatnio rynek sprzętu przełączającego i rutującego.

Literatura:

[1] Ballew S. M.: Zarządzanie sieciami IP za pomocą ruterów Cisco, Wydawnictwo RM, Warszawa 1998

[2] Rybaczyk P.: *Podręcznik Inżynierii Internetu*, Wydawnictwo PLJ, Warszawa 1999

[3] Slattery T., Burton B.: *Zaawansowane trasowanie IP w sieciach Cisco*, Wydawnictwo PLJ, Warszawa 2000

[4] Lewis C.: *Routing Cisco TCP/IP dla profesjonalisty*, Wydawnictwo PLJ, Warszawa 1999

[5] Malkin G.: *RIP Version 2*, RFC 2453

[6] Hedrick C. L.: Routing Information Protocol, RFC1058

[7] Moy J.: *OSPF Version 2*, RFC 2328

[8] Rekhter Y., Li T.: A Border Gateway Protocol 4 (BGP-4), RFC 1771

[9] Bates T., Chandra R., Chen E.: BGP Route Reflection - An Alternative to Full Mesh IBGP, RFC 2796

[10] GNU Zebra. Free routing software distributed under GNU General Public License - www.zebra.org

[11] Ishiguro K.: Dokumentacja GNU Zebra - wersja html, http://www3.math.spbu.ru/~nformatycznych, 1999.

Recenzent: dr inż. Janusz Furtak
Praca wpłynęła do redakcji: 10.10.2001