# Rutowanie pakietów

## Rutowanie statyczne a rutowanie dynamiczne.

Zastosowanie rutingu statycznego niesie ze sobą wiele korzyści. Przede wszystkim jest przewidywalne, ponieważ wcześniej administrator sam ustala tablicę rutowania i wie dokładnie, jaką drogę przebędzie pakiet, aby osiągnąć miejsce docelowe. W przypadku zastosowania rutingu dynamicznego nie jest już takie proste określenie trasy pakietu, ponieważ protokół sam podejmuje taką decyzję na podstawie otrzymanych danych o stanie łączy oraz odległości od innych routerów. Jak już napisałem wyżej protokoły rutingu dynamicznego rozsyłają, co pewien czas informacje, na podstawie, których podejmują odpowiednie decyzje którędy przesłać pakiet, aby dotarł najszybciej do miejsca docelowego. Informacje te w pewnych okolicznościach mogą w znacznym stopniu zmonopolizować dostępne pasmo w danej sieci. Zależy to w dużym stopniu od ilości ruterów i zastosowanego protokołu rutowania dynamicznego.

Na przykład w sieci składającej się z 200 segmentów, co 30 sekund, zgodnie ze specyfikacją protokołu RIP, rutery powinny wysyłać informacje aktualizacyjne zawierające opis dostępności wszystkich 200 segmentów sieci. Jak widać rutery w ciągu 30 sekund muszą wysłać przez każdy ze swoich interfejsów informację, która zawiera około 3 Kb danych. Mnożąc to przez 200 wychodzi około 600 Kb informacji wysyłanych przez rutery w ciągu 30 sekund – przy wolnych łączach modemowych może to znacznie obciążyć pasmo w sieci.

Poza tym ruting statyczny jest prosty do skonfigurowania w małych sieciach, gdzie administrator musi jedynie powiadomić rutery o każdym z segmentów, do których dany ruter nie jest bezpośrednio podłączony.

Podstawową wadą rutingu statycznego jest to, że jest on praktycznie nie do zastosowania w dużych skomplikowanych sieciach składających się z dużej ilości ruterów i różnych rodzajów łącz. W takim przypadku niezastąpiony jest protokół rutingu dynamicznego. Głównymi zaletami rutowania dynamicznego w stosunku do rutowania statycznego są skalowalność i zdolność dopasowywania się do zmieniających się połączeń sieci.

Rutery wykorzystujące ten protokół są w stanie „uczyć się" topologii danej sieci po przez wymianę informacji o stanie łącz i podłączonych segmentach do innych ruterów.

W takim przypadku sieć może sama reagować na zachodzące w niej uszkodzenia i na podstawie informacji zawartych w uaktualnieniach rozwiązywać te problemy.

Oczywiście protokół dynamicznego rutingu ma tez swoje wady, największą wadą jest zawiłość działania sieci, aby ruter był w stanie określić najlepszą
i najkrótszą trasę do docelowego segmentu sieci, musi przetworzyć dużo informacji nadchodzących od innych rutenów. Ponadto, aby ruter reagował
na zmieniające się warunki w sieci musi mieć możliwość usuwania starych
i bezużytecznych informacji o trasach ze swojej tablicy rutingu. Sposób, w jaki będzie to robił jeszcze bardziej komplikuje budowę takiego protokółu.

Stopień komplikacji protokółu prowadzi do błędów w jego poprawnej implementacji lub różnić w interpretacji tego protokółu w urządzeniach różnych producentów.

## Klasyfikacja dynamicznych protokołów rutowania.

Protokoły rutingu dynamicznego można klasyfikować na kilka sposobów:

- protokoły zewnętrzne w porównaniu z protokołami wewnętrznymi;
- protokoły typu dystans-wektor w porównaniu z protokółami stanu łącza.

Pierwsza klasyfikacja opiera się na tym, w jakiej części sieci je stosujemy, druga opiera się na rodzaju informacji, jakie protokół wymienia oraz sposobie, w jaki każdy z ruterów podejmuje decyzję o wprowadzeniu do swojej tablicy rutowania otrzymanych informacji.

## Protokoły zewnętrzne a wewnętrzne.

Dynamiczne protokoły rutowania są klasyfikowane jako, Exterior Gateway *Protocol (*EGP) lub *Interior Gateway Protocol* (IGP).

Zewnętrzny protokół odpowiada za wymianę informacji o rutowaniu pomiędzy dwiema niezależnymi sieciami, takimi jak sieci dwóch korporacji. Każda z tych jednostek ma niezależną infrastrukturę sieciową i wykorzystuje EGP
do przesyłania informacji o rutowaniu do innych podobnych jednostek. Najpopularniejszym obecnie zewnętrznym protokołem jest *Border Gateway Protocol* (BGP). Jest on podstawowym protokołem stosowanym pomiędzy sieciami tworzącymi globalna sieć Internet i specjalnie w tym celu został opracowany.

W przeciwieństwie do protokołu opisanego powyżej IGP jest stosowany wewnątrz sieci lub pomiędzy blisko współpracującymi sieciami. Protokół ten został tak stworzony, aby jego prosta budowa jak w najmniejszym stopniu obciążała rutery. Główną wadą tego typu protokołów jest to ze nie są one
w stanie obsługiwać rozrastających się sieci. Najczęściej stosowanymi w sieciach IP protokołami są:

- *Ruting Information Protocol (RIP),*
- *Open Shortest Path First (OSPF),*
- *Enhanced Interior Gateway Routing Protocol*

Pierwsze dwa protokoły są otwartymi standardami, które zostały

użyte lub wymyślone przez społeczność sieci Internet a trzeci jest protokołem firmowym, opracowanym przez firmę CISCO Systems i stosowane w ruterach tej firmy.

