PN/N/17

Słociński Michał

Pokrywka Paweł

Tolerowanie uszkodzeń w sieciach komputerowych

Systemy tolerujące uszkodzenia – projekt
INF 2003/2004

Prowadzący:

Dr hab. inż. Stanisław J. Piestrak prof. PWr

Spis treści

1. Wstęp

W dzisiejszym świecie największą wartość ma informacja. Dlatego też, coraz większego znaczenia nabiera przesyłanie danych na duże odległości. Znanych jest coraz więcej przypadków wymagających wysokiej niezawodności oraz pewności, że przesyłana informacja dotarła do adresata w niezmienionej formie. W sieciach komputerowych stosowane są dwa, komplementarne podejścia do tego problemu: unikanie uszkodzeń, oraz ich tolerowanie. Pierwsza metoda polega na zwiększaniu niezawodności urządzeń sieciowych i mediów transmisyjnych. Drugi sposób to takie zaprojektowanie architektury sieci, aby występujące uszkodzenia nie wpływały na funkcjonowanie usług dostarczanych użytkownikom.

Na przestrzeni lat zmieniało się zarówno podejście do projektowania sieci pod kątem metod transmisji, jak i podejście do występujących w nich uszkodzeń. W pierwszym przypadku obserwujemy trend polegający na stopniowym wypieraniu sieci komutowanych przez sieci pakietowe. Główną zaletą sieci pakietowych jest cena, znacznie niższa niż sieci komutowanych. Taką samą tendencję możemy zaobserwować w zapewnianiu niezawodności sieci. Tańsze okazało się tolerowanie uszkodzeń, niż zapobieganie im. Jak wiadomo niemożliwe jest uzyskanie stuprocentowej niezawodności w każdej z dziedzin nauki i techniki, także tolerowanie uszkodzeń staje się koniecznością przy dostarczaniu usług wysokiej jakości.

2. Problemy pojawiające się w sieciach komputerowych

Aby przeanalizować metody tolerowania uszkodzeń w sieciach komputerowych należy przyjrzeć się bliżej typom awarii, które występują w tych sieciach.

Występują dwie zasadnicze klasy uszkodzeń:

1. Sprzętu aktywnego oraz medium transmisyjnego.
2. Przeciążenia sieci.

Do pierwszej klasy zaliczają się takie problemy jak:

- Niestabilne napięcie sieci energetycznej,
- Niedoskonałości oprogramowania urządzeń aktywnych (koncentratory, przełączniki, routery),
- Promieniowanie elektromagnetyczne (dot. w szczególności sieci bezprzewodowych, a także sieci światłowodowych),
- Uszkodzenia powodowane przez czynniki zewnętrzne (burze, pożary, powodzie, gryzonie),
- Wandalizm.

Przeciążenia w sieciach komputerowych dotyczą:

- Sytuacji, w której do sieci szkieletowej podłączonych jest zbyt wiele stacji, w stosunku do jej przepustowości,
- Pętli w sieci (dot. sieci ethernet zbudowanych w oparciu o tanie urządzenia),
- Robaków sieciowych (dot. bardziej Internetu niż sieci lokalnych, jednak zjawisko to może także występować w sieciach LAN),
- Przeciążenie aplikacji serwerów.

3. Metody i algorytmy tolerowania uszkodzeń.

3.1. STP (Spanning Tree Protocol)

Spanning Tree Protocol [8] opracowany został dla zastosowań w drugiej warstwie modelu sieciowego ISO/OSI, co oznacza że jest niezależny od protokołów wyższych warstw takich jak IP (ang. *Internet Protocol*) oraz TCP (ang. *Transmission Control Protocol*). Ideą, dla której został opracowany STP, było zwiększenie niezawodności środowisk sieciowych składających się z kilku segmentów, połączonych ze sobą przy pomocy redundantnych mostków sieciowych (ang. *network bridge*) oraz przełączników (ang. *network switch*).

W tego typu sieciach, bardzo łatwo może dojść do wystąpienia niepożądanego zjawiska, jakim są tzw. „pętle". Zjawisko to można łatwo opisać, definiując środowisko złożone z dwóch segmentów sieciowych **SEG1** i **SEG2**, połączonych ze sobą przy pomocy dwóch redundantnych, równolegle umieszczonych przełączników **SW1** oraz **SW2** w celu zapewnienia niezawodnej łączności pomiędzy segmentami.
Sytuacja, w której powstaje pętla ma miejsce, gdy np. z segmentu **SEG1** wysłany zostanie pakiet na adres rozgłoszeniowy. Odebrany zostanie on zarówno przez przełącznik **SW1** jak i **SW2** poprzez porty podłączone do segmentu **SEG1**. Oprogramowanie przełączników przekopiuje pakiet do bufora nadawczego portów podłączonych do segmentu **SEG2** oraz przeznaczy je do wysłania. Gdy pakiety zostaną wysłane, trafią do swojego miejsca przeznaczenia czyli segmentu **SEG2**, ale także pojawią się po raz kolejny na portach przełączników **SW1** i **SW2**, tym razem od strony segmentu **SEG2**.

W tym momencie dochodzi do zjawiska lawinowego powielania pakietów adresowanych do całej sieci, co prowadzi do uniemożliwienia prawidłowej komunikacji, a nawet do unieruchomienia całej sieci.

STP został opracowany przez IEEE (**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers) oraz opisany został w dokumencie 802.1D [8]. Idea działania algorytmu STP, polega na utworzeniu drzewa opisującego topologię sieci, w którym pomiędzy każdymi dwoma elementami istnieje tylko jedna ścieżka, po której może odbywać się komunikacja. Na szczycie drzewa znajduje się główny przełącznik – korzeń drzewa (ang. *root*), zarządzający topologią sieci. Określone zostają także inne ścieżki początkowo nieaktywne, mogących jednak służyć jako nadmiarowe na wypadek awarii. Jeśli ścieżka staje się niedostępna (np. w wyniku uszkodzenia), następuje próba rekonfiguracji topologii sieci oraz inna ścieżka staje się aktywną.

Ponieważ mostki i przełączniki, działają w drugiej warstwie sieci, identyfikują elementy na podstawie ich adresów MAC (ang. *Media Access Control*). Z każdym interfejsem sieciowym (czy też portem w przypadku przełącznika) powiązana jest tabela adresów MAC osiągalnych poprzez ten interfejs/port. W środowisku sieci, w której mostki sieciowe wykorzystują algorytm STP, komunikacja pomiędzy nimi odbywa się poprzez protokół BPDU (ang. *Bridge Protocol Data Unit*).

Informacje przesyłane protokołem BPDU dotyczą:

- Określenia pojedynczego przełącznika głównego, stanowiącego korzeń drzewa
- Określenia przełącznika znajdującego się najbliżej przełącznika *root*, który będzie przekazywał pakiety bezpośrednio do niego
- Określenia przełączników obsługujących konkretne segmenty sieci
- Określenia portów tych przełączników tworzących razem drzewo topologii sieci
- Rekonfiguracji sieci
- Usunięcia pętli w sieci poprzez umieszczenie nadmiarowych portów przełączników w stanie zapasowym „backup"

Algorytm STP na podstawie danych nadsyłanych w pakietach BPDU takich jak: adresy przełączników oraz ich portów, priorytety, koszty dotarcia do portów przełącznika, określa który z przełączników sieciowych ma pełnić rolę głównego przełącznika, będącego korzeniem tworzonego drzewa.

Topologia sieci tworzona jest na podstawie:

- Unikalnych adresów skojarzonych z każdym przełącznikiem oraz jego portami
- Kosztów dotarcia z portów pozostałych przełączników do przełącznika *root*

Po inicjalizacji sieci (włączeniu przełączników w ich domyślnej konfiguracji), głównym zostaje automatycznie ten, który posiada najniższy adres MAC. Zwykle konfiguracja taka nie jest optymalna i może ona zostać zmieniona po ustaleniu priorytetów portów przełączników na podstawie przepustowości danego łącza. Po korekcji tych wartości nastąpi automatyczna rekonfiguracja topologii sieci tak, aby ścieżki w sieci zostały wybrane optymalnie.

Każdy przełącznik posiada swoją bazę adresów MAC dostępnych przez jego porty. Każdy wpis posiada czas starzenia się. W domyślnych konfiguracjach, zaleca się aby wartość ta ustalona została na 300 sekund. Jeśli przez ten czas z każdego adresu MAC nie zostanie przesłana żadna ramka, wpis w pamięci wygasa a przełącznik przystępuje do badania sieci. Często zdarza się jednak że wartość ta jest zbyt duża, szczególnie w przypadkach gdy awarii ulegnie łącze będące aktywną ścieżką pomiędzy dwoma przełącznikami. W takim przypadku, aktywowana zostaje zapasowa ścieżka, a wszystkie przełączniki zostają poinformowane o konieczności skrócenia czasu postarzania wpisów w swojej pamięci podręcznej, zalecaną wartością jest w tym przypadku 15 sekund.

Przełącznik, który wykryje konieczność zmiany topologii sieci, wysyła specjalny pakiet BPDU (zwany TCN) w kierunku głównego przełącznika. Przełącznik który go odbierze, potwierdza nadającemu odebranie jego wiadomości (pakiet TCA) oraz sam kontynuuje przesyłanie pakietu TCN w kierunku głównego przełącznika. Gdy punkt docelowy zostanie osiągnięty, przełącznik *root* rozpoczyna rekonfigurację sieci, wysyłając pakiety TC na adres rozgłoszeniowy do wszystkich pozostałych przełączników.

Algorytm STP zaimplementowany został w wielu urządzeniach sieciowych firm takich jak Cisco oraz w systemie Linux począwszy od wersji jądra 2.3.47 [9].

