Państwowa Uczelnia im. Stefana Batorego

Konfiguracja routingu statycznego i dynamicznego

Administracja Sieciami Lokalnymi

Wykonał:

Patryk Kaniewski,

2 rok Informatyka stacjonarne

Spis treści

1. Wprowadzenie

Routing jest to proces wybierania ścieżki w sieci lub pomiędzy rożnymi sieciami. Przykładami routingu jest telefonia (dawniej operator fizycznie przełączający wtyczkę z naszym telefonem z innym gniazdem) jak i sieci komputerowe np. Internet. W sieciach komputerowych zazwyczaj jest to rozwiązane za pomocą routerów. Router posiada w sobie informacje o sąsiednich sieciach i na podstawie tych informacji oraz źródła, miejsca docelowego i innych zasad (np. filtrowania) przekazuje pakiety do następnego routera lub komputera.

1.1 Cel ćwiczenia

Konfiguracja tras statycznych i uruchomienie a następnie testowanie i monitorowanie zachowań protokołów routingu dynamicznego oraz zbadanie ich możliwości w różnych typowych i awaryjnych sytuacjach.

1.2 Wymagania wstępne

- Podstawy CLI (np. Linux/Bash)
- Konfiguracja interfejsów sieciowych
- Konfiguracja usługi świadczącej routing dynamiczny (Linux: *quagga*)
- routing na podstawie stanu łącza - OSPFv2 (RFC 2328)
- routing na podstawie wektora odległości - RIPv2 (RFC 2458)
- Narzędzia diagnostyczne sieci (*tcpdump,traceroute,ping*)

1.3 Zakres ćwiczenia

W tym ćwiczeniu wzięte jest zagadnienie routingu w sieciach komputerowych używające modelu TCP/IP na podstawie MAC (Ethernet, WiFi). Pod uwagę wzięte są tylko protokoły bram wewnętrznych (ang. *Interior gateway protocol*) ze względu na zakres materiału w protokołach bram zewnętrznych (protokoły tę obejmują zagadnienia routingu pomiędzy systemami dostawców usług internetowych,miast, krajów, kontynentów oraz zagadnienia takie jak cache, content delivery network w celach usprawniania działania sieci).

Ćwiczenie może być wykonane na maszynach wirtualnych (przykłady na podstawie GNU/Linux Debian 9 i *quagga* 1.2.4)

1.4 Zagadnienia

1. Jak działa routing w sieci komputerowej
2. Routing statyczny
3. Problemy routingu statycznego
4. Routing dynamiczny
   4.1. Wektor odległości (Przykład: RIPv2)
   4.2. Stan łącza (Przykład: OSPFv2)
5. Wady i zalety rozwiązań routingu dynamicznego
6. Dostosowanie rozwiązań do zapotrzebowań w różnych okolicznościach

1.5 Dodatkowe definicje

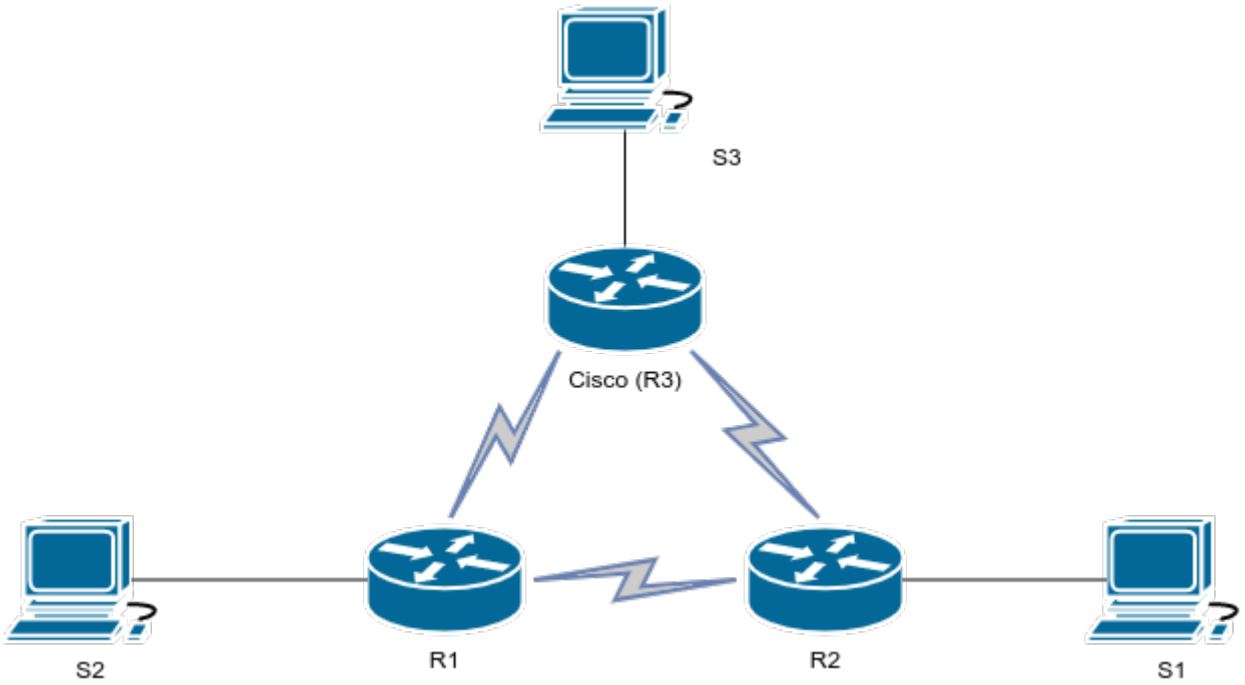
System autonomiczny (ang. Autonomous system) – jest to zbiór sieci połączonych z sobą protokołami routingu bram wewnętrznych. Takie systemy natomiast łączą się z innymi za pomocą protokołów routingu bram zewnętrznych

VLSM (ang. Variable Length Subnet Mask) – jest to odejście od klasowej maski sieci w adresowaniu IP (24,16,8 dla klas C,B,A odpowiednio) i stosowanie dowolnej długości maski aby lepiej wykorzystać przestrzeń adresową (256 może być za mało adresów dla sieci firmowej a 16 tysięcy za dużo, tak samo z połączeniami point-to-point)

Multicast (IP) – jest to metoda wysyłania datagramów do grupy innych hostów. Realizowana jest poprzez specjalną grupę adresów na których zainteresowany host nasłuchuje. W sieciach IP jest to realizowane poprzez adresy zaczynające się od 0xE. Multicast jest używany m.in. do rozprowadzania mediów do dużej ilości urządzeń (IPTV) oraz różnych protokołów routingu

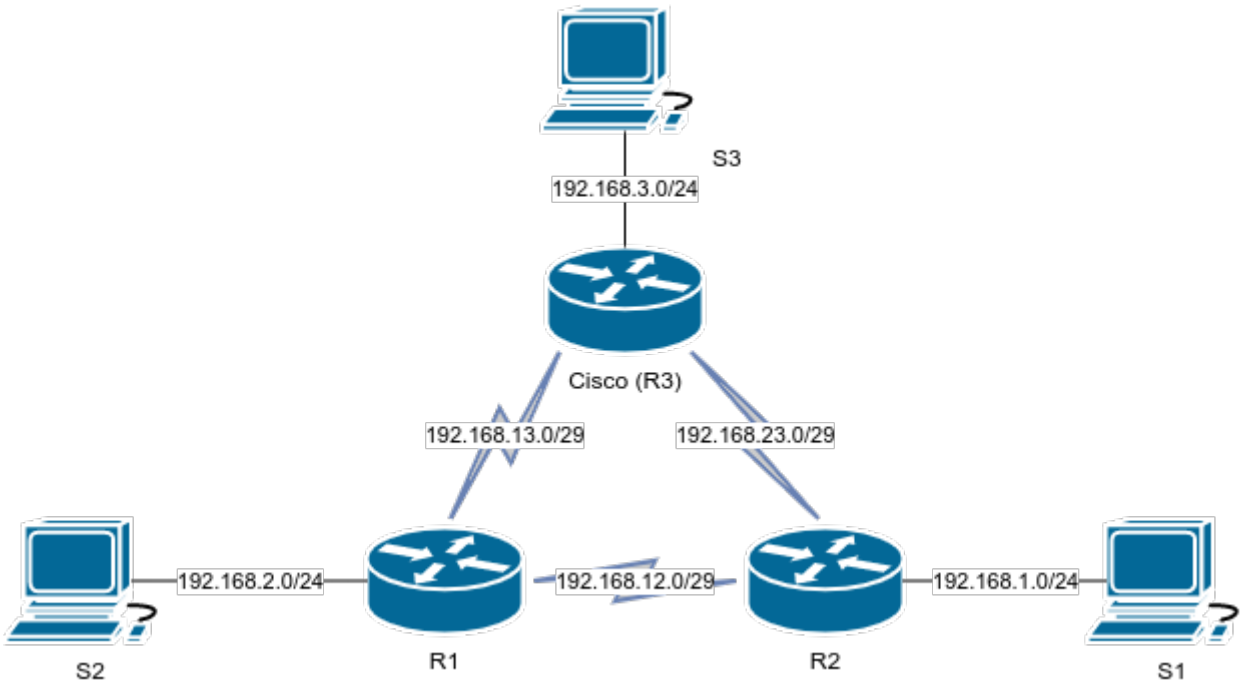
2. Zagadnienia routingu

Fizyczna konfiguracja sieci:



Rys 2.1

Logiczna konfiguracja sieci:



Rys 2.2

Wyjaśnienie:

Routery są łączone w sieciach z długą maska sieciową ponieważ nie jest potrzebne wiele adresów do sieci point-to-point (używana jest maska /29 by ułatwić logowanie poprzez zewnętrznego sniffera). Maszyny robocze są zostawione w domyślnych podsieciach klasy C. Za pomocą tego będzie możliwa weryfikacja jak protokoły radzą sobie z maską sieci różnej długości (VLSM).

