Sieci – sprawozdanie

6 czerwca 2017

Spis treści

1 Szybki problem – szybka odpowiedź

Innymi słowy, gdzie co szybko znaleźć!

Podstawy konfigurowania, Vlan, interfejs Vlan, Trunk, Port-bassed, Tagged, VTP, DTP – Lab 1

VTY, konfigurowanie przez SSH, STP, wartość cost, propagacja MAC, EtherChannel, ARP, inne konfiguracyjne switch'a – Lab 2

Konfigurowanie Router'a, DHCP, mostki w Router'ach, interfejsy serial, PPP, PAP, CHAP, Multilink PPP, HDLC, ACL – Lab 3

Redystrybucja, RIP, OSPF, EIGRP, ogólnie IGP, Tunele GRE, redystrybucja pomiędzy Vlan'ami – Lab 4

NAT overloading, transmisje statyczne DMZ, oraz z przekierowaniem portów TCP NAPT, konfigurowanie interfejsów ipv6, RIPng, OSPFv3, EIGRP w wersji ipv6, tunelowanie ipv6 przez ipv4, – Lab 5

IGMP, Multicast dense mode i sparse mode, Policy Based Routing, Cisco HSRP, VRRP, IP SLA – Lab 6

Internal i External BGP, Route Reflector, Konfederacja systemów autonomicznych, prefix, AS_PATH, multihoming – Lab 7

Budowa datagramu IP, podsłuchiwanie protokołów – Lab 8

WAN, ATM PVC, ATM T3/E3, – Lab 9

PPP nad łączem ATM (PPPoA), PPP na Ethernetem (PPPoE), instalacja SHDSL CO-CPE, – Lab 10

HP ProCurve, konfigurowanie przez EtherSwitch (ESW), – Lab 11

Cisco Catalyst 5500, CatOS, RSM, – Lab 12

2 Lab 1 Switch I

Zajmujemy się tutaj podstawami konfigurowania urządzeń Cisco (póki co Switch), również poprzez telnet lub SSH. Obsługujemy parametry odpowiednich portów, spróbujemy utworzyć kilka Vlan'ów, przypiszemy interfejs Vlan (to on odpowiada za telnet komunikację z nierutowalnym Switch'em), następnie zajmiemy się utworzeniem Tagged Vlan's czyli konfigurowanie portów Trunk (z użyciem IEEE 802.1Q).Jest jeszcze sprawa Native Trunk'ów, Na koniec zajmiemy się jeszcze kwestią protokołu VTP, przesyłającego informacje o sieciach Vlan z switch'a – servera to switch'y – client'ów.

2.1 Teoria

Konfigurujemy podpinając się do konsolki przez TP lub z PC podłączonego do tej samej sieci co interfejs Vlan Switch'a poprzez telnet lub SSH. Zwykle poleceniem „enable" przechodzimy do trybu administratora i dalej

wpisując „configuration terminal” zmieniamy odpowiednie aspekty urządzenia (więcej w części praktycznej).

Vlan tworzymy gdy w urządzeniu chcemy utworzyć kilka odseparowanych od siebie logicznych sieci. Vlan to nie interfejs Vlan, ten drugi istnieje tylko po to, aby z nierutowalnym Switch'em można było połączyć się zdalnie. Domyślnie switch posiada jeden Vlan (o numerze 1), apropos i zakres to 1-4095, dodatkowo ciekawosta (numer 0 oznacza brak Vlan). Odpowiednie porty grupujemy do odpowiednich Vlan i korzystamy z wybranej metody ich komunikacji.

Komunikacja port-bassed zakłada łączenie każdego Vlan'a osobnym kabelkiem poprzez jeden port (tracimy port i duży koszt). Lepszą metodą są tagged Vlan's, wykorzystujące trunk. Trunk to port służący do komunikacji z innymi urządzeniami DTE (switch). Niektóre przełączniki wymagają jawnego podania sposobu enkapsulacji.
Native Trunk wykorzystują zwykłe ramki, zamiast Q-ramek, ale może być tylko jeden taki, aby nie nastąpiły nam problemy (wtedy np. mielibyśmy problemy z STP). Oba przełączniki muszą wiedzieć który port to native trunk dla którego vlan'a, aby nie nastąpił przeciek pomiędzy Vlan'ami.

Dzięki VTP, możemy przesyłać informacje o Vlan'ach pomiędzy switch'ami przez domeny (informacje przesyłane w ramach jednej domeny), każdy przełącznik może wystąpić w trybie server(jego informacje o Vlan są przekazywane), transparent(sam nie pobiera informacji, ale podaje je dalej), client(konfiguruje swoje Vlan na podstawie otrzymanych informacji). Pomiędzy przełącznikami nie może być urządzeń innych firm lub w innej domenie VTP. Przełącznik dopinany(nawet w trybie server) ZAWSZE traci zastępuje poprzednią wiedzę o Vlan tą z przypiętej domeny. Łącza do komunikacji muszą być w trybie Trunk. Nazwy domen muszą być unikatowe. Można ustalić hasła, ale jeżeli będą niezbieżne, to trzeba clienta wstawić w tryb transparent, ustalić wszystkim te same hasła i będzie śmigać. Przy błędzie „revision number” zmieniamy nazwę domeny i wracamy do poprzedniej.

Powiedzmy jeszcze pare słów o protokole DTP. Gdy tryb portu jest ustawiony na dynamic, automatycznie kojarzony jest rodzaj portu po drugiej stronie. Czyni to także gdy nie jest dynamic, ale pod warunkiem, że kabel nie prowadzi do kilku innych switch'y jednocześnie poprzez przełącznik innego producenta nie wspierający DTP. Najlepiej wyłączyć wtedy DTP dla tego portu (wpisać mu nonegotiate), bo przy korzystaniu jeszcze z VTP, sprzeczne komunikaty mogą spowodować błąd i wyłączenie portu.

2.2 Praktyka

Najpierw zajmiemy się podstawami operowania ze switch'ami Cisco, następnie wykonamy Vlan'y najpierw w jednym Switch'u, a następnie przy pomocy metody tagged Vlan oraz protokołu VTP stworzymy całą architekturę Vlan.

2.2.1 Podstawowa konfiguracja

Łączymy się z konsolą Switch'a i wchodzimy do trybu uprzywilejowanego (exec) oraz od razu do trybu konfiguracji.

```
Switch>enable
Switch#configuration terminal
```

Switch warstwy drugiej może posiadać interfejs (tożsamość) Vlan, która służy tylko i wyłącznie do konfigurowania zdalnego, nie jest związana z samym dzieleniem urządzenia na poszczególne Vlan'y. Switch'owi możemy przypisać ip dla interfejsu Vlan i łączyć się z nim przez telnet lub SSH, przy pomocy komend:

```
Switch(config)#interface vlan1
Switch(config-if)#ip address 200.200.200.1 255.255.255.0
Switch(config-if)#no shutdown
```

Karetka „Switch(config)#” informuje nas o tym, że znajdujemy się w trybie konfiguracyjnym terminala. Komenda „interface vlan1” pozwala wejść do trybu konfiguracji wybranego interfejsu. Polecenie „no

shutdown" służy do uruchomienia administracyjnego, w tym przypadku danego interfejsu. Oczywiście adres przypisanego Switch'a musi pasować do sieci, w której funkcjonuje. Co więcej, aby Vlan został uruchomiony, przynajmniej jeden interfejs fizyczny, musi znajdować się w trybie Forwarding, więc musimy np. przypisać tam PC.

Porty mogą pracować w trzech możliwych trybach (access, trunk, dynamic) i tryb ten ustala rodzaj ramki.

```
Switch(config)#interface fa 0/5
Switch(config-if)#switchport mode access
```

„fa 0/5" oznacza skróconą nazwę rzeczywistego identyfikatora portu Fast Ethernet o numerze 5. 0 oznacza, że port ten znajduje się bezpośrednio w obudowie urządzenia. Domyślnie ustalany jest tryb „dynamic", może to powodować problemy opisane w teorii (port wyłączony i komunikat „%DTP-5-DOMAINMISMATCH: Unable to perform trunk negotiation on port Fa0/5 because of VTP domain mismatch."

DTP wyłączamy DTP przy pomocy:

```
Switch(config)#interface fa 0/5
Switch(config-if)#switchport nonegotiate
```

Do przeglądania ustawień interfejsów IP switch'a służą polecenia:

```
Switch#show ip interface brief
Switch#show ip interface vlan 1
Switch(config)#do show ip interface brief
Switch(config)#do show ip interface vlan 1
```

Zwracamy uwagę na aktywność interfejsów (up/down/administratively down). W trybie konfiguracyjnym możemy użyć poleceń z trybu uprzywilejowanego dodając przed poleceniem „do".

2.2.2 Konfigurowanie Vlan

Proste polecenie do przeglądania aktualnego stanu bazy Vlan.

```
Switch#show vlan
```

Tworzenie Vlan jest dosyć proste, musimy tylko wybrać liczbę z zakresu $< 2, 4095 >$, Vlan 1 jest domyślny i do niego trafiają wszystkie nieprzypisane przez nas porty.

```
Switch(config)#vlan 11
Switch(config-vlan)#exit
```

Jeśli używamy VTP i switch jest w trybie Client VTP musimy zmienić tryb np. na transparent, aby coś zmieniać.

```
Switch(config)#vtp mode transparent
```

Możemy przypisywać porty pojedynczo lub przez zakres portów.

```
Switch(config)#interface fa0/2
Switch(config-if)#no shutdown
Switch(config-if)#switchport mode access
Switch(config-if)#switchport access vlan 11
Switch(config)#interface range fa0/15 - 17
Switch(config-if-range)#switchport mode access
Switch(config-if-range)#switchport access vlan 20
```

Przy czym po range spacje przez i po znaku „-" są obowiązkowe. W starszym sprzęcie konfigurowanie odbywało się poprzez użycie komendy „vlan database" z trybu uprzywilejowanego.

2.2.3 Trunks

Poprzednio ustalaliśmy Vlan dla pojedynczego switch'a, teraz skonfigurujemy trunk pomiędzy np. dwoma switch'ami do których możemy podpiąć PC do różnych Vlan, a komunikacja będzie mogła przebiegać tylko w tych samych Vlan. Wchodzimy do interfejsów portów (w obu switch'ach znaczy) poprzez które złączyliśmy switch'e, ustalamy tryb portu, jeżeli trzeba enkapsulację oraz zakres vlan'ów, które ma obsługiwać:

```
Switch(config)#interface fa 0/1
Switch(config-if)#switchport mode trunk
Switch(config-if)#no shutdown
Switch(config-if)#switchport trunk encapsulation dot1q
Switch(config-if)#switchport trunk allowed vlan 1-100
```

Możemy też usunąć zezwolenie na obsługiwanie Vlan'a:

```
Switch(config-if)#switchport trunk allowed vlan remove 10
```

Jeden wyróżniony trunk (gdy mamy ich więcej), przez który przechodzi najwięcej ramek, możemy skonfigurować jako „native".

```
Switch(config)#int fa 0/1
Switch(config-if)#switchport trunk native vlan 10
```

Ważne, aby w obu router'ach jako native został wybrany ten sam vlan!

2.2.4 Vlan i VTP

Zanim cokolwiek zaczniemy, możemy włączać i wyłączać diagnostykę:

```
Switch#debug sw-vlan vtp events
Switch#no debug sw-vlan vtp events
```

Swoją drogą dodanie „no" przed komendą zazwyczaj oznacza wyłączenie jej działania lub jego odwrócenie.
W switch'u serverze konfigurujemy:

```
Switch(config)#vtp domain domena
Switch(config)#vtp mode server
```

Następnie ustalamy klienta:

```
Switch(config)#vtp domain domena
Switch(config)#vtp mode client
```

Ustalmy kilka vlan w serverze, a następnie po odczekaniu czasu migracji sprawdźmy stan vlan'ów. Switch można przestawić również w tryb „transparent", aby podawał informacje o vlan'ach dalej, ale sam swoich nie aktualizował.

```
Switch(config)#vtp mode transparent
```

Metodą wymuszenia aktualizacji VTP jest usunięcie i dodanie Vlan w serverze. Możemy wyświetlić stan VTP przy pomocy:

```
Switch#show vtp status
```

Liczba Revision number informuje nas o kolejnym numerze aktualizacji VTP.

3 Lab 2 Switch II

Rozszerzamy zagadnienia konfigurowania Switch'a Cisco, przyjrzymy się dokładniej jak przygotować urządzenie pod pracę z SSH, rozpoznamy i spróbuje sprostać problemowi pętli oraz wykonamy agregację łącz fizycznych w łącze logiczne – EtherChannel.

3.1 Teoria

Gdy mamy już interfejs Vlan możemy skonfigurować linie terminali wirtualnych VTY. Jest to logiczna linia do komunikacji telnet lub SSH, którą switch po prostu udziela. Zwykle jest ich 16, musimy sami ustalić hasło oraz możliwy port dla danych linii. Dodatkowo należy zwracać uwagę na tryb autoryzacji new-model - powinien być wyłączony aby wprowadzić komendę „login". Możemy ustalić nazwę domeny dla switch'a oraz jego hostname (a nie jest to wyłącznie karetka, bo korzystają z niego np. protokoły CDP). W przypadku SSH generujemy również klucz RSA, uruchamiamy new-model (teraz nie można użyć polecenia login, problem z telnetem wcześniej, a to dlatego że hasło z linii VTY nie jest już obsługiwane). Musimy zdefiniować w switch'u użytkownika z loginem, hasłem (wraz z oznaczeniem 7 - szyfrowane, 0 - plain text) i priorytetem. Poprzez ACL możemy ograniczać, ale to będzie dogłębniej opisane w lab 3.