## Protokoły dystans – wektor a protokoły stanu łącza.

Innym sposobem klasyfikowania dynamicznych protokołów rutowania jest opieranie się na informacjach jakie przekazują pomiędzy sobą rutery oraz
na sposobie w jaki wykorzystują one informacje znajdujące się w ich tablicach rutowania. Większość protokołów należy do jednej z wymienionych kategorii.
W protokołach dystans-wektor rutery regularnie wysyła do sąsiadów dwie części informacji, które posiada na temat adresów przeznaczenia do których zna drogę.
Pierwsza część informacji mówi sąsiadom rutera jak daleko jest adres przeznaczenia, a druga informuje o tym w jakim kierunku (wektorze) należy kierować pakiety aby dotarły do punktu przeznaczenia.[1] Ruter kolejnego przeskoku wskazuje kierunek , który należy wykorzystać, aby pakiety osiągnęły punkt przeznaczenia, a wymieniona informacja zwykle przyjmuje formę: *„wyślij to do mnie, bo ja wiem , jak to przesłać dalej”.*

Na przykład: uaktualnienia tras RIP zawierają po prostu listę adresów do których rozgłaszający je ruter zna trasę, a także odległość, w jakiej te adresy się znajdują.

Na podstawie odbieranych uaktualnień inny ruter wnioskuje że adresem kolejnego przeskoku prowadzącego do danego miejsca w sieci jest rozgłaszający informacje ruter. Uaktualnienie może jednak przyjąć formę przekazu typu: *„prześlij to do innego rutera, który wie, jak się tam dostać.”*

Ta druga forma uaktualnienia jest zwykle wykorzystywana wtedy, kiedy ruter przez który można dotrzeć do danego miejsca, nie może lub nie będzie mógł rozgłaszać informacji z powodu awarii sieci. Nie wszystkie jednak protokoły rutowania obsługują ten

typ uaktualnienia.

Druga część protokołu, którą jest informacja o odległości, stanowi o różnicy między protokołem dystans-wektor a innymi protokołami. W każdym z przypadków protokół używa pewnej *miary,* aby poinformować odbierające informacje rutery o tym, jak daleko jest adres przeznaczenia. Miara ta może być prawdziwym wskaźnikiem określającym odległość (na przykład okresowe sprawdzanie czasu podróży pakietu do miejsca przeznaczenia), czymś, co w przybliżeniu określa odległość (tak jak liczba przeskoków), lub może to być inna wartość nie związana wcale z odległością. Zamiast tego można na przykład mierzyć koszty danej drogi do miejsca przeznaczenia. Określanie tej wartości może być również wykonywane na podstawie skomplikowanych obliczeń, w których brane są pod uwagę czynniki takie jak obciążenie sieci, pasmo łącza, opóźnienie łącza i inne wartości opisujące ruter. Wartość ta może również zawierać wagę, określaną przez administratora sieci w celu wskazania jednej z tras jako preferowanej w stosunku do innych.

W każdym z przypadków wartość miary kosztu pozwala ruterowi wybrać najlepszą trasę ze wszystkich informacji, jakie do niego docierają w postaci rozgłaszanych informacji o trasach. Wybór dokonywany jest na podstawie porównania odległości podanej w różnych rozgłaszanych trasach. Sposób, w jaki dokonywane jest to porównanie, zależy od tego, jak liczona lub określana jest wartość przekazywanej miary. Na przykład miary w trasach przekazywanych w uaktualnieniach RIP są określone jako liczba przeskoków, gdzie jeden przeskok oznacza obsługę pakietu przez jeden ruter na drodze do miejsca przeznaczenia. Miejsce przeznaczenia z podaną liczbą przeskoków równą 16 uznaje się za nieosiągalne. Kiedy jakiś ruter odbiera uaktualnienia RIP od różnych ruterów, odnoszące się do tej samej sieci, wybiera trasę, która ma najmniejszą miarę. Jeśli miara ta jest mniejsza od tej, którą przechowuje w swojej tablicy rutowania, ruter wymienia trasę do danej sieci na

nową, zakładając, że uzyskane z innego rutera informacje są aktualne.

Aby informacja o trasach do różnych podsieci mogła być propagowana poprzez sieć każdy ruter umieszcza w rozgłaszanych komunikatach wszystkie kierunki, do których jest bezpośrednio dołączony, a także trasy do miejsc przeznaczenia, o których dowiedział się od innych ruterów. Kiedy ruter zaczyna przekazywać dalej informacje o trasach, o których dowiedział się od innych ruterów, to konieczny jest algorytm wchodzący w skład protokołu rutowania, który dokona odpowiedniego zwiększenia miary dla danej trasy. W przypadku protokołu RIP oznacza to, że zanim ruter rozpowszechni informację, którą wcześniej uzyskał z innego rutera, do metryki każdej z tych informacji dodaje jeden przeskok. Dzięki takiemu algorytmowi miara rośnie, gdy zwiększa się odległość od miejsca przeznaczenia wskazywanego przez zapis w tablicy rutowania.