Konfiguracja interfejsów:

ręcznie

za pomocą skryptu: https://github.com/p7tryk/administracja/blob/master/network.sh

Instalacja programów diagnostycznych (jeśli potrzeba):

 # apt install tcpdump traceroute ping

Włączenie routingu na R1,R2:

Edycja */etc/sysctl.conf*

\+ net.ipv4.ip_forward = 1

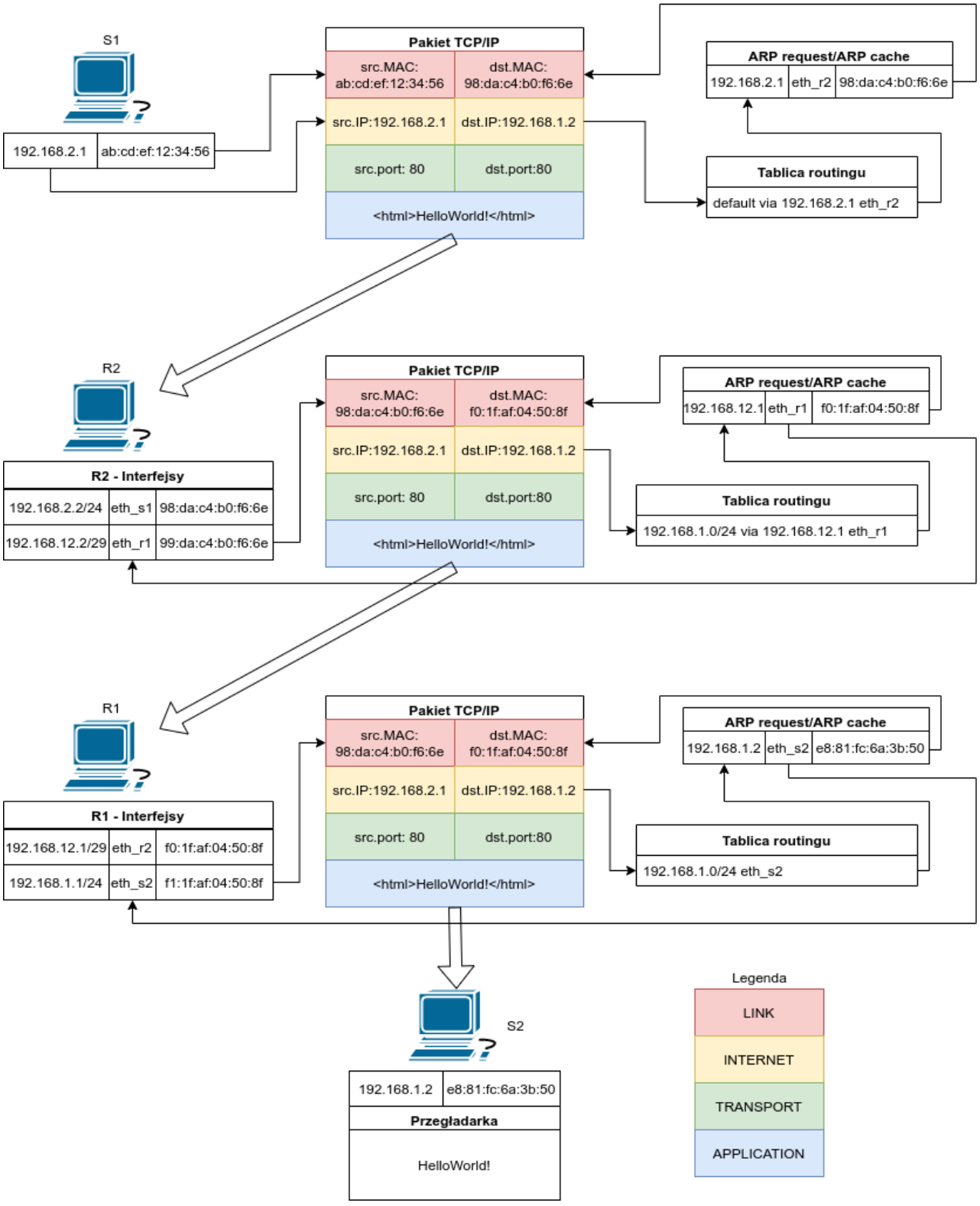
2.1 Routing

Ogólny schemat działania routingu w TCP/IP:

1. Router otrzymuje pakiet TCP/IP
2. Sprawdza adres MAC do którego jest zaadresowany.

3. Jeżeli pokrywa się z adresem urządzenia zdejmuje ramkę Ethernet
4. Sprawdza adres IP do którego jest zaadresowany
5. Jeżeli jest zaadresowany do innej sieci, porównuje adres tej sieci z tablicą routingu
6. Konsultowane jest ARP Cache lub wysyłany jest zapytanie ARP żeby znaleźć adres fizyczny związany z adresem IP
7. Po otrzymaniu adresu fizycznego zakładana jest ramka Ethernet z tym adresem i adresem routera
8. Nowo skonstruowany pakiet wysyłany jest przez interfejs podany przez tablice routingu.

Lab1: Konfiguracja routingu statycznego i dynamicznego



Rys 2.3

Uproszczony schemat działania routingu w TCP/IP

Link,Internet,Transport,Application

2.2 Routing statyczny

Routing statyczny jest to najprostsze rozwiązanie problemu routingu. Zwykle polega na manualnym wprowadzeniu rekordów do tablicy routingu. Administrator sieci musi ręcznie konfigurować routing dla każdego routera. Zwykłe używane w małych sieciach i prostych sieciach.

Specjalną drogą jest droga domyślna (default gateway) trafiają tam wszystkie pakiety których droga nie jest specyficznie wypisana w tablicy routingu

2.2.1 Konfiguracja routingu statycznego (Linux)

Konfiguracja routingu statyczny ze stacji roboczej (S1,S2,S3) do przyłączonego do niej routera.

Tablica routingu przed:

```
192.168.1.0/24 dev ens4 proto kernel scope link src 192.168.1.1
192.168.122.0/24 dev ens3 proto kernel scope link src 192.168.122.249
```

2.2.1.1 ip route add

ip route [-6] add $adress *[opcje] via* $adress *[opcje] dev* $interfejs

```
ip route add 192.168.2.0/24 via 192.168.1.2 dev ens4
```

tablica routingu:

```
192.168.1.0/24 dev ens4 proto kernel scope link src 192.168.1.1
192.168.2.0/24 via 192.168.1.2 dev ens4
192.168.122.0/24 dev ens3 proto kernel scope link src 192.168.122.249
```

2.2.1.2 Default route

ip route add default via $adress dev $interfejs

```
ip r add default via 192.168.1.2 dev ens4
```

```
default via 192.168.1.2 dev ens4
192.168.1.0/24 dev ens4 proto kernel scope link src 192.168.1.1
192.168.122.0/24 dev ens3 proto kernel scope link src 192.168.122.249
```

edycja */etc/network/interfaces*

```
iface ens4 inet static
        address 192.168.1.1
        netmask 255.255.255.0
        gateway 192.168.1.2
```

2.3 Routing dynamiczny

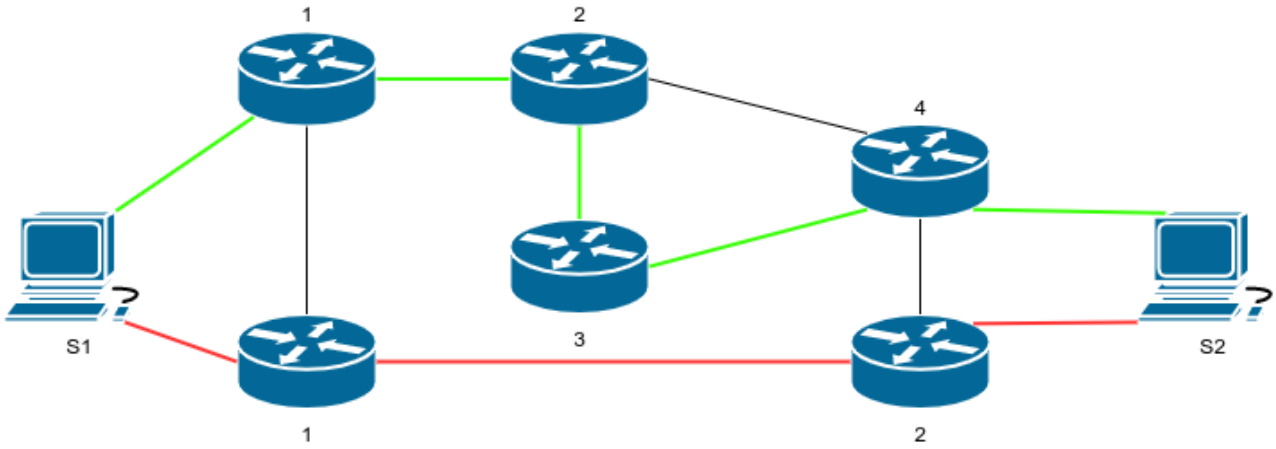
W miarę wzrostu złożoności sieci komputerowych narosła potrzeba bardziej inteligentnego i *dynamicznego* systemu który mógłby reagować na zmiany w ruchu sieciowym czy awarie na połączeniach.

Ogólna idea się nie zmienia (punkt 2.1) ale nasze tablice routingu są teraz generowane i aktualizowane automatycznie.