W sieci nie może występować pętla, ponieważ ramki broadcast zalewałyby nam sieć uniemożliwiając jakąkolwiek komunikację. Do walki z tym problemem wystawiamy protokół STP oraz jego pochodne RSTP (od razu tworzy alternatywy w przypadku padnięcia czegoś) i MSTP (wersja oferująca zbieżną konfigurację dla zakresu VLAN). Definiowany jest Root Bridge, który staje się korzeniem drzewa STP i przez niego przechodzi komunikacja pomiędzy jego poddrzewami, natomiast w poddrzewach nie występują cykle. Wyłączne są połączenia, które powodują pętle i są nieoptymalne.

Root switch jest wybierany gdy ma najniższy BID (Bridge-ID), składający się z: dwa bajty priorytetu i osiem bajtów MAC (mniejszy MAC, no to ROOT, MAC'i są unikatowe). W wersji obsługującej Vlan (PVST – osobne drzewo dla każdego Vlan w sieci), w części priorytetu jest 4 bity priorytetu i 12 bitów numeru sieci Vlan, dlatego priorytety to wielokrotności 4096. Root ma najmnieszy BID – zasada(to on wysyła Superior BPDU, jak sam dosatnie superior to oddaje root'a). Jedyną możliwością nie oddania root'a, jest zmniejszenie swojego priorytetu na wypadek przyjścia informacji o istnieniu switch'a o mniejszym BID (Guard root). Root port – do root bridge, designated port – od root. (może być siele desg, jeden root). Można wykluczyć port z STP (Portfast), wyłączenie portów uni-directional (Guard loop). Jeżeli podłączymy wadliwy kabel, jedynym ratunkiem jest UDLD (komunikaty echo wysyłane), STP poradzi sobie jednak, jeżeli kabel uszkodzi się po połączeniu już.

Cost jest metryką odległości od Root'a. Ustalana w kierunku do Root'a w każdym switch'u. Jest wersja 32 bitowa i 16 bitowa (ta pierwsza ma domyślne wartości nieco bardziej rozbieżne, daje większą elastyczność). (domyślne np. 16-bit: 10Mbps = 100, 100Mbps = 19, 1Gbps = 4, 10Gbps = 2, 32-bit: 10Mbps = 2 000 000, 100Mbps = 200 000, 1Gbps = 20 000, 10Gbps = 2 000, 100Gbps = 200). Można je ręcznie zmieniać.

Istnieje również metryka priorytetu portu – Port priority. Składa się z części konfigurowalnej – 4 przesunięte bity (stąd wielokrotności 16) oraz numeru portu. Ustalana jest w kierunku od root'a. Służy do wytypowania innym switch'om, najlepszego portu od root'a.

Kolejność przyznania root portu: Obliczony koszt łącze (niższe pierwszeństwo), gdy równe to priorytet portu (niższe pierwszeństwo), gdy równe to numer portu. Koszt łącza jest obliczany jako suma kosztu BPDU i ostatniego odcinka ścieżki.

Stany portów w kolejności: Listening – odbiera BPDU nie wysyła ramek, Learning – tylko wymiana MAC, Forwarding – normalna praca, przekazuje ramki, Blocking – powoduje pętle, Discarding – wstrzymanie pracy w rapid, Broken – błędy STP. Dodatkowo w PVST i RSTP ALternate Port – wie że do root'a jest lepsza droga przez inny, Backup Port – inny port w tej samej sieci ma lepszą drogę.

Zwielokrotnienie linii – EtherChannel. W Cisco łączymy kilkoma kabelkami, najlepiej dajemy że są trunk'ami i stają się jednym kabelkiem logicznym z mniejszym kosztem, bo rozdziela przesył na wszystkie fizyczne. Może być też nazywany Port Trunk.

ARP jest protokołem, który pozwala zebrać do jakich MAC wysyłać ramki, gdy przyszły pod adres IP, dzieje się to poprzez zapytania ARP i tworzenie tablicy ARP w switch'u, na podstawie adresów przychodzących na port od urządzeń w jego sieci, stąd wie gdzie wysyłać gdy przyjdzie zapytanie o jakieś IP.

3.2 Praktyka

3.2.1 Konfigurowanie przez SSH i inne konfiguracyjne

Przed połączeniem się przez SSH, musimy w switch'u określić nazwę domeny, wygenerować klucz RSA

```
Switch(config)#ip domain-name nazwaDomeny
Switch(config)#crypto key generate rsa // następnie podajemy wielkość klucza (360-2048 bitów)
```

Przełączamy tryb autoryzacji użytkowników z domyślnego servera „AAA radius" (dzięki temu w telnecie używaliśmy komendy „login") , na wykorzystanie lokalne systemu kont.

```
Switch(config)#aaa new-model
```

Teraz musimy stworzyć użytkownika (gdzie 15 to wartość priorytetu użykownika, 0 oznacza jawny tekst hasła, 7 szyfrowałoby hasło).

```
Switch(config)#username sieci priv 15 password 0 sieci
```

Na koniec podobnie jak w telnecie, przypisujemy wybrane linie vty na korzystanie z SSH.

```
Switch(config)#line vty 0 15
Switch(config-line)#transport input ssh
Switch(config-line)#no transport input telnet //jednocześnie blokujemy dostep przez telnet
```

Do linii można dopisywać ograniczenia do poszczególnych IP wykorzystując reguły ACL, omówienie teoretyczne znajduje się w pierwszej laboratorce z routingu, tutaj tylko przykład:

```
Switch(config)#access-list 55 permit 192.168.1.0 0.0.0.255
Switch(config)#line vty 0 15
Switch(config-line)#access-class 55 in
Switch(config-line)#exit
```

Jeszcze kilka chwytliwych rzeczy i komend:

- kiedy zrobimy literówkę switch domyślnie będzie poszukiwał adresu ip błędnego symbolu, no bo nie wpisaliśmy komendy poprawnie, ta komenda ochłodzi jego zapędy,

  ```
  Switch(config)#no ip domain-lookup
  ```

- `Switch(config)#hostname nowyHost // zmienia nazwę hosta, wraz z karetką`
- `Switch#show running-config // sprawdza aktualną konfigurację`
- ```
  Switch#write memory // utrwala obecne ustawienia w pamięci Flash
  Switch#write erase // kasowanie ustawień z Flash i przejście do fabrycznych
  ```
- ```
  Switch(config-line)#exit // przejście w ścieżce o jeden w górę
  Switch(config-line)#end //przejście w ścieżce na samą górę
  ```

I pamiętajmy o złotej zasadzie, aby wyłączyć lub usunąć efekt jakiejś komendy, dopisujemy „no" przed tą komendą.

3.2.2 STP

Najpierw proste komendy do sprawdzania STP w router'ach Cisco, dowiadujemy się z nich dane Root'a, dane bieżącego Root port i designed ports:

```
Switch#show spanning-tree
Switch#show spanning-tree detail
Switch#show spanning-tree vlan 1-10
Switch#debug spanning-tree events // gdy chcemy być na bieżacą z przebudową STP
```

Na porty nie podłączone do naszych Switch'y możemy zastosować poniższą komende, aby nie zepsuli nam STP poprzez spreparowane ramki BPDU:

```
Switch(config-if)#spanning-tree bpduguard disable //oczywiście będąc w odpowiednim interfejsie
```

Możemy zmienić root'a na dwa sposoby:

- ustalenie, że ten switch ma być primary lub secondary przy użyciu komendy i wtedy dzieje się automatycznie:

  ```
  Switch(config)#spanning-tree vlan 1 root primary // secondary
  ```

- bezpośrednie manipulowanie wartością priorytetem switch'a, poprzez określenie czterech nastarszych bitów BID, a więc możemy przypisywać liczby (32768-4096), ale z krokiem 4096:

  ```
  Switch(config)#spanning-tree vlan 1 priority 16384 //im mnniejsza wartość tym wyższy priorytet
  ```

Można blokować możliwość utraty statusu Root'a ustalając guard na porcie (tutaj nie chodzi o spreparowane BPDU, ale np. o BPDU switchy, które ktoś nam nieopatrznie podłączył do naszej sieci czy coś):

```
Switch(config-if)#spanning-tree guard root
```

Generalnie koszty dla połączeń powinny odpowiadać zasadzie (10Mbps = 100, 100Mbps = 19, 1Gbps = 4, 10Gbps = 2).Przypisywanie swojego kosztu do interfejsu:

```
Switch(config)#int fa 0/1
Switch(config-if)#spanning-tree cost 13
Switch(config-if)#spanning-tree vlan 1 cost 13 //wersja z określeniem Vlan
```

Gdy koszty równe o tym które połączenie odciąć decyduje priorytet portu, a gdy priorytety równe, to numer portu w switch'u. Zmiana priorytetu portu:

```
Switch(config)#int fa 0/1
Switch(config-if)#spanning-tree port-priority 64
Switch(config-if)#spanning-tree vlan 1port-priority 64  //wersja z określeniem Vlan
```

Przełączanie wersji STP i jego wyłączenie:

```
Switch(config)# spanning-tree mode pvst
Switch(config)# spanning-tree mode rapid-pvst
Switch(config)#spanning-tree portfast default // wyłączenie w całym switch'u
Switch(config-if)#spanning-tree portfast // wyłącznie dla portu
```

3.2.3 Propagacja MAC

W switch'u nierutowalnym nie można używać list ACL, ale możemy mieć coś analogicznego z adresami MAC, i możemy poniższą komendą dodawać takie różne pomysły np. zablokować stację PC o określonym MAC:

```
Switch(config)#mac-address-table static 0013.72b9.89fe vlan 1 drop
```

Sprawdzenie zawartości tablicy MAC switch'a:

```
Switch1#show mac-address-table
Switch1#show mac address-table // Cisco 2960
Switch1#show mac-address-table interface fa 0/1 //lista MAC danego interfejsu
```

Możemy ręczne własnych MAC dla interfejsów biorąc pod uwagę Vlan:

```
Switch(config)#mac-address-table static 1111.2222.3333 vlan 1 int fa0/5
```

3.2.4 EtherChannel

Ważna uwaga, że zanim zaczniemy spinać, należy odpiąć po jednej ze stron (lub powyłączać zaangażowane porty), aby nie zaczęło nam STP odcinać portów w trakcie konfiguracji.

Tworzenie interfejsu Port-Channel, port-mode we wszystkich zaangażowanych musi być ustawione na access:

```
Switch(config)#interface Port-channel 1
```

Przypisujemy wybrane porty do channel-group, number po channel-group musi być ten sam co stworzonego wcześniej portu Port-channel:

```
Switch(config)#interface range FastEthernet 0/1 - 2
Switch(config-if)#switchport mode access
Switch(config-if)#channel-group 1 mode on
```

Diagnostyka:

```
Switch#debug etherchannel
```

W przypadku błędów misconfig:

```
Switch(config)#int Po 1
Switch(config)#shut
Switch(config)#no shut
```

4 Lab 3 Route I

Zaczynamy sprawy z konfigurowaniem Router'ów Cisco, razem z tworzeniem w nich mostków i opowiemy trochę o listach kontrolnych (Aceelkach), będą również protokoły łącz szeregowych (SLIP, PPP, HDLC), będzie też konfigurowanie DHCP, i dwa słowa o ICMP i SNMP.

4.1 Teoria

Zaczynamy zabawy z Router'ami. Na początek kilka różnic w stosunku do switch'y, numerowanie interfejsów zaczyna się tutaj od 0, a nie od 1, interfejsy w chassis (obudowie router'a), znajdują się w module o numerze 0, dobudowywane moduły mogą mieć pod moduły, stąd możliwy jest np. interfejs fe 1/2/1. Numeracje dla różnych typów interfejsów nie nachodzą na siebie (wszystkie fe i wszystkie seriale zawsze od zera). Można tworzyć loopback'i, czyli interfejsy symulujące jakąś tam sieć. Router musi znajdować się na przecięciu różnych sieci, więc każdy interfejs musi być w innej sieci IP.

Możliwe jest utworzenie mostka w routerze, wtedy staje się on lub jego część jakby zwykłym switch'em, są trzy metody:

- legacy bridging - router staje się switch'em w całości,
- CRB (Concurrent Routing and Bridging) - cześć interfejsów staje się interfejsami switch'a, a część router'a, natomiast nie ma między nimi połączenia,

- IRB (Integrated Routing and Bridging) - CRB tylko z dodatkowym specjalnym interfejsem wirtualnym BVI (Bridge Virtual Interface) dzięki któremu może odbywać się rutowanie pomiędzy częścią router'a rutującą i switch'em, adres musi być zgodny z siecią, w której znajduje się switch.

Generalnie listy ACL służą do kontroli i blokowania ruchu lub czynności. Mogą być standardowe (1-99) lub rozszerzone(100-199), są też nazwane, ale te i tak określamy później do jednej z poprzednich, po prostu zamiast numerka mają nazwę. Grupy list przypisujemy do interfejsu. W standardowych listach możemy np. zezwolić na ruch tylko z określonych sieci ip, poprzez permit, lub zablokować kilka poprzez deny i na końcu zezwalając permit na wszystkie pozostałe. Możemy też przypisać ACL konfigurując interfejs, wtedy możemy dopowiedzieć czy rodzi o ruch in lub out. Rozszerzone pozwalają na wyszczególnienie cech nadawcy i odbiorcy, pozwalają korzystać ze szczegółowych kryteriów filtrowania wykorzystując np. protokół tcp (np. zablokować jakiś tam numer portu TCP). Maski sieci zawsze zapisujemy w inwersji bitowej! W grupie ACL jest też zabawa z ustawianiem poszczególnych ACL w środku...