3.2. MPLS (MultiProtocol Label Switching)

MPLS [10] reprezentuje następny poziom w ewolucji standardów w połączeniu technologii przełączania w warstwie 2 (ang. *data link layer*) z technologią routingu w warstwie 3 (ang. *network layer*). Głównym celem tego procesu standaryzacyjnego jest stworzenie elastycznej struktury sieciowej, która będzie miała większą efektywność niż dotychczasowe sieci oraz będzie w większym stopniu skalowalna. W związku z tym, muszą być w niej zawarte właściwości inżynierii ruchu (ang. *traffic engineering*), która zaoferuje, np. aspekty zapewnienia określonej jakości usług sieciowych (ang. *Quality of Service (QoS) / Class of Service (CoS)*) i ułatwi użycie wirtualnych sieci prywatnych (ang. *Virtual Private Networks (VPNs)*). MPLS jest zaprojektowany do pracy z różnorodnymi mechanizmami transportu, jednakże, początkowa implementacja skupia się na dwóch głównych technologiach ATM i Frame Relay, które już są zaimplementowane w dużych sieciach operatorów.

MPLS [10] zapewnia w pełni połączeniową, niezawodną oraz odporną na uszkodzenia transmisję danych. MPLS stosowany jest najczęściej w sieciach szkieletowych, odpowiadając za transportowanie datagramów IP pomiędzy brzegami tego szkieletu, w celu ich wymiany pomiędzy sieciami IP. Dzięki takiemu podejściu, bezpołączeniowy oraz zawodny protokół IP zyskuje nową jakość umożliwiającą stosowanie nadmiarowych, zarządzalnych i w pełni przewidywalnych ścieżek po których przekazywane są pakiety, a także wprowadzenie zaawansowanych algorytmów QoS oraz CoS.

Aby ruch IP mógł zostać przekazywany przez sieć MPLS, w miejscach łączących obie architektury sieci umieszczane są routery LSR (ang. *Label Switching Router*) potrafiące przekształcać w odpowiedni sposób pakiety IP tak, aby dostosować je do wymagań MPLS, a po przejściu przez tę sieć przetworzyć je na zwykłe pakiety IP które będą mogły być przekazywane dalej przez sieć o takiej właśnie architekturze.

Koncepcja działania protokołu MPLS zaczerpnięta została z sieci ATM (ang. *Asynchronous Transfer Mode*) oraz Frame Relay. Technologie przełączania oparte na ATM i Frame Relay, wykorzystują całkiem inny algorytm przekazywania w stosunku do IP, który jest właściwie algorytmem wymiany etykiet (ang. *label -swapping*). Z powodu prostoty algorytmu, jest on implementowany w sprzęcie, przynosząc korzyść w postaci lepszej ceny oraz wydajności, w porównaniu go do tradycyjnego routingu IP.

ATM i Frame Relay są technologiami zorientowanymi połączeniowo, co oznacza, że ruch między dwoma punktami jest przenoszony wtedy, gdy zostanie ustalona (predefiniowana) ścieżka. W związku z tym, technologie zorientowane połączeniowo tworzą sieć bardziej przewidywalną i lepiej zarządzaną.

W klasycznej sieci IP, detekcja awarii może zająć od ułamków sekundy, nawet do kilku minut. W sieciach MPLS, czas wykrycia awarii jest zwykle rzędu 60ms. W ciągu tego czasu, algorytm MPLS dodatkowo stara się określić czy jest w stanie zapewnić zapasową ścieżkę zapewniającą nie mniejszy poziom QoS oraz przełączyć całą transmisję na nowo wybraną ścieżkę [10].

3.2.1. Wykrywanie awarii

Stosowane są dwie metody wykrywania awarii sieci [10]. Pierwszą z nich jest monitorowanie na bieżąco stanu kolejnych urządzeń sieciowych, drugą zaś są komunikaty o błędach. Pierwsza metoda zużywa więcej zasobów sieciowych, jest w zamian w niektórych przypadkach znacznie wydajniejsza od metody drugiej.

Metoda opisywana jako bieżące monitorowanie stanu urządzeń, polega na cyklicznym raportowaniu swojego stanu przez wszystkie urządzenia sieciowe do urządzenia kontrolującego działanie sieci. Gdy w pewnej chwili nie zostanie dostarczona informacja o sprawności konkretnego urządzenia, zostaje ono uznane za uszkodzone oraz następuje natychmiastowe wykluczenie go z topologii sieci. W drugiej metodzie awaria wykrywana zostaje przez sąsiadujące z uszkodzonym urządzenie, które rozpoczyna rozsyłanie komunikatów o awarii sąsiada do całej sieci, umożliwiając innym uwzględnienie zmiany w topologii.

Wykorzystanie jednego z dwóch podejść do detekcji awarii, zależne jest od poziomu niezawodności który ma zostać osiągnięty kosztem wydajności sieci.

W sieciach routowanych, takich jak IP podczas przesyłania zwykłych danych, nie wymagających interaktywności oraz wysokiego stopnia QoS detekcja awarii trwająca kilkanaście sekund oraz ponowne przesłanie informacji może być akceptowalne. Problem pojawia się dopiero w przypadku przesyłania danych wymagających wysokiego poziomu QoS, takich jak przesyłanie dźwięku czy obrazu w czasie rzeczywistym w przypadku telemedycyny. Na potrzeby takich usług, opracowane zostały protokoły takie jak IGP potrafiące w bardzo szybkim czasie wykryć awarię oraz zapewnić alternatywną drogę dla danych. Protokół IGP nie potrafi jednak zapewnić odpowiednio wysokiego poziomu QoS. W tym miejscu na ratunek przychodzi MPLS.

Architektura MPLS potrafi zapewnić odpowiedni poziom QoS w zależności od typu transmisji i jej priorytetu, co więcej, potrafi przełączyć transmisje o niższym priorytecie na inne ścieżki tak, aby zapewnić transmisji o większym priorytecie odpowiednio wysoki poziom jakości usługi.

Takie działanie możliwe jest dzięki implementacji w protokole MPLS architektury zarządzającej ruchem (ang. *Traffic Engineering*) zdefiniowanej przez IETF (ang. *Internet Engineering Task Force*) w dokumencie [15] określanej jako *DiffServ*. Działanie *DiffServ* opiera się na redefinicji pola TOS (ang. *Type Of Service*) w nagłówku pakietu IP. Implementacje IP rzadko obsługiwały ten element protokołu, co utrudniało zarządzanie ruchem oraz jakością usług świadczonych przy pomocy IP. *DiffServ* umożliwia klasyfikowanie ruchu IP w zależności od jego zachowania oraz przypisywanie go do określonych grup współdzielących ten sam poziom jakości usługi, nie zapewnia jednak samej kontroli jakości.

Tę drugą cechę posiada połączeniowy MPLS, co wspólnie umożliwia przypisanie każdego pakietu do określonej klasy oraz zarządzanie klasami poprzez na podstawie ścieżek wyznaczanych przez MPLS [14].

Zakłada się, że ruch przypisany do konkretnej klasy musi spełniać pewne wymagania dotyczące jakości oraz priorytetu usługi. Podczas tworzenia połączenia MPLS pomiędzy punktami końcowymi, obliczana jest ścieżka po której będą poruszać się pakiety z danymi, która spełni wymagania dotyczące żądanej przez usługę przepustowości. Wraz z podstawową ścieżką, obliczane są ścieżki zapasowe (niekoniecznie w pełni rozłączne), które będą mogły zostać wykorzystane po wykryciu awarii aktualnie wybranej ścieżki.

3.3. VRRP (Virtual Router Redundancy Protocol), CARP(Common Address Resolution Protocol)

Typowe, lokalne sieci Ethernet/IP budowane są z wykorzystaniem jednego routera, który zapewnia połączenie z innymi sieciami, w tym z Internetem. W takiej konfiguracji mamy do czynienia z pojedynczym punktem awarii jakim jest router. W sytuacji, gdy urządzenie to ulegnie uszkodzeniu - przestanie przekazywać pakiety – wówczas cała sieć traci możliwość komunikacji z otoczeniem.
Aby tolerować ten typ uszkodzeń stosuje się redundancję sprzętu. Uruchamia się kilka routerów fizycznych, które tworzą jeden „wirtualny" router z wykorzystaniem protokołów VRRP lub CARP.
Zanim protokoły te zostaną omówione należy wyjaśnić jak odbywa się komunikacja z otoczeniem zewnętrznym w sieciach Ethernet [1].
Każda stacja posiada unikalny adres IP oraz adres fizyczny MAC. Komunikacja między dwoma stacjami jest możliwa pod warunkiem, że każda ze stacji zna adres IP oraz MAC stacji drugiej. Adres IP zazwyczaj jest uzyskiwany poprzez usługę DNS, natomiast do zdobywania adresu MAC służy protokół ARP (ang. *Address Resolution Protocol*).
Zakładamy, że adres IP stacji docelowej (B) jest znany, natomiast stacja nadawcza (A) nie posiada informacji o adresie MAC stacji B. W tym celu A wysyła zapytanie ARP do wszystkich stacji w sieci, w którym żąda adresu MAC stacji B. Wszystkie stacje ignorują ten komunikat, z wyjątkiem stacji B. B wysyła do A pakiet-odpowiedź, który zawiera jej adres MAC. Od tej chwili obie stacje znają swoje adresy MAC i mogą się komunikować.
Aby wysłać pakiet do innej sieci, stacja wysyła go na adres MAC domyślnego routera, którego adres IP ma zapisany w konfiguracji. Gdy router odbiera pakiet adresowany do niego w warstwie MAC, ale o IP różnym od jego adresu to przesyła go do odpowiedniej sieci.
Protokół VRRP został stworzony przez firmę Cisco [2]. Jego działanie polega na stworzeniu wirtualnego routera przez utrzymywanie wirtualnego adresu MAC. Ten adres jest przypisany routerowi fizycznemu, który w danej chwili pełni funkcję MASTER.
Każdy router nadaje cyklicznie pakiety do pozostałych routerów. Pakiety te informują pozostałe routery o tym, że router który je wysyła działa. W każdym pakiecie rozgłoszeniowym jest zapisany priorytet routera wysyłającego. Gdy nastąpi uszkodzenie to router, który jemu uległ przestaje nadawać. Jest to sygnał dla pozostałych routerów, które wybierają spośród siebie nowego MASTERa. W tryb MASTER przechodzi router z najwyższym priorytetem.
W momencie gdy router, który wcześniej pełnił rolę MASTERa zostaje naprawiony zaczyna nadawać pakiety rozgłoszeniowe. Jako, że pakiety te mają wyższy priorytet niż pakiety aktualnego MASTERa, router przejmuje jego funkcje.
Należy zauważyć, że protokół VRRP (odnosi się to również do protokołu CARP, który za chwilę będzie omówiony) pozwala na tolerowanie uszkodzeń nie tylko routerów, ale także sieci komputerowej (okablowanie, urządzenia aktywne). Można sobie wyobrazić geograficznie rozproszoną strukturę routerów w jednej, dużej sieci. Gdy nastąpi uszkodzenie okablowania, sieć zostanie podzielona na dwie, nie komunikowalne części.

Jeśli zdaży się, że każda z tych części będzie miała łączność z przynajmniej jednym z routerów fizycznych to okaże się, że routery w obu częściach sieci przejdą w tryb MASTER i łączność z otoczeniem zostanie zachowana. Po naprawieniu uszkodzenia obie części sieci połączą się w jedność, zaś router MASTER o niższym priorytecie przekaże swoją funkcję routerowi o wyższym priorytecie.
Protokół CARP został zaprojektowany dla i zaimplementowany w systemie OpenBSD przez jego twórców [3]. Powstał on jako odpowiedź na protokół VRRP, którego używanie w oprogramowaniu Open Source jest ograniczone przez prawo patentowe. Jest to nowy protokół (pierwsza wersja została upubliczniona 17.10.2003) dlatego jego specyfikacja może podlegać jeszcze nieznacznym zmianom.
Działanie protokołu CARP, podobnie jak VRRP jest oparte na istnieniu wirtualnego routera. Jednak w przeciwieństwie do swojego opatentowanego konkurenta, CARP nie stosuje jednego, wspólnego adresu MAC tylko wspólny adres IP. Każdy router fizyczny rozgłasza do pozostałych informacje o swojej dostępności. Ten który posiada najwyższy priorytet przechodzi w tryb MASTER. Tryb MASTER uprawnia do wysyłaniu odpowiedzi na zapytania ARP o wspólny adres IP. Pozostałe routery nie mają do tego prawa.
W przypadku uszkodzenia routera MASTER, pozostałe routery odnotowują fakt, że nie dostają od niego komunikatów i wybierają nowego MASTERa spośród siebie. Od tej chwili na zapytania ARP odpowiada nowy MASTER, co powoduje, że ruch do innych sieci jest routowany przez niego.
Protokół CARP zapewnia większe bezpieczeństwo niż VRRP, ponieważ używa kluczy uwierzytelniających komunikaty rozgłoszeniowe – dzięki temu skutecznie blokuje ataki polegające na podszywaniu się pod router MASTER przez stacje atakującego. Dodatkowo CARP pozwala na dzielenie obciążenia pomiędzy kilka routerów. Jest to ważna funkcja, ponieważ w klasycznym przypadku przeciążenie pojedynczego routera powoduje defekt – użytkownik sieci nie jest w stanie korzystać z sieci zewnętrznych. W przypadku CARP router fizyczny może być inny dla każdej stacji. Jest to uzyskiwane przez wysyłanie różnych odpowiedzi ARP dla różnych stacji z wykorzystaniem algorytmu karuzelowego z wagami (ang. *Weighted Round Robin*). Dzięki temu, przy zastosowaniu równych wag, każdy router fizyczny przyjmuje na siebie obciążenie 1/n liczby stacji, gdzie n jest liczbą routerów.
Niestety protokół CARP posiada w stosunku do VRRP wadę, która objawia się opóźnieniem odczuwalnym przez użytkowników w przypadku, gdy funkcję MASTER przejmuje inny router fizyczny. Dzieje się tak, ponieważ stacje zapamiętują adres MAC routera MASTER; po zmianie adresu MAC routera MASTER, stacje wysyłają przez pewien czas pakiety pod stary adres MAC. Ten adres należy do routera, który już nie działa, więc komunikacja zamiera. Dopiero to powoduje, że stacje wysyłają nowe zapytania ARP i tym razem uzyskują adres MAC aktualnego MASTERa.

3.4. Retransmisja w protokole TCP

Protokół TCP, niezwykle popularny w dzisiejszych sieciach komputerowych zarówno lokalnych jak i globalnych takich jak Internet, zapewnia niezawodną warstwę transportową. Jest to możliwe, dzięki potwierdzeniom wysyłanym przez jedną stronę połączenia, dotyczących odebrania danych wysyłanych przez stronę drugą. Jak zostało wspomniane, TCP potrafi zapewnić niezawodność transmisji danych, w przypadku gdy protokół niższej warstwy jakim jest IP zawiedzie i zaginięciu ulegną segmenty z danymi

TCP zabezpiecza się przed taką możliwością poprzez określanie czasu oczekiwania (ang. *Timeout*) podczas wysyłania danych: jeśli dane nie zostaną potwierdzone przed upływem

określonego czasu oczekiwania, następuje ich retransmisja. Krytycznym elementem każdej implementacji tego protokołu jest strategia czasu oczekiwania i retransmisji. Należy odpowiedzieć przy tym na pytania w jaki sposób określany jest przedział czasu oczekiwania i jak często występuje retransmisja.

Podstawą działania retransmisji w TCP jest mierzenie czasu podróży pakietów (RTT) w danym połączeniu. Można spodziewać się, że czas ten ulega zmianie tak samo, jak zmienia się trasa, jaką przesyłane są pakiety i ruch w sieci. TCP powinien śledzić te wszystkie zmiany i na podstawie aktualnych informacji stosować odpowiednie czasy oczekiwania.

Oryginalna specyfikacja TCP [11] określa szacowaną wartość RTT w chwili i na podstawie następującego filtra:

$$RTT(i) = RTT \cdot (i - 1) + (1 - \alpha) \cdot M$$

gdzie α jest współczynnikiem, którego zalecana wartość to 0,9. Szacowana w ten sposób wartość RTT uaktualniana jest za każdym razem, kiedy dokonywany jest pomiar. Dziewięćdziesiąt procent nowo oszacowanej wartości pochodzi z poprzedniej wartości, a 10% z nowego pomiaru.

Mając tak oszacowaną wartość, która zmienia się wraz ze zmianą RTT, zaleca się [11] żeby czas retransmisji (RTO) określany był jako:

$$RTO = R \cdot \beta$$

gdzie β jest zmienną opóźnienia o rekomendowanej wartości równej 2.

3.4.1. Algorytm Jacobsona

W dokumencie [12] szczegółowo omówiono te kwestie, bazując na opisanym wyżej podejściu i udowodniono, że stosując taki algorytm nie można nadążyć za zmianami RTT, co powoduje niepotrzebne retransmisje. Jak stwierdzono, niepotrzebna dodatkowa retransmisja obciąża sieć, która i tak jest już mocno obciążona.

W [12] stwierdzono, że oprócz szacowania RTT naprawdę potrzebna jest funkcja śledzenia różnic w mierzonych RTT. RTO policzone w oparciu o obie wartości pozwoli na znacznie lepsze dostosowanie czasu do zmian w sieci niż algorytm liczenia RTO oparty o stałą wielokrotność różnicy czasów.

Średnie odchylenie jest dobrym przybliżeniem odchylenia standardowego [12], a jednocześnie jest ono łatwiejsze do policzenia. Takie rozumowanie prowadzi do następujących równań, stosowanych przy pomiarze M dla każdego RTT.

$$Err = M - A$$
$$A = A + g \cdot Err$$
$$D = D + h \cdot (|Err| - D)$$
$$RTO = A + 4 \cdot D$$

gdzie *A* jest wygładzoną tłumioną wartością RTT (oszacowaniem wartości średniej), a *D* jest wygładzoną wartością odchylenia średniego. *Err* jest różnicą pomiędzy zmierzoną właśnie wartością , a obecnie stosowaną wartością RTT. Zarówno *A* jak i *D* używane są do obliczenia kolejnej wartości czasu retransmisji (RTO). Wartość przyrostu *g* jest stosowana do uśrednienia i zwykle ustawiona jest na 0,125. Przyrost odchylenia określony jest stałą *h*, której wartość to 0,25. Duży przyrost odchylenia powoduje, że RTO rośnie szybciej niż zmienia się RTT.

3.4.2. Algorytm Karna

Retransmisja pakietów dostarcza także nowych problemów. Jeśli pakiet jest retransmitowany i zachodzi przekroczenie czasu oczekiwania, RTO jest wtedy definiowane zgodnie z powyższym opisem, następuje retransmisja pakietu z dłuższym RTO i odbierane jest potwierdzenie. W tym momencie, nie wiadomo czy potwierdzenie dotyczy pierwszej transmisji czy też drugiej. Taka sytuacja nazywana jest problemem niejednoznaczności transmisji (ang. r*etransmission ambiguity problem*).

Jeśli wystąpi przekroczenie czasu i retransmisja, to nie można uaktualniać RTT, dopóki nie nadejdzie potwierdzenie retransmitowanych danych [13]. Nie wiadomo bowiem, której transmisji dotyczy odebrane właśnie potwierdzenie. (Być może opóźnieniu uległa pierwsza transmisja i nie została ona odrzucona, możliwe jest również, że opóźnieniu uległo potwierdzenie dla pierwszej transmisji).

Ponadto, kiedy dane uległy retransmisji i zastosowano wykładniczy algorytm dla RTO, to policzony w ten sposób RTO będzie ponownie użyty w następnej retransmisji. Dopóki nie nadejdzie potwierdzenie dla segmentu, który nie był retransmitowany, nie jest liczona kolejna wartość RTO.

3.4.3. Zegar Persist

Odbiorca danych TCP może informować nadawcę o ilości danych które jest w stanie odebrać. Służy do tego specjalne pole w nagłówku TCP określane jako rozmiar okna. Nadawca może wysyłać dane do odbiorcy tylko wtedy, gdy rozmiar okna ustawiony jest na wartość dodatnią, natomiast gdy wynosi on 0, oznacza to że odbiorca nie chce w danej chwili otrzymywać żadnych nowych danych, ponieważ nie przetworzył jeszcze poprzednio przesłanych. Gdy bufor odbiorcy zwolni się, wysyła on do nadawcy informację zawierającą dodatni rozmiar okna określający ilość bajtów która jest w stanie w danej chwili odebrać.

W sytuacji w której z jakiegokolwiek powodu w sieci zaginie informacja o nowym rozmiarze okna, może dojść do takiego stanu w którym odbiorca oczekuje na nowe dane od nadawcy, a nadawca oczekuje na informację określającą dodatni rozmiar okna.

Aby zapobiec tego typu sytuacjom, nadawca wykorzystuje zegar *persist*, który wymusza regularne odpytywanie odbiorcy o to, czy rozmiar okna został zwiększony. Segmenty wysyłane przez nadawcę w wyniku działania tego zegara, nazywane są sondowaniem okna (ang. *Window probes*).

3.5. Równoważenie obciążenia i metody traktowania uszkodzeń łącz internetowych

Przy ograniczonym budżecie i zapotrzebowaniu na szybki dostęp do globalnej sieci (np. w przypadku dostarczycieli Internetu) często zakup łącza o dużej przepustowości nie wchodzi w rachubę ze względu na wysoki koszt. Okazuje się, że wykupienie kilku słabszych łącz jest tańsze, a uzyskana sumaryczna przepustowość zbliżona. Niestety zazwyczaj tanie łącza są bardziej awaryjne, zaś umowa SLA (ang. *Service Level Agreement* – umowa o zapewnieniu jakości usługi) w ogóle nie wchodzi w grę. Poważnym problemem jest także rozkładanie obciążenia (ang. *Load Balancing, LB*) proporcjonalnie do parametrów łącza. Te dwa problemy nakładają się: należy zapewnić tolerowanie uszkodzeń przy jednoczesnym rozkładaniu obciążenia. LB stanowi element, który zwiększa złożoność algorytmów tolerujących uszkodzenia.

3.5.1. Podejście klasyczne

Istnieje rozwiązanie, stosowane powszechnie przez dużych operatorów, które pozwala na tolerowanie uszkodzeń i rozkładanie obciążenia jednocześnie. Jest to protokół BGP (ang. *Border Gateway Protocol*) [16, 17]. Jego działanie opiera się na sieci Systemów Autonomicznych AS (ang. *Autonomous System*). System Autonomiczny to sieć (która może składać się z wielu podsieci), posiadająca jeden lub więcej routerów zapewniających komunikację z sieciami zewnętrznymi. Każdy router rozgłasza do innych routerów komunikaty o sieciach z którymi posiada łączność. W ten sposób na wszystkich routerach obsługujących BGP są budowane tablice routingu. Jeśli do jednej sieci prowadzi wiele tras, to router wybiera optymalną, kierując się zbiorem parametrów związanych z tymi trasami. Podstawowym parametrem jest liczba AS'ów, przez które prowadzi trasa – im liczba ta jest niższa tym trasa lepsza.
W przypadku uszkodzenia jednego z łącz, router który je zauważy rozgłasza tą informację do routerów obsługujących najbliższe systemy autonomiczne, wraz z informacjami na temat tras alternatywnych. Routery sąsiadujące modyfikują swoje tablice routowania, co powoduje, że ruch skierowany do AS'a z uszkodzonym łączem jest kierowany inną trasą i uszkodzenie jest maskowane.
BGP zapewnia rozkładanie obciążenia, jednak nie jest to realizowane optymalnie. BGP nie posiada mechanizmów, które pozwoliłyby na wybranie więcej niż jednej scieżki naraz do jednej sieci, co powoduje, że przy intensywnej komunikacji z jedną siecią występuje przeciążenie jednego łącza, zaś pozostałe nie są wykorzystywane. Lekarstwem na bolączki BGP jest wchodzący protokół MPLS, który został opisany w innym rozdziale.
Niestety, podstawową wadą BGP jest cena. Założenie własnego AS jest kosztowne, opłaty abonamentowe również. Dodatkowo, często trzeba się liczyć z sytuacją, że BGP nie jest dostępne w ogóle (np. w mniejszych miejscowościach).

W przeciwieństwie do BGP, które pozwala na kontrolę ruchu w obie strony, a co za tym idzie współpracy między routerami operatorów (co zazwyczaj zmusza operatora usługobiorcę do płacenia usługodawcy), rozwiązania alternatywne bazują na rozkładaniu obciążenia generowanego przez ruch wychodzący. Ten sposób posiada pewne ograniczenia:

1. Bez współpracy z zewnętrznymi routerami nie można wybierać tras optymalnych w czasie rzeczywistym.
2. Nie sposób dokładnie przewidzieć jaki będzie ruch przychodzący na podstawie ruchu wychodzącego, dlatego nie jest możliwy optymalny wybór trasy dla połączeń wychodzących.

3. Usługi dostępne z zewnątrz są widziane pod różnymi adresami IP dla każdego z łącz, ponieważ adresy te nie są wzajemnie routowalne; w sytuacji uszkodzenia jednego z łącz, aby skorzystać z usługi trzeba użyć innego adresu – tak więc występuje defekt, ponieważ użytkownik musi ręcznie wprowadzić inny adres.
4. Z tego samego powodu nie jest możliwe optymalne rozłożenie obciążenia łącz ruchem inicjowanym z zewnątrz.

Okazuje się jednak, że problemy te w wielu zastosowaniach nie są krytyczne i przedstawione poniżej rozwiązania są używane powszechnie np. w sieciach osiedlowych. Sieci te charakteryzują się małym natężeniem ruchu wychodzącego oraz intensywnym ruchem przychodzącym. Jednak ruch przychodzący niemal w całości jest generowany przez odpowiedzi na pakiety wysyłane z sieci wewnętrznej; ruch inicjowany z sieci zewnętrznych jest marginalny, dlatego problemy 3) i 4) tracą na istotności.

3.5.2. Metody statyczne

Pierwszą klasą rozwiązań LB są metody statyczne.

Najprostszy sposób to zdefiniowanie statycznej tablicy routingu[1], czyli przydzielenie każdemu z łącz sektorów Internetu (np. poszczególnych państw) i routowaniu ruchu do danego sektora przez dane łącze.

Sposobem podobnym jest routing na podstawie adresu IP nadawcy [18] z sieci wewnętrznej (ang. *source-routing*). Ten sposób jest najczęściej wykorzystywany w sieciach osiedlowych. Polega on na tym, że sieć jest dzielona na tyle podsieci iloma łączami dysponuje operator. Następnie każdej podsieci zostaje przypisane łącze, z którego korzysta.

Kolejny sposób polega na rozdziale ruchu między łącza w zależności od usługi. Np. usługi interaktywne są routowane łączem #1, zaś ruch masowy łączem #2.

Te trzy statyczne metody można ze sobą dowolnie łączyć, jednak mimo tego efekty, które z ich pomocą się uzyskuje są mierne zarówno w kwestii LB jak i tolerowania uszkodzeń. Zostały one tutaj wymienione tylko dlatego, że są najbardziej rozpowszechnione.

3.5.3. Metody statystyczne

Drugą klasą są metody statystycznego rozkładania obciążenia; algorytmów w tej klasie jest bardzo wiele, dlatego przedstawimy tylko najbardziej popularne.

Wielotorowy, równowagowy routing (ang. *Equal Cost Multipath*) [18] to zaimplementowany w jądrze systemu Linux algorytm LB, który polega na dynamicznym wyborze łącza na podstawie funkcji skrótu (ang. *hash*) z adresu źródłowego i docelowego pakietu oraz z aktualnego czasu (mierzonego z pewną, niezbyt wysoką rozdzielczością). Do tego dochodzi tablica tymczasowych tras (ang. *routing cache*), która zapamiętuje ostatnio używane trasy, aby z powodu zmiany czasu nie zmieniać łącza dla już zestawionego połączenia. Taka zmiana jest niedopuszczalna przy translacji adresów NAT (ang. *Network Address Translation*), której wykorzystanie jest niemal regułą w sieciach osiedlowych.

Opisany algorytm jest wydajny, ponieważ zarówno zaimplementowana funkcja skrótu jak i przeszukiwanie tablicy cache są szybkie i prawie niezależne od obciążenia.

Wadą tego rozwiązania jest możliwość wystąpienia sytuacji, w której adres hosta źródłowego (wewnętrznego) i docelowego zostają przypisane do silnie obciążonego w danej chwili łącza. W tym momencie użytkownik praktycznie nie jest w stanie połączyć się z hostem docelowym, a ponieważ komputer co pewien czas ponawia próby łączenia się, wpis z tablicy cache nie jest usuwany, co powoduje, że pakiety użytkownika cały czas są routowane tym samym łączem – jest to defekt.

Problem ten częściowo rozwiązuje algorytm dynamicznego rozdzielania połączeń. Jest on dostępny np. w systemach Linux[38] i OpenBSD [20]. Polega na śledzeniu połączeń (ang. *connection tracking*) w filtrze pakietów, oraz współpracy tegoż z podsystemem

routowania. Każde połączenie jest rejestrowane w specjalnej strukturze danych w pamięci jądra oraz odpowiednio oznaczane. Oznaczenie to jest dziedziczone przez pakiety wchodzące w skład tego połączenia, zaś podsystem routowania wybiera łącze którym wyśle pakiet na podstawie tego oznaczenia.
Rozwinięciem tej metody jest dynamiczne rozdzielanie połączeń w zależności od obciążenia. Algorytm został zaimplementowany w systemie Linux [21, 22], są dostępne także implementacje dla innych systemów [23]. Sposób ten polega na ciągłym badaniu obciążenia łącza i przydzielaniu nowych połączeń najmniej obciążonemu.

3.5.4. Monitorowanie łącz

Wymienione metody (statyczne i statystyczne) polegają na ciągłym sprawdzaniu dostępności łącz. Jeśli jedno z łącz przestaje być dostępne następuje rekonfiguracja i obciążenie uszkodzonego łącza przejmują łącza pozostałe. Mogą być także podejmowane działania, które mają na celu naprawę uszkodzenia.
Metody badania stanu łącza można podzielić na pasywne, aktywne i aktywne lokalnie.
Sposoby pasywne opierają się na analizie ruchu na routerze. Jeśli router stwierdzi, że mimo wysyłania pakietów przez dane łącze, ruch przychodzący nie pojawia się, oznacza to najpewniej, że wystąpiło uszkodzenie.
Metody aktywne polegają na ciągłym wysyłaniu pakietów kontrolnych do sieci zewnętrznych i oczekiwaniu odpowiedzi. Mogą to być np. pakiety żądania echa ICMP (ang. *Internet Control Message Protocol Echo-Request*). Inną metodą jest zastosowanie algorytmu badania trasy przez modyfikację pola TTL (ang. *Time To Live*) w nagłówku pakietów IP (algorytm ten został zaimplementowany w programie traceroute [24]) – dzięki niemu router posiada informacje nie tylko o uszkodzeniu, ale także o prawdopodobnym miejscu jego wystąpienia. To w połączeniu z ogólną wiedzą na temat infrastruktury sieciowej dostawcy łącza może służyć do dokładniejszego diagnozowania uszkodzeń i odpowiedniego ich traktowania.
Metody aktywne lokalnie ograniczają monitorowanie do sieci wewnętrznej, tak aby pakiety kontrolne nie obciążały łącz. W przypadku, gdy łącza są podłączone do routera z wykorzystaniem sieci ethernet mogą to być pakiety ARP WHO-HAS. Innym przykładem może być badanie stanu łącza modemu DSL (ang. *Digital Subscriber Line*) przez połączenie telnet/snmp lub połączenie szeregowe. Modem bada stan linii w warstwie I oraz II i udostępnia te informacje przez zdalną konsolę. Program na routerze nawiązuje połączenie lokalne z modemem (a więc nie obciąża łącza) i zczytuje te informacje, aby następnie podjąć decyzję o ewentualnej rekonfiguracji tablicy routowania.
Sposoby automatycznej naprawy uszkodzeń ograniczają się zazwyczaj do zdalnego resetowania urządzeń dostępowych. Dokonuje się tego np. za pośrednictwem zdalnej konsoli, i/lub wyspecjalizowanych urządzeń typu netdog (nazwa wzięła się od układów watchdog stosowanych w urządzeniach elektronicznych). Przykładami netdogów są urządzenia firmy Osso[25] oraz Cyberbajt Reseter [26].
Poważną wadą wszystkich metod monitorujących jest bezwładność. Każdy pomiar odbywa przez pewien okres czasu, zaś czas który upływa między kolejnymi pomiarami nie może być zbyt mały, szczególnie w przypadku metod aktywnych. To powoduje, że wystąpienie uszkodzenia jest rejestrowane zawsze z pewnym opóźnieniem, zaś to z kolei jest przyczyną defektów, ponieważ router przez ten czas nie jest rekonfigurowany i ruch jest routowany przez uszkodzone łącze. Jeśli czas między pomiarami wynosi n, to rekonfiguracja zajmuje średnio n/2+m, gdzie m jest sumą czasu rekonfiguracji oraz samego pomiaru.

3.5.5. Metoda niezależna od monitorowania

Ciekawą metodą, której nie można zaliczyć do klasy metod statycznych ani statystycznych jest redundatne wysyłanie pakietów inicjujących połączenie. Tą metodę można zaimplementować z wykorzystaniem wirtualnego interfejsu [27]. Mając do dyspozycji n łącz, router typu statefull wybiera pakiety inicjujące połączenie (w przypadku protokołu TCP są to pakiety z ustawioną flagą SYN, pakiety innych protokołów muszą być konfrontowane z tablicą stanów połączeń) oraz zwielokratnia je n-razy. Każdą kopię pakietu opatruje innym, źródłowym adresem IP oraz routuje innym łączem. Następnie router czeka na pierwsze potwierdzenie pakietu inicjującego od zdalnego hosta. Potwierdzenie to przychodzi najmniej obciążonym w danej chwili łączem. W tablicy stanów routera połączenie jest zestawiane i kojarzone z tym łączem, zaś odpowiedzi, które spłyną po pewnym czasie pozostałymi łączami są ignorowane.

Ten algorytm wybiera łącze, dla którego początkowe opóźnienie jest najmniejsze. Z punktu widzenia tolerowania uszkodzeń ten algorytm posiada dwie poważne zalety.

Pierwszą jest fakt, że monitorowanie łącz nie jest niezbędne dla prawidłowego działania algorytmu. Oczywiście monitorowanie jest cały czas przydatne dla administratora, który powinien wiedzieć o uszkodzeniach. Jednak algorytm monitorowania może być mniej agresywny niż w przypadku metod statycznych i statystycznych – monitorowanie może być całkowicie pasywne.

Drugą, istotną zaletą jest to, że uszkodzenia są maskowane natychmiast po ich wystąpieniu. Jeśli założymy, że pomijamy defekty będące skutkiem przeciążenia łącza oraz zerwania instniejących sesji, to algorytm zapewnia, że dopóki jest dostępne choć jedno łącze to nie występują błędy, zaś użytkownik nie odczuwa defektu – cały czas jest w stanie nawiązywać nowe połączenia (w przeciwieństwie do metod opartych na monitorowaniu). Algorytm jest wysoce odporny na uszkodzenia wielokrotne, zaś użytkownik nie jest w stanie zauważyć momentów w których wystąpiło uszkodzenie.

Wadą metody jest generowanie zbędnego ruchu w Internecie – pakietów inicjujących połączenia, które są później ignorowane. Przy normalnym wykorzystaniu pakiety te stanowią nikły procent ruchu, jednak możliwe są nadużycia prowadzące do zablokowania łącz. Aby tego uniknąć należy limitować liczbę nowych połączeń użytkownika na jednostkę czasu.

3.6. Globalna tablica NAT i śledzenie połączeń

W czasach rosnącej przestępczości elektronicznej oraz zalewu wirusów i robaków internetowych koniecznością staje się zabezpieczenie sieci przed atakami.

Firewall to urządzenie które zapewnia podstawową ochronę. Jest to router wyposażony w funkcje filtrowania pakietów. Administrator konfiguruje zestaw reguł (czasami bardzo duży), które firewall stosuje do każdego pakietu, który jest przez niego routowany i decyduje, czy go zaakceptować czy odrzucić.

Powyższy opis dotyczy firewalli nie śledzących połączeń (ang. *stateless*). Bardziej zaawansowane firewalle utrzymują w pamięci tablice połączeń i pozwalają na uwzględnianie w regułach ich stanu. Są to firewalle typu statefull. Zapewniają większe bezpieczeństwo, ponieważ proste filtry pakietów typu stateless można łatwo oszukać. Dodatkowo takie firewalle z reguły mają wbudowaną funkcję translacji adresów NAT, która służy do udostępniania dostępu do Internetu wielu maszynom w sieci wewnętrznej za pomocą małej liczby adresów publicznych (z reguły jednego).

Ze względów bezpieczeństwa firewall stanowi jedyną drogę dostępu do Internetu dla sieci wewnętrznej. Z punktu widzenia tolerowania uszkodzeń firewall jest pojedynczym punktem awarii sieci.

Najczęściej spotykanym rozwiązaniem tego problemu jest redundancja sprzętu. Do sieci wpina się równoległe drugi firewall (lub kilka firewalli), który pracuje w trybie slave do momentu awarii firewalla master. Wtedy firewall slave przełącza się w tryb master. Można do tego celu wykorzystać omówione w innych rozdziałach rozwiązania VRRP, CARP lub Hearbeat.
Opisane rozwiązanie sprawuje się doskonale, kiedy mamy do czynienia z firewallami typu stateless (np. w systemie Linux z VRRP [43]). Jedynym zagadnieniem w tym przypadku jest synchronizacja reguł firewalla. Jeśli administrator zmieni reguły na firewallu głównym, zmiana ta powinna rozpropagować się na pozostałe. Nie jest to poważny problem. Administratorzy najczęściej stosują do tego celu proste skrypty shellowe wykorzystujące programy do kopiowania plików w sieci.
Sytuacja komplikuje się w przypadku firewalli typu statefull. Zastosowanie powyższego rozwiązania spowoduje, że tylko firewall główny będzie utrzymywał tablice połączeń. Kiedy ulegnie uszkodzeniu, firewall zapasowy które przejmie jego funkcje będzie miał pustą tablice połączeń. Doprowadzi to do zerwania wszystkich nawiązanych sesji użytkowników. W przypadku krótkich połączeń HTTP nie stanowi to dużego problemu, jednak zerwanie sesji protokołów zorientowanych połączeniowo (np. SSH) będzie stanowić poważny defekt.
Istnieją dwa rozwiązania tego problemu [41, 42].

3.6.1. Nasłuchujący firewall slave

Idea określona przez jej twórce jako „tolerowanie awarii dla ubogich" polega na klastrze firewalli, z których tylko jeden (aktualny master) przekazuje pakiety (ma włączoną funkcję ip_forwarding w jądrze). Pozostałe firewalle slave tylko analizują ruch wchodzący i wychodzący z firewalla master. Nasłuchiwanie jest możliwe tylko w klasycznym, nieprzełączanym ethernecie – czyli opartym na zwykłym hubie.
Firewalle slave analizując ruch modyfikują swoje tablice połączeń, dzięki czemu jest ona zsynchronizowana z tablicą połączeń firewalla master.
Metoda ta jest trywialnie prosta w implementacji. Polega na dołożeniu do jądra prostego kodu, który zapewnia śledzenie połączeń w trybie nasłuchowym.
Niestety ta prostota jest okupiona wadami:

1. Mimo jednorodnej konfiguracji sprzętowej i programowej klastra może się zdażyć, że nastąpi rozsynchronizowanie stanów z powodów losowych – szczególnie pod wysokim obciążeniem.
2. Przerwa w działaniu firewalla nasłuchowego powoduje, że traci on z tablicy te z połączeń, które zostały zainicjowane podczas jego niedostępności. To powoduje utratę synchronizacji tablicy stanów i w momencie uszkodzenia firewalla master – defekt.
3. Nie jest możliwa synchronizacja stanów NAT. Dzieje się tak, ponieważ każdy z firewalli może przydzielić inny port źródłowy jednemu połączeniu.

3.6.2. Replikacja tablicy połączeń

Jest to metoda stosowana w firewallach Cisco [47] do utrzymywania globalnej tablicy NAT. Istnieją także nieoficjalne poprawki dla systemu OpenBSD, które zapewniają dość prymitywną replikację stanów połączeń [46]. W systemie Linux pojawiają się pierwsze wersje kodu [45] implementujące w sposób pełny protokół Heralda Welte [44], który jest opisany poniżej.
Firewall master rozgłasza informacje o zmianach tablicy stanów do wszystkich firewalli slave. Do tego celu jest wykorzystywany protokół IP Multicast, dzięki czemu komunikat jest wysyłany tylko raz, a odbierają go wszystkie maszyny (i protokół ten działa na

standardowych przełącznikach ethernetowych). W celu wyeliminowania błędów transmisji każdy pakiet ma numer sekwencyjny (inkrementowany cyklicznie) oraz sumę kontrolną. Jeśli firewall slave odbierze pakiet z numerem sekwencyjnym większym o dwa lub więcej niż pakiet poprzedni, oznacza to, że nie odebrał przynajmniej jednego komunikatu. Slave wysyła wtedy do mastera prośbę o ponowne przesłanie zagubionych pakietów.
Zalecana implementacja polega na wykorzystaniu wątków jądra, ponieważ pozwalają one na bezpośredni dostęp do funkcji operujących na tablicy połączeń, są asynchroniczne oraz szybkie.

3.7. DRS (Dynamic Routing System)

DRS [4] to rozwiązanie sprzętowo-programowe umożliwiające tolerowanie uszkodzeń sieci w klastrach, czyli sieci komputerów wykonujących wspólnie obliczenia.
W tego typu sieciach tolerowanie uszkodzeń jest niezbędne, a czas samo-naprawy powinien być możliwie najkrótszy. Maskowanie uszkodzeń powinno odbywać się przed wystąpieniem defektu, a nie post-factum.
Istniejące standardy w postaci protokołów routowania nie spełniają powyższych wymagań. RIP (ang. *Routing Information Protocol*) oraz OSPF (ang. *Open Shortest Path First*) to protokoły używane w sieciach lokalnych; charakteryzują się wysoką bezwładnością czasową, co jest niedopuszczalne w klastrach. Protokoły EGP (ang. *Exterior Gateway Protocol*) i BGP (ang. *Border Gateway Protocol*) mają zastosowanie w sieciach rozległych, ale w przypadku klastrów także nie spełniają swej roli.
DRS spełnia wszystkie powyższe założenia. Umożliwia stworzenie niezawodnej sieci komunikacyjnej typu każdy z każdym (ang. *Peer to Peer*) wielokrotnie niższym kosztem.
DRS opiera się na redundancji sprzętowej. Sieć komunikacyjna oraz interfejsy sieciowe komputerów są powielone. Każdy host posiada dwie karty sieciowe, z których każda jest podłączona do innej, fizycznej sieci.
W systemie DRS na każdym komputerze musi być zainstalowane specjalne oprogramowanie. Oprogramowanie to zajmuje się utrzymywaniem aktualnej tablicy routingu oraz komunikacją z instancjami DRS na innych hostach. Oprogramowanie to realizuje także cykliczne sprawdzanie osiągalności pozostałych hostów w obydwu sieciach z wykorzystaniem pakietów ICMP echo-request (popularnych pingów).
Istnieje wymaganie, aby do jednego hosta prowadziły dwie, różne trasy. W przypadku braku uszkodzeń jedna trasa prowadzi przez pierwszy, druga przez drugi interfejs.
Zbiór uszkodzeń, które są uwzględniane przez DRS to uszkodzenia:

1. Przełącznika/koncentratora ethernetowego.
2. Kabla sieciowego.
3. Interfejsu sieciowego hosta.
4. Sterownika interfejsu sieciowego hosta.
5. Stosu sieciowego systemu operacyjnego hosta.
6. Jądra systemu hosta.
7. Oprogramowania DRS.

Jeśli wystąpi którekolwiek z powyższych uszkodzeń następuje błąd w procesie sprawdzania osiągalności hostów. Np. jeśli uszkodził się przełącznik ethernetowy w sieci #1, to każdy host wykaże, że nie może się skomunikować z pozostałymi w tej sieci. Jednak nie dojdzie do defektu, ponieważ DRS na każdej maszynie uaktualni tablice routowania tak, aby od tej chwili komunikacja odbywała się z wykorzystaniem sieci #2. DRS pozwala na komunikację pośrednią, tzn. z wykorzystaniem jednego hosta jako routera do drugiego. Tak więc warunek

o dwóch trasach zostanie spełniony, ponieważ jedną trasą będzie trasa bezpośrednia, zaś drugą trasa pośrednia prowadząca przez jakiś host.
DRS toleruje uszkodzenia pojedyncze, oraz podwójne proste i złożone. Uszkodzenie pojedyncze zostało omówione wyżej. Uszkodzenie podwójne proste to takie, które DRS rozpoznaje jako pojedyncze. Jest to np. jednoczesne uszkodzenie kabla i portu na przełączniku, do którego ten kabel jest podłączony. DRS nie jest w stanie zweryfikować który element został uszkodzony, więc taka sytuacja jest traktowana jak uszkodzenie pojedyncze.
Uszkodzenie podwójne złożone zaś składa się z dwóch uszkodzeń pojedynczych. Np. uszkodzenie interfejsu sieciowego do sieci #1 hosta A oraz uszkodzenie portu na przełączniku #2, do którego jest podłączony host B.
W tej sytuacji hosty A i B tracą możliwość bezpośredniej komunikacji. Konieczne jest zestawienie alternatywnych tras pośrednich. W tym przykładzie rozważymy zachowanie DRS na hoście A (na hoście B to zachowanie będzie analogiczne).
DRS rozsyła wiadomość do wszystkich hostów w obydwu sieciach (z tym, że wiadomość pojawi się tylko w sieci #2, ponieważ łączność A z siecią #1 jest przerwana). W wiadomości tej umieszcza zapytanie, czy jest w sieci taki host, który posiada łączność z hostem B. Pierwszy host, który się zgłosi jest wpisywany do tablicy routingu jako router do hosta B. Operacja ta jest powtarzana, aby spełnić warunek o dwóch trasach.
Wadą DRS jest duże zapotrzebowanie na pasmo transmisyjne (spowodowane wysyłaniem pakietów sprawdzających osiągalność hostów), które rośnie razem z dwoma zmiennymi: szybkości reakcji na uszkodzenia i liczbie hostów w sieci. Np. przy 162 hostach w sieci i 10-cio sekundowej reakcji na uszkodzenie pasmo transmisyjne sieci Ethernet 10Mbit/s jest zajmowane w 25%.
Autorzy systemu DRS wdrożyli go w dużej firmie telekomunikacyjnej w USA, co zaowocowało całkowitym wyeliminowaniem defektów powodowanych przez uszkodzenia sieci w okresie 12 miesięcy.

3.8. Metody traktowania uszkodzeń w systemach klastrowych z równoważeniem obciążenia

Usługi internetowe, które wystawione są na duże obciążenie coraz częściej rezydują nie na pojedyńczym serwerze, ale na klastrze. Klastry zapewniają równoważenie obciążanie oraz traktowanie uszkodzeń. Ważnym zadaniem jest zapewnienie klientowi przezroczystości, tzn. wrażenia że łączy się z jednym serwerem, a nie kilkoma maszynami tworzącymi klaster.
Istnieje wiele metod, które pozwalają rozwiązać ten problem [28].

3.8.1. Metoda przekierowań HTTP

Ten sposób wykorzystuje mechanizm przekierowań wbudowany w protokół HTTP [29]. Polega on na tym, że gdy główny serwer jest zbytnio obciążony, to w momencie otrzymania zapytania przekierowuje klienta na adres innego, mniej obciążonego serwera. Przeglądarka klienta w sposób dla niego przezroczysty nawiązuje połączenie z drugim serwerem i ponawia zapytanie, tym razem uzyskując odpowiedź.
Wadą tego podejścia jest generowanie pewnej ilości niepotrzebnego ruchu w sieci, związanego z obsługą przekierowań za każdym razem, gdy klient ma skorzystać z serwera innego niż główny.
Z punktu widzenia tolerowania uszkodzeń, sposób ten sprzyja powstawaniu defektów z powodu faktu, że serwer główny jest pojedynczym punktem awarii. Jeśli on zawiedzie, cały klaster zostanie odłączony i wszystkie nowe połączenia będą odrzucane. Aby zapewnić dostateczną dostępność należy tak zaprojektować klaster, aby główny serwer był jak

najbardziej niezawodny. Stopień obciążenia, którego osiągnięcie powoduje, że główny serwer zaczyna wysyłać przekierowania powinien być niższy niż pozostałych serwerów.

3.8.2. Rozkładanie obciążenia realizowane po stronie klienta

Metoda ta polega na wbudowaniu funkcji rozkładania obciążenia i tolerowania uszkodzeń w program kliencki danej usługi. Jako przykład można podać przeglądarkę Netscape Navigator. Gdy użytkownik zechce połączyć się ze stroną główną firmy Netscape to przeglądarka wybiera losowo liczbę N z zakresu 1-32 a następnie łączy się z hostem wwwN.netscape.com [30].
Metoda zapewnia tolerowanie uszkodzeń, użytkownik nie odczuwa defektów dopóki przynajmniej jeden serwer jest osiągalny. Poważna wadą metody jest jej ograniczone zastosowanie – metoda może być wykorzystywana tylko przez firmy, które kontrolują aplikacje wszystkich swoich klientów.
W przypadku, gdy zawęzimy rozważania do usług sieci web, można skorzystać z dynamicznie ładowanych aplikacji działających po stronie klienta – apletów. Aplet na podstawie różnych parametrów serwerów (obciążenia, odległości itp.) wybiera z nich ten, który zapewni optymalne połączenie [31]. Aplet może zostać umieszczony na dowolnej stronie www, dlatego metoda może być szeroko stosowana.
Ponieważ aplety uruchomione na stacjach klienckich muszą dokonać pomiarów, wykonują połączenia które obciążają sieć i serwery – jest to wada tej metody.
Drugim minusem wszystkich metod opartych na kliencie jest możliwość nadużyć. Oprogramowanie uruchamiane na kliencie jest pod jego kontrolą, więc nieuczciwy użytkownik może je zmodyfikować tak, aby w skrajnym przypadku zamiast rozkładania obciążania, centralizowało je. To może prowadzić do odmowy usług przez serwer (DoS) – czyli defektu, który dotknie także pozostałych użytkowników.

3.8.3. Metoda dynamicznego DNS

Sposób ten polega na skonfigurowaniu serwera DNS (ang. *Domain Name Service*), aby wykonywał translację adresów symbolicznych na adresy IP w sposób dynamiczny [32]. Najpopularniejszym algorytmem jest algorytm karuzelowy (ang. *Round-Robin*). Polega on na tym, że gdy przychodzi zapytanie o translacje to z puli dostępnych serwerów wybierany jest jeden i jego adres IP jest zwracany. Adresy IP są wybierane z puli cyklicznie i stąd nazwa algorytmu.
Metoda ta ma szerokie zastosowanie, ze względu na prostotę implementacji.
Niestety posiada także szereg wad spowodowanych bezwładnością systemu DNS w Internecie. Zapytania DNS są buforowane przez serwery pośredniczące oraz hosty klientów, co powoduje, że raz wybrany IP jest skojarzony z domeną przez czas dłuższy niż życzyłby sobie tego projektant. W przypadku uszkodzenia jednego z serwerów, jego adres jest zbuforowany na dużej liczbie komputerów i następują próby nawiązywania połączeń zakończone niepowodzeniem – występuje defekt. Z tego samego względu rozkładanie obciążenia tą metodą także jest dalekie od perfekcji.

3.8.4. Wirtualne IP

Metoda bazuje na zapewnieniu klientowi dostępu do całego klastra serwerów pod jednym adresem IP. Jest więc najbardziej przezroczysta z wymienionych, ponieważ klient ma złudzenie, że łączy się z jednym serwerem. Dodatkowo wybór serwera który obsłuży danego klienta jest realizowany całkowicie po stronie usługodawcy, klient nie ma możliwości wskazania serwera z którym chce się połączyć. Dzięki temu praktycznie niemożliwe stają się próby „nadwyrężania" wybranych elementów klastra przez nieuczciwych uzytkowników.
Istnieje wiele implementacji wirtualnego IP:

3.8.4.1. TCP Router

Jest to rozwiązanie zaproponowane przez firmę IBM [33, 34], które bazuje na specjalnym urządzeniu, które znajduje się między klastrem serwerów a podłączeniem do sieci zewnętrznej. Urządzenie to ma przypisany adres IP klastra i odbiera pakiety do niego skierowane z Internetu. Po odebraniu każdego pakietu zmienia jego adres docelowy na adres wybranego serwera (metodą sprawdzania obciążenia) w klastrze. Pakiety należące do jednego połączenia TCP mogą być skierowane tylko do tego serwera z którym połączenie to jest nawiązane, dlatego urządzenie utrzymuje w pamięci tablice połączeń TCP – stąd jego nazwa – TCP Router.
Pakiety powrotne, czyli odpowiedzi serwerów skierowane do klientów pomijają TCP Router, ale żeby połączenie przebiegało prawidłowo adres źródłowy połączeń musi być zmieniany na adres routera. Jest to realizowane na każdym z serwerów osobno przez modyfikację stosu TCP/IP (rozwiązanie oryginalne) lub przez alias IP (metoda wprowadzona wraz z Network Dispatcherem firmy IBM [35])

3.8.4.2. NAT

Jest to rozwinięcie poprzedniej metody – tym razem to router zmienia zarówno adresy docelowe pakietów przychodzących jak i adresy źródłowe pakietów wychodzących z klastra. Zaletą w porównaniu z TCP Routerem jest pełna przeźroczystość dla serwerów – nie jest potrzebna ich nietypowa konfiguracja.
Przykładami produktów, które implementują ten sposób rozkładania obciążenia są Cisco Local Director [36], Magic-Router [37], oraz filtry pakietów w systemach operacyjnych Open Source jak Linux [38] i systemy *BSD [20, 39, 40].
Jest to najpopularniejsza metoda, jednak ma dwie wady. Po pierwsze, wymaga specjalnego traktowania protokołów sieciowych, w których adres IP jest przesyłany w warstwie wyższej niż trzecia. Są to np. FTP, H.323. Aby te protokoły działały poprawnie router wykonujący translację adresów musi analizować i zmieniać pole danych (ang. *payload*) pakietów. Może to być realizowane za pomocą proxy w przestrzeni użytkownika (powolne) lub w przestrzeni jądra. Niestety ten proces komplikuje architekturę routera i zmniejsza jego wydajność oraz niezawodność.
Drugą wadą, szczególnie istotną z punktu widzenia tolerowania uszkodzeń jest fakt, że router stanowi pojedynczy punkt awarii. Uszkodzenie routera oznacza powstanie defektu jakim jest niedostępność całego klastra.
Można z tym walczyć przez redundancję sprzętu – skonfigurowanie drugiego routera, który włączy się w przypadku awarii pierwszego. Do tego celu wykorzystuje się np. VRRP lub CARP, które są opisane w innym rozdziale.
Można też wyeliminować router całkowicie, co jest opisane w następnym rozwiązaniu.

3.8.4.3. Podejście anty-centralne [28]

Każdy serwer klastra posiada dwa adresy IP. Jeden z nich to jego wewnętrzny, unikalny adres w klastrze. Drugi jest identyczny dla wszystkich serwerów – jest to wspólny adres IP klastra. Gdy przychodzi pakiet inicjujący połączenie trafia on do wszystkich serwerów naraz. Każdy z nich oblicza funkcję skrótu z adresu źródłowego pakietu i w ten sposób determinuje czy obsłużyć to połączenie, czy je zignorować. Funkcja skrótu mapuje czterobajtowy adres IPv4 na liczbe z zakresu <1,n>, gdzie n jest liczbą serwerów. Wynikiem funkcji skrótu jest numer serwera, który ma obsłużyć połączenie z hostem o danym adresie IP.
Niewątpliwą zaletą tego podejścia jest brak centralnego punktu awarii. W tej metodzie nie ma zaawansowanego routera, który rozdzielałby połączenia. Klaster łączy się z Internetem za pomocą prostego, i tym samym rzadko podlegającego uszkodzeniom routera.
Ważnym elementem jest monitorowanie stanu serwerów. Jeśli jeden z nich ulegnie uszkodzeniu to funkcja skrótu jest przeliczana tak, aby nie wybierała uszkodzonego serwera. Dzięki temu nowe połączenia są przypisywane tylko do działających serwerów. Oczywiście stare połączenia są zrywane, co jest defektem, jednak w większości zastosowań jest akceptowalne. W przypadku usług www opracowano jednak zaawansowane algorytmy, które pozwalają na uniknięcie nawet tych defektów. Jest to opisane w rozdziale „Metody traktowania uszkodzeń w usługach www".

3.9. LVS - Linux Virtual Server

3.9.1. Przegląd systemu

Linux Virtual Server jest projektem, mającym za zadanie dostarczyć szkielet na bazie którego będą mogły być budowane wysoce skalowalne oraz niezawodne usługi sieciowe [48]. Stos TCP/IP systemu operacyjnego Linux, może zostać rozszerzony w celu wsparcia dla trzech różnych techniki rozkładania obciążenia, pozwalając tym samym na zrównoleglenie pracy kilku serwerów świadczących pewien rodzaj usług oraz pojawiających się pod jednym adresem IP. Skalowalność tego rozwiązania polega na łatwym i transparentym dodawaniu oraz usuwaniu węzłów systemu.

Aplikacje klienckie współpracują z tak zaprojektowanym systemem klastrowym jak z pojedynczym serwerem, dlatego też nie wymagają one odrębnej konfiguracji.

3.9.2. Architektura systemu

Architektura systemu LVS składa się z trzech zasadniczych ściśle powiązanych poziomów.

Pierwszym z nich, jest **zarządca ruchu** (ang. *load balancer)*. Jest to podsystem, odpowiedzialny za przechwytywanie przychodzących żądań oraz kierowanie ich do określonego serwera znajdującego się w puli dostępnych serwerów. Decyzja określająca konkretny serwer podejmowana jest na podstawie jednego z dostępnych algorytmów rozkładania obciążenia. Gdy do zarządcy przychodzi pierwszy pakiet sesji należący do danego połączenia, tworzony jest wpis w tablicy połączeń tak, aby pozostałe pakiety tego połączenia mogły być kierowane do tego samego adresu docelowego. Operacje te mają

miejsce na poziomie jądra systemu operacyjnego, dlatego też są wykonywane efektywnie. Zarządca na bieżąco monitoruje poziom dostępności kolejnego poziomu.

Kolejnym poziomem jest **pula dostępnych serwerów** (ang. *server pool*). Na tym poziomie znajduje się dowolna liczba serwerów świadczących określony rodzaj usługi (np. serwisy WWW). Wszystkie serwery posiadają tę samą konfigurację oraz potrafią się wzajemnie zastępować. Serwery na tym poziomie są faktycznym „sercem" systemu, to one świadczą usługi końcowym użytkownikom. Ich konfiguracja zapewnia wysoki poziom dostępności poprzez redundancję zarówno sprzętu jak i oprogramowania. Zarządca ruchu posiada mapę serwerów znajdujących się na tym poziomie. Potrafi wykryć awarię jednego z nich oraz wykreślić go z listy dostępnych serwerów, wykryć nowo podłączony serwer lub też jeden z uszkodzonych, który został naprawiony i może powrócić do normalnej pracy. Bardzo popularne usługi mogą spowodować wystąpienie „wąskiego gardła" co może zaowocować obniżeniem wydajności całego systemu. Gdy zarządca ruch wykryje sytuację w której następuje nasycenie ilości żądań użytkowników, może dołączyć kilka nieaktywnych, zapasowych serwerów rozkładając w ten sposób obciążenie.

Ostatnim poziomem jest **magazyn danych** (ang. *backend storage*) gromadzący wszystkie dane udostępniane przez usługi świadczone przez poprzedni poziom. Takie rozwiązanie, polegające na oddzieleniu faktycznych danych od serwerów przez które są one dostępne umożliwia zapewnienie niezawodności oraz spójności informacji świadczonych przez wszystkie z serwerów. Najczęściej stosuje się do tego celu jeden z rozproszonych, tolerujących uszkodzenia systemów plików takich jak GFS [50], Intermezzo [51] czy Coda [52]. Dzięki takiemu rozwiązaniu, wszystkie serwery widzą system plików tak jak gdyby był on dostępny lokalnie, mając do nich pełny dostęp, nie zdają sobie one sprawy z tego że jest on współdzielony z innymi systemami. Rozproszone systemy plików potrafią wyeliminować konflikty zapisu/odczytu wprowadzane przez jednoczesny dostęp do danych wielu aplikacji serwerowych.

3.9.3. Techniki tolerowania uszkodzeń na poszczególnych poziomach systemu LVS

System LVS zaprojektowany został z myślą o zapewnieniu niezawodności poprzez redundancję sprzętu oraz oprogramowania jako całość. Jednak na poszczególnych poziomach architektury, można podnieść stopień bezpieczeństwa niezależnie od całego środowiska.

3.9.3.1. Heartbeat

Patrząc na ogólną architekturę systemu LVS, można dostrzec że najsłabszym jego elementem jest poziom na którym znajduje się zarządca ruchu. Zasadniczo definiuje się ten poziom jako pojedynczy komputer regulujący obciążenie oraz monitorujący poszczególne serwery. Jednakże również on może ulec awarii, co zaowocuje niedostępnością usług niezależnie od stanu w jakim znajdują się serwery.

Rozwiązaniem, jest wprowadzenie redundancji sprzętu i oprogramowania również na tym właśnie poziomie. Proponowanym rozwiązaniem jest moduł projektu Linux High Availability o nazwie Heartbeat. Nazwa projektu sugeruje że dostępny jest on jedynie dla systemu operacyjnego Linux, jednak został on także z sukcesem przeniesiony na inne platformy

systemowe takie jak Solaris oraz rodzina BSD, co umożliwia stosowanie go heterogenicznych środowiskach.

Heartbeat [49] operuje poniżej warstwy modelu OSI odpowiadającej za sesję aplikacji, dlatego też może zapewniać swoje usługi dla szerokiej gamy serwisów sieciowych. Działanie tego systemu, opera się na wzajemnym monitorowaniu aktywności oraz dostępności dwóch komputerów. Połączone ze sobą przez jeden z wielu obsługiwanych interfejsów takich jak łącza szeregowe, ethernet, linie telefoniczne. Oprogramowanie zainstalowane na dwóch systemach wysyła wzajemnie informacje o swojej dostępności. Zasada działania podobna jest do systemów typu „watchdog", jeśli do zapasowego systemu w określonym czasie nie napłynie potwierdzenie o dostępności drugiej strony połączenia, przejmuje on jego adres IP oraz rozpoczyna pracę w charakterze systemu podstawowego. W celu zapewnienia większej niezawodności połączenia sieciowego pomiędzy tymi dwoma systemami zaleca się użycie dwóch lub więcej połączeń sieciowych pomiędzy nimi.

Dzięki swej niezależności od aplikacji uruchamianej na serwerach, Heartbeat jest szeroko stosowany w wielu dziedzinach usług sieciowych takich jak serwery poczty, serwisy WWW, serwery nazw DNS, jednym z miejsc gdzie znakomicie może spełniać swoją rolę jest poziom zarządzania ruchem w systemie LVS.

3.9.3.2. Poziom puli serwerów

Dzięki redundancji zarówno sprzętu jak i oprogramowania serwerów świadczących usługi końcowym użytkownikom można zapewnić wysoki poziom niezawodności oraz dostępności usług. W dzisiejszych czasach gdy ruch w sieciach komputerowych stale wzrasta należy rozszerzyć znaczenia pojęcia „awaria". Nie powinno się ono odnosić jedynie do sytuacji, w której unieruchomieniu ulega punkt świadczący usługę, lecz powinno także obejmować wszystkie sytuacje gdy wydajność spada poniżej pewnego określonego punktu krytycznego. Rozwiązanie polegające na zwiększeniu wydajności poprzez zrównoleglenie pracy jest po pierwsze wydajne, po drugie zwiększa niezawodność całości systemu, po trzecie tanie, ponieważ koszt kilku - kilkunastu przeciętnej wydajności serwerów jest niższy od ceny jaką należałoby przeznaczyć na kupno jednego superkomputera.

3.9.3.3. Magazyn danych

Zastosowanie wydzielonego poziomu architektury na magazyn danych niesie ze sobą wiele korzyści. Pierwszą z nich jest zapewnienie spójności informacji udostępnianych przez serwery usług sieciowych. Niezależnie od ilości serwerów, będą one miały dostęp zawsze do tych samych danych. Kolejną korzyścią jest wydzielenie następnego elementu całego systemu w którym mogą występować awarie. Dekompozycja jednego problemu na kilka mniejszych, zawsze przynosi wymierne skutki w postaci uniezależnienia od siebie poszczególnych poziomów. Scentralizowany magazyn, w przeciwieństwie do kopii tych samych danych na każdym z serwerów, powoduje także oszczędność przestrzeni dyskowej co ma duże znaczenie przy większej ilości serwerów.

Jednym z podejść stosowanych w celu zapewnienia niezawodnego, zcentralizowanego magazynu danych są rozproszone systemy plików takie jak Coda [52], InterMezzo [51] oraz OpenGFS [50]. Umożliwiają one wydajny oraz niezawodny dostęp do informacji wielu

systemom tak, jak gdyby system plików dostępny był lokalnie. Ponadto, systemy te posiadają mechanizmy chroniące przed konfliktami odczytu/zapisu przez kilka aplikacji w tym samym czasie oraz zwiększające wydajność poprzez stosowanie lokalnego magazynu danych podręcznych (ang. *cache*). Kolejnym z rozwiązań, jest umieszczenie wszystkich potrzebnych danych w transakcyjnej bazie danych. Obecne systemy baz danych mogą przechowywać dowolne typy obiektów, włączając w to dowolne pliki binarne, mogą również tworzyć wspólnie z innymi bazami danych jeden rozproszony magazyn.

3.9.4. Techniki sterowania ruchem

W Linuksie, sterowanie przepływem informacji na poziomie zarządcy ruchu, może być osiągnięte na trzy sposoby (LVS/NAT, LVS/TUN, LVS/DR), posiadające swoje wady i zalety, które przedstawione zostaną w poniższych sekcjach.

3.9.4.1. Translacja adresów sieciowych

Translacja adresów sieciowych NAT (ang. *Network Address Translation*) to technika polegająca na modyfikowaniu, w zależności od potrzeb, adresów źródłowych/docelowych, w protokole IP w celu uzyskania zamierzonego efektu [38]. Jednym z nich jest główny cel, dla którego powstał NAT, polegający na wprowadzeniu oszczędności w przydzielaniu adresów IP instytucjom publicznym oraz prywatnym.

W systemie LVS, zastosowanie tej techniki, polega na przypisaniu jednego, publicznego adresu IP systemowi zarządcy ruchu. Wszystkie serwery, znajdujące się na kolejnym poziomie, posiadają adresy IP prywatne wewnętrznej sieci w której się znajdują, niedostępne dla świata zewnętrznego (takiego jak chociażby Internet). Klienci kontaktują się z adresem IP zarządcy ruchu, ten zaś podmienia adres docelowy pakietów IP na adres jednego z obsługiwanych przez siebie serwerów oraz odsyła pakiet właśnie do niego. Dzięki takiemu rozwiązaniu, zarówno klient jak i serwer mają wrażenie sytuacji, w której komunikują się ze sobą bez pośrednika w postaci zarządcy ruchu, dzięki czemu nie wymaga się od nich dodatkowej konfiguracji. Gdy pierwszy pakiet danego połączenia pojawia się w systemie zarządcy ruchu, tworzony zostaje specjalny wpis w pamięci, informujący przyszłe pakiety należące do tego połączenia gdzie mają być kierowane. Pakiet odsyłany przez serwer do klienta ponownie trafia do maszyny wykonującej translację, która tym razem podmienia adres źródłowy pakietu będący adresem prywatnym serwera na swój własny, publiczny adres IP.

Takie rozwiązanie jest proste oraz skuteczne, jednak jego słabym punktem jest konieczność modyfikowania pakietów przez zarządcę oraz przepisywanie go pomiędzy interfejsami zarówno dla ruch kierowanego do serwera jak i dla ruchu będącego odpowiedziami do klientów. Dla większej ilości serwerów (ponad 20), może to spowodować znaczny spadek wydajności, jednak jest to również zależne od wydajności systemu wykonującego translację.

3.9.4.2. Tunele IP

Tunelowanie IP [53] jest techniką polegającą na umieszczaniu pakietów IP w innych pakietach IP, dzięki czemu pakiety przeznaczone docelowo do określonego adresu IP mogą zostać opakowane i przesłane w pierwszej fazie do innego IP. Dzięki takiemu rozwiązaniu,

zarządca ruchu może opakować żądania klientów do jednego wybranych serwerów oraz przekazać je do serwera, który może posiadać publiczny adres IP i tym samym być zdolnym do bezpośredniego wysłania odpowiedz do klienta.

Minusem tej metody jest konieczność dodatkowej konfiguracji każdego z serwerów końcowych oraz konieczność wsparcia na danej platformie systemowej dla tunelowania IP. Na dzień dzisiejszy jest to jednak na tyle powszechna technologia, że wyposażone są w nią wszystkie popularne sieciowe systemy operacyjne. Zaletą tej techniki w przeciwieństwie do NAT, jest zmniejszenie obciążenia zarządcy ruchu, dzięki temu że serwery mogą wysyłać odpowiedzi do klientów bezpośrednio.

3.9.4.3. Routing bezpośredni

Routing bezpośredni DR (ang. *direct routing*) jest technologią zaczerpniętą z rozwiązania NetDispatcher firmy IBM. Jej działanie, polega na połączeniu serwerów oraz zarządcy w jedną sieć wewnętrzną, przypisaniu serwerom określonego, tego samego adresu IP oraz przekierowywaniu ruchu od zarządcy bezpośrednio do adresu sieciowