0. Konfiguracja routingu statycznego i dynamicznego

Wykonał: Patryk Kaniewski, 2 rok Informatyka ST PUSB Skierniewice.

1. Wprowadzenie

Routing jest to proces wybierania ścieżki w sieci lub pomiedzy 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 rozwiazane za pomoca routerów. Router posiada w sobie informacje o sąsiednich sieciach i na podstawie tych informacji i innych zasad (np. filtrowania, źrodła) i 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 wstepne

- Podstawy CLI (Linux, ~~Cisco IOS~~)
- Konfiguracja interfejsów sieciowych
- Konfiguracja usługi swiadczącej routing dynamiczny (Linux: *quagga*)
- routing na podstawie stanu łącza - OSPFv2 (RFC 2328)
- routing na podstawie wektora odległości - RIPv2 (RFC 2458)
- Narzedzia diagnostyczne sieci (*tcpdump*,*traceroute*,*ping*)

1.3 Zakres ćwiczenia

W tym ćwiczeniu zajmiemy się docelowym routingiem w sieciach komputerowych używających modelu TCP/IP na podstawie MAC (Ethernet, WiFi itp.) Pod uwagę weźmiemy tylko protokoły bram wewnętrznych (ang. *Interior gateway protocol*) ze względu na zakres materiału w protokołach bram zewnetrznych (protokoły tę obejmują zagadnienia routingu pomiedzy 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

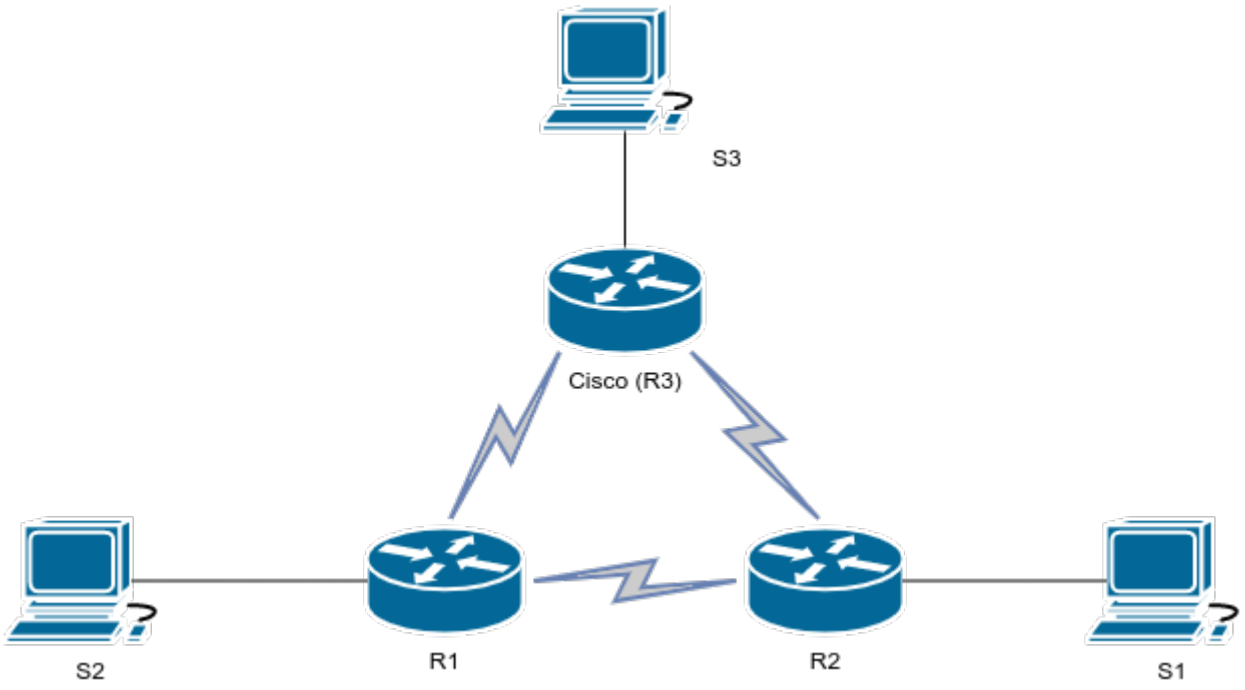
1.5 Dodatkowe definicje

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

VLSM (ang. Variable Length Subnet Mask) – jest to odejscie 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)

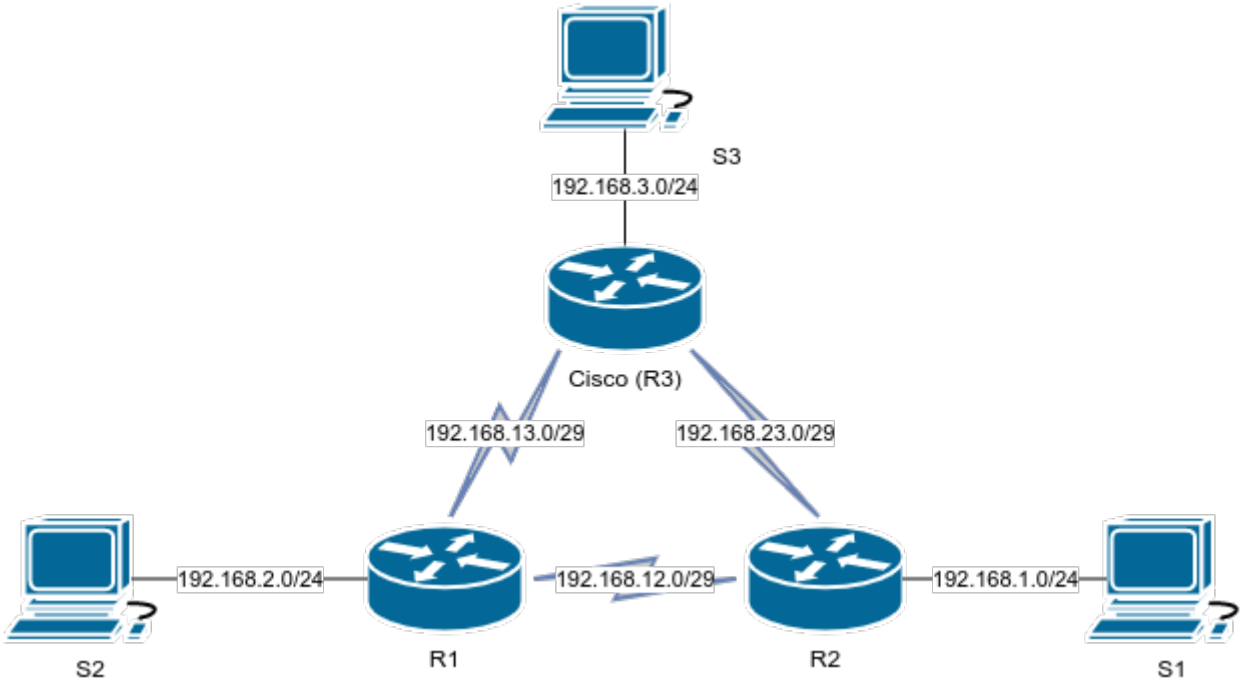
2. Przebieg ćwiczenia

Fizyczna konfiguracja sieci:



Rys 2.1

Logiczna konfiguracja sieci:



Rys 2.2

Wyjaśnienie:

Routery łączymy w sieciach z długą maska sieciową ponieważ nie potrzebujemy wiele adresów do sieci point-to-point. Maszyny robocze zostawiamy w domyślnych podsieciach klasy C. Za pomocą tego możemy zweryfikować jak protokoły radzą sobie z maska sieci różnej długości (VLSM).

Konfiguracja interfejsów:

recznie

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

Instalacja programów diagnostycznych (jeśli potrzeba):

 # apt install tcpdump traceroute ping

Właczenie routingu na R1,R2:

Edycja sysctl.conf

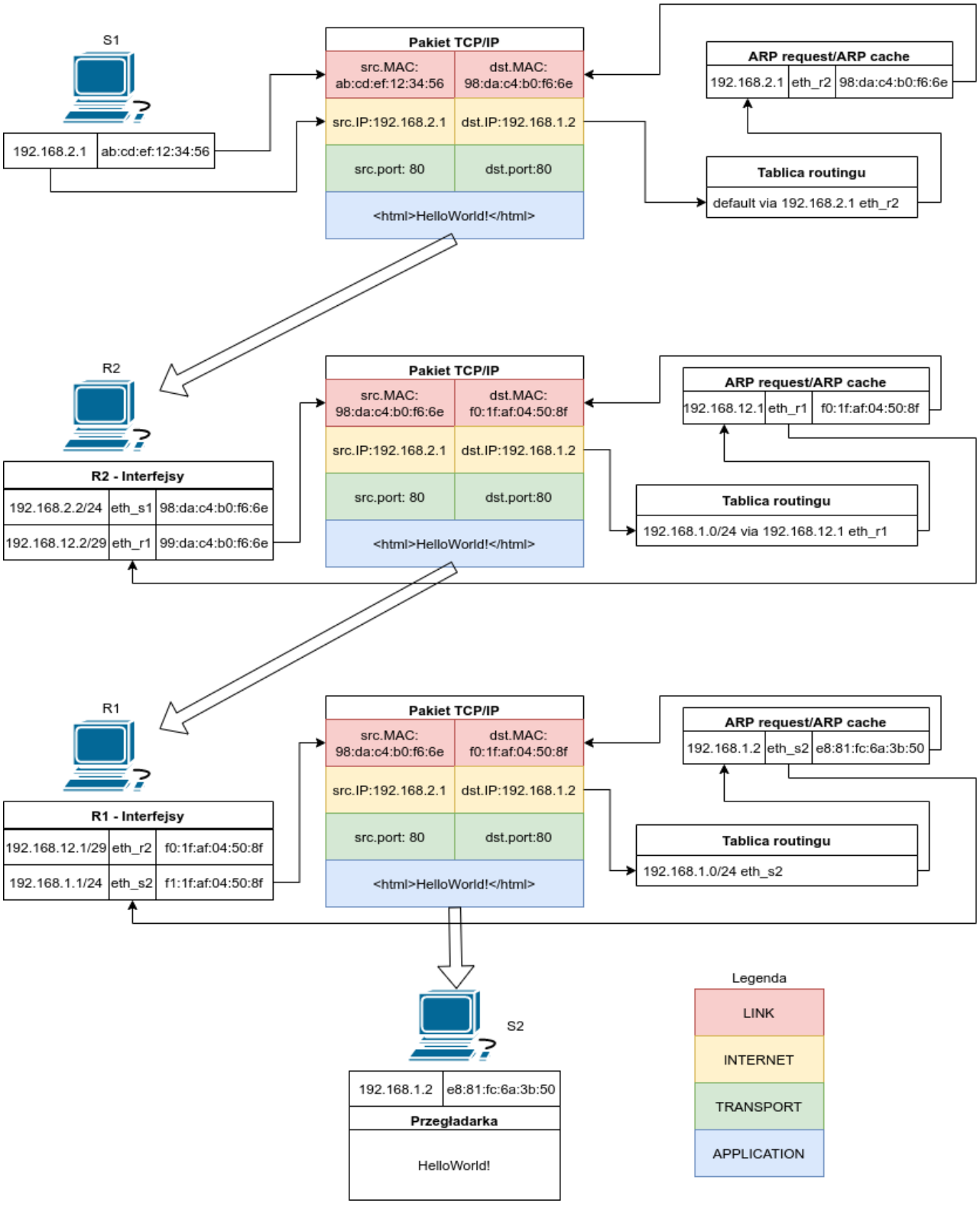
+ net.ipv4.ip_forward = 1

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

Link,Internet,Transport,Application

Patryk Kaniewski GPLv3
https://github.com/p7tryk

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 Linux

Konfigurujemy routing statyczny ze stacji roboczej (S1,S2,S3) do przyłaczonego do niej routera.

2.2.1.1 ip route add

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

2.2.1.2 Default route

ip route add default via $adress dev $interfejs

<zdjecie>

edycja /etc/network/interfaces

<zdjecia>

2.2.2 Weryfikacja

Działanie połaczenia:

<zdjecie ping>

Sprawdzanie tablic routingu:

<zdjecie ip r>

2.3 Routing dynamiczny

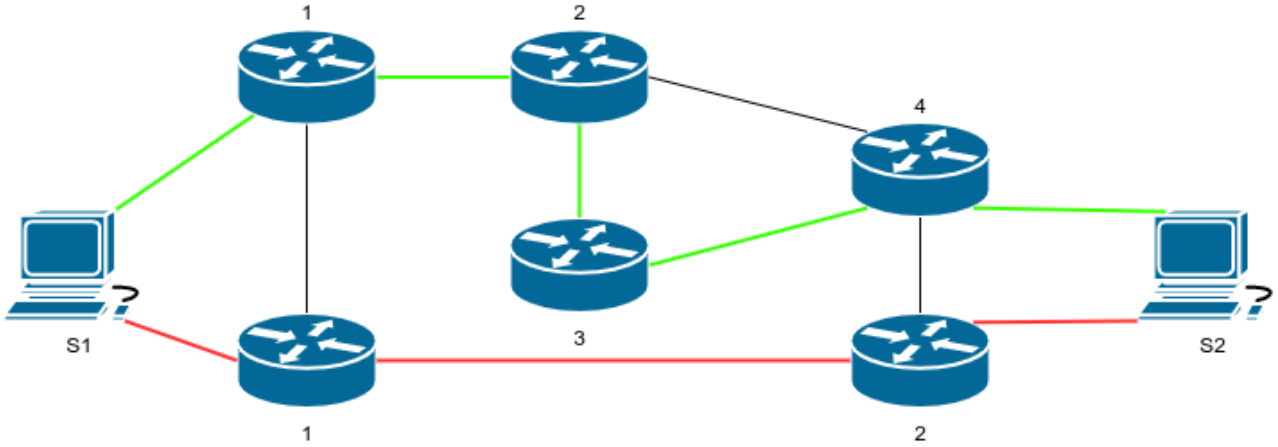
W miare wzrostu złożoności sieci komputerowych narosła potrzeba bardziej inteligentnego i *dynamicznego* systemu który mogłby reagować na zmiany w ruchu sieciowym czy awarie na połączeniach.

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

Dwoma głownymi metodami routingu dynamicznego w sieciach wewnetrznych 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 przepustowościa 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 (szczegolnie VLSM) ale zachowując prostote protokołu.

- Protokoł 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 pomiedzy routerami oraz nie pozwala na balansowanie ruchu sieciowego zdala od słabych połączeń
- Protokół nie rozróżnia pomiedzy 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 (awaria). RIP rozwiązuje to za pomocą regularnych aktualizacji domyślnie co 30 sekund (*$update*). Jeżeli router nie odpowiada po

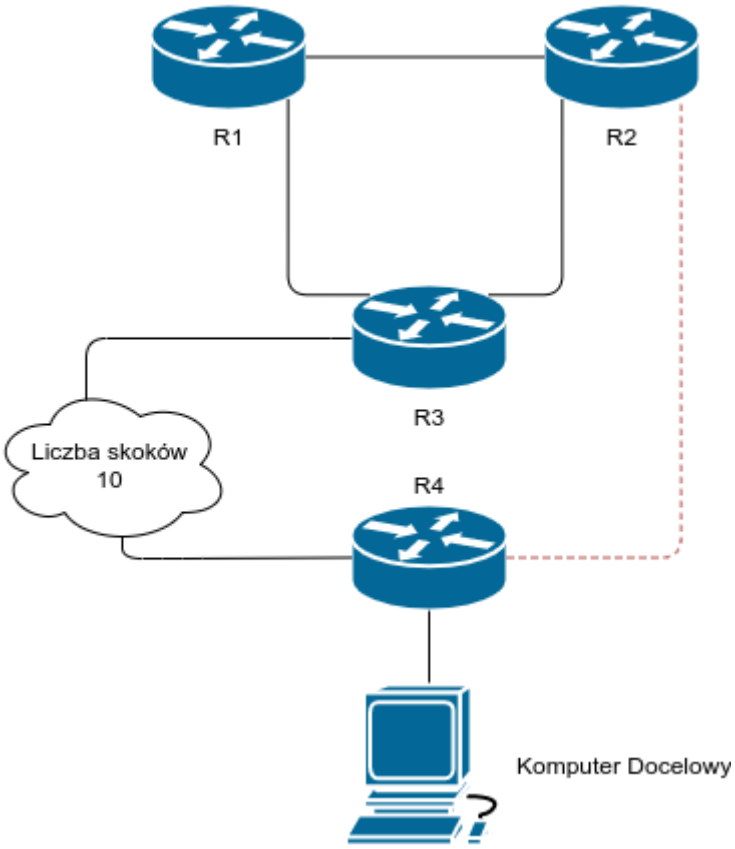
domyślnie 180 sekundach (*$dead*) to ta droga jest zaznaczana jako nieważną a po następnych 180s ($garbagecollect) usuwana z tabeli.

RIP używa pakietów UDP i zawiera od 1 do 25 rekordów RIP

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  command (1)  |  version (1)  |       must be zero (2)        |
+---------------+---------------+-------------------------------+
|                                                               |
~                        RIP Entry (20)                         ~
|                                                               |
+---------------+---------------+---------------+---------------+
```

RIP packet RFC2453

2.3.1.2 Problemy z distance-vector routing



rys 2.5

Weźmy 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 krótrsza. Stanie się to zawsze (chyba że nowa

droga nie będzie możliwa – skoki >15) ale może zająć czas kiedy sieć będzie niedostępna. Problem ten nazywa się 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 droge dostępną przez dany router do *tego routera* powoduje to że routery sobie ufaja i może zająć dużo czasu zanim fałszywa droga (pętla) bedzię gorsza niż droga alternatywna.

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

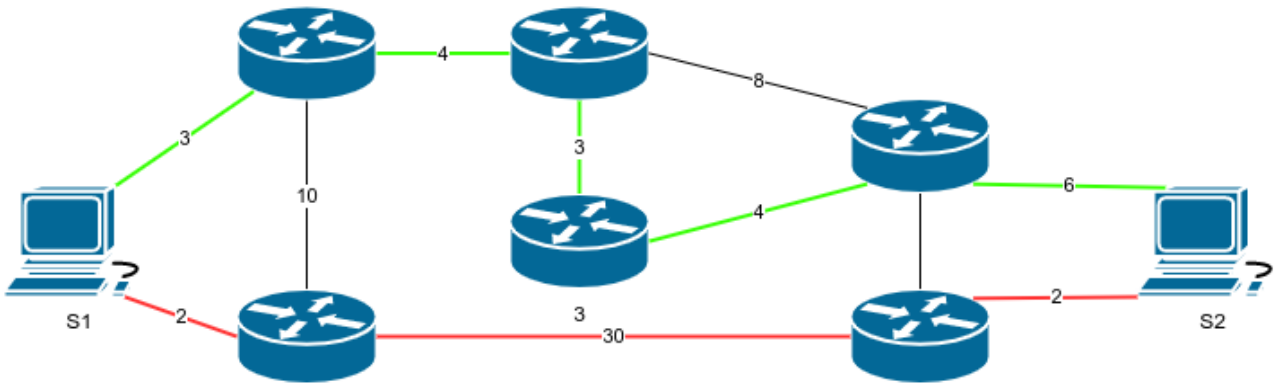
Lepszym rozwiązaniem jest zatruwanie (ang. poisoning) takich dróg. Ustawiamy takie drogi na maksymalną ilość skoków (15). Powoduje to na natychmiastowe policzenie do *nieskończoności* i przerwanie pętli. Rozwiązuje to wszystkie pętle 2 routerów ale w systemach z większą liczbą routerów które mogą się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

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 oraz obciążenia. Rozwiązuje to problem obwodnicy ale zdecydowanie komplikuje działanie takiego systemu.



Rys 2.4

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

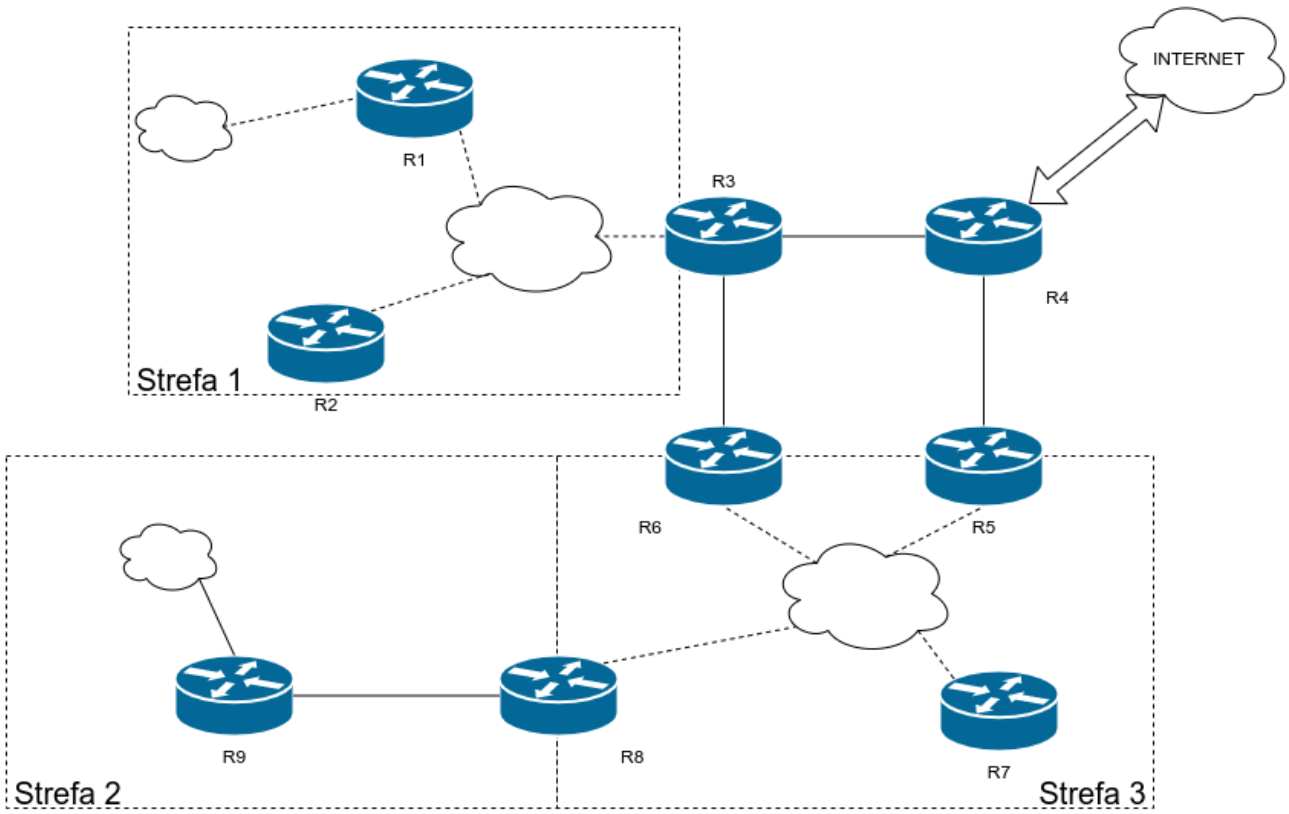
OSPFv2 jest implementacją link state routing dla bram wewnętrznych. Router OSPF ma swój unikalny numer 32bitowy (w systemie 1 bramy wewnętrznej) i przechowuje w sobie mapę topologii sieci wraz z wagami przypisanymi do każdego połączenia. Routery wymieniają pomiedzy sobie informacje o swoich bazach stanu. Za pomocą tej bazy danych (ang. link-state database) każdy router buduje drzewo nakrótszych dróg zaczynając od samego siebie (jest korzeniem tego drzewa). W tym drzewie lisciami są informacje zewnetrzne 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 którą można dostosować do wymagań systemu. 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 – ma intefejsy podłaczone do strefy 0 i co najmniej jednej innej strefy.
granicy systemu autonomicznego – posiadu drogi zewnętrzne do systemu autonomicznego
backbone – połączone z strefą 0



rys.2.5
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

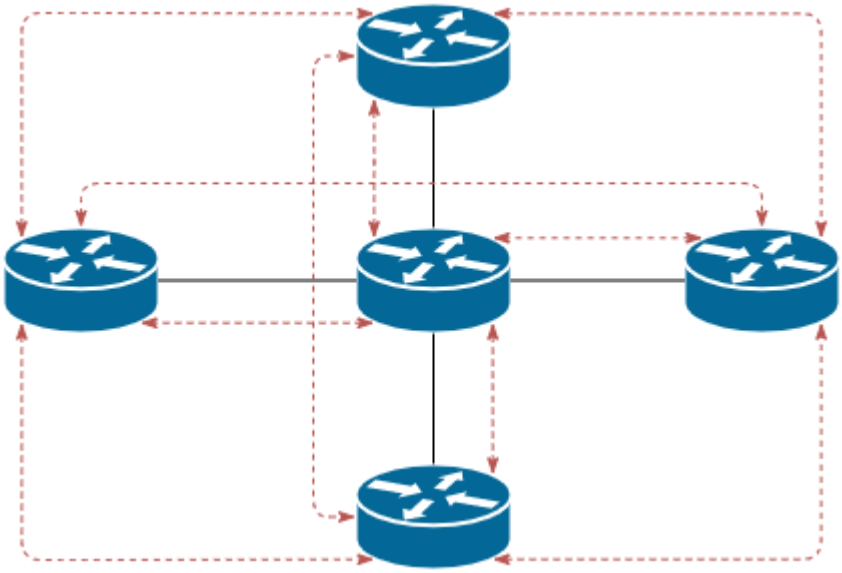
Jest mechanizmem za pomocą którego router ogłasza sieci do niego podłączone.

1. Router LSA – router ogłasza sieć podłączoną bezpośrednio niego
2. Network LSA - Desginated 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 przez granice systemu autonomicznego

opisywać więcej? stubby/nonstubby area

2.3.2.3 Desginated router

Desginated router i backup desginatated router – mechanizm stworzony by ograniczniu wysyłaniu LSA pomiedzy wszystkimi routerami

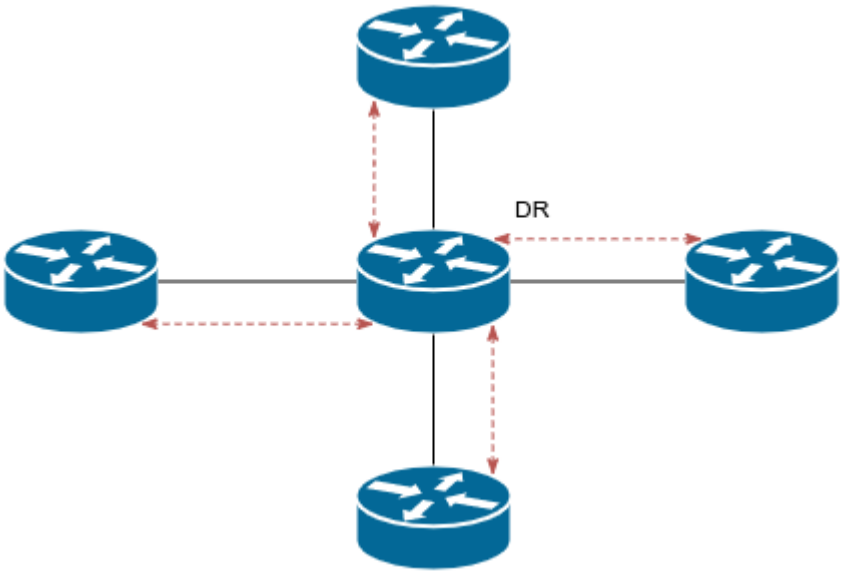


rys. 2.6

routery wymieniające informacje pomiędzy sobą (czerwone linie)

Jeden router jest wyznaczany jako desginated router i LSA są wysyłane tylko pomiędzy routerem normalnym a DR.

Jeżeli DR przestanie odpowiadać to zapasowy DR zostanie DR.

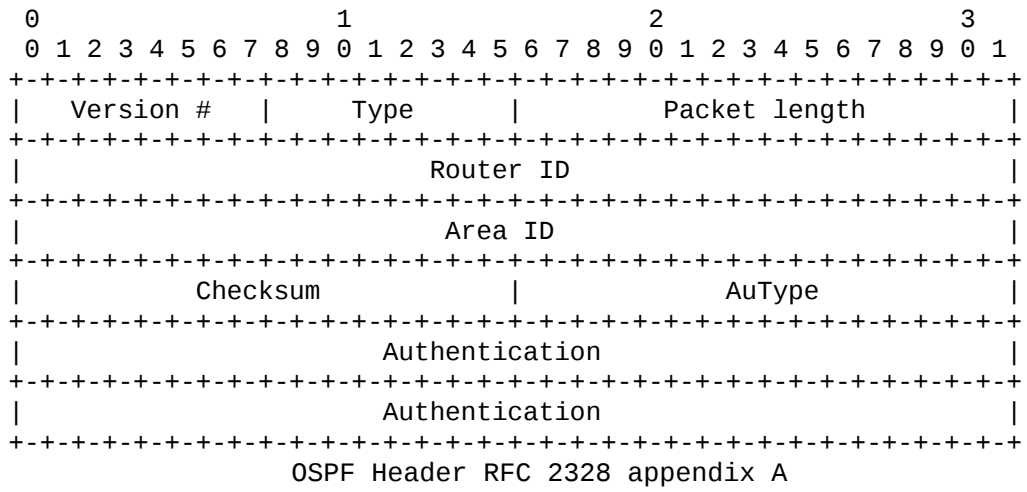


Rys 2.7

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

2.3.2.2 Komunikacja pomiędzy routerami OSPF

OSPF używa jednego 24 bajtowego nagłowka:



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

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.

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 start ospfd.service //dla systemów z systemd jeżeli twój system używa innego systemu init skonsultuj się z jego konfiguracją

Łaczymy się do servera ospfd za pomoca *telnet* lub *vtysh:*

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

$ *enable* //zeby przejsc do trybu uprawnionego

configure terminal //żeby przejsc do trybu konfiguracji

router ospf //żeby przejsc do konfiguracji OSPF

network $adres_sieci/$subnet *area* $area //podajemy sieć którą będziemy ogłaszać

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

2.3.3.3 Konfiguracja zaawansowana (OSPF)

do punktu 3.

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

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

 # hello-interval $sekundy //specyfikuje co jak czas pakiet *hello* będzie wysyłany

 # priority $numer //priorytet routera

 # transmit-delay $sekundy //dodaje czas do pola czasu w pakiecie LSA

retransmit-interval $sekundy //ustawienie częstotliwości retransmisji dopóki pakiet LSA nie zostanie potwierdzony

2.3.3.4 Konfiguracja podstawowa (RIP)

uruchamiamy usługę:

 # systemctl start ripd.service

Łaczymy się do servera ripd za pomoca *telnet* lub *vtysh:*

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

$ *enable* //zeby przejsc do trybu uprawnionego

configure terminal //żeby przejsc do trybu konfiguracji

router rip //żeby przejsc do konfiguracji OSPF

rip ip send version [1,2,1 2] //żeby wybrać wersje RIP której uzywamy

[no] *network* $adres_sieci/$subnet //podajemy sieć którą będziemy ogłaszać

[no] *network* $nazwainterfejsu //podajemy sieć którą będziemy ogłaszać (przez interfejs)

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

2.3.3.4 Konfiguracja zaawansowana (RIP)

do punktu 3.

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

 # timers basic [$update $timeout $garbagecollect] //pokazuje/zmienia czasy

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

//$timeout – czas do unieważnienia drogi

//$garbagecollect – czas do usunięcią nieważnej drogi

2.3.4 Weryfikacja dzialania protokolu

<Ping, traceroute, ip route show, tcpdump (pakietow kontrolnych)? pomiedzy stacjami roboczymi, routing table routerów>

<zdjecia>

3. Testowanie „dynamiczności” routingu

3.1 Metodyka testowania

Używamy tylko jednego daemona (OSPF lub RIP)

1. Weryfikujemy że pakiety pomiedzy S1 i S2 są wysyłane przez R1 i R2 (traceroute)
2. Wywołujemy ping pomiedzy S1 i S2 i odwrotnie
3. Odłączamy interfejs pomiedzy R1 i R2 (*set link down* na jednym z routerów lub fizycznie odłączamy sieć)
4. Logujemy komunikacje pomiedzy 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)

TBD przykład rys2.5

3.2 Wyniki

TBD

4. Wnioski

TBD

5. Bibliografia

TBD TBD

RFC 2453 RIP Version 2

RFC 2328 OSPF Version 2

https://www.nongnu.org/quagga/docs

cisco docs