Protokół stanu łącza nie przekazuje informacji od ruterów o miejscach, które można za ich pośrednictwem osiągnąć. Zamiast tego przekazuje informację o topologii sieci. Informacja ta składa się z listy segmentów sieci lub łączy, do których dołączony jest dany ruter, oraz stanu tych łączy (czy funkcjonują, czy też nie). Informacje takie są następnie przepuszczane przez sieć. Przepuszczając te informacje coraz dalej w sieci każdy ruter jest w stanie zbudować sobie własny obraz sieci i bieżącego stanu wszystkich tworzących ją łączy. Ponieważ każdy ruter w sieci widzi te same informacje, wszystkie stworzone w opisany wyżej sposób obrazy sieci powinny być identyczne. Na podstawie takiego obrazu sieci każdy ruter wylicza najlepszą dla siebie trasę do poszczególnych miejsc w sieci i na tej podstawie tworzy tablicę rutowania. To, w jaki sposób ruter określa, która trasa jest najlepsza, zależy od algorytmu zastosowanego wdanym protokole. W najprostszych rozwiązaniach ruter może po prostu policzyć ścieżkę wykorzystując najmniejszą liczbę przeskoków. W bardziej złożonych protokołach informacje o stanie łącza

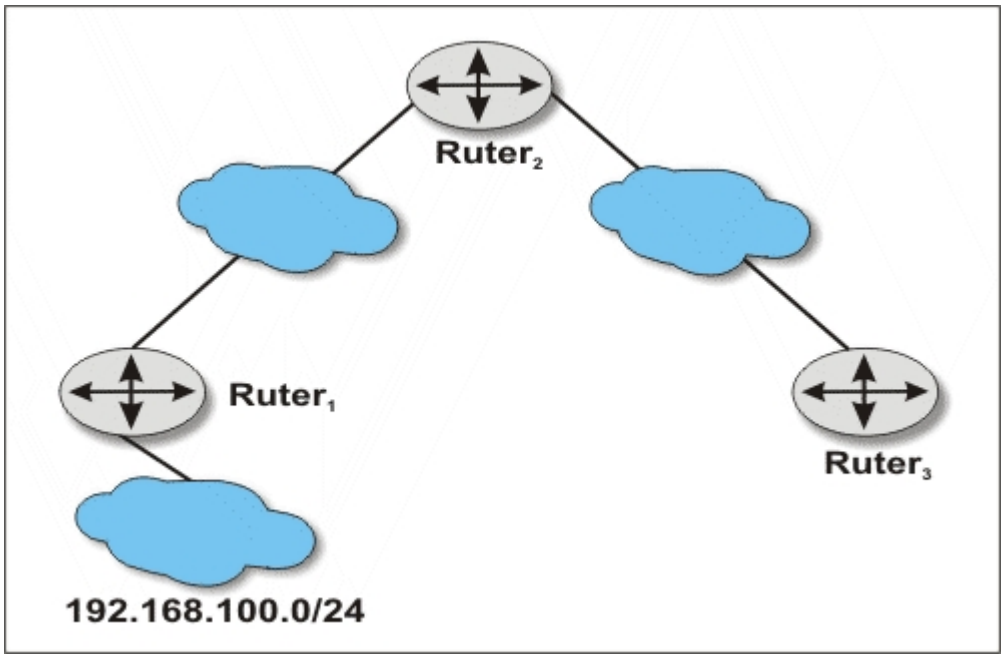
mogą zawierać dodatkowe dane, które pomogą ruterowi określić najlepszą ścieżkę. Informacje takie mogą zawierać dane na temat pasma łącza, bieżącego obciążenia tego łącza, współczynników administracyjnych, a nawet ograniczenia przesyłania niektórych pakietów przez pewne łącza. Na przykład jakieś łącze w sieci może nie być wykorzystywane do przesyłania informacji tajnych.

Protokoły dystans-wektor oraz stanu łącza mają swoje dobre i złe strony. W poprawnie funkcjonującej i skonfigurowanej sieci każdy z tych protokołów poprawnie określi najlepszą trasę pomiędzy dwoma punktami.

## Wady protokołów dystans-wektor.[2]

Ogólnie rzecz biorąc, protokoły typu dystans-wektor są łatwiejsze w konfigurowaniu niż protokoły stanu łącza. Łatwiej też zrozumieć ich działanie. W mniejszym stopniu obciążają one również procesor, co pozwala ruterowi zająć się innymi zadaniami, takimi jak przełączanie pakietów. Główne wady tych protokołów wynikają często z ich prostej budowy. Jedną z największych wad jest to, że rutery nie przekazują informacji o tym, skąd dowiedziały się o danej trasie, którą umieściły w komunikacie zawierającym uaktualnienia. Rozważmy np. prostą sieć zbudowaną z użyciem trzech ruterów, pokazaną na rysunku 1.5.1. $Ruter_1$ informuje $Ruter_2$ o sieci 192 .168 .100 .0/24. $Ruter_2$ będzie oczywiście informował $Router_2$ o tej sieci, ale taką samą informację może przekazać również do $Rutera_1$ $Router_2$ także może poinformować $Ruter_2$ o tym, że wie, jak dostać się do tej samej sieci, nawet jeśli trasa będzie prowadziła przez $Ruter_2$.

**Rysunek 1.5.1:** Prosta sieć złożona z trzech ruterów




Zwykle sytuacja taka nie jest problemem, ponieważ każdy ruter będzie porównywał miary tras, o jakich dowiaduje się z sieci, z miarami tras, które ma zapisane w tablicy rutowania, i na tej podstawie będzie wybierał najkorzystniejszą trasę. Co się jednak stanie, kiedy $Ruter_1$ straci połączenie z siecią 192.168.100.0/24 z powodu uszkodzenia sprzętu? Przestanie ona informować $Ruter_2$ o swoim istnieniu i w końcu zapis trasy do tej podsieci zostanie usunięty z tablicy rutowania tego rutera (trasa zostanie usunięta w wyniku upłynięcia określonego czasu lub na podstawie komunikatu przekazanego przez $Ruter_1$). Kiedy to nastąpi, $Ruter_2$ może usłyszeć od $Routera_2$ o istnieniu takiej sieci i doda tę „nową” podsieć do swojej tablicy rutowania, przekazując o tym informację $Ruterowi_1$. Oczywiście informacja wysłana zostanie również do $Routera_2$, który odkryje, że trasa prowadząca przez $Ruter_2$ jest gorsza od zapisanej poprzednio. Nie zważając na to, ruter uaktualni swoją tablicę rutowania i miarę, z którą będzie teraz rozgłaszał te informacje, wysyłając je do $Routera_2$. Odebranie tego kolejnego uaktualnienia przez $Ruter_2$ spowoduje, że ogłosi on tę trasę (z trochę gorszą miarą) $Ruterowi_2$, który następnie zwróci informację do $Routera_2$ z jeszcze większą miarą. W końcu rutery osiągną wartość miary, która jest zdefiniowana w danym protokole jako

„nieskończoność”. Kiedy to się stanie, wszystkie rutery usuną tę trasę ze swoich tablic rutowania.

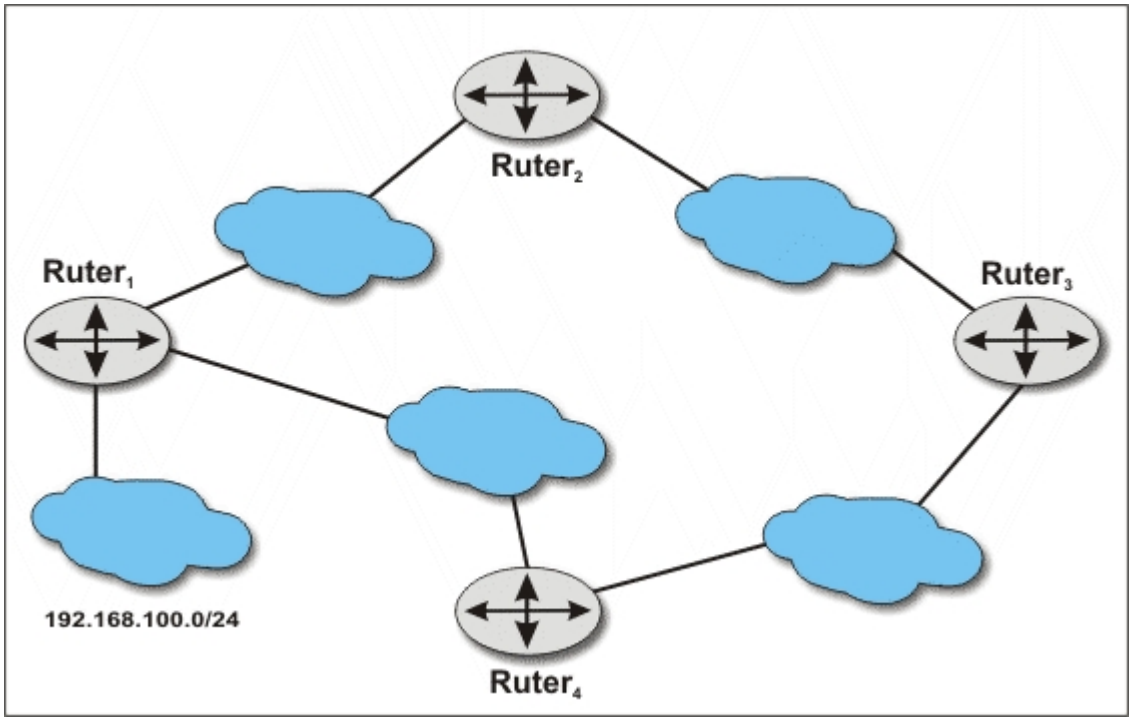
W zależności od tego, jak duża jest wartość „nieskończoności” określona w danym protokole, oraz od tego, jak często rutery wysyłają sobie uaktualnienia nawzajem, okres niestabilności i błędnego rutowania pakietów może trwać od kilku sekund do kilku minut. Niewątpliwie nie chcesz, aby tablice rutowania Twoich ruterów były niestabilne przez całe minuty za każdym razem, kiedy uszkodzeniu ulegnie jakaś część sieci! Większość protokołów typu dystans-wektor ma dodatkowe funkcje, które obsługują takie przypadki i zapobiegają przedłużaniu się czasu niestabilności. Pierwszą rzeczą, którą się zwykle dodaje, jest coś, co nazywane jest *domknięciem horyzontu.* W procedurze tej w momencie, kiedy ruter tworzy uaktualnienie dotyczące konkretnego interfejsu, pomija w nim wszystkie odniesienia do tras, których nauczył się od ruterów dostępnych przez ten interfejs.

W naszym przypadku oznacza to, że $\text{Ruter}_2$ poinformuje $\text{Router}_2$ o podsieci 192.168.100.0/24, której nauczył się z $\text{Rutera}_1$, ale pominie wszelkie odwołania do tej sieci, kiedy będzie wysyłał uaktualnienie do $\text{Rutera}_ι$. $\text{Router}_2$ także powstrzyma się od informowania $\text{Rutera}_2$ o tej sieci, ponieważ to właśnie od tego rutera uzyskał o niej informacje. Niewielką modyfikacją metody „domkniętego horyzontu” jest metoda *„poison reverse„*. W metodzie tej zamiast pomijania informacji o sieciach, których ruter nauczył się z danego interfejsu, ruter dołącza te informacje do rozsyłanego uaktualnienia, ale dodaje do nich znacznik informujący, że taka sieć jest nieosiągalna. Taka informacja powoduje, że odbierający ją ruter posługujący się niewłaściwą trasą może ją usunąć ze swojej tablicy nitowania

Wynikiem działania w sieci opisanych wyżej metod jest fakt, że prosta niestabilność sieci opisana wcześniej nie może się w tej sieci zdarzyć. Niestety, ani jedna, ani druga metoda nie rozwiązują wszystkich problemów. Jeśli w sieci jest przejście

łączące $Ruter_1$ z $Ruterem_2$, być może przez $Ruter_4$, jak pokazano na rysunku 1.5.2, możliwe jest wystąpienie pętli rutowania, nawet jeśli uruchomione są algorytmy opisanych wyżej metod. W takiej sieci $Ruter_1$ informuje $Ruter_2$ i $Ruter_4$ o swoim połączeniu z siecią 192 .168.100.0/24, podając w obu przypadkach prawdopodobnie taką samą wartość miary. Rutery te z kolei poinformują $Router_2$ o trasie prowadzącej do tej sieci, stosując prawdopodobnie tę samą miarę. $Router_2$ wybierze jedną z tych tras (prawdopodobnie tę, którą odbierze jako pierwszą) i umieści ją w swojej tablicy rutowania.

**Rysunek 1.5.2:** Standardowe metody nie zapobiegają występowaniu pętli rutowania w sieciach, w których połączenia tworzą pierścień




Załóżmy, że $Router_2$ wybierze trasę prowadzącą przez $Ruter_2$. Ponieważ działa metoda *„poison reverse„*, zgodnie z logiką działania tej metody wyśle on informację o tej trasie do $Rutera_2$ z miarą informującą o tym, że adres jest nieosiągalny. Ponieważ jednak zdecydował, że nie używa trasy prowadzącej przez $Ruter_4$, nie zastosuje wymienionej wyżej metody dla tego łącza, ale zamiast tego dołączy trasę do sieci

192.168.100.0/24 przez $Ruter_2$ do uaktualnienia wysyłanego do $Rutera_4$, który z kolei zignoruje to uaktualnienie i wybierze trasę prowadzącą przez $Ruter_1$.

Wszystko będzie działało dobrze do czasu, kiedy łącze pomiędzy $Ruterem_1$ i siecią 192.168.100.0/24 nie ulegnie uszkodzeniu. Wtedy $Ruter_1$ przestanie rozgłaszać tę trasę do $Rutera_2$ i $Rutera_4$. Rutery te z kolei przestaną rozgłaszać trasę do $Routera_2$, ale możliwe jest, że $Ruter_4$ usłyszy komunikat rozgłoszeniowy od $Routera_2$, zanim opisany wyżej proces dobiegnie końca. Ponieważ ruter ten nie ma informacji o trasie w swojej tablicy rutowania, umieści ją tam i poinformuje $Ruter_1$, że ma nową trasę. Następnie, zgodnie z działaniem algorytmu, $Ruter_1$ poinformuje o tej trasie $Ruter_2$, który prześle informację dalej, do $Routera_2$.

Pętla ta zostanie w końcu przerwana, kiedy każdy z ruterów, zwiększając miarę przy każdym przesyłaniu informacji o trasie w pętli, zwiększy ją do pewnej granicznej wartości dla danego protokołu, którą określaliśmy wcześniej jako wartość „nieskończoności”. Ile czasu zajmie ruterom tak zwane „odliczanie do nieskończoności”, zależy w dużym stopniu od tego, jak często wymieniają między sobą uaktualnienia, jaka jest wartość graniczna dla używanego w sieci protokołu i ile ruterów uczestniczy w tej pętli. Rozwiązaniem opisanego problemu jest wprowadzenie czasu blokowania. Kiedy ruter dowie się, że jakiś adres nie jest już dostępny dla ścieżki, której używał wcześniej, rozpoczyna odliczanie czasu, w trakcie którego ignoruje wszelkie inne informacje o lutowaniu dotyczące tego adresu. Czas ten wprowadzony jest po to, aby inne rutery mogły dowiedzieć się o wystąpieniu uszkodzenia, zanim ruter odliczający czas zacznie wykorzystywać ich trasy prowadzące do tego adresu docelowego. W naszym przypadku kiedy $Ruter_1$ stwierdza, że nie może dostać się do sieci 192.168.100.0/24, rozpoczyna odliczanie czasu blokowania tego zapisu w tablicy rutowania. W czasie odliczania ignoruje

wszelkie uaktualnienia nadsyłane przez Router2. Jeśli czas blokowania jest wystarczająco długi, to zanim Ruter$_1$ zacznie znowu słuchać, Router2 stwierdzi, że jego trasa nie jest już popraw- na i nie będzie jej więcej rozgłaszał.

Wadą czasu blokowania jest to, że trudno jest określić, ile powinien on wynosić. Ile czasu zajmie rozpropagowanie informacji, że trasa nie jest już poprawna, do wszystkich ruterów, od których dany ruter otrzymuje uaktualnienia? Czasy te są szczególnie długie w przypadku protokołu takiego jak RIP. W swojej prostszej wersji RIP rozsyła uaktualnienia tablicy rutowania co 30 sekund. Ponieważ uaktualnienia te nie są potwierdzane przez odbiorców, możliwe, że niektóre z nich są gubione w sieci. Ponadto kiedy w uaktualnieniu znajduje się informacja o tym, jakie adresy są osiągalne, to nie zawsze wiadomo, które już osiągalne nie są. Nie ma więc żadnej wskazówki dla rutera, że powinien usunąć ze swojej tablicy rutowania trasę, która nie jest już dłużej poprawna.

Aby umożliwić wykrywanie zagubionych w sieci uaktualnień, RIP ustawia zegar dla każdej trasy, której się nauczył. Za każdym razem, kiedy RIP słyszy uaktualnienie dotyczące tej trasy, zegar jest zerowany. Jeśli ruter nie odbierze uaktualnienia w ciągu 180 sekund, usuwa trasę ze swojej tablicy rutowania i przestaje rozgłaszać ją swoim sąsiadom. W rezultacie jeśli jakieś uaktualnienie zostanie zagubione, rutery nie będą natychmiast usuwały tras ze swoich tablic rutowania. Prawdopodobnie trasy te znajdą się w kolejnym uaktualnieniu i ich zegary zostaną wyzerowane.

W praktyce procedura opisana wyżej oznacza, że rozgłoszenie zmiany w topologii sieci i zapisanie jej w tablicach rutowania wszystkich ruterów, które w tej sieci pracują, może zająć sporo czasu. Zastanów się jeszcze raz nad działaniem sieci z trzema ruterami, pokazanej na rysunku 1.5.1. Kiedy Ruter$_1$ stwierdzi, że utracił połączenie z siecią 192.168.100.0/24, to po prostu przestał rozgłaszać tę sieć w swoich uaktualnieniach

wysyłanych do Rutera2. Mimo to przez kolejne 3 minuty od ostatniego komunikatu $Ruter_2$ nadal wierzył, że ma trasę prowadzącą do tej sieci i wysyłał informację o tym w uaktualnieniach kierowanych do $Routera_2$. Po trzech minutach $Ruter_2$ stwierdza, że $Ruter_1$ musiał utracić tę trasę i usuwa zapis trasy do sieci 192.168.100.0/24 ze swojej tablicy rutowania, informując o tym $Router_2$. Mimo to $Ruter_3$ nadal będzie wykorzystywał starą, nieaktualną już informację przez kolejne trzy minuty

Rozważmy teraz, co się będzie działo, jeśli taka procedura odliczania czasów na kolejnych ruterach wykonywana będzie w sieci składającej się z kilkunastu ruterów. Jeśli każdy z ruterów musi odczekać trzy minuty od czasu, kiedy najbliższy mu ruter przestał rozgłaszać daną trasę, to oczywiste staje się, że trasa może nie zniknąć całkowicie z sieci przez około 45 minut! Nierozsądne jest więc określanie tak długiego czasu blokowania rekordów w tablicy rutowania. Czas ten powinien stanowić niewielką część tych trzech minut. Aby redukować czas, kiedy w sieci występuje stan niespójności informacji o nitowaniu, protokół dystans-wektor umożliwia ruterom rozsyłanie informacji o *osiągalności negatywnej* dla tras, które zostały przez te rutery rozgłoszone, ale nie są już dłużej osiągalne. Informacje takie pozwalają ruterom na szybkie stwierdzenie faktu, że jakaś trasa nie jest dłużej dostępna. Dla protokołu RIP informacja o negatywnej osiągalności jest po prostu informacją o trasie z miarą ustawioną na wartość 16. Inne protokoły oznaczają taką informację we właściwy sobie sposób

Rozgłaszanie negatywne pomaga przyspieszyć przekazywanie informacji o uszkodzeniach tras, ale nie eliminuje opóźnień. Kiedy $Ruter_1$ odkryje, że jego połączenie z podsiecią 192.168.100.0/24 zostało przerwane (lub odtworzone), przekaże tę informację do $Routera_2$ w kolejnym uaktualnieniu. W przypadku stosowania protokołu RIP jest to realizowane poprzez wysłanie

uaktualnienia i może upłynąć do 30 sekund, zanim zostanie ono wygenerowane.

Ponadto jeśli $Ruter_2$ dostanie wiadomość od $Rutera_1$, to może również odczekać do 30 sekund, zanim powiadomi o zmianie $Router_2$, który z kolei odczeka do 30 sekund itd. Nawet jeśli informacja o zmianie stanu łącza przesłana zostanie przez sieć dość szybko, zwłaszcza w porównaniu z czasem, jaki jest potrzebny do wygaśnięcia zapisu w tablicy rutowania, to nadal może to zająć kilka minut, zanim wszystkie rutery w sieci dowiedzą się o tej zmianie i odpowiednio uaktualnią swoje tablice rutowania. Opóźnienie pomiędzy czasem wystąpienia zmiany stanu łącza w sieci a chwilą, kiedy wszystkie rutery w tej sieci dopasują swoje tablice rutowania, określane jest mianem *czasu zbieżności.* Długi czas zbieżności jest niewątpliwie problemem dla każdego protokołu rutowania

Aby zminimalizować czas konwergencji, protokół dystans-wektor może uruchomić wysyłanie uaktualnień *typu flash* lub *triggered.* Uaktualnienie *triggered* wysyłane jest za każdym razem, kiedy tablica nitowania danego rutera zmieni się w sposób, który może wpływać na rozsyłanie uaktualnień innych tras tego rutera. Jeśli każdy ruter używa uaktualnień tego typu i umieszcza w nich informacje o negatywnej osiągalności, to możliwe jest, że informacja o uszkodzeniu połączenia z $Rutera_1$ do sieci 192.168.100.0/24 zostanie przekazana do wszystkich ruterów pracujących w sieci w ciągu kilku sekund. Dzięki temu znacznie zmniejszy się czas zbieżności oraz czas, jaki ruter odczekuje przed usunięciem zapisu z tablicy rutowania.

Ten mechanizm nie jest prosty. Jeśli dodatkowe uaktualnienia nie będą dokładnie kontrolowane, to chwilowe uszkodzenie może powodować rozsyłanie w sieci tam i z powrotem różnych uaktualnień, co będzie zajmowało pasmo i moc obliczeniową procesorów w ruterach, które będą się zajmowały przetwarzaniem uaktualnień, a nie przełączaniem pakietów. Powszechnie stosowanym rozwiązaniem jest nieznaczne wydłużenie czasu

odczekiwania przed usunięciem zapisu z tablicy rutowania oraz dodanie krótkiego czasu oczekiwania, który ustawiany jest po każdym uaktualnieniu typu *flash.* W czasie tego oczekiwania ruter nie przyjmuje żadnych innych uaktualnień, co pomaga złagodzić efekty faktycznych uszkodzeń

Kolejną dużą wadą protokołu typu dystans-wektor jest wada wynikająca z faktu, że nie jest to protokół zbyt skomplikowany. Ponieważ topologia sieci może ulec zmianie, w wyniku uszkodzenia łącza lub dodania albo usunięcia segmentu sieci, wszystkie dynamiczne protokoły rutowania muszą przekazywać do ruterów informacje o tych zmianach. W protokole dystans-wektor uaktualnienia wykonywane są zwykle po- przez okresowe rozsyłanie pakietów typu *broadcast (lub multicast*) poprzez niektóre lub wszystkie interfejsy rutera. Często uaktualnienia te zawierają pełną informację o nitowaniu, którą posiada ruter wysyłający to uaktualnienie. Okresowe uaktualnienia są przydatne, gdyż pozwalają ruterom pracującym w danym segmencie sieci informować się wzajemnie. Niestety, komunikaty te generują dodatkowy ruch w sieci nawet wtedy, kiedy sieć pracuje stabilnie (co, mamy nadzieję, stanowi większość czasu pracy sieci). Niektóre nowsze protokoły dystans-wektor, takie jak *Cisco EIGRP*, rozgłaszają tylko zmiany zachodzące w tablicach rutowania, ale protokół ten nadal jest rzadko stosowany.

Podczas gdy protokół dystans-wektor jest raczej nieskomplikowany oraz łatwy w obsłudze dla procesora rutera, prostota ta może prowadzić do nietypowych zachowań w wyniku uszkodzeń sieci i długich czasów zbieżności sieci. W sieci obsługiwanej przez ten protokół czas pomiędzy wystąpieniem uszkodzenia jednego z komponentów sieci a ustaleniem trasy obejściowej obsługiwanej przez poprawnie pracujące rutery może być dość długi. Działanie tego protokołu może również prowadzić do dużego wykorzystania pasma sieci i znacznego obciążenia procesora rutera nawet wtedy, gdy sieć pracuje stabilnie. Choć zmiany dokonywane w samym protokole mogą

zmniejszyć te problemy, to po dodaniu dodatkowych funkcji rozgłaszania, obsługi czasów oczekiwania itd. protokół przestanie być zrozumiały i nieskomplikowany i znacznie trudniej będzie śledzić jego działanie.

## Wady protokołów stanu łącza

Protokoły stanu łącza mają kilka ważnych zalet. Ponieważ obliczają trasy nitowania na podstawie znajomości topologii sieci, o której dowiadują się z uaktualnień informujących go o stanie łączy, nie mogą tworzyć pętli w wyniku częściowego uszkodzenia sieci, jak to zdarzało się w przypadku protokołów typu dystans-wektor. Ponieważ zmiany stanu łączy przekazywane są przez sieć natychmiast po ich wystąpieniu i docierają do wszystkich ruterów, które następnie uaktualniają swoje mapy topologii i tablice nitowania, to czas zbieżności sieci obsługiwanej przez taki protokół jest minimalny. Ostatnią zaletą, o której należy wspomnieć, jest fakt, że większość protokołów stanu łącza jest opracowana tak, by wysyłała uaktualnienia stanów łącza tylko wtedy, kiedy stan ten się zmieni, co sprawia, że protokoły tego typu oszczędzają pasmo i moc procesorów w czasie, kiedy sieć jest stabilna.

Choć protokoły stanu łącza zapobiegają powstawaniu pętli, skracają czasy zbieżności sieci i stopień wykorzystania zasobów sieci, mają też wady. Główną wadą jest ich złożoność. Złożoność jest głównym aspektem implementacji protokołu, ale często daje o sobie znać również podczas konfigurowania sprzętu. Tak naprawdę protokół OSPF, uważany za protokół wewnętrzny, jest znacznie bardziej skomplikowany od BGP, który stosowany jest jako protokół zewnętrzny. Na szczęście w typowej konfiguracji większość skomplikowanych funkcji ukryta jest przed użytkownikiem.

Dlaczego protokół stanu łącza jest tak złożony? Rozważmy jeszcze raz to, co mówiliśmy o sposobie, w jaki rutery określają swoje trasy. Zbierają one wszystkie uaktualnienia stanów łącza nadsyłane przez inne rutery i na ich podstawie

budują mapę topologii sieci. Wykorzystując tę mapę rutery obliczają następnie najlepsze trasy do różnych miejsc w sieci. Pierwszym problemem jest generowanie mapy topologii. Choć człowiek może dość szybko narysować mapę połączeń sieci, bazując na informacjach o tym, co jest z czym połączone, to komputer musi mieć jakiś sposób zapisu tego ludzkiego rysunku w elektronicznej formie pozwalającej na dalsze przetwarzanie tych informacji. Standardowym sposobem zapisu tych informacji jest wykorzystanie jednego z wielu rodzajów grafów sieci.

Każdy rodzaj grafów ma pewien zestaw działań, które dobrze obsługuje, i zestaw funkcji, których nie obsługuje prawie wcale. Przeprowadzono wiele badań w celu opisania różnych typów grafów i funkcji, które te grafy obsługują. Bardzo często specyfikacja protokołu nie określa sposobu, w jaki ma być on implementowany. Możliwe, że w specyfikacji nie określa się nawet rodzajów danych, jakie będą konieczne do poprawnej pracy tego protokołu. Nawet jeśli rodzaje danych określone są w specyfikacji, to sposób w jaki dane te są reprezentowane (tzn. jaki rodzaj grafu zostanie użyty) pozostawia się temu, kto implementuje protokół. Zły wybór grafu może doprowadzić do trudno rozpoznawalnych uszkodzeń i błędów w kodzie oprogramowania rutera.

Drugą trudnością związaną z implementacją protokołu stanu łącza jest sposób liczenia najlepszej trasy do wszystkich miejsc w sieci. Choć istnieją algorytmy obliczające najlepszą ścieżkę za pomocą różnych typów grafów i miar, to nadal jest to kwestia odpowiedniej implementacji. Popełnione w procesie implementacji błędy dają ciekawe rezultaty w czasie działania produktu końcowego, jakim jest protokół rutowania w sieci.

Złożoność implementacji nie powinna być jednak przedmiotem zainteresowania administratora sieci, jeśli kod wynikowy, jaki otrzymał wraz z ruterami, działa poprawnie. Nawet jeśli kod jest poprawny, to skomplikowana implementacja wymaga zwykle większej mocy procesora i większej pamięci w ruterze. Na przykład wygenerowanie grafu topologii będzie zajmowało trochę

czasu, a graf ten należy przecież jeszcze gdzieś zapisać. Musi on być przechowywany w dość bezpiecznym miejscu, ponieważ uaktualnienia stanu łączy zawierają tylko informacje o zmianach, jakie nastąpiły w topologii sieci. Dodatkowe wymagania dotyczące pamięci i mocy procesora sprawiają, że niektórzy administratorzy sieci trzymają się z dala od protokołów stanu łącza, ale nie jest to jedyny powód takiego postępowania. Ważniejszym powodem jest złożoność tych protokołów lub założenie, że są one skomplikowane i trudno je konfigurować.

Większość protokołów stanu łącza jest znacznie trudniejsza w konfiguracji niż protokoły typu dystans-wektor. Jeśli jednak interfejs konfiguracyjny jest dobrze zaimplementowany i jeśli zawiera zestaw właściwie określonych parametrów domyślnych, to możliwe jest skonfigurowanie protokołu stanu łącza przy niewiele większym nakładzie pracy niż dla protokołu dystans-wektor.

Zarówno protokół stanu łącza, jak protokół dystans-wektor będą działały poprawnie, jeśli rutowanie w stabilnej sieci będzie bezbłędne. Powinny one ponadto zmienić rutowanie na inne w sytuacji, kiedy w sieci wystąpi jakieś uszkodzenie.

---

[1] Matematyczna definicja wektora określa, że musi on mieć kierunek i długość. Niestety kiedy sieciowcy posługują się określeniem wektora w przypadku protokołów dystans-wektor, to myślą wyłącznie o jego kierunku.

[2] Scott M. Ballew *„Zarządzanie sieciami IP za pomocą ruterów CISCO”,* O’Reilly 1998r.