Łączom serial możemy przypisać rodzaj protokołu łącza szeregowego wraz z ewentualnym uwierzytelnianiem. Serial może łączyć asynchronicznie lub synchronicznie. Chodzi o to, aby nie korzystać z Ethernet'u lub Token Ring w połączeniach Point to Point. Transport pomiędzy dwoma stacjami roboczymi może być zapewniony przez prosty Slip zastąpiony przez PPP, które jest wyposażone w LCP, które obsługuje wszystkie szczegóły połączenia, np. rozłączenie, stabilizację czy utrzymanie. PPP może być uwierzytelniane przez PAP, wtedy klient wysyła hasło i nazwę i czeka na odpowiedz lub CHAP wtedy server wysyła challenge i klient sprawdza czy pasuje, jak pasuje to się łączą. Są też warianty obustronnego uwierzytelnienia. Możliwe jest trochę analogiczne do EtherChannel zagregowanie wielu równoległych fizycznych łącz w jedno logiczne poprzez Multilink PPP.

Można tez korzystać z HDLC, w nim wyróżniane są dwa typy urządzeń primary i secondary. W wariancie unbalanced możliwe jest wiele secondary do jednego primary, w balanced podłaczone są dwa równoprawne terminale - peers.

DHCP(Dynamic Host Configuration Protocol) służy do automatycznego przyznawania adresów IP z puli i obsługi tego procesu. Nowy klient jest uzgadniany tak, że klient wysyła DHCP Discover, server odpowiada DHCP Offer, klient daje DHCP Request, i server odpowiada DHCP ACK. Dla klienta nie nowego, ale po restarcie w celu podtrzymania adresu, klient wysyła DHCP Offer i swój poprzedni adres, a server odpowiada DHCP ACK lub DHCP NACK. Klient i server znają szczegóły czasowe dzierżawy. Przy przedłużaniu dzierżawy, klient wysyła DHCP Request, a server odpowiada DHCP ACK lub DHCP NACK. Aby zwolnić adres przed końcem dzierżawy, klient może wysłać DHCP Release.

I dwa słowa o ICMP. Jest to protokół czwartej warstwy OSI, głównie służy do raportowania. Jeśli komputer docelowy nie odpowiada, to system wykrywający problem wysyła Destination Unreachable. Jeśli wysłał to router to znaczy, że adres IP nie istnieje - Host-unreachable, lub gdy nie może dostarczyć do tej sieci Network-unreachable. Jeśli wysyła to komputer, to znaczy że nie obsługuje on protokołu Protocol-unreachable lub nie pozwala działać na tym porcie TCP/UDP Port-unreachable. Obsługuje również sytuacje gdy odbiorca nie nastarcza z obróbką danych, przekierowania na lepszą ścieżkę, sprawdzenie obecności, no i sytuacja gdy zejdzie do zera wartość TTL pakietu, wtedy wysyłany jest do nadawcy Time-exceeded.

SNMP służy do zarządzania siecią. Monitoruje zasoby, wysyła zapytania do urządzeń, niekiedy nazywanych agentami. Agent udostępnia komponenty MIB (Management Information Base). Komunikuje się na porcie 161, pułapki na porcie 162. Pułapki obsługują takie zdarzenia jak np. odłączenie kabla i są wtedy wysyłane z agenta automatycznie. w PDU są takie polecenia jak GetRequest, GetNextRequest, SetRequest, GetResponse, Trap.

4.2 Praktyka

4.2.1 Konfigurowanie Router'a

Jeżeli karetką było „rommon" wpisujemy „reset". „Chodzenie" po poziomach zagnieżdżenia jak przy switch'u. Możemy łączyć się przy użyciu telnetu i SSH bardzo analogicznie, kilka komend:

```
Router (config)#ip http server // uruchomienie usług servera HTTP
Router (config)#username sieci privilege 15 password 0 sieci // nowy użytkownik
Router(config)#no ip domain-lookup // wyłączenie zapytan o DNS
Router(config)#no service config // wyłączenie konfiguracji przez TFTP
Router#terminal length 0 // wyłączenie stronicowania
Router#show running-config // sprawdzenie bieżącej konfiguracji
Router#show arp // no wiadomo
```

Kilka komend protokołu CDP

```
Router#sh cdp // dwie komendy protokołu CDP zbierającego info o sąsiadach
Router#sh cdp neighbors
Router(config)# cdp run // włączenie cdp
Router(config)#interface fa 0/1
Router(config)#no cdp enable // wyłączenie w konkretnym interfejsie
Router#clear cdp table // kasowanie informacji o innych urządzeniach
Router(config)#cdp timer 10 // zmiana czasu nadawania
Router(config)#cdp holdtime 90 // zmiana czasu uaktualnienia
```

Tworzenie loopbacka:

```
Router(config)#interface Loopback 0
Router(config-if)#ip address 10.0.0.1 255.255.255.0
Router(config-if)#no shutdown
```

Wysyłanie długiego pinga (przerwanie przez CTRL SHIFT 6)

```
Router#ping 192.168.1.1 rep 1000
```

4.2.2 DHCP

Włączenie poprzez:

```
R1(config)#service dhcp
```

Konfiguracja parametrów, pula adresów DHCP i adres interfejsu router'a, przez który będzie prowadzone konfigurowanie muszą znajdować się w tej samej sieci w niej będą przydzielane adresy IP. Adres tego interfejsu jest jednocześnie adresem domyślnej bramki z sieci.

```
Router(config)#ip dhcp pool nazwa
Router(dhcp-config)#network 10.10.10.0 255.255.255.0
Router(dhcp-config)#default-router 10.10.10.1
Router(dhcp-config)#dns-server 123.123.123.3
Router(dhcp-config)#domain-name domena.pl
Router(dhcp-config)#exit
Router(config)#ip dhcp excluded-address 10.10.10.10 10.10.10.20
```

Interfejsy klientów konfigurujemy przez:

```
R2(config)#interface FastEthernet 0/0
R2(config-if)#ip address dhcp
```

Diagnostyka:

```
Router#show ip dhcp binding
Router#debug ip dhcp server events
```

4.2.3 Mostki w Routerze

Przy legacy bridging usuwamy adresację IP z interfejsów i wyłączamy rutowanie, interfejsy łączymy w mostek:

```
Router(config)#int fa 0/0
Router(config-if)#brigde-group 1
Router(config-if)#no ip address
Router(config)#exit
Router(config)#int fa 0/1
Router(config-if)#brigde-group 1
Router(config-if)#no ip address
Router(config)#exit
Router(config)#no ip routing
```

Przy CRB rutowanie musi być włączone, uruchamiamy mostkowanie crb (przy poprzednim stanie interfejsów):

```
Router(config)# ip routing
Router(config)# bridge crb
Router(config)# no bridge 1 route ip // bo domyślnie włączony
Router(config)# bridge 1 bridge ip
```

No i najlepsze, IRB:

```
Router(config)# ip routing
Router(config)# bridge irb
Router(config)# bridge 1 bridge ip
Router(config)# bridge 1 route ip
```

I tworzenie BVI (number BVI musi być zgodny z numerem mostka):

```
Router(config)#interface BVI 1
Router(config-if)#ip address 200.200.200.3 255.255.255.0
Router(config-if)#no shutdown
```

4.2.4 Protokoły łącz szeregowych

Najpierw konfigurujemy połączenie serial, wybieramy rodzaj enkapsulacji, adresację IP, a router'ze DCE ustalamy clock rate jeżeli mamy połączenie synchroniczne.

```
Router(config)#interface serial 0/0
Router(config-if)#encapsulation HDLC // np
Router(config-if)#ip address 200.200.200.1 255.255.255.0
Router(config-if)#no shutdown
```

Teraz uwierzytelnianie, najpierw PAP (czyli nie HDLC tylko PPP). Tutaj musimy stworzyć użytkownika z hasłem, w router'ze który będzie akceptował połączenie serial. I od razu ustalamy, uwierzytelnianie PAP. Po obu stronach musi być zgodność.

```
Router1(config)#username uzytkownik password haslo
Router1(config)#int serial 0/0
Router1(config-if)#PPP authentication PAP
```

W drugim router'ze ustalamy, aby pozwalano na połączenie gdy przyjdzie odpowiednie hasło, to z pierwszego:

```
Router2(config)#int serial 0/0
Router2(config-if)#PPP PAP sent-username uzytkownik password haslo
```

No to teraz protokół CHAP. W obu router'ach definiujemy odpowiedni unikatowy hostname i w każdym z nich tworzymy użytkownika z nazwą przeciwległego hostname. Oba hasła użytkowników muszą byc identyczne:

```
Router1(config)#hostname R1
Router2(config)#hostname R2
R1(config)#username R2 password haslo
R2(config)#username R1 password haslo
```

W interfejsach wprowadzamy już tylko:

```
R1(config)#int serial 0
R1(config-if)#PPP authentication CHAP
R2(config)#int serial 0
R2(config-if)#PPP authentication CHAP
```

Można włączyć debugowanie:

```
Router#debug PPP authentication
```

4.2.5 PPP Multilink i łącze asynchroniczne

W obydwu router'ach tworzymy wirtualny interfejs multilink:

```
R1(config)#interface multilink 1
R1(config-if)#ip address 200.200.205.1 255.255.255.0
R2(config)#interface multilink 1
R2(config-if)#ip address 200.200.205.2 255.255.255.0
```

Następnie podpinamy do multilinka fizyczne interfejsy. Fizycznym nie nadajemy już adresacji IP, ta została nadana multilinkowi:

```
Router(config)#int serial 0/3/0
Router(config-if)#encapsulation ppp
Router(config-if)#ppp multilink
Router(config-if)#ppp multilink group 1
Router(config-if) clock rate 1000000
Router(config-if)# no shut
```

Diagnostyka:

```
Router#show ppp multilink
Router#debug ppp multilink events
```

I jeszcze pare komend jeżeli mamy połączyć asynchronicznie np. poprzez RS-232. Konfigurujemy linię:

```
Router(config)#line aux 0
Router(config-line)#modem InOut
Router(config-line)#transport input all
Router(config-line)#flowcontrol hardware
```

I konfigurujemy interfejs Async router'a:

```
Router(config)#interface Async 65
Router(config-line)#ip address 200.200.200.1 255.255.255.0
Router(config-line)#encapsulation ppp
Router(config-line)#async default routing // aktywuje rutowanie IP
Router(config-line)#async mode dedicated // określa przeznaczenie linii RS-232
```

Są też interfejsy hssi, wtedy analogicznie tylko nazwa interfejsu to hssi.

4.2.6 ACL

Najpierw tworzenie standardowych:

```
Router(config)#access-list 90 deny any // 90 id nowej listy,
Router(config)#access-list 70 deny host 10.10.10.1
Router(config)#access-list 70 permit any // czyli 3 komendy z nowymi zakazami i pozoleństwami
Router(config)#access-list 70 remark To jest lista usuwająca 10.10.10.1
```

Przypisanie listy do interfejsu:

```
Router(config)#int fe 0/0
Router(config-if)#ip access-group 90 in // in to przychodzące do interfejsu
```

Usuwanie listy i samego przypisania do interfejsu:

```
Router(config)#no ip access-list standard 90
Router(config-if)#no ip access-group 90 in
```

Przebudowanie numeracji listy (60 to numer listy, 20 to nowy początek numeracji tej listy, a 10 to kolejny przyrost):

```
Router(config)#ip access-list resequence 60 20 10
```

Dodawanie reguły w określone miejsce w liście:

```
Router(config)#ip access-list standard 70
Router(config-std-nacl)#26 deny host 10.10.10.1
Router(config-std-nacl)#no 26 // usuwanie reguły
```

Diagnostyka:

```
Router#show access-list interface fa 0/0 in
Router#show access-list 70
```

Przykład tworzenia listy rozszerzonej(gdzie tcp to identyfikator protokołu nad IP, any oznacza dowolny źródłowy adres IP, 10.10.10.0 0.0.0.255 oznacza dowolnego hosta docelowego w sieci 10.10.10.0/24, natomiast eq 23 oznacza dalsze ograniczenia związane z protokołem TCP (w tym przypadku: port docelowy TCP=23 czyli telnet)):

```
Router(config)#access-list 190 deny tcp any 10.10.10.0 0.0.0.255 eq 23
```

Przypisywanie i diagnostyka analogicznie. No i jeszcze notka z nazwanymi, tutaj zamiast liczby identyfikowanie łańcuchem znaków, przykład utworzenia standardowej nazwanej i rozszerzonej nazwanej:

```
Router(config)#IP access-list standard moja_lista
Router(config-std-nacl)#deny 10.10.10.1 0.0.0.0
Router(config-std-nacl)#permit 10.10.10.2 0.0.0.0
Router(config-std-nacl)#exit
Router(config)#IP access-list extended moja_lista2
Router(config-ext-nacl)#deny TCP 10.10.10.1 0.0.0.0 20.10.10.1 0.0.0.0 eq www
Router(config-ext-nacl)#exit
```

Przypisanie do interfejsu:

```
Router(config-line)#access-group moja_lista2 in
```

5 Lab 4 Route II

Głównie zajmiemy się tutaj technikami rutowania dynamicznego wewnątrz systemu autonomicznego oraz rutowania pomiędzy różnymi protokołami również. Warto przyjrzeć się również takiemu dziwakowi jak tunel GRE oraz kwesti zmieszania wcześniejszych rzeczy, czyli Vlan'ów i kwestii rutowania pomiędzy tymi Vlan'ami.

5.1 Teoria

Routery muszą wiedzieć gdzie posyłać pakiety, zawsze ustala się domyślną bramkę, gdy nie wiadomo co zrobić z danym pakietem, ale generalnie router'y powinny wiedzieć w jaki sposób dosłać pakiet gdy np. docelowy host znajduje się w dwie hopki, wybierając przy tym najdogodniejszy interfejs. Dlatego przeprowadzana jest redystrybucja, a router'y uczą się od siebie nawzajem gdzie posyłać dane pakiety poprzez rutowanie dynamiczne. Jest również rutowanie statyczne („Router(config)#ip route adresstatyczny maskategoadresu adresInterfejsuRuteraOściennegoNaKtóryWysyłamy"), niemniej wiele czasu mu nie poświęcimy.

Router posiada RIB - tablicę rutowania oraz forwarding table. W forwarding table znajdują się wpisy w formacie, literka odpowiadająca za typ, adres docelowy i maskę, domyślną bramkę dla tego typu adresów lub interfejs, jeżeli nie została podana bramka. W tablicy rutowania mogą znajdować się jeszcze metryki i flagi.

Możemy rutować pomiędzy Vlan'ami, wtedy nie ustalamy adresu IP dla interfejsu tylko tworzymy jego podinterfejsy dla każdego Vlan'a i dopiero wtedy przypisujemy adresy. Dużo zabawy jest też z klasowością adresów. Zawsze wybierany będzie ta reguła rutowania, gdzie adres jest bardziej sprecyzowany.

Trasy mogą być obliczane poprzez distance-vector czyli używana jest jakaś metryka do określonego celu np. liczba przeskoków, jest też metoda link-state gdzie ustalana jest informacja kompletnego drzewa ścieżek np. w obszarze OSPF. Pierwsza metoda może prowadzić do zapętleń, ponieważ nie mamy informacji o trasie do celu, a jedynie o kierunku, trasa może później faktycznie rzeczywiście nie istnieć, dodatkowo może również wystąpić problem, gdy gdzieś na trasie zostanie odłączony wcześniej istotny router, bo informacja o nim będzie jeszcze przez długi czas niejako znajdowała się na początku trasy, ratunkiem metoda Split Horizont (brak możliwości wysłania informacji o trasie do router'a, z którego ta informacja wyszła).

W protokole RIP router żąda informacji od innych router'ów z ich tablic routingu i aktualizuje własną. Dodatkowo każdy router informuje o swojej obecności. Protokół UDP i port 520. Bezklasowość dopiero od wersji drugiej. Metryką jest liczba przeskoków, ograniczona do 15. W drugiej wersji wspierany jest też multicast i szyfrowanie przy identyfikacji router'ów. RIPng to wariant dla ipv6.

EIGRP bezklasowy, maksymalna topologia na 50 router'ów. 32 bitowa metryka ze wskazaniem na jakość łącza. Używa algorytmu DUAL do poszukiwania trasy. Określa trasę podstawową i zapasową oraz wewnętrzną(z systemu autonomicznego i z tego protokołu) i zewnętrzną (tutaj te dodane spoza tego protokołu).

OSPF bazuje na algorytmie Djikstry do poszukiwania najlepszej ścieżki, typu link-state. W domenie OSPF maksymalnie 500 węzłów. Wyróżnia się w nim strefy oraz router'y ABR pomiędzy strefami oraz ASBR wyjściowy z OSPF na coś innego. Granica stref zawsze przebiega przez router i zawsze istnieje area 0 (backbone), do której musi być podłączony każdy inny obczar, możliwe jest to również przez virtual link. Komunikaty LSA do komunikacji pomiędzy router'ami (1,2 dotyczą wewnątrz danej area, 3 od ABR, 4 reprezentuje ASBR, 5 trasy poza OSPF). Możliwe jest utworzenie stub area, czyli tylko informacje z danego OSPF totally stub area, czyli tylko z danej area, oraz NSSA, gdzie obszar dostaje informacje o swojej area oraz spoza OSPF. Możliwe jest włączenie uwierzytelniania.

Tunel posiada adres IP, prowadzony jest przez jakieś tam router'y w internecie, a nas interesuje tylko to, że pakiet ma dotrzeć na drugi koniec tunelu. Częstym zabiegiem jest przesyłanie ruchu ipv6 przez ipv4 właśnie przy użyciu tuneli.

5.2 Praktyka

5.2.1 RIP

Włączamy rutowanie i od razu możemy zastrzec bezklasowość (co prawda dopiero RIP w wersji drugiej zadziała bezklasowo, ale podamy oba warianty):

```
Router(config)#ip routing
Router(config)#ip classless
```

Włączamy RIP, podajemy sieci bezpośrednio podłączone i wyłączamy generalizację

```
Router(config)#router rip
Router(config-router)#network 200.200.200.0
Router(config-router)#network 0.0.0.0 // dodaje wszystkie bezpośrednio podłaczone
Router(config-router)#no auto-summary
```

Diagnostyka:

```
Router#show ip route
Router#show ip protocols
Router#debug ip rip
Router#debug ip routing
```

W wersji drugiej, możemy podać uwierzytelnianie kluczem (taki sam we wszystkich router'ach):

```
Router(config)#router rip
Router(config-router)#version 2
Router(config)#key chain nazwa
Router(config-keychain)#key 2 // 2 to identyfikator klucza
Router(config-keychain-key)#key-string 0987654321
Router(config-if)#ip rip authentication key-chain nazwa //przypisujemy do interfejsu
Router(config-if)#ip rip authentication mode md5
```

W wersji 2 dopuszczone jest włączenie przetwarzania komunikatów wersji 1.

```
Router1(config-if)#ip rip send version 2
Router2(config-if)#ip rip receive version 1
```

FailOver to dodatkowa trasa statyczna dodana, na wypadek gdyby przestały działać wszystkie trasy RIP'a. Musimy podać Administrative Distance większy niż 120, aby nie był brany pod uwagę przed RIP'em.

```
Router(config)#ip route 200.200.200.0 255.255.255.0 200.200.100.5 150 // 150 to ten distance
```

5.2.2 OSPF

Potrzebna jest domyślna bramka:

```
Router(config)#ip route 0.0.0.0 0.0.0.0 192.168.1.1
```

Włączamy OSPF

```
Router(config)#router ospf 150 // 150 to identyfikator procesu OSPF
```

Przypisujemy sieci bezpośrednio przyłączone (maska w bitowej inwersji):

```
Router(config-router)#network 200.200.200.0 0.0.0.255 area 0
Router(config-router)#network 200.200.201.0 0.0.0.255 area 0
```

Diagnostyka:

```
Router#show ip protocols
Router#show ip interface brief
Router#show ip ospf interface fa 0/0
```

W przypadku sieci Non Broadcast Multiple Access należy dodatkowo dopisać sąsiadów:

```
Router(config-router)#neighbor 200.200.100.1
Router(config-router)#neighbor 200.200.200.1
```

Tworzenie virtual link pomiędzy area (tam nie ma adresu ip, tylko router ID, który zdobywamy przy pomocy komendy „sh ip ospf interface"):

```
RABR1(config)#router ospf 150
RABR1(config-router)# area 1 virtual-link 5.5.5.5 // tutaj jest własnie ten router id
RABR2(config)#router ospf 150
RABR2(config-router)# area 1 virtual-link 6.6.6.6
```

Tworzenie kolejnego obszaru (musi być połączony poprzez router ABR lub virtual link z area 0):

```
Router(config)#int loopback 5
Router(config-if)#ip addr 200.200.101.1 255.255.255.0
Router(config-if)#exit
Router(config)#router ospf 150
Router(config-router)#network 200.200.101.0 0.255.255.255 area 1
```

5.2.3 EIGRP

Włączamy rutowanie, ustalić bramkę domyślną i protokół EIGRP:

```
Router(config)#ip routing
Router(config)#ip route 0.0.0.0 0.0.0.0 192.168.1.1 // adres bramki
Router(config)#router eigrp 1234 // tutaj jest identyfikator systemu autonomicznego EIGRP
Router (config-if)#bandwidth 128000  // wpływa na obliczanie metryk
```

Rejestrujemy sieci bezpośrednio podłączone:

```
Router(config-router)#network 200.200.200.0
Router(config-router)#network 200.200.201.0
```

Jeśli nie mamy wspierania multicastu, musimy jawnie podać sąsiadów:

```
Router1(config-router)#neighbor 200.200.200.1 fa 0/0 // interfejs do sąsiada
Router2(config-router)#neighbor 200.200.200.2 fa 0/0
```

Diagnostyka:

```
Router#sh ip eigrp topology
Router#sh ip eigrp topology all-links
Router#show ip eigrp topology 200.200.100.0
Router#sh ip eigrp neighbors
Router#sh ip eigrp traffic
Router#debug ip routing
Router#show ip route
Router#show ip protocols
```

5.2.4 Redystrybucja pomiędzy protokołami IGP

Np. mamy router z OSPF, jakiś z RIP i między nimi taki z OSPF i RIP i chodzi o ustalenie co ten środkowy musi mieć, aby poprawnie wymieniać informacje pomiędzy protokołami.

Redystrybucja w kierunku RIP do OSPF:

```
R2(config)#router ospf 4
R2(config-router)# redistribute rip metric 170 subnets // subnets żeby przesyłać też adresy classless
```

Redystrybucja w kierunku OSPF do RIP:

```
R2(config)#router rip
R2(config-router)#redistribute ospf 4 metric 11 // tutaj to ma być ilość hopek w RIP
```

5.2.5 Tunel GRE

W router'ach początku i końca tunelu wpisujemy np.:

```
R1(config)#interface Tunnel 0
R1(config-if)#tunnel source FastEthernet0/0
R1(config-if)#ip address 192.168.5.1 255.255.255.0
R1(config-if)#tunnel destination 200.200.201.1
R3(config)#interface Tunnel 0
R3(config-if)#tunnel source FastEthernet0/0
R3(config-if)#ip address 192.168.5.2 255.255.255.0
R3(config-if)#tunnel destination 200.200.200.1
```

Zmiana fragmentacji (wartości po obu stronach identyczne):

```
R1(config)#interface Tunnel 0
R1(config-if)#ip mtu 1460
R1(config-if)#ip tcp adjust-mss 1430
```

5.2.6 Rutowanie pomiędzy Vlan

Włączamy interfejs do tych Vlan'ów i usuwamy jego adres IP:

```
Router(config)#int fa 0/0
Router (config-if)#no ip address
Router (config-if)#no shutdown
Router (config-if)#exit
```

Definiujemy pod interfejsy i im ustalamy adres IP oraz rodzaj enkapsulacji:

```
Router(config)#int fa 0/0.1
Router(config-subif)#ip addr 10.1.0.1 255.255.0.0
Router(config-subif)#encapsulation dot1Q 2
Router(config)#int fa 0/0.2
Router(config-subif)#ip addr 10.2.0.1 255.255.0.0
Router(config-subif)#encapsulation dot1Q 4
```

Numeracja pod interfejsów jest zgodna z numeracją odpowiednich Vlan'ów. (w sensie to po kropeczce). Należy sprawdzić możliwośc rutowania pomiędzy Vlan'ami.

Możliwe jest również pod interfejsy nad fizycznymi łączami. Dzięki temu, można mieć w jednym połączeniu warstwy pierwszej, wiele połączeń warstwy drugiej. Identyfikatory VID pod interfejsów muszą być parami identyczne, a adresacja IP zgodna z zasadami, np:

```
R1(config)#interface fa 0/1.4
R1(config-subif)#encapsulation dot1Q 2
R1(config-subif)#ip address 11.0.0.1 255.255.255.0
R1(config-subif)#exit
R1(config)#interface fa /1.7
R1(config-subif)encapsulation dot1Q 4
R1(config-subif)ip address 13.0.0.1 255.255.255.0
R2(config)#interface fa 0/1.10
R2(config-subif)#encapsulation dot1Q 2
R2(config-subif)#ip address 11.0.0.2 255.255.255.0
R2(config-subif)#exit
R2(config)#interface fa /1.11
R2(config-subif)#encapsulation dot1Q 4
R2(config-subif)#ip address 13.0.0.2 255.255.255.0
```

6 Lab 5 NAT i IPv6

Zajmiemy się dwiema grubszymi kwestiami, jedna to będzie użytkowanie NAT poprzez wykorzystanie dobrodziejstw TCP i potężnych router'ów, drugą sprawą będą podstawy IPv6 wraz z rutowaniem dynamicznym oraz tunelowaniem go w IPv4.

6.1 Teoria

NAT (Network Address Translation) służy do konwersji (masquerade), ukrycia sieci wyizolowanej. Adresy źródłowe z tej sieci są zamieniane na inny, ustalony np. bramki z tej sieci. Możliwe są konwersje jeden do jeden, wielu do jeden, jak i wielu do wielu. Zwykle operuje się na trzech blokach adresów (Klasa A : 10.0.0.0 – 10.255.255.255, Klasa B : 172.16.0.0 – 172.31.255.255, Klasa C : 192.168.0.0 – 192.168.255.255).

Overloading pozwala na konwersję jeden do wielu. Wyróżniamy adres local, inside czyli w sieci prywatnej, oraz outside jest to adres w sieci publicznej, pod którym w niej występuje, są też adresy global, inside to adres interfejsu IP w sieci prywatnej, nadawcy datagramów do publicznej po konwersji, przy overloadingu również jest to adres interfejsu outside w router'ze NAT, a outside to rzeczywisty adres hosta w sieci publicznej.

Istnieją także translacje statyczne host-host, które definiujemy sami. Jest dwukierunkowa (to właściwie taka podmianka tylko). Możemy ustalić ograniczenie, aby z zewnątrz dało się np. tylko poprzez konkretny port TCP połączyć przez interfejs outside, wtedy w sieci inside możliwe jest świadczenie usług sieciowych.

IPv6 posiada adresy 128 bitowe. Pełna notacja to dwukropki co 16 bitów, podawana heksagonalnie. Można raz skrócić wszystkie zera podwójnym dwukropkiem. Maskę podajemy po znaku „/". Można utworzyć adres, podając adres MAC urządzenia i pośrodku wartość 0xFFFE(EUI-64). Interfejs może mieć wiele adresów IPv6, mogą być trwałe lub tymczasowe, dlatego każdy ma ustalony czas życia, domyślnie ustalony na nieskończoność. Ma własny stos TCP/IP.

Można mapować adresy IPv4 w IPv6. W wersji 6to4 adres to 2002:xxxx:xxxx::/16, a w miejsca x'ów wstawiamy przeliczony na szesnatkowe adres IPv4. Drugim sposobem jest 6 mapped 4, i tutaj możemy albo mieć przedrostek 0:0:0:0:0:FFFF:x.x.x.x i tutaj adres może być napisany szesnastkowo i maska /96 lub 0:0:0:0:0:0:x.x.x.x i adres szesnastkowo, maska taka sama. I to się przydaje np. przy tunelowaniu.

6.2 Praktyka

6.2.1 Overloading

Zestawiamy wszystko tak, aby istniał router obsługujący NAT, ustalamy dla niego gdzie jest sieć inside, a gdzie outside. Tworzymy listę ACL, zezwającą na ruch wychodzacy z sieci inside i przypisujemy do NAT po stronie wewnętrznej. np:

```
RouterNAT(config)#access-list 5 permit 10.0.0.0 0.0.0.255
RouterNAT(config)#ip nat inside source list 5 interface fa 0/1 overload
```

Określamy stronę NAT dla interfejsu po stronie wewnętrznej i zewnętrznej:

```
RouterNAT(config-if)#ip nat inside
RouterNAT(config-if)#ip nat outside
```

Diagnostyka:

```
RouterNAT#debug ip nat
RouterNAT#show ip nat translations
RouterNAT#show ip nat statistics
```

6.2.2 Translacje statyczne DMZ

Definiowanie reguły statycznej (10.0.0.3 z inside zostanie zamieniony i w global będzie widoczny jako 200.200.200.3):

```
RouterNAT(config)# ip nat inside source static 10.0.0.3 200.200.200.3
RouterNAT(config)# ip nat inside source static 10.0.0.4 200.200.200.4
```

Możliwe jest nawet ustalenie translacji do wirtualnego adresu z sieci niepodłączonej bezpośrednio, ale wtedy musimy sami ustalić regułę rutowania do fizycznego interfejsu NAT outside router'a. Możemy także zdefiniować przekierowanie, ustalić port na który ma przyjść do router'a oraz port na który ma trafić już przetranslatowany z router'a na danego hosta Robimy to po stronie inside, ze względu na dwukierunkowość.

```
RouterNAT(config)# ip nat inside source static tcp 10.0.0.5 23 200.200.200.1 100
RouterNAT(config)# ip nat inside source static tcp 10.0.0.5 80 200.200.200.1 200
```

Do diagnostyki (można spróbować np. połączyć się telnetem na porcie 100, podając ten adres z 200...):

```
RouterNAT#debug ip nat
RouterNAT#sh ip nat translatios
R3#debug ip tcp transactions
```

6.2.3 Podstawy IPv6

Zdefiniowanie adresu dla interfejsu:

```
Router(config-if)#ipv6 address 1111:2222:1111:2222::/64 eui-64
Router(config-if)# ipv6 address 1:1::1/64  //ze skróceniem
```

Diagnostyka:

```
Router#show ipv6 int brief
Router#show ipv6 route
```

Definiowanie statycznych reguł rutowania bardzo analogiczne do v4:

```
Router(config)#ipv6 route 3:1::/64 2:1::1
```

6.2.4 RIPng i OSPFv3

Uruchomienie rutowania i RIPng:

```
Router(config)# ipv6 unicast-routing
Router(config)# ipv6 router rip ripper // ripper to nazwa procesu RIPng
```

Wchodzimy do interfejsu i tam aktywujemy RIPng:

```
Router(config)#int fa 0/0
Router(config-if)#ipv6 rip ripper enable
Router(config-if)#no shutdown
Router(config-if)#exit
```

Diagnostyka:

```
Router#show ipv6 int fa 0/0
Router#show ipv6 route
Router#show ipv6 rip
Router#clear ipv6 route * //czyszczenie forwarding table
```

OSPFv3 uruchomienie:

```
Router(config)#ipv6 router ospf 10 // 10 to identyfikator procesu OSPF
```

Ręcznie konfigurujemy router-id, pamiętając o ich unikatowości:

```
Router(config-router)#router-id 1.1.1.1
```

Przypisujemy do OSPF interfejsy router'a:

```
Router(config)#interface FastEthernet 0/0
Router(config-if)#ipv6 ospf 10 area 0.0.0.0
```

Obszary i diagnostyka analogiczna:

```
Router# debug ipv6 ospf
Router#sh ipv6 protocols
Router#sh ipv6 route
Router#sh ipv6 ospf neighbor
```

I jeszcze na szybko uruchomienie EIGRP:

```
Router(config)# ipv6 router eigrp 1234 // tutaj to identyfikator systemu autonomicznego
```

Przypisanie zaangażowanych interfejsów:

```
Router(config)#int fa 0/0
Router(config-if)#ipv6 eigrp 10
```

Oczywiście przed uruchomieniem nowego systemu rutowania dynamicznego, poprzedni należy wyłączyć.

6.2.5 Tunelowanie IPv6 w IPv4

W routerze, który ma obsługiwać wyłącznie IPv6 można użyć takiej konfiguracji:

```
ipv6 unicast-routing //domyślnie wyłączona
ipv6 router rip ripper
interface Ethernet 0/1
no ip address
ipv6 address 2111:1:1:1:1:1:1:1112/112
ipv6 rip ripper enable
```

W routerze bramce, który ma rozpocząć tunelowanie (czyli w routerze początku i końca tunelu), włączamy i ipv4 i ipv6, skonfigurować tunel podając adres przeciwległego końca i definiujemy rutowanie dla sieci ipv4, aby dało się przez imitowany Internet przejść, przykładowo:

```
ipv6 unicast-routing
ipv6 router rip ripper
interface FastEthernet 0/0
 ip address 200.200.200.2 255.255.255.0
exit
interface FastEthernet 0/1
 no ip address
 ipv6 address 2111:1:1:1:1:1:1:1111/112
 ipv6 rip ripper enable
exit
router ospf 1
 log-adjacency-changes
 network 200.200.200.0 0.0.0.255 area 0
 router-id 200.200.200.2
exit
interface Tunnel0
 no ip address
```

```
 ipv6 address 3111:1:1:1:1:1:1:1111/112
 ipv6 rip ripper enable
 tunnel source FastEthernet 0/0
 tunnel destination 200.200.201.2
 tunnel mode ipv6ip
exit
```

Plus konfiguracja np. jednego router'a imitującego Internet:

```
ip subnet-zero
ip classless
interface FastEthernet 0/0
 ip address 200.200.200.1 255.255.255.0
exit
interface FastEthernet 0/1
 ip address 200.200.201.1 255.255.255.0
exit
router ospf 1
 log-adjacency-changes
 network 200.200.200.0 0.0.0.255 area 0
 network 200.200.201.0 0.0.0.255 area 0
 router-id 200.200.201.1
exit
```

Diagnostyka:

```
Router#show ipv6 int brief
Router2#show ip route
Router2#show ipv6 route
```

Powinno dać się komunikować przez tunel.

7 Lab 6 Route III

Głównie zajmiemy się tutaj zabawami związanymi z multicastem w ipv4, więc protokół IGMP (MLD jest dla multicastu ipv6), również pobawimy się z protokołowami rutowania multicastu. Drugim tematem będzie kilka pojedynczych wspomożeń takich jak protokół HSRB czy IP SLA służący do monitorowania rutowania czy informacji o stanie obciążenia sieci.

7.1 Teoria

Jeżeli ktoś chce być odbiorcą multicast'u, musi zapisać się poprzez protokół IGMP na bezpośrednio podłączonym router'ze. Router sprawdza wtedy na których interfejsach znajdują się zarejestrowani użytkownicy i to tam rozsyła ruch multicast. Działa Reverse path forwarding czyli transmisja danych od celu do źródła, liczy się adres nadawcy, a nie odbiorcy. Pula adresów to $224.0.0.0 - 239.255.255.255$ (w ipv6 to $FF :: /112$). W warstwie trzeciej datagram multicastu jest rozpoznawany poprzez specjalny MAC (części odpowiedzialnej za producenta) nadawcy $01 : 00 : 5E : 0...$ (po kropkach reszta to najmłodsze bity jest z adresu ip, tutaj może wystąpić pokrycie więc niektóre adresy nie są do końca bezpieczne, jeżeli adresy różnią się tylko na pięciu nastarszych bitach to będą uznane za jedną grupę). W warstwie drugiej wartości po kropkach są uzupełniane przez F, a ostatnie zero jest zamieniane na siódemkę (np. 01:00:5E:F:FF:FF).

IGMP służy więc głównie do zajmowania się takimi grupami multicastowymi. Dla ipv6 istnieje MLD. Host zgłasza się do grupy multicastu przez membership report, router dodatkowo może wysyłać membership query, aby wiedzieć do kogo nadal wysyłać, w przypadku nie otrzymywania membership report od żadnego użytkownika danej grupy multicastu usuwa ją.

IGMP działa w lokalnym segmencie sieci, pomiędzy sieciami odbywa się z pomocą protokołów multicast. Proste przełączniki rozsyłają ramki multicastu wszędzie, ale jeśli mają opcję Snooping, nie zaleją niepotrzebnych urządzeń.

PIM (Protocol Independent Multicast) dwa warianty - sparse mode gdzie są drzewa odbiorców każdej grupy odbiorca (router) musi czynnie wysłać Ŝądanie zarejestrowania go jako potencjalnego odbiorcy multicastu, służy do tego specjanie wybrany Rendevous Point - dense mode każdy ruter dostaje ruch IP Multicast, chyba ze czynnie zabroni wysyłania do niego takiego ruchu (wyśle komunikat prune), są też hybrydy gdzie RP wybierany automatycznie.

W tablicy rutowania multicastu znajdują się dwa drzewa komunikatów, Shared Tree prowadzi do RP od odbiorców (np. *,239.0.0.1), a w Source Tree wpis dotyczy konkretnego hosta będącego źródłem datagramów (np. 200.200.200.1, 239.0.0.1), drzewo do źródła. Dla pierwszego drzewa ustalany jest w tablicy rutowania multicastowej IncomingInterface, dla drugiego OutgoingInterface, znajdują się też informacje o wybranym protokole PIM. Istnieje coś takiego jak MOSPF, taki multicast przez OSPF (pozwala router'om w grupach OSPF dzielić się wiedzą na temat multicastu) czy MBGP czyli multicast w BGP.

Policy Based Routing wspomaga proces rutowania przez dodanie dodatkowych informacji takich jak QoS, flagi itp. W oparciu o ACL buduje się odpowiednie Route-Map, w których można określić np. adres next-hop, czy interfejs z którego ma przyjść datagram.

Możemy monitorować ruch w sieci przy pomocy IP SLA (Service Level Aggrement) czyli zestaw mechanizmów do nasłuchiwania lub np. konfigurowania zdalnego. Wyłączy nam to port z normalnego użytkowania.

HSBR (Hot Standby Routing Protocol) to protokół pozwalający na stworzenie „zapasowego" router'a na wypadek padnięcia głównego w sposób jakby hot-swap, bez wyraźnych opóźnień. Rutery są łączone w logiczną grupę o ustalonym numerze i współdzielą dodatkowy wirtualny adres IP. Jeden z ruterów jest w stanie active i on aktualnie obsługuje wirtualny adres IP, odpowiadając na wszelki ruch z nim związany. Gdy ruter ten ulegnie awarii - adres wirtualny migruje do kolejnego rutera bez powodowania przerwy w ruchu. Podobnie VRRP (Virtual Routing Redundancy Protocol).

7.2 Praktyka

Uwaga przed tym wszystkim, zanim zaczniemy routing multicast, musi stać i działać poprawny routing unicast!

7.2.1 IGMP Snooping w przełącznikach Ethernet

Podłączamy źródło mutlicastu i odbiorcę multicastu do switch'a wyposażonego w mechanizm snooping i pamiętamy o podłączeniu dodatkowego router'a. W switch'u włączamy IGMP snooping

```
Switch(config)#ip igmp snooping
```

Musimy podłączyć router, który będzie pełnił funkcję IGMP querier (do niego muszą iść komunikaty IGMP i on je musi rozumieć)

```
Router(config)#ip multicast-routing
Router(config)# int fa 0/0
Router(config-if)#ip pim dense-mode //przykładowo
```

Uruchamiamy nadawanie i odbieranie. Dobrze byłoby teraz podłączyć coś dodatkowego i zobaczyć czy ruch multicast jest też kierowany do niego (snooping powinien zablokować, o ile nie będzie wysyłał próśb o multicast). Do diagnostyki:

```
Switch#show ip igmp snooping
Switch#show ip igmp snooping group //rejestr grup
Switch#show mac-address-table multicast vlan 1 igmp-snooping
Switch#show ip igmp snooping querier //informacje o routerz'e
Switch#show ip igmp snooping mrouter
```

W routerz'e kasowanie grup IGMP

```
Router#clear ip igmp group
```

7.2.2 IP PIM dense mode

Czyli automatyczne rozgłaszanie do wszystkich router'ow, chyba że któryś wyśle informację że nie chce (prune).

Stwórzmy kilka sieci i włączym multicast:

```
Router(config)#ip multicast-routing
```

W każdym interfejsie na drodze multicastu:

```
Router(config)# int fa 0/0
Router(config-if)#ip pim dense-mode
```

Diagnostyka:

```
Router#show ip mroute
Router#show ip igmp groups
Router#show ip igmp membership
Router #show ip mroute summary
Router #show ip mroute active
Router #show ip mroute count
Router #show interfaces summary
```

Warto się pobawić, włączyć odbiorcę, wyłączać odbiorców, patrzeć jak router, gdy widzi że w swojej sieci nie ma odbiorców wysyła prune itp.

7.2.3 IP PIM sparse mode

Tutaj jest tak, że router widzi, że ma kogoś chętnego multicastu, to wysyła do Rendevous Point informację, żeby do niego nadawał (router'y po drodze wtedy też muszą przechwytywać ten multicast na tej drodze), w momencie otrzymania zgłoszenia IGMP od potencjalnego odbiorcy ruchu IP multicast (host zgłasza się w ruterze do grupy IP multicast) ruter rozpoczyna pobieranie od RP takiego ruchu i przekazywanie go do segmentu zawierającego wspomnianego hosta IP. W konfiguracji IP PIM sparse mode kaŜdy z ruterów musi posiadać informację o adresie IP rutera Rendezvous Point (nawet on sam).

```
Router(config)#ip multicast-routing
```

Teraz w interfejsach:

```
Router(config)# int fa 0/0
Router(config-if)#ip pim sparse-mode
```

Wskazujemy router RP:

```
Router(config)#ip pim rp-address 200.200.200.2
```

Diagnostyka:

```
Router#sh ip pim rp
Route#sh ip pim rp mapping
Router#sh ip pim nei
```

I znów warto się trochę pobawić!

7.2.4 Route Maps i Policy Based Routing

Przykładowe ACL dla Route Maps:

```
R1(config)#access-list 105 permit ip 200.200.202.0 0.0.0.255 200.200.201.0 0.0.0.255
R1(config)#access-list 105 deny ip any any
```

Definiowanie mapy:

```
R1(config)#route-map mapa1 permit 10
R1(config-route-map)#match ip address 105
R1(config-route-map)#set interface fa 0/1
R1(config-route-map)#set ip next-hop 200.200.200.2
```

Gdzie next-hop to adres kolejnego interfejsu na trasie, jaka ma być wybrana, 105 to ID listy ACL, interface fa 0/0 to interfejs otrzymujący ruch (ten oznaczony na rysunku strzałką), a interface fa 0/1 to interfejs na który ruch jest przekierowywany dalej (przez route map).

Przypisanie mapy do interfejsu:

```
R1(config)#interface fastethernet 0/0
R1(config-if)#no ip route-cache
R1(config-if)#ip policy route-map mapa1
```

Diagnostyka:

```
Router#show ip policy
Router#debug ip policy
Router#show route-maps
Router#show access-lists
```

7.2.5 Cisco HSRP i VRRP

Przygotowujemy dwa Router'y, jeden będzie w trybie active, drugi standby. Ich interfejsy muszą być w tych samych sieciach IP, w tej samej sieci musi znajdować się ich wspólny, wirtualny adres standby:

```
R1(config)#int fa 0/1
R1(config-if)#ip address 192.168.123.150 255.255.255.0
R1(config-if)#standby 1 ip 192.168.123.160
R1(config-if)#standby 1 priority 140
R1(config-if)#standby 1 preempt
R2(config)#int fa 0/1
R2(config-if)#ip address 192.168.123.151 255.255.255.0
R2(config-if)#standby 1 ip 192.168.123.160
R2(config-if)#standby 1 priority 90
R2(config-if)#standby 1 preempt
```

Protokół określa protokół active za sprawą priorytetu, którym możemy manipulować:

```
R1(config-if)#standby 1 priority 30
```

Najprzyjemniejsza część, zepsujmy router active i oglądajmy jak ruch przechodzi przez standby. A teraz wytestujmy drugi sposób.

Przed uruchomieniem VRRP wyłączmy HSRP:

```
R1(config)#int fa 0/1
R1(config-if)#no standby 1
R2(config)#int fa 0/1
R2(config-if)#no standby 1
```

Znów określamy wirtualny adres, cechę i priorytet.

```
R1(config)#int fa 0/1
R1(config-if)#vrrp 1 ip 192.168.123.160
R1(config-if)#vrrp 1 priority 110
R1(config-if)#vrrp 1 preempt
R2(config)#int fa 0/1
R2(config-if)#vrrp 1 ip 192.168.123.160
R2(config-if)#vrrp 1 priority 100
R2(config-if)#vrrp 1 preempt
```

Powyższe protokoły umożliwiają przeprowadzenie autentyfikacji stron:

```
R1(config)#key chain klucz
R1(config-keychain)# key 1
R1(config-keychain-key)#key-string klucz
R1(config-keychain-key)#exit
R1(config-keychain)#exit
R1(config)#int fa 0/1
R1(config-if)#vrrp 1 authentication md5 key-chain klucz
```

7.2.6 IP SLA

Skonfigurujemy router, który jeżeli nie będzie miał odpowiedzi od danego hosta, przeniesie ruch na innego automatycznie.

```
Router(config)#ip sla monitor 5
Router(config-sla-monitor)#type echo protocol ipicmpEcho 200.200.200.1
source-interface FastEthernet0/0
Router(config-sla-monitor)#timeout 1000
Router(config-sla-monitor)#threshold 2
Router(config-sla-monitor)#frequency 3
Router(config-sla-monitor)#exit
Router(config)#ip sla monitor schedule 5 life forever start-time now
```

gdzie 5 to numer definiowanej właśnie operacji monitorowania IP SLA.

W nowszych wersjach będzie to:

```
Router(config)#ip sla 5
Router(config-sla-monitor)# icmp-echo 200.200.200.1 source-interface
FastEthernet0/0
Router(config-sla-monitor)#timeout 1000
Router(config-sla-monitor)#threshold 2
Router(config-sla-monitor)#frequency 3
Router(config-sla-monitor)#exit
Router(config)#ip sla schedule 5 life forever start-time now
```

Diagnostyka:

```
Router#show track
Router#show ip sla monitor operational-state 1
Router#show ip sla monitor statistics
Router#debug ip routing
```

8 Lab 7 BGP

Zajmiemy się protokołami BGP działającymi wewnątrz jednego systemu autonomicznymi, jak i tą wersją operującą pomiędzy systemami autonomicznymi, oraz redystrybucją tras pomiędzy nimi.

8.1 Teoria

Poznamy teraz kolejne protokoły rutowania dynamicznego. BGP (Border Gateway Protocol) ma swoją wersję wewnątrz systemu autonomicznego Interior i na zewnątrz Exterior, pracuje na TCP port 179, identyfikator systemu autonomicznego to szesnastobitowa liczba. Router'y w BGP komunikują się ze soba (Open, Keepalive, update, notification). TTL w eBGP wynosi 1, w celu zapobiegania pętlom. Zwykle musimy określić sąsiadów BGP, po uruchomieniu go.

Prefix to zakres IP z maską i innymi cechami, przesyłany w komunikatach BGP. Mają klasy (np. Well-known Mandatory, Well-known Discretionary) określające gdzie które mają trafić. Atrybuty takie jak Origin

(skąd dany prefix pochodzi), As_Path (kolejka systemów autonomicznych przez które trzeba przejść by dotrzeć do prefixu), Next-hop (gdzie bezpośrednio kierować kolejny skok dla prefixu), Weight (priorytet lokalny określający którędy będzie wybrana trasa), Local-preference (określany dla AS, biorący pod uwagę ruch wychodzący z AS), MED (Multi Exit Discriminator, określa preferowane wyjście, gdy dwa AS mają wiele połączeń i nie wiadomo, przez które powinna prowadzić droga) itp.

Router korzysta z bazy danych prefixów i na tej podstawie określa trasę, leci w kolejności: Weight, Local_Preference, Origin, As_Path, Med. Kieruje się też regułami, aby nie kierować gdzieś gdzie next-hop nie jest osiągalny, nie brać pod uwagę trasy iBGP jeśli sesja nie jest aktywna. Możliwy jest multihoming - definiowanie kilku dobrych tras, z podziałem ruchu. Wiele AS może zostać połączone w Community, które łączy je np. pod względem geograficznym.

W eBGP przeciwko pętlom chroni wartość TTL, ale w iBGP trzeba w tym celu utrzymywać wariant full-mesh. Można to zrobić wykorzystując Route Reflector, jeden router (można też kilka...), do którego klienci (RRC), muszą się podłączać i prosić o info. Drugą techniką jest tworzone konfederacji AS, czyli takich trochę fikcyjne AS, które wymieniają ze sobą informacje bezpiecznie.

Przy wykorzystaniu loopback pojawia się konieczność skonfigurowania w BGP procesu przekazywania informacji o tym interfejsie do innych ruterów (BGP nie robi tego automatycznie).

8.2 Praktyka

8.2.1 iBGP full mesh

Tworzymy pare sieci w jednym systemie autonomicznym:

```
Router(config)#ip routing
Router(config)#router bgp 65005
```

Jeżeli w router'ach nie ma jeszcze adresacji ip, musimy dodatkowo okreslić BGP id:

```
Router(config-router)#bgp router-id 200.200.205.1
```

Rejestracja sąsiednich router'ów:

```
R1(config-router)#neighbor 200.200.201.1 remote-as 65005
R1(config-router)#neighbor 200.200.203.2 remote-as 65005
```

Rejestracja sieci bezpośrednio podłączonych (każda sieć przynajmniej w jednym routerz'e):

```
R1(config-router)#network 200.200.205.0
R2(config-router)#network 200.200.206.0
R3(config-router)#network 200.200.204.0 //sieci nieklasowe trzeba z maską
Router(config-router)#network 10.10.10.0 mask 255.255.255.0
```

Diagnostyka:

```
Router#show ip bgp
Router#show ip bgp neighbors
Router#show ip protocols
Router#show ip bgp update-group
```

8.2.2 iBGP Route Reflector

Teraz ustalimy jeden router, do którego będą podłączać się klienci po informacje o trasowaniu w AS (w przykładach RR).

```
RR(config)#router bgp 65005
RR(config-router)# no synchronization
RR(config-router)# no auto-summary
RR(config-router)#neighbor 200.200.202.2 remote-as 65005
RR(config-router)#neighbor 200.200.202.2 route-reflector-client
```

```
RR(config-router)#neighbor 200.200.201.2 remote-as 65005
RR(config-router)#neighbor 200.200.201.2 route-reflector-client
RR(config-router)#end
```

Pozostałe router'y konfigurujemy tak jak poprzednio.

8.2.3 iBGP konfederacja AS

W ruterach mających komunikować się z innymi AS będących w konfederacji należy ten fakt zgłosić:

```
R1(config)router bgp 65001
R1(config-router)#bgp confederation identifier 100
R2(config)router bgp 65002
R2(config-router)#bgp confederation identifier 100
```

Informujemy router'y o istnieniu innych AS:

```
R1(config)router bgp 65001
R1(config-router)# bgp confederation peers 65002
R2(config)router bgp 65002
R2(config-router)# bgp confederation peers 65001
```

Następnie w każdym z konfedereatów wykonujemy już sprawy jakbyśmy byli w jednym BGP.

8.2.4 Konfigurowanie eBGP

Tym razem montujemy kilka osobnych systemów autonomicznych (żeby się nie pomylić może w nich być np. po jednym routrz'e), i znów ustalamy np.:

```
R2(config)#ip routing
R2(config)#router bgp 65002
```

Rejestrujemy sąsiednie router'y z innych systemów:

```
R1(config-router)#neighbor 200.200.201.1 remote-as 65002
R2(config-router)#neighbor 200.200.201.2 remote-as 65001
R2(config-router)#neighbor 200.200.202.2 remote-as 65003
R3(config-router)#neighbor 200.200.202.1 remote-as 65002
Router(config)#no parser cache //w razie problemów
```

W router'ach rejestrujemy widoczne dla nich sieci:

```
Ruter(config-router)#network 200.200.201.0 mask 255.255.255.0
```

Diagnostyka jak w pierwszym podrozdziale.

8.2.5 eBGP manipulowanie atrybutami tras

Atrybuty modyfikujemy poprzez route-maps.

```
Router(config)#route-map mapa
Router(config-route-map)#set metric 123456
Router(config-route-map)#set origin incomplete
```

I możemy zmieniać wybrane atrybuty np:

```
R1(config)#route-map mapa1
R1(config-route-map)#set local-preference 18
R1(config-route-map)#set weight 19
R1(config-route-map)#set metric 333
R1(config-route-map)#set as-path prepend 200 100
R1(config-route-map)#exit
```

8.2.6 eBGP policy-based routing

Blokowanie wysyłania ogłoszeń BGP

```
R3(config)#access-list 15 deny 200.200.230.0 0.255.255.255
R3(config)#access-list 15 permit 0.0.0.0 255.255.255.255
R3(config)# route-map nie_wysylaj permit 10
R3(config-route-map)#match ip address 15
```

gdzie 10 to numer kolejny route-map branej pod uwagę, 15 to identyfikator ACL opisujących route-map i zgłaszamy route-map w BGP:

```
R3(config-router)# neighbor 200.200.200.1 route-map nie_wysylaj out
```

Blokowanie przyjmowania:

```
R3(config)#access-list 16 deny 200.200.220.0 0.255.255.255
R3(config)#access-list 16 permit 0.0.0.0 255.255.255.255
R3(config)#route-map nie_odbieraj permit 10
R3(config-route-map)#match ip address 16
R3 (config)#router bgp 65003
R3 (config-router)#neighbor 200.200.200.1 route-map nie_odbieraj in
```

8.2.7 Redystrybucja do i z RIP

RIP do BGP:

```
R1(config)#router bgp 65005
R1(config)#redistribute rip metric 2000
```

BGP do RIP:

```
R1(config)#router rip
R1(config)# redistribute bgp 65001 metric 12
R1(config-router)# redistribute-internal
```

9 Lab 8 Traffic

9.1 Teoria

Nagłówek datagram IP składa się z:

- Wersja – [4 bity] – numer wersji protokołu IP,
- IHL – [4 bity] – (Internet Header Length) długość nagłówka w słowach,
- ToS (Type of Service) opisuje jakość wymaganej usługi. Zawiera flagi definiujące pierwszeństwo, niezawodność, opóźnienie krytyczne,
- DSCP/ECN – [8 bitów] 8 bitów podzielono na 6 bitowy Differential Services Code Point identyfikujący klasę QoS oraz 2 bitowy kod Explicit Congestion Notification obsługujący przypadki przeciążeń ruterów,
- Długość całkowita – [16 bitów] – jest długością pakietu IP w bajtach (nagłówek i dane),
- Identyfikator – [16 bitów] – wartość identyfikacyjna przypisana nadawanemu pakietowi przed fragmentacją, W przypadku fragmentacji określa ona przynależność fragmentu do datagramu,
- Flagi – [3 bity] – zerowy bit musi być zero, pierwszy jak 1 to nie wolno fragmentować, jak zero to wolno, drugi jak jeden to znaczy że jest jeszcze więcej datagramów, zero że nie ma,

- Przesunięcie fragmentacji – [13 bitów] – pole to wskazuje, do którego miejsca pakietu danych naleŜy ten fragment, jest mierzone w jednostkach 8 po bajtów,
- Czas Życia – [8 bitów] – TTL (Time-to-Live) - pole to wskazuje maksymalny czas przebywania pakietu w Internecie wyrażony w hop'ach przez bramki,
- Protokół – [8 bitów] – pole to wskazuje numer protokołu warstwy wyższej, do którego zostaną przekazane dane z tego pakietu (istnieje spójny system rejestracji protokołów warstw wyższych np. ICMP = 1),
- Suma kontrolna – [16 bitów] – obliczana i sprawdzana za kaŜdym razem, gdy dany nagłówek jest przetwarzany,
- Adres źródła i przeznaczenia – po [32 bity] – adres IP źródła danych i komputera docelowego.

Każdy rodzaj sieci ma określony maksymalny rozmiar pakietu MTU (Maximum Transmission Unit). W trakcie przekazywania danych, może się okazać, że MTU właściwy dla jednej z sieci, jest zbyt duży dla następnej. Datagram jest ubijany i wysyłane jest ICMP z informacją o konieczności fragmentacji.

TCP (Transmission Control Protocol) zapewnia połączeniową transmisję danych. Strumienie danych przesyłane w datagramach IP, brak ograniczeń czasowych dla strumienia i możliwość sterowania przepływem. Potwierdzanie pozytywne (Po otrzymaniu wiadomość zwrotna do nadawcy, jeśli nadawaca takowej nie otrzyma następuje automatyczna retransmicja niepotwierdzonych danych, stąd pewność że ostatecznie odbiorca poskłada je tak jak trzeba i niczego po drodze nie straci). Pacing, przez sourse quench, jeżeli odbiorca nie nadąża. Operuje na szesnastobitowych portach. PMTUD (Path MTU Discovery) wykrywa MTU na całej ścieżce, tak aby nie robić niepotrzebnych fragmentacji, tylko dobrać rozmiar segmentu TCP i datagramu IP, aby TCP datagram IP zmieścił się w jednej ramce.

Budowa nagłówka segmentu TCP:

- Numer portu źródłowego i docelowego – po [16 bitów] – port źródłowy i adres źródłowy IP funkcjonują jako adres, port docelowy zawiera adres interfejsu aplikacji w komputerze odbiorcy,
- Numer sekwencji – [32 bity] – wykorzystywany przez komputer odbierający do zrekonstruowania rozproszonych, rozbitych, podzielonych danych i przywrócenia im pierwotnej postaci,
- Numer potwierdzenia (ACK), - [32 bity] pierwszy oktet danych zawarty w następnym oczekiwanym segmencie,
- Wyrównanie danych [4 bity] pole przechowuje rozmiar nagłówka TCP, którego miarą jest 32-bitowa struktura danych, znana jako "słowo",
- Flagi [6 bitów] - określają rzeczy takie jak pilność, zerowanie, zakończenie połączenia itp,
- Rozmiar okna [16 bitów] - używa host docelowy w celu poinformowania komputera źródłowego o tym, ile danych jest gotów przyjąć w jednym segmencie TCP,
- Suma kontrolna [16 bitów] (na podstawie segmentu) i wypełnienie to zera dodane po to, aby segment był wielokrotnością 32.

UDP - User Datagram Protocol zapewnia bezpołączeniową transmisję. Nie następuje sprawdzanie danych, musi zmieścić się w datagramie IP, dobry gdy nie interesuje nas każdy docierający pakiecik, ale najnowsze dane (np. transmisje multimedialne). Nie ma tutaj strumieniowania i nie ma też w związku z tym niezawodności.

Budowa nagłówka segmentu UDP:

- Numer portu źródłowego i docelowego – po [16 bitów] – jak przy TCP,
- Suma Kontrolna [16 bitów] - generowana na podstawie segmentu do detekcji błedów,
- Długość komunikatu [16 bitów] - informacja dla komputera docelowego, w celu dodatkowego sprawdzania błędów.

Asocjacja zawiera rodzaj protokołu, adres węzła oferującego usługę number portu lokalnego (do tego miejsca też przy półasocjacji) oraz number węzła zdalnego i adres jego portu.

W warstwie drugiej i trzeciej operuje MPLS (MultiProtocol Label Switching).

9.2 Praktyka

Część praktyczna polegała tutaj głównie na porównywaniu zawartości ramek oglądanych przez WireShark z teoretycznymi. Pakiety ARP, IP, wraz ze wszystkimi elementami opisanymi w teorii. Ale też śledzenie i wykazywanie różnic w protokołach rutowania dynamicznego RIP, RIPv2, OSPF, EIGRP. Również protokołów pomocniczych ICMP (np. wiadomość Network Unreachable), a także usług TELNET, SSH, HTTP, HTTPS, SSL, FTP, usług DHCP, DNS, POP3 i SNMP.

Co możemy znaleźć w:

1. Ramce ARP
    - Długość protokołu wyższej warstwy,
    - Rodzaj operacji (Zapytanie, Odpowiedź, Zapytanie odwrotne, odpowiedź odwrotna),
    - Adres sprzętowy źródła (SHA) - nadawcy,
    - Adres sprzętowy przeznaczenia (THA) - odbiorcy,
2. Datagramie RIP
    - Rodzaj polecenia (żądanie czy odpowiedź),
    - Numer wersji - no tutaj 1,
    - Uaktualnienia przez UDP na porcie 520,
3. Datagramie OSPF
    - Wersja i typ (typ zapytania czy odpowiedź czy zapytanie),
    - Identyfikatory router'a i obszaru,
    - Kod uwierzytelniania,
    - Porty i adresy...

10 Lab 9 WAN I

Krótki wstęp do sieci rozległych, zajmiemy się łączami ATM, takim klasycznym połączeniem oraz połączeniem T3/E3. (T-carrier, E-carrier czyli wysoko przepustowe połączenia PPP).

10.1 Teoria

ATM (Asynchornous TransferMode), operuje na celkach (53 bajty), zapewnia że celki nie mogą się wyprzedzić, celki przechodzą pomiędzy przełącznicami ATM, poprzez wirtualny interfejc VCC (Virtual Channel Connection) stworzony pomiędzy stacjami źródłową, a docelową, zestaw takich kanałów tworzy ścieżkę (virtual Path Connection). Połączenia te funkcjonują w drugiej warstwie OSI, więc w łaczu fizycznym może istnieć wiele logicznych. UNI (User-Network Interface) - interfejs do urządzenia brzegowego, NNI (Network-Network Interface) - pomiędzy przełącznicami (Też są dwie celki dla nich, w UNI jest pole GFC okresla CoS). W celce nie ma adresów końca tylko identyfikatory VPI (Virtual Path Identifier), VCI, (Virtual Channel Identifier), przez które ma przejść przez ATM.

AAL (ATM Adaptation Layer) określa zasady transmisji przez sieć ATM dotyczace fragmentacji, kontroli błędów i przepływu, na podstawie typu pochodzącego zgłoszenia, pomaga określić technologię wspierające transmisję i dostosowuje jednostki informacyjne warstw wyŝszych do sposobu przesyłania informacji w standardzie ATM (Frame Relay, SONET, Gigabit Ethernet).

W ATM mogą zostać zestawione połączenia PVC (Permament Virtual Connection) - trwałe, zestawiane ręcznie w przełącznicy ATM, oraz SVC (Switched Virtual Connection) - zestawiane na podstawie protokołu rutowania potrzebne są adresy AESA NSAP ATM dla wyjść z sieci (protokół ILMIi PNNI).

Technika IMA (Inverse Multiplexing over ATM) polega na zestawieniu kilku łącz fizycznych w jedno logiczne łącze ATM (ATM Cell Scrambling), czyli zwielokrotnienie dające nam zwielokrotnienie kanałów i niezawodność, chociaż nadal celki nie będą się wyprzedzać. I tak dodatkowo: SONET (Synchronous Optical NETwork) i SDH (Synchronous Digital Hierarchy).

10.2 Praktyka

10.2.1 Konfiguracja prostego połączenia PVC

Porty modułów, które moga korzystać z ATM łączymy światłowodem i włączamy rutowanie.

```
Router(config)#ip routing
```

Konfigurujemy ATM, nadając adresację IP w interfejsie:

```
Router(config)# interface atm 2/0
Router(config-if)# ip address 200.200.200.1 255.255.255.0
```

W jednym z końców określamy zegarownie linii jako wewnętrzne, on będzie wtedy nadawał na podstawie sygnałów z router'a:

```
Router(config-if)#atm clock internal
```

Teraz w obydwu utworzony zostanie pvc, musimy podać identyfikator wirtualnej ścieżki i kanału:

```
Router(config-if)# pvc 1/32 // gdzie 1 to VPI, 32 to VCI
Router(config-if-atm-vc)#
```

Po obu stronach musimy zezwolić na ruch do określonego hosta, można również zastosować metodę dodatkowego rozgłaszania broadcast:

```
Router(config-if-atm-vc)#protocol ip 200.200.200.2 broadcast
```

Diagnostyka:

```
Router#show interface atm 2/0
Router#show controller atm1/0
Router#show atm vc
Router#show atm vp
```

10.2.2 Ppołączenie T3/E3 ATM

Porty modułów T3/E3 łączymy kablami koncentrycznymi pamiętając o krzyżowaniu (TX ¡=¿ RX). Włączamy routing i określamy adresację w interfejsach ATM.

```
Router(config)#ip routing
Router(config)#interface atm 2/0
Router(config-if)#ip address 200.200.200.1 255.255.255.0
Router(config-if)#no sh
```

Podobnie jak wyżej zostaje utworzone PVC i musimy podać hostów (lub trick z broadcastem):

```
Router(config-if)# pvc 1/32
Router(config-if-atm-vc)#protocol ip 200.200.200.2 broadcast
```

Diagnostyka analogiczna.

11 Lab 10 WAN II

Kontynuujemy temat WAN, teraz będziemy nadbudowywać DSL nad ATM, a później postaramy się stworzyć łącza PPP nad ATM i na Ethernet'em.

11.1 Teoria

DSL (Digital Subscriber Line) modem, przeprowadza łącza hurtowo do ATM, zasięg kilku kilometrów, szerokopasmowa transmisja danych. SHDSL (Symmetric High-speed Digital Subscriber Line) wykorzystuje telefoniczną linię dierżawioną, dla router'ów może tworzyć mostkek. Wyróżniana jest strona master CO (Central Office Equipment), oraz slave CPE (Customer Premises Equipment).

PPPoA oraz PPPoE pozwalają nam na zestawienie połączeć PPP enkapsulowanych w Ethernecie lub ATM z wszystkimi jej dobrami (np. autentykacja PAP, CHAP). Wyróżniamy stronę servera oraz klienta. Po stronie klienta tworzony jest interfejs wirtualny Dialer, połaczony z fizycznym interfejsem ATM lub Ethernet, po stronie servera tworzony jest Virtual-Template, który dla dialerów-klientów tworzy odpowiednie Virtual-Access, które będą miały ten sam adres IP co template (oszczędnośc!).