Dwoma głównymi metodami routingu dynamicznego w sieciach wewnętrznych jest routing oparty o *stan łącza (link-state routing)* i *wektor odległości (distance-vector routing).*

2.3.1 Distance-vector routing

Distance-vector routing polega na wyliczeniu liczby skoków (routerów) przez które pakiet przechodzi na danej drodze. Problemem w takiej sytuacji może być droga z małą przepustowością może być krótsza niż droga z dużą przepustowością (analogia świata rzeczywistego: jechanie przez centrum miasta vs. jechanie szybciej obwodnicą)



Rys 2.4

drogi pomiędzy S1 i S2

droga zielona ma 4 skoki a czerwona 2 skoki

2.3.1.1 RIPv2

RIPv2 jest protokołem opartym na algorytmie Bellman-Ford. Jest rozwinięciem protokołu RIP dodając nowe możliwości (szczególnie VLSM) ale zachowując prostotę protokołu.

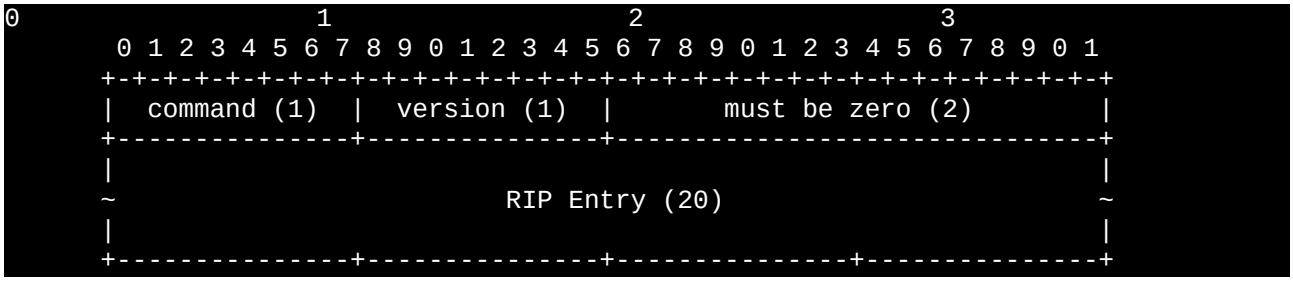
- Protokół jest ograniczony do 15 skoków. Jest to zrobione specjalnie gdyż twórcy uważają że większe sieci nie powinny być konstruowane z tak prostym protokołem
- Protokół używa tylko jednej metryki – ilości skoków. Powoduje to problemy w sieciach z dużymi różnicami przepustowości pomiędzy routerami oraz nie pozwala na balansowanie ruchu sieciowego z dala od słabych połączeń

- Protokół nie rozróżnia pomiędzy sieciami a hostami (jeżeli potrzebne są specjalne drogi dla hostów)

Router tworzy tabele z rekordami drogi do każdej możliwej lokalizacji i przechowuje jej długość i router który jest następnym skokiem. Co jakiś czas router wysyła aktualizacje do swoich sąsiadów. Sąsiad otrzymując taką aktualizacje dodaje do informacji wysłanej liczbę skoków do sąsiada i porównuje ze swoją tabelą, jeżeli jakaś droga jest krótsza to dopisuje ją do swojej tablicy zamiast poprzedniej.

Problemem jest jeżeli topologia sieci się zmieni (np. awaria). RIP rozwiązuje to za pomocą regularnych aktualizacji domyślnie co 30 sekund (*$update*). Jeżeli router nie odpowiada po domyślnie 180 sekundach (*$dead*) to ta droga jest zaznaczana jako nieważną a po następnych 180s (*$garbagecollect*) usuwana z tabeli.

RIP wysyła datagramy UDP na adres multicast `224.0.0.6:520`

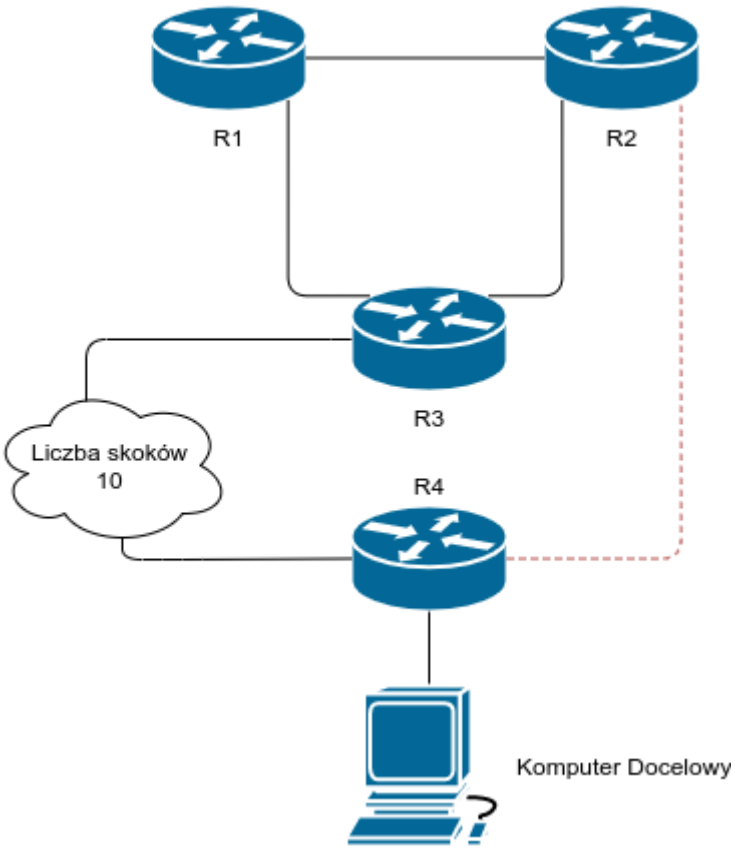


RIP packet RFC2453

```
20:00:19.190558 IP (tos 0xc0, ttl 1, id 42159, offset 0, flags [DF], proto UDP
(17), length 92)
    192.168.13.1.520 > 224.0.0.9.520: [udp sum ok]
        RIPv2, Response, length: 64, routes: 3 or less
          AFI IPv4,     192.168.1.0/24, tag 0x0000, metric: 2, next-hop: self
          AFI IPv4,     192.168.2.0/24, tag 0x0000, metric: 1, next-hop: self
          AFI IPv4,    192.168.12.0/29, tag 0x0000, metric: 1, next-hop: self
        0x0000:  0202 0000 0002 0000 c0a8 0100 ffff ff00
        0x0010:  0000 0000 0000 0002 0002 0000 c0a8 0200
        0x0020:  ffff ff00 0000 0000 0000 0001 0002 0000
        0x0030:  c0a8 0c00 ffff fff8 0000 0000 0000 0001
```

przykład pakietu RIP wyświetlony za pomoca tcpdump -vvv

2.3.1.2 Problemy z distance-vector routing



rys 2.5

pętla routingu w RIP

Rozważmy ten przykład (rys 2.5) z dokumentu RFC. Czerwona droga została przerwana. R2 unieważnia drogę bo R4 nie odpowiada. Jednak R3 i R1 nadal wysyłają że mogą dostać się do sieci docelowej przez R2 (3 skoki). Nawet jeżeli dowiedzą się że ich droga przez R2 nie jest ważna to będą myśleć ze mogą się dostać przez odpowiednio R1 i R3. Będą one liczyć samych siebie coraz wyżej aż alternatywna droga (przez R3) stanie się krótsza. Stanie się to zawsze (chyba że nowa droga nie będzie możliwa – skoki >15) ale może zająć czas kiedy sieć będzie niedostępna. Problem ten jest nazywany liczeniem do *nieskończoności*(naszą nieskończonością na szczęście jest maksymalna liczba skoków = 15)

2.3.1.3 Split Horizon

Jednym z problemów jest to że routery ogłaszają drogę dostępną przez dany router do *tego routera* powoduje to pętle zaufania i może zająć dużo czasu zanim fałszywa droga (pętla) będzie gorsza niż droga alternatywna.

Prostym sposobem jest ominięcie w swoich ogłoszeniach dróg które usłyszeliśmy z hosta do którego wysyłamy.

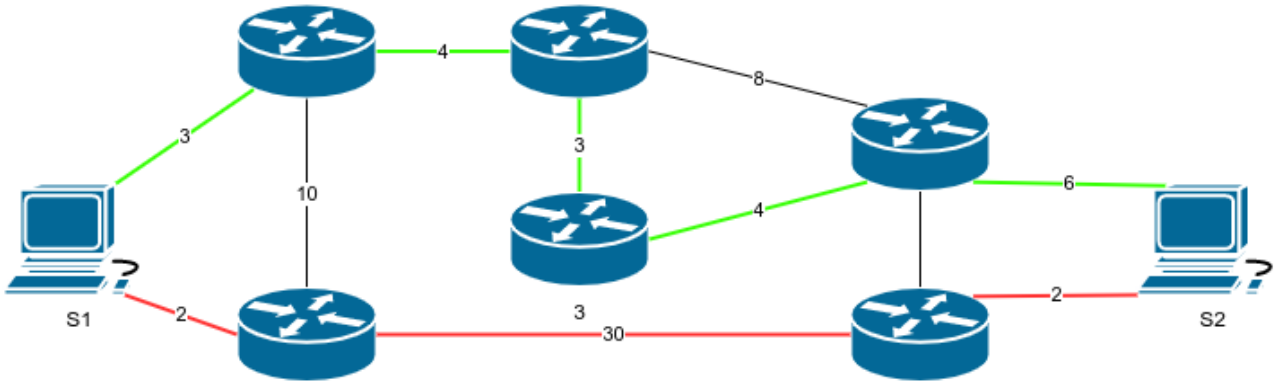
Lepszym rozwiązaniem jest zatruwanie (ang. *poisoning*) takich dróg. Ustawiane są takie drogi na maksymalną ilość skoków (15). Powoduje to na natychmiastowe policzenie do *nieskończoności* i przerwanie pętli. Rozwiązuje to wszystkie pętle 2 routerów ale w systemach z większą liczbą routerów które mogą się połączyć w taką pętlę ten problem nadal może się zdarzyć.

2.3.1.4 Triggered Updates.

Aby rozwiązać problem większych pętli dodano możliwość wysyłania aktualizacji od razu po wykryciu utraty drogi. W najlepszym przypadku natychmiastowo rozprowadzi informacje i zapobiegnie tworzeniu fałszywej drogi. Jeżeli jednak czasowa aktualizacja zdarzy się w podobnym czasie taka droga nadal może powstać. W takiej sytuacji rzadziej może zdarzyć się większa pętla więc ochrona przez split horizon (2.3.1.3) może zdecydowanie obniżyć szanse przestoju w sieci.

2.3.2 Link-state routing

Link-state routing polega na nadaniu danemu routerowi wagi zwykle oparte na szybkości połączenia i urządzenia i/lub obciążenia. Rozwiązuje to problem „obwodnicy” ale zdecydowanie komplikuje działanie takiego systemu.



Rys 2.6

drogi pomiędzy S1 i S2

droga zielona ma koszt 3+4+3+4+6=20

droga czerwona ma koszt 2+30+2=34

2.3.2.1 OSPFv2

OSPFv2 jest implementacją link-state routing dla bram wewnętrznych. Router OSPF ma swój unikalny numer 32bitowy (w systemie jednej bramy wewnętrznej) i przechowuje w sobie mapę topologii sieci wraz z wagami przypisanymi do każdego połączenia. Routery wymieniają pomiędzy sobie informacje o swoich bazach stanu. Za pomocą tej bazy danych (ang. *link-state database*) każdy router buduje drzewo najkrótszych dróg zaczynając od samego siebie (jest

korzeniem tego drzewa). W tym drzewie lisicami są informacje zewnętrzne dla systemu (takie otrzymane z protokołów bram zewnętrznych).

System ten może łączyć sieci w strefy (ang. area) by zredukować ilość informacji które są wysyłane pomiędzy routerami od siebie odległymi (równoznacznie: rozdzielenie systemu autonomicznego na części). Specjalna strefą jest strefa 0 (ang. backbone) która stanowi centrum systemu

Koszt drogi nie jest zdefiniowany jako prędkość a jedynie jako wartość całkowita którą można dostosować do wymagań systemu o zakresie 1-255. Cisco domyślnie używa 10^8/przepustowość, czyli 100Mb/s jest kosztem 1 a 10Mb/s jest kosztem 10.

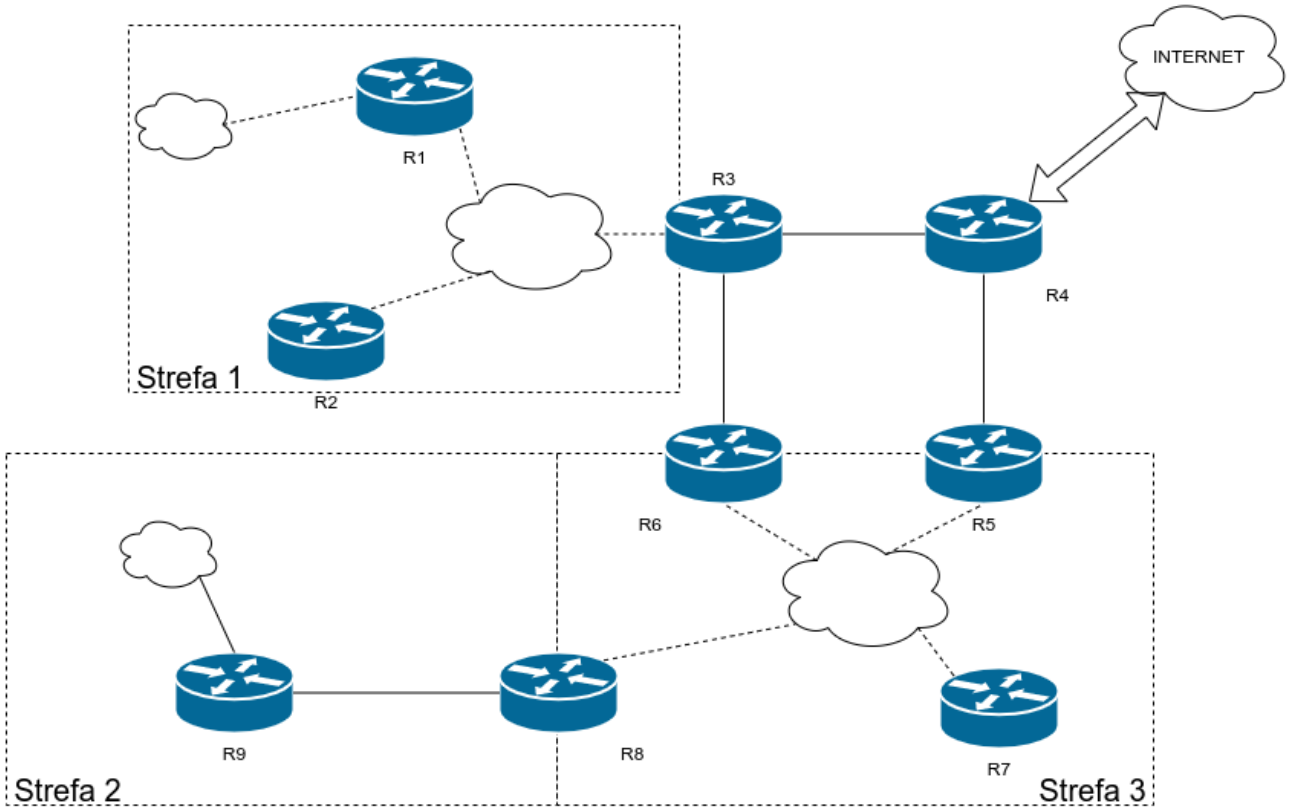
Rodzaje routerów:

wewnętrzne – połączone tylko z sieciami w jednej strefie

granicy stref (ang. *area boundary*) – ma interfejsy podłączone do strefy 0 i co najmniej jednej innej strefy.

granicy systemu autonomicznego(ang. a*utonomic system boundary*) – posiada drogi zewnętrzne do systemu autonomicznego.

podpory (ang. backbone)– połączone z strefą 0.



rys.2.7

R1,R2,R7,R9 – wewnętrzne

R3,R5,R6,R8 – granicy stref

R3,R5,R6,R4 – backbone

R4 - granicy systemów autonomicznych

2.3.2.2 Link State Advertisment

Jest to mechanizm za pomocą którego router ogłasza sieci do niego podłączone.

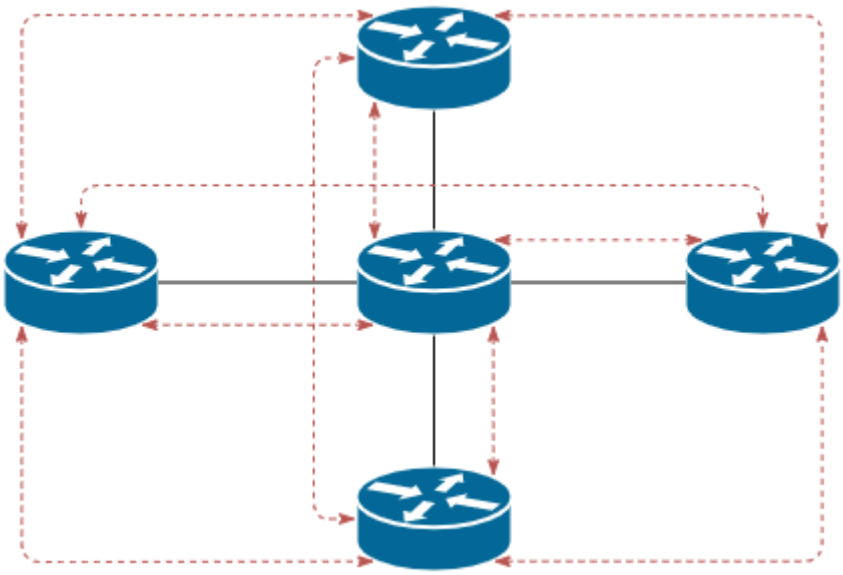
Wyróżnione typy LSA:

1. Router LSA – router ogłasza sieć podłączoną bezpośrednio niego.
2. Network LSA - Designated router ogłasza listę routerów które są z nim związane (propaguje się tylko lokalnie).
3. Summary LSA (area boundary router) – jeżeli ogłoszenia przechodzą przez różne strefy są sumaryzowane zanim przekroczą te bariery.
4. Summary LSA (autonomous area boundary router) – ogłasza drogę do routera z wyjściem z systemu autonomicznego.
5. External LSA (autonomous area boundary router summary) – ogłasza drogi dostępne zewnętrznie poprzez granice systemu autonomicznego.

7. NSSA LSA – działa podobnie jak typ 5 w NSSA(2.3.2.4) po wyjściu ze strefy konwertowane na standardowy typ 5.

2.3.2.3 Designated router

Designated router i backup designatated router (z ang, Dedykowany router) – mechanizm stworzony by ograniczyć liczbę wysyłanych LSA pomiędzy wszystkimi routerami w sieci.

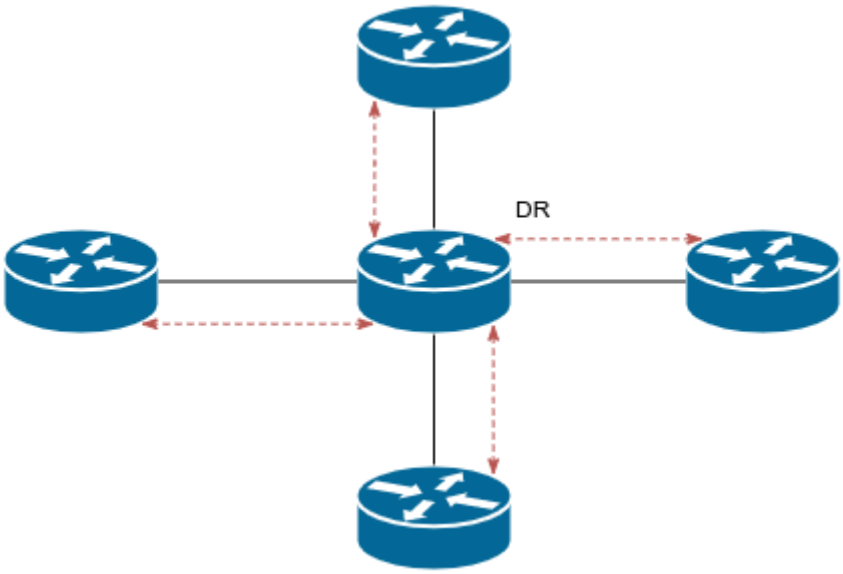


rys. 2.8

routery wymieniające informacje pomiędzy sobą (czerwone linie)

Jeden router jest wyznaczany jako designated router (według priorytetu ustanawianego przez administratora) i LSA są wysyłane tylko pomiędzy routerami podrzędnym a DR.

Jeżeli DR przestanie odpowiadać to zapasowy DR zostanie DR.



Rys 2.9

Routery komunikują się tylko z DR a nie między sobą (czerwone linie)

2.3.2.4 Typy stref

W celu ograniczeniu wielkości tablic routingu strefy mogą być specjalnie wydzielone aby blokować specyficzne LSA zewnętrzne dla tej strefy.

1. Stubby area.

   Zewnętrzne drogi (z poza systemu autonomicznego) nie będą tam rozprowadzane (LSA typ 5).

2. Totally Stubby area.

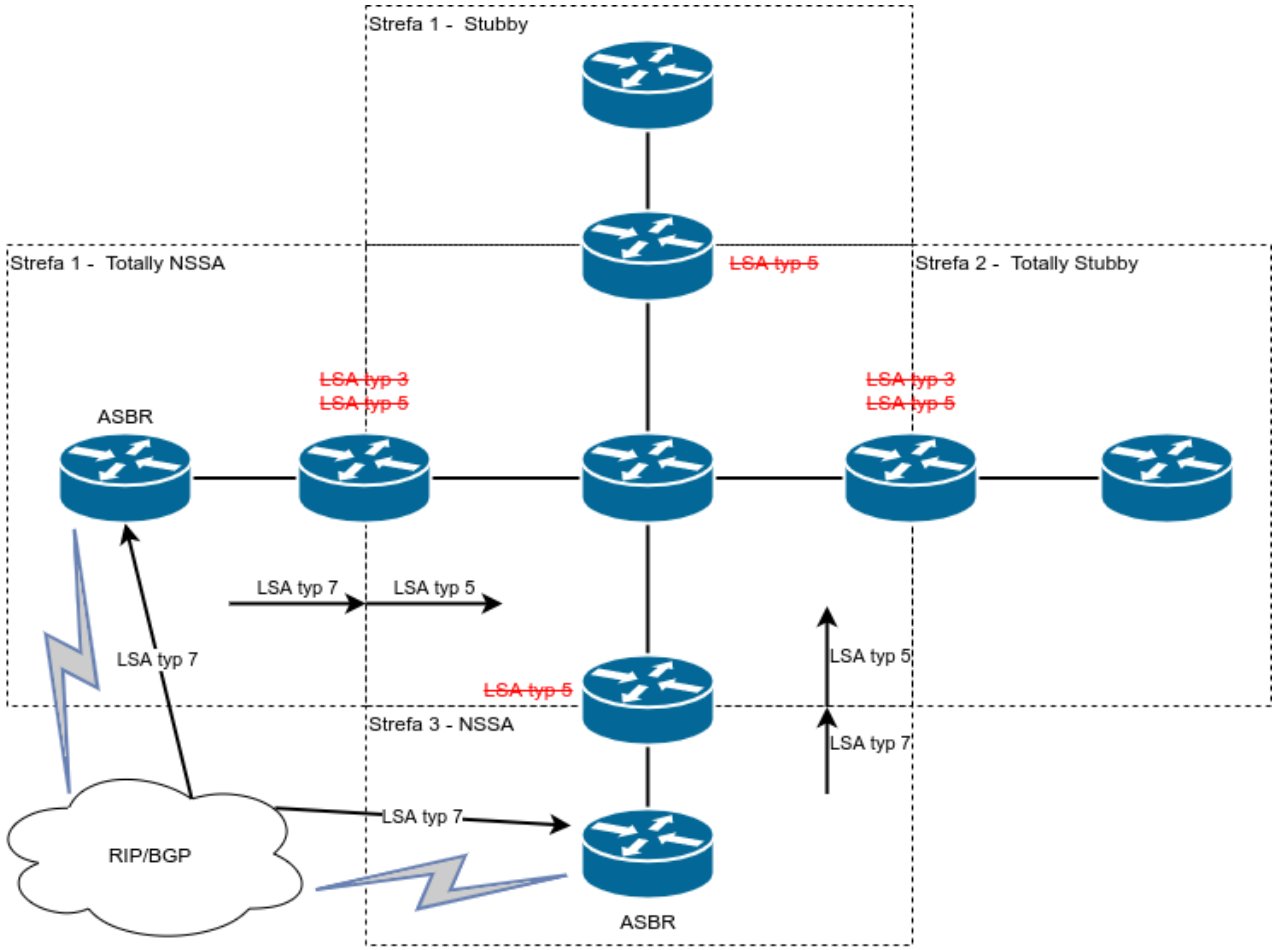
   Stubby area (bez LSA5) + Informacje o drogach pomiędzy strefami nie będą tam rozprowadzanie (LSA typ 3).

3. Not so Stubby area(NSSA).

   Jak Stubby Area, ale może posiadać ASBR a jego drogi są rozprowadzane przez LSA typ 7 (konwertowane na LSA5 przy wyjściu ze strefy).

4. Totally NSSA.

   Jak NSSA, ale może posiadać ASBR a jego drogi są rozprowadzane przez LSA typ 7 (konwertowane na LSA5 przy wyjściu ze strefy).



Rys 2.10

Różne rodzaje stub area, widać że NSSA dostaje drogi zewnętrzne przez ASBR (typ7)

2.3.2.5 Komunikacja pomiędzy routerami OSPF

Inaczej niż inne protokołu routingu OSPF nie używa protokołu powłoki transportu (jak TCP czy UDP). OSPF tworzy własne datagramy IP z portem 89, tworzy on własne 24bajtowe nagłówki i wysyła je na adres multicast 224.0.0.5 (większość) i 224.0.0.6 (designated router)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Version #   |     Type      |         Packet length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Router ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Area ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |            AuType             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Authentication                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
|                       Authentication                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                    OSPF Header RFC 2328 appendix A
```

Version – wersja protokołu (1/2)
Type – jeden z pięciu typów pakietów OSPF

1. Hello
   odpowiedzialny za nawiązywanie i utrzymywanie sąsiadów. Wysyłane regularnie (*hello-interval*)
2. Database Description
   Pakiet synchonizujący bazy sąsiednich routerów, posiadają listę LSA. Jeżeli sąsiadni router zobaczy wpis nowszy niż ma w swojej bazie wysyła LS Request
3. Link State Request
   Zapytanie wysyłane przez router kiedy sąsiedni DD ma LSA nowsze niż we własnej bazie.
4. Link State Update
   Każdy pakiet zawiera routing, i informacje o topologi części sieci
5. Link State Ack
   Pakiet używany do zatwierdzania innych pakietów OSPF. Wiele pakietów może być zatwierdzonych jednym pakietem.

```
10:54:02.723655 IP (tos 0xc0, ttl 1, id 60218, offset 0, flags [none], proto
OSPF (89), length 68)
    192.168.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        Router-ID 192.168.13.1, Backbone Area, Authentication Type: none (0)
        Options [External]
          Hello Timer 10s, Dead Timer 40s, Mask 255.255.255.248, Priority 1
          Designated Router 192.168.12.2, Backup Designated Router 192.168.12.1
          Neighbor List:
            192.168.23.2
```

Przykładowy pakiet hello pokazany za pomocą tcpdump -vvv

2.3.3 Konfiguracja daemona routingu dynamicznego (quagga)

2.3.3.1 Instalacja quagga

 # apt install quagga quagga-doc

edycja /etc/quagga/daemons

+ zebra=yes

+ ospfd=yes

+ ripd=yes

stworzenie pustych plików konfiguracyjnych (inaczej usługi nie zostaną uruchomione *dependencies not met*)

 # touch /etc/quagga/{zebra.conf,ospfd.conf,ripd.conf}

2.3.3.2 Konfiguracja podstawowa (OSPF)

uruchamiamy usługę:

 # systemctl enable ospfd.service //dla systemów z systemd

Łączymy się do servera ospfd za pomocą *telnet* lub *vtysh:*

numer portu możemy poznać za pomoca *nmap localhost | grep ospf*

`$ enable` //aby przejść do trybu uprawnionego

`# configure terminal` //żeby przejść do trybu konfiguracji

`# interface $interface` //żeby przejść do trybu konfiguracji interfejsów

`# router ospf` //żeby przejść do konfiguracji OSPF

`(config-router)# network $adres_sieci/$subnet area $area` //podajemy sieć którą będziemy ogłaszać

Po dodaniu podłączonych sieci do rozgłoszenia wychodzimy z konfiguracji i komendą *write* zapisujemy obecną konfiguracje do pliku.

2.3.3.3 Konfiguracja zaawansowana (OSPF)

`(interface)# ip ospf cost $koszt` //manualne ustawienie kosztu prześcia przez interfejs (domyslnie kalkulowane z trybu prędkości interfejsu (np. 100 BASE-T)

`(interface)# dead-interval $sekundy` //po jakim czasie nie odpowiadający router sąsiedni będzie uznany za martwy

`(interface)# hello-interval $sekundy` //specyfikuje co jak czas pakiet *hello* będzie wysyłany

`(interface)# priority $numer` //priorytet routera

`(interface)# transmit-delay $sekundy` //dodaje czas do pola czasu w pakiecie LSA

`(interface)# retransmit-interval $sekundy` //ustawienie częstotliwości retransmisji dopóki pakiet LSA nie zostanie potwierdzony

2.3.3.4 Konfiguracja podstawowa (RIP)

Usługę możemy uruchomić za pomocą:

 # systemctl enable ripd.service

Połączenie do konsoli ripd jest udostępnione za pomocą *telnet* lub *vtysh:*

numer portu możemy poznać za pomocą *nmap localhost | grep ripd*

```
$ enable          //przejście do trybu uprawnionego
# configure terminal      //przejście do trybu konfiguracji
# router rip          //Przejscie do konfiguracji RIP
# rip ip send version [1,2,1 2]  //żeby wybrać wersje RIP której używamy
# network $adres_sieci/$subnet //Podanie sieci(lub hosta) do rozgłoszenia
# network $nazwainterfejsu  //podajemy sieć którą będziemy ogłaszać (przez interfejs)
```

Po skonfigurowaniu sieci do rozgłoszenia należy wyjść z konfiguracji i komendą write zapisać obecna konfiguracje do pliku

2.3.3.4 Konfiguracja zaawansowana (RIP)

 # distance [1-255] [$adress/$maska] //ustawienie maksymalnego dystansu (liczby skoków) dla sieci

 # timers basic [$update $timeout $garbagecollect] //pokazuje/zmienia liczniki serwera rip

//$update – czas wysyłania aktualizacji do *neigbour*(sąsiadów)

//$timeout – czas do unieważnienia drogi

//$garbagecollect – czas do usunięcia nieważnej drogi

2.3.4 Weryfikacja działania protokołu

RIP tablica routingu (R2):

```
default via 192.168.122.1 dev ens3
192.168.1.0/24 dev ens9 proto kernel scope link src 192.168.1.2
192.168.2.0/24 via 192.168.23.3 dev ens8 proto zebra metric 20
192.168.12.0/29 dev ens4 proto kernel scope link src 192.168.12.2
192.168.13.0/29 via 192.168.23.3 dev ens8 proto zebra metric 20
192.168.23.0/29 dev ens8 proto kernel scope link src 192.168.23.2
```

OSPF tablica routingu (R2):

```
default via 192.168.122.1 dev ens3
192.168.1.0/24 dev ens9 proto kernel scope link src 192.168.1.2
```

```
192.168.2.0/24 via 192.168.12.1 dev ens4 proto zebra metric 20
192.168.12.0/29 dev ens4 proto kernel scope link src 192.168.12.2
192.168.13.0/29 proto zebra metric 20
        nexthop via 192.168.12.1 dev ens4 weight 1
        nexthop via 192.168.23.3 dev ens8 weight 1
192.168.23.0/29 dev ens8 proto kernel scope link src 192.168.23.2
```

Traceroute pomiędzy S1 a S2:

```
traceroute to 192.168.2.2 (192.168.2.2), 30 hops max, 60 byte packets
 1  192.168.1.2 (192.168.1.2)  0.987 ms  0.997 ms  1.028 ms
 2  192.168.12.1 (192.168.12.1)  5.077 ms  4.690 ms  4.168 ms
 3  192.168.2.2 (192.168.2.2)  3.911 ms  3.787 ms  3.774 ms
```

3. Testowanie „dynamiczności” routingu

W tych ćwiczeniach przetestujemy typowe sytuacje z którą routing dynamiczny ma sobie radzić, awaria połączenia(lub urządzenia) sieciowego. Pełne logi i pliki konfiguracyjne dostępne są w $nazwamaszny/$numercwiczenia np. *`$CWD/r3/3.1/ospf`*

3.1 Awaria interfejsu

Używamy tylko jednego daemona (OSPF lub RIP)

1. Weryfikujemy że pakiety pomiędzy S1 i S2 są wysyłane przez R1 i R2 (traceroute)
2. Wywołujemy ping pomiędzy S1 i S2 i odwrotnie
3. Odłączamy interfejs pomiędzy R1 i R2 (*set link down* na jednym z routerów lub fizycznie odłączamy sieć)
4. Logujemy komunikacje pomiędzy routerami
5. Obserwujemy kiedy połączenie powróci i weryfikujemy że używa R3 (traceroute)
6. Podłączamy interfejs
7. Obserwujemy kiedy (i jeżeli) połączenie przełączy się na bezpośrednie R1-R2 (traceroute)

3.1.1 RIP

Traceroute przed odłączeniem sieci:

```
traceroute to 192.168.2.2 (192.168.2.2), 30 hops max, 60 byte packets
 1  192.168.1.2 (192.168.1.2)  0.987 ms  0.997 ms  1.028 ms
 2  192.168.12.1 (192.168.12.1)  5.077 ms  4.690 ms  4.168 ms
 3  192.168.2.2 (192.168.2.2)  3.911 ms  3.787 ms  3.774 ms
```

następnie wywołujemy ping i odłączamy sieć i obserwujemy *ping*:

```
64 bytes from 192.168.2.2: icmp_seq=33 ttl=62 time=2.93 ms
```

```
64 bytes from 192.168.2.2: icmp_seq=34 ttl=62 time=3.64 ms
64 bytes from 192.168.2.2: icmp_seq=35 ttl=62 time=4.01 ms
From 192.168.1.2 icmp_seq=75 Destination Host Unreachable
From 192.168.1.2 icmp_seq=76 Destination Host Unreachable
From 192.168.1.2 icmp_seq=77 Destination Host Unreachable

...

From 192.168.1.2 icmp_seq=195 Destination Host Unreachable
From 192.168.1.2 icmp_seq=196 Destination Host Unreachable
From 192.168.1.2 icmp_seq=197 Destination Host Unreachable
64 bytes from 192.168.2.2: icmp_seq=201 ttl=61 time=5.26 ms
64 bytes from 192.168.2.2: icmp_seq=202 ttl=61 time=4.04 ms
64 bytes from 192.168.2.2: icmp_seq=203 ttl=61 time=5.06 ms
```

Jak widać stracone były pakiety pomiędzy 36 a 200 (164 sekundy).

Następnie weryfikujemy drogę:

```
traceroute to 192.168.2.2 (192.168.2.2), 30 hops max, 60 byte packets
 1  192.168.1.2 (192.168.1.2)  0.876 ms  0.880 ms  0.939 ms
 2  192.168.23.3 (192.168.23.3)  3.286 ms  3.168 ms  3.040 ms
 3  192.168.13.1 (192.168.13.1)  3.716 ms  3.519 ms  3.472 ms
 4  192.168.2.2 (192.168.2.2)  6.903 ms  7.032 ms  6.748 ms
```

3.1.2 OSPF

Logicznie postępujemy tak samo z OSPF, *traceroute* przed:

```
traceroute to 192.168.2.2 (192.168.2.2), 30 hops max, 60 byte packets
 1  192.168.1.2 (192.168.1.2)  1.214 ms  1.232 ms  1.209 ms
 2  192.168.12.1 (192.168.12.1)  3.737 ms  3.061 ms  2.734 ms
 3  192.168.2.2 (192.168.2.2)  5.272 ms  5.149 ms  5.175 ms
```

Ustawiamy *ping* i odłączamy sieć:

```
64 bytes from 192.168.2.2: icmp_seq=32 ttl=62 time=3.56 ms
64 bytes from 192.168.2.2: icmp_seq=33 ttl=62 time=4.11 ms
64 bytes from 192.168.2.2: icmp_seq=34 ttl=62 time=2.67 ms
From 192.168.1.2 icmp_seq=40 Destination Host Unreachable
From 192.168.1.2 icmp_seq=41 Destination Host Unreachable
From 192.168.1.2 icmp_seq=42 Destination Host Unreachable
```

…

```
From 192.168.1.2 icmp_seq=66 Destination Host Unreachable
From 192.168.1.2 icmp_seq=67 Destination Host Unreachable
From 192.168.1.2 icmp_seq=68 Destination Host Unreachable
64 bytes from 192.168.2.2: icmp_seq=70 ttl=61 time=5.26 ms
64 bytes from 192.168.2.2: icmp_seq=71 ttl=61 time=5.31 ms
64 bytes from 192.168.2.2: icmp_seq=72 ttl=61 time=3.25 ms
```

W tym przypadku straciliśmy tylko pakiety pomiędzy 34 a 70 (46 sekund). Oczywiście ciężko porównać te wyniki gdyż nie wiemy kiedy odłączyliśmy sieć czy to się stało 1 sekundę po wysłaniu aktualizacji czy 39 sekund (40s dead dla OSPF, 120s timeout RIP).

Traceroute po:

```
traceroute to 192.168.2.2 (192.168.2.2), 30 hops max, 60 byte packets
 1  192.168.1.2 (192.168.1.2)  0.811 ms  0.802 ms  0.696 ms
 2  192.168.23.3 (192.168.23.3)  2.840 ms  6.961 ms  6.680 ms
 3  192.168.13.1 (192.168.13.1)  6.427 ms  6.275 ms  6.143 ms
 4  192.168.2.2 (192.168.2.2)  6.540 ms  10.063 ms  10.070 ms
```

W logach komunikacji pomiędzy r2 i r3 możemy zobaczyć pakiet LS-Update:

```
10:55:12.730015 IP (tos 0xc0, ttl 1, id 60242, offset 0, flags [none], proto
OSPF (89), length 140)
    192.168.23.2 > 224.0.0.5: OSPFv2, LS-Update, length 120
        Router-ID 192.168.23.2, Backbone Area, Authentication Type: none (0), 2
LSAs
          LSA #1
          Advertising Router 192.168.23.2, seq 0x8000000b, age 1s, length 40
            Router LSA (1), LSA-ID: 192.168.23.2
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 192.168.12.0, Mask: 255.255.255.248
                topology default (0), metric 10
              Neighbor Network-ID: 192.168.23.3, Interface Address: 192.168.23.2
                topology default (0), metric 10
              Stub Network: 192.168.1.0, Mask: 255.255.255.0
                topology default (0), metric 10
          LSA #2
          Advertising Router 192.168.23.2, seq 0x80000001, age 3600s, length 12
            Network LSA (2), LSA-ID: 192.168.12.2
            Options: [External]
            Mask 255.255.255.248
            Connected Routers:
              192.168.13.1
              192.168.23.2
```

Widząc że R2 wysyła jako Network LSA że ma dostęp do R1 poprzez siec 192.168.13.x (siec R1, R3).

Tą aktualizacje można zobaczyć w tablica routingu R2:

```
default via 192.168.122.1 dev ens3
192.168.1.0/24 dev ens9 proto kernel scope link src 192.168.1.2
192.168.2.0/24 via 192.168.23.3 dev ens8 proto zebra metric 20
192.168.12.0/29 dev ens4 proto kernel scope link src 192.168.12.2
192.168.13.0/29 via 192.168.23.3 dev ens8 proto zebra metric 20
192.168.23.0/29 dev ens8 proto kernel scope link src 192.168.23.2
```

3.2 Przeciążenie na sieci (OSPF)

Przeciążenie sieci (lub wolne połączenie) zasymulujemy dodając interfejsowi wysoki koszt

Ćwiczenie do OSPF:

1. Weryfikujemy że pakiety pomiędzy S1 i S2 są wysyłane przez R1 i R2 (traceroute)
2. Nadajemy wysoki koszt interfejsowi R1→R2
3. Logujemy komunikacje pomiędzy routerami
4. Sprawdzamy czy komunikacje w dwie strony przechodzi tą samą drogą (*traceroute)*
5. Jeżeli nie, to nadajemy wysoki koszt R2→R1
6. Weryfikujemy że obie drogi przechodzą przez R3 (*traceroute)*

3.2.1 Wyniki

Weryfikujemy połączenie:

```
traceroute to 192.168.2.2 (192.168.2.2), 30 hops max, 60 byte packets
 1  192.168.1.2 (192.168.1.2)  1.214 ms  1.232 ms  1.209 ms
 2  192.168.12.1 (192.168.12.1)  3.737 ms  3.061 ms  2.734 ms
 3  192.168.2.2 (192.168.2.2)  5.272 ms  5.149 ms  5.175 ms
```

następnie wchodzimy w konfiguracje R2 (*vtysh 2604*)

```
r2# configure terminal
r2(config)# interface ens4
r2(config-if)# ip ospf cost 100

r2# write
```

po pewnym czasie możemy zaobserwować zmianę w połączeniu S1 → S2:

```
traceroute to 192.168.2.2 (192.168.2.2), 30 hops max, 60 byte packets
 1  192.168.1.2 (192.168.1.2)  0.942 ms  0.737 ms  0.746 ms
 2  192.168.23.3 (192.168.23.3)  5.265 ms  5.011 ms  4.783 ms
 3  192.168.12.1 (192.168.12.1)  4.652 ms  4.519 ms  4.385 ms
 4  192.168.2.2 (192.168.2.2)  4.793 ms  4.660 ms *
```

natomiast droga S2 → S1 nadal pozostaje S2 → R1→ R2→ S1:

```
traceroute to 192.168.1.1 (192.168.1.1), 30 hops max, 60 byte packets
 1  192.168.2.1 (192.168.2.1)  1.543 ms  1.518 ms  3.285 ms
 2  192.168.23.2 (192.168.23.2)  22.530 ms  22.335 ms  21.911 ms
 3  192.168.1.1 (192.168.1.1)  21.428 ms  21.186 ms  19.434 ms
```

Zmieniamy też koszty na R1:

```
r1# configure terminal
r1(config)# interface ens8
r1(config-if)# ip ospf cost 100
```

```
r1# write
```

i znowu po pewnym czasie

```
traceroute to 192.168.1.1 (192.168.1.1), 30 hops max, 60 byte packets
 1  192.168.2.1 (192.168.2.1)  0.611 ms  0.654 ms  0.643 ms
 2  192.168.13.3 (192.168.13.3)  3.370 ms  2.847 ms  2.755 ms
 3  192.168.23.2 (192.168.23.2)  2.993 ms  2.946 ms  3.603 ms
 4  192.168.1.1 (192.168.1.1)  4.248 ms * *
```

3.3 Sprawdzanie Overhead-u

W tym ćwiczeniu sprawdzimy wpływ ilości pakietów kontrolnych na działanie sieci porównując je z domyślnymi ustawieniami. Ćwiczenie te może być niereprezentowane ze względu na duża ilość zmiennych ciężkich do kontroli bez użycia fizycznego sprzętu (CPU Boost, hypervisor scheduler itp.).

Maszyna fizyczna to i7-4600U @2.1 GHz (2C/4T) z Ubuntu 20.04.1 LTS

Maszyny wirtualne to Debian 9 z jednym logicznym rdzeniem i 1GB RAM

Skrypty:

https://github.com/p7tryk/administracja/blob/master/routing/netcatclient.sh

https://github.com/p7tryk/administracja/blob/master/routing/netcatserver.sh

Testem szybkości będzie wysłanie wygenerowanego pliku z losowymi danymi 1GB (*dd if=/dev/urandom of=pliktestowy.bin bs=4M count=256*) za pomocą *netcat*
Poniższe kroki wykonać dla RIP i OSPF:

1. Zweryfikować połączenie pomiędzy S1 i S2 przez R2 i R1.
2. Otworzyć serwer *netcat* na S2.
3. Zarejestrować czas i rozpocząć rejestrowanie zużycia procesora.
4. Wysłanie pliku.
5. Zarejestrować czas i zatrzymać rejestrowanie zużycia procesora.

3.3.1 Kontrola

RIP domyślne ustawienia:

```
pwsz@s2:~$ ./netcatclient.sh 192.168.1.1
0 min 44s
```

```
pwsz@s1:~$ ./netcatclient.sh 192.168.2.2
0 min 46s
```

OSPF domyślne ustawienia:

```
pwsz@s2:~$ ./netcatclient.sh 192.168.1.1
0 min 46s

pwsz@s1:~$ ./netcatclient.sh 192.168.2.2
0 min 46s
```

3.3.2 RIP

Zmienimy licznik *$update* serwera RIP na 1 (*domyślnie 30*) a następnie wykonamy kroki z 3.3.1.

```
pwsz@s2:~$ ./netcatclient.sh 192.168.1.1
0 min 47s

pwsz@s1:~$ ./netcatclient.sh 192.168.2.2
0 min 47s
```

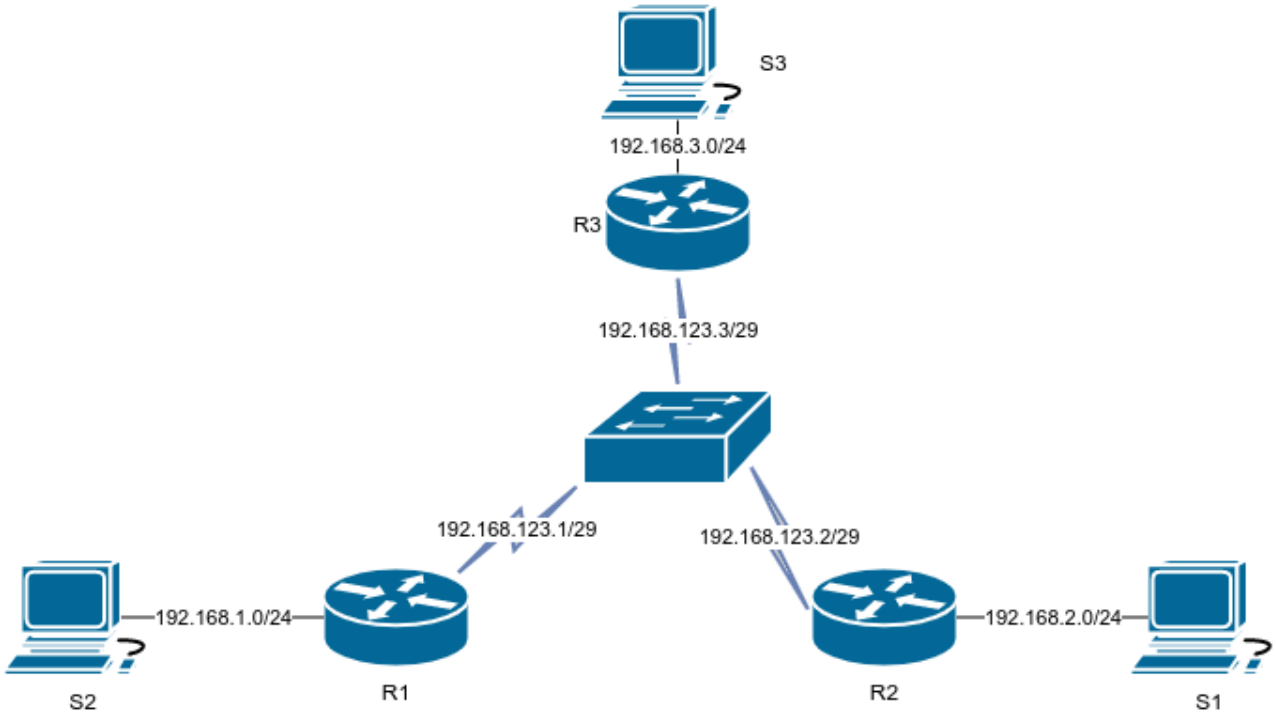
3.3.3 OSPF

Zmienimy licznik *$hello* serwera OSPF na 1 *(domyślnie 10*) a następnie wykonamy kroki z 3.3.1.

```
r1# configure terminal
r1(config)# interface ens4
r1(config-if)# ip ospf hello-interval 1

pwsz@s1:~$ ./netcatclient.sh 192.168.2.2

0 min 49s

pwsz@s2:~$ ./netcatclient.sh 192.168.1.1

0 min 46s
```

3.4 Designated router (OSPF)

Żeby zademonstrować mechanizm DR wymagana jest zmiana topologi sieci (routery połączone nie peer-to-peer).



Rys 3.1

schemat sieci do ćwiczenia 4

Ćwiczenie do OSPF:

1. Sprawdzamy który router jest DR (*show ip ospf neighbour)*
2. Weryfikujemy że routery komunikują się tylko z DR (*tcpdump*)
3. Podwyższamy priorytety routerów które **nie są DR** do różnych wartości.
4. Sprawdzamy czy nowy DR będzie wynegocjowany.
5. Jeżeli nie, restartujemy usługi (lub serwery)
6. Sprawdzamy czy DR i backup DR zostały wynegocjowane wg naszych priorytetów.

3.4.1 Wyniki

Po zmianie topologi można zauważyć że R3 stał się DR a R1 stał się Backup DR

```
r1# show ip ospf neighbor

Neighbor ID     Pri State           Dead Time Address         Interface
192.168.123.2     1 Full/DROther      33.352s 192.168.123.2   ens4:192.168.123.1
192.168.123.3     1 Full/DR           39.073s 192.168.123.3   ens4:192.168.123.1
```

zatem nadajemy R1 priorytet 100 a R3 priorytet 50:

```
r1# configure terminal
r1(config)# interface ens4
r1(config-if)# ip ospf priority 100
```

```
r3# configure terminal
r3(config)# interface ens4
r3(config-if)# ip ospf priority 50
```

oczekiwany wynik jest taki że R1 stanie się DR a R3 Backup DR.

```
r2# show ip ospf neighbor

Neighbor ID     Pri State           Dead Time Address         Interface
192.168.123.1   100 Full/DR           38.287s 192.168.123.1   ens4:192.168.123.2
192.168.123.3    50 Full/Backup       35.846s 192.168.123.3   ens4:192.168.123.2
```

komenda *show ip ospf neighbor* pokazuje nam również priorytety wszystkich routerów sąsiednich.

4. Wnioski

W powyższym ćwiczeniu zademonstrowane zostały zastosowania, możliwości routingu docelowego w sieciach komputerowych zarówno rozwiązań statycznych jak i dwóch popularnych rozwiązań dynamicznych (OSPFv2 i RIPv2)

4.1 Routing statyczny vs dynamiczny

Jak można było zauważyć, na niskim poziome routing statyczny i dynamiczny używa dokładnie tego samego mechanizmu a zmienia się w nich jedynie sposób wpisywania danych do tablicy routingu. Zauważyć również można było zalety takiego systemu dynamicznego który potrafił ominąć awarie sieci a nawet dostosować się do przepustowości sieci. Systemy takie po wstępnej konfiguracji są praktycznie nie widoczne dla użytkowników (w zależności od ustawień: od minut do kilku sekund przestoju). Z tych względów zastosowania routingu statycznego są bardzo ograniczone do prostych i szybkich rozwiązań jak np. droga domyślna (default route) a nawet to rozwiązanie jest zastępowane przez serwer DHCP którego jedną z możliwości jest rozprowadzanie drogi domyślnej.

4.2. RIP vs OSPF

W przypadku porównania RIP (w wersji 2) i OSPF (w wersji 2) widać dużą rozbieżność w podejściu do zagadnienia routingu dynamicznego. Bardzo proste szybkie rozwiązanie takie jak RIP jest w stanie pokazać moc propagowania dynamicznych zmian w sieciach które pozwalają na podwyższenie niezawodności tych sieci. OSPF używa natomiast innego podejścia by osiągnąć ten cel. Jest to bardzo rozbudowane rozwiązanie z możliwościami podziału topologi na strefy (area) by ograniczyć ilości informacji routingowych propagowanych w sieciach i wielokrotnością ustawień związanych z zarządzeniem nimi.

RIP i OSPF różnią się nie tylko na podstawie dodatkowych narzędzi do tworzenia bardziej skomplikowanych struktur ale i również samej podstawie znajdowania tej drogi. RIP używa metody distance-vector która liczy tylko ilość skoków(urządzeń) pomiedzy dwoma punktami. OSPF używa

bardziej złożonej metody link-state która nie tylko bierze pod uwagę odległość ale również koszt (zdefiniowany przez administratora np. szybkość interfejsu) i tworzy skomplikowane tablice by wyliczyć nowa drogę.

4.3 Ustawienia

W sprawach bezpieczeństwa oba rozwiązania domyślnie oba rozwiązania nie oferują uwierzytelniania. Zarówno OSPFv2 jak i RIPv2 oferują możliwość haszowania przez MD5 lecz niestety jest to funkcja która została dawno pokonana i z tego powodu nie oferuje żadnego wzrostu w bezpieczeństwie.

W ćwiczeniu 1 zostały zbadane domyślne ustawienia w zakresie zapominania połączeń w przypadku braku odpowiedzi przez część sieci. Oba rozwiązania w przeciągu kilku minut przywróciły działanie sieci. W tej sytuacji rozważyć trzeba dwa główne przypadki: tymczasowe wyłączenie sieci np. restart routera (czy promieniowanie kosmiczne) oraz po prostu awaria sieci (która musi zostać rozwiązana przez administratora). W pierwszym przypadku raczej nie potrzebna jest zmiana dróg w całej sieci a w przypadku drugim jak najbardziej. Balans pomiędzy tymi kwestiami a ograniczeniami przestoju lub balansowania ruchu jest dość skomplikowanym tematem który każda instytucja musi ustali sama. Domyślne ustawienia (kilkadziesiąt sekund do kilku minut) mieszczą się w takiej typowej sytuacji dla użytkownika np. nie działa sieć; wstanę rozprostować nogi/pójdę po kawę/pójdę do toalety a po powrocie cała sieć działa poprawnie.

W ćwiczeniu 3 zostały zbadane ustawienia w zakresie znajdowania nowych urządzeń (*hello*). Wszystkie wyniki były w granicy błędu i nie było rozstrzygnąć tej kwestii w nie perfekcyjnie przygotowanym laboratorium. Logicznie zmiana z jednego pakietu *hello* na 10sekund do jednego na sekundę nie powinna wpłynąć na pracę komputera w obecnych czasach. Nawet jeżeli wzięta jest pod uwagę wielkość tablic w dużych nie podzielonych sieciach, to prędkości procesorów i sieci w obecnych czasach są na tyle duże że, decyzja ustawienia *hello* na 10s (30s w RIP) jest sprawą przeszłości i jest bardziej kwestią stosunku ile pakietów hello może być zgubionych zanim przełączmy drogę

4.4 Dopasowanie do wymagań i sytuacji

Routing statyczny – praktycznie ograniczony do połączeni peer-to-peer i całkowitych podstaw sieciowych (a tutaj drogi domyślne rozprowadzone przez DHCP załatwiają większość problemów)

RIP i OSPF – w praktyce w dzisiejszych czasach w zastosowaniach mniejszych topologi porównywalne, OSPF ze względu na ilość opcji pozwala na większe rozbudowanie takiej sieci poprzez rozdzielenie stref. Mimo ogromu opcji oferowanych przez oba rozwiązania, podstawowe konfiguracje (takie jak zostały zaprezentowane w ćwiczeniach) są bardzo podobne merytorycznie i różnią się tylko składnią oraz zajmują podobny czas do implementacji.

5. Bibliografia

(1) RFC 2453 RIP Version 2 listopad 1998

(2) RFC 2328 OSPF Version 2 kwiecień 1998

(3) https://www.nongnu.org/quagga/docs dostęp 2020-04-09

(4) https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_ospf/configuration/xe-16/iro-xe-16-book/iro-cfg.html#GUID-4AABEB56-2125-488B-B5A4-A5650F3159BB dostęp 2020-04-09

(5) https://www.cisco.com/c/en/us/support/docs/ip/open-shortest-path-first-ospf/9237-9.html#q3a dostęp 2020-04-09

(6) https://www.ciscopress.com/articles/article.asp?p=26919 dostęp 2020-04-09

(7) https://www.freeccnaworkbook.com/workbooks/ccna/configuring-basic-ospf dostęp 2020-04-09