11.2 Praktyka

11.2.1 Instalacja SHDSL w trybie CO-CPE

Najpierw przygotowujemy mostek IRB:

```
Router(config)bridge irb
```

Określamy zasady filtrowania protokołów dla DSL w mostku:

```
Router(config)#bridge 1 protocol ieee
Router(config)#bridge 1 route ip
Router(config)#ip forward-protocol spanning-tree
```

W interfejsie, w którym tworzymy mostek oczywiście musimy wyłączyć adresację IP i przypisać go do mostka:

```
Router(config-if)#no ip address
Router(config-if)#bridge-group 1
```

Teraz konfigurujemy interfejs ATM zawierający dodatkowo konfigurajce linii DSL:

```
Router(config)#int atm 0/0
//wybieramy protokół kodowania DSL (zgodny z obu stron)
Router(config-if)#dsl operating-mode gshdsl symmetric annex A
Router(config-if)# dsl equipment-type co //lub cpe, strona konfiguracji
Router(config-if)# dsl linerate auto // prędkośc łącza
Router(config-if)# dsl linerate 520 // można ręcznie ustawić prędkość linii,
//wtedy trzeba wyłączyć i włączyć dla ustalenia
```

Dodanie ruchu do łącza DSL oraz konfiguracja PVC:

```
Router(config)#int atm 0/0
Router(config-if)#no ip address
Router(config-if)#bridge-group 1
Router(config-if)#pvc 0/40
// lub Router(config-if)#pvc tunnel 0/40
Router(config-if-atm-vc)#encapsulation aal5snap // po stronie CO
```

Na sam koniec dołaczamy interfejsy BVI, aby umożliwić dostęp z samego router'a do mostka. (Ale to już było opisane! 4.2.3).

11.2.2 PPPoA

Łączymy standardowo poprzez interfejsy ATM. Teraz po stronie będącej serverem tworzymy użytkownika, do autentyfikacji PAP.

```
R1(config)#aaa new-model
R1(config)#username sieci password sieci
R1(config)#aaa authentication ppp default local // niekiedy konieczne
```

Tworzymy PVC w interfejsie ATM, określając że powstaje jako fragmentacja dla PPP:

```
R1(config)#interface ATM 0/0
R1(config-if)#no shut
R1(config-if)#pvc 1/35
R1(config-if-vc)#encapsulation aal5mux ppp Virtual-Template 1
```

Tworzymy Virtual-Template

```
R1(config)#interface Virtual-Template 1
R1(config-if)#ip address 200.200.200.1 255.255.255.0
R1(config-if)#ppp authentication pap
```

Teraz w kliencie również tworzymy PVC i określamy, że PPP ma iść jako fragmenty w ATM, ale tutaj przypisujemy jeszcze dany interfejs ATM do danego dialer pool:

```
R2(config)#interface ATM 0/0
R2(config-if)#no shut
R2(config-if)#pvc 1/35
R2(config-if-vc)#encapsulation aal5mux ppp Dialer
R2(config-if-vc)#dialer pool-member 1
```

Teraz konfigurujemy samego Dialer'a:

```
R2(config)#interface Dialer 1
R2(config-if)#ip address 200.200.200.3 255.255.255.0
R2(config-if)#dialer pool 1
R2(config-if)#encapsulation ppp
R2(config-if)#ppp pap sent-username sieci password sieci
```

Dodatkowo w serverze możemy dodać rozszerzenie automatycznego nadawania adresu IP klientów (więc w kliencie będziemy musieli uprzednio usunąć adres IP):

```
//W serverze zakładamy pulę adresów, musi należeć do sieci interfejsu
R1(config)#ip local pool pula 200.200.200.3 200.200.200.10
R1(config)#interface Virtual-Template 1
R1(config-if)#peer default ip address pool pula
//W kliencie usuwamy adres ip
R2(config)#interface Dialer 1
R2(config-if)#no ip address
R2(config-if)#ip address negotiated
```

11.2.3 PPPoE

Będziemy działali podobnie, z tym że pobawimy się dodając przełącznik Ethernet i od razu kilku klientów, wraz z możliwym monitorowaniem.

W przełączniku konfigurujemy interfejs dla SPAN (ten do analizy, wyłączony z normalnego ruchu):

```
Switch(config)#monitor session 1 source interface Fa0/1 both
Switch(config)#monitor session 1 destination interface Fa0/4
```

Znów zakładamy klienta w serverze:

```
R1(config)#aaa new-model
R1(config)#username sieci password sieci
```

Tworzymy też grupę BBA (BroadBand Aggregation Group), z nią będzie powiązany interfejs Virtual-Template:

```
R1(config)# bba-group pppoe global //global domyślna, można też swoje
R1(config-bba-group)virtual-template 1
```

Konfigurujemy wybrany interfejs Ethernet, dla Virtual-Template i zakładamy pulę adresów dla klientów:

```
R1(config)#interface Fa 0/0
R1(config-if)#no shut
R1(config-if)#pppoe enable group global
R1(config)#ip local pool pula 200.200.200.3 200.200.200.10
```

Tworzymy Virtual-Template o numerze z grupy BBA:

```
R1(config)#interface Virtual-Template 1
R1(config-if)#ip address 200.200.200.1 255.255.255.0
R1(config-if)#ppp authentication pap
R1(config-if)#peer default ip address pool pula
R1(config-if)#mtu 1492 // dla PPP konieczne zmniejszenie MTU
```

W przykładowym kliencie konfigurujemy wybrany interfejs Ethernet:

```
R2(config)#interface Fa 0/0
R2(config-if)#no shut
R2(config-if)#pppoe-client dial-pool-number 1
```

Tworzymy Dialer z liczbą identyfikującą dialer pool:

```
R2(config)#interface Dialer 1
R2(config-if)#ip address negotiated
R2(config-if)#mtu 1492
R2(config-if)#dialer pool 1
R2(config-if)#encapsulation ppp
R2(config-if)#ppp pap sent-username sieci password sieci
```

Jeżeli chcemy posiadać ruch pomiędzy klientami, będzie on przechodził przez server (hub-on-spoke). Konieczne jest nadanie odpowiednich reguł rutowania.

Sprawdzanie stanu sesji:

```
R1#show pppoe session
R1#show pppoe statistics
R1#show pppoe summary
```

12 Lab 11 CORE

Będziemy obsługiwali Vlan'y w przełączniku rutującym Cisco oraz zajmiemy się przełącznikiem rutującym HP ProCurve, gdzie stworzymy Vlan'y, wraz z rutowaniem pomiędzy nimi, oraz odpalimy VTP.

12.1 Teoria

Teraz zajmiemy się nieco solidniejszym sprzetem. Na pierwszy ogień router z serii 3700 z dobudowanym modułem przełącznika. Może definiować fizyczne interfejsy IP jak i sieci Vlan (a więc ustalamy czy dany port będzie obsługiwał wartwę drugą czy trzecią, tak jakby). Głównie zauważenie, że wszystko z wcześniejszych rzeczy w stylu Vlan i Trunk działa.

Na drugi ogień idzie sprzęt rodu Core, Hp ProCurve 2650. Prawdziwym problemem okazało się jednak połączenie zdalne do tego przełącznika poprzez TAS (Terminal Access Server). Na jednym PC konfigurujemy, drugi służy do sprawdzania, oba łączymy do sieci laboratorium i konieczne jest utworzenie nowego interfejsu Vlan w przełączniku... cuda nie widy!

12.2 Praktyka

12.2.1 Moduł ESW w ruterze Cisco + Vlan Trunk i VTP

Umożliwienie konfiguracji przez telnet:

```
Router(config)#enable password cisco
Router(config)#ip http server
Router(config)#ip http authentication local
Router(config)#username sieci privilege 15 password 0 sieci
Router(config-line)#line console 0
Router(config-line)#login local
Router(config-line)#exit
Router(config)#line vty 0 4
Router(config-line)#privilege level 15
Router(config-line)#login local
Router(config-line)#transport input telnet
```

Tworzenie Vlan (nie możemy być w trybie VTP client):

```
Router#vlan database
Router(vlan)#vtp transparent
Router(vlan)#vlan 10 //tworzenie nowego
Router(vlan)#no vlan 10 //likwidacja
```

Przydzielenie portu do Vlan:

```
Router(config)# int fa 4/18
Router(config-if)# switchport mode access
Router(config-if)# switchport access vlan 10
Przydzielenie zakresu portów do wybranego VLAN:
Router(config)# int range fa 4/1 - 10
Router(config-if)# switchport mode access
Router(config-if)# switchport access vlan 10
```

Definiowanie IP dla Vlan:

```
Router(config)#interface vlan 10
Router(config-if)# ip address 2.2.2.2 255.255.255.0
```

Definiowanie trunk w przełączniku:

```
Router(config)#interface fa 4/1
Router(config-if)#switchport mode trunk
Router(config-if)#switchport trunk allowed vlan 1,10-11,1002-1005 //musimy wszystkie domyślne
```

Możliwe używanie vtp (co było widać już wcześniej):

```
Router(config)#vtp domain domena
Router(config)#vtp mode server
Router(config)#vtp mode client // transparent
```

12.2.2 Konfiguracja ProCurve 2650 i zabezpieczenia

Gdy uda nam się już jakoś zdalnie z tym połaczyć... Zabezpieczenia:

```
Switch(config)#telnet-server
Switch(config)#web-management
Switch(config)#ip ssh key-size 1024
Switch(config)#crypto key generate
Switch(config)#ip ssh port 129 //inny niż SSH
Switch(config)#ip ssh
```

Diagnostyka:

```
Switch(config)#show telnet
Switch(config)#show ip ssh
```

Konfiguracja community serwera SNMP i inne:

```
Switch(config)#snmp-server community "public" unrestricted
Switch(config)#trunk 1-5 nazwa //konfigurowanie trunk
Switch(eth-4)#qos priority 6 // musimy być w danym porcie
```

12.2.3 ProCurve Vlan i rutowanie pomiędzy Vlan

Tworzenie i diagnostyka Vlan:

```
Switch(config)#vlan 20
Switch(config)#show vlan
```

Przypisywanie portów do Vlan:

```
Switch(vlan-20)#tagged 8 // do komunikacji z innym DCE
Switch(vlan-20)#untagged 9-16
```

Adresy Ip dla Vlan'ów:

```
Switch(vlan-20)#ip address 200.200.0.1 255.255.255.0 //lub:
Switch(config)#vlan 1 ip address 200.200.0.1 255.255.255.0
```

Włączenie rutowania:

```
Switch(config)#ip routing
```

Ustalamy każdej Vlan interfejs Ip, i dodajemy reguły rutowania, np.:

```
Switch(config)#vlan 2
Switch(config)#vlan 3
Switch(vlan-2)#ip address 192.168.123.160/24
Switch(vlan-3)#ip address 192.168.124.160/24
Switch(config)#ip route 10.0.4.0 255.255.255.255 10.0.1.1
Switch(config)#no ip route 10.0.4.0 255.255.255.255 10.0.1.1
Switch(config)# ip route 0.0.0.0 0.0.0.0 192.168.123.254 //domyślna reguła
```

13 Lab 12 EDGE

13.1 Teoria

Zajmiemy się konfiguracją przełącznika Cisco Catalyst 5500, i jego modułu RSM. Stworzymy sieć Vlan z module supervisor'a i spróbujemy okiełznać kilka modułów RSM. Konieczne jest adresowanie portów przełącznika z użyciem numeracji modułów – a te są montowane w numerowanych gniazdach chassis.

13.2 Praktyka

13.2.1 Vlan w Supervisor i łączenie RSM

Definiowanie adresacji dla interfejsu sc0 modułu supervisor i domyślna bramka router'a:

```
Cat5500> (enable) set interface sc0 192.168.123.215 255.255.255.0
Cat5500> (enable) set ip route 0.0.0.0 192.168.123.254
```

Obsługa vTP:

```
Cat5500> (enable) show vtp domain
Cat5500> (enable) set vtp domain mode transparent
Cat5500> (enable) set vtp domain cisco
```

Definiowanie Vlan i dodanie portów:

```
Cat5500> (enable) set vlan 2 //clear vlan 6 usuwanie
Cat5500> (enable) set vlan 2 7/1-10
Cat5500> (enable) show vlan //diagnostyka
```

gdzie 6 to numer konfigurowanego VLAN, 7 to numer kart (line card), 1-10 to zakres portów na tej karcie.
Diagnostyka STP:

```
Cat5500> (enable) show spantree active
Cat5500> (enable) show spantree summary
Cat5500> (enable) show spantree 1 //dla konkretnego Vlan'u
```

Łaczymy się z wybranym modułem RSM i zapamiętujemy znak powrotu:

```
Cat5500> (enable) session 5
```

Tam działamy już sobie standardowo jak w normalnym routerz'e.

13.2.2 Funkcjonalność testowa boot router'a

W pamięci wewnętrznej bootflash zainstalowana jest wersja testowa (tzw. boot version) systemu operacyjnego router'a. W pamięci na karcie PCMCIA (slot0:) - wersja pełna. Sprawdzenie zawartości:

```
Router#dir slot0:
Router#dir bootflash:
```

Gdy karta jest włoŜona do gniazda PCMCIA - ruter uruchomi pełną wersję systemu operacyjnego, gdy karty nie ma - ruter uruchomi wersję testową (z ROM).

Zabawa tutaj polega na wyjęciu karty PCMACIA i zresetowaniu całosci. (#reload). Do testów komendy:

```
Cat5500> (enable) ping 200.200.200.1
Cat5500> (enable) trace 200.200.200.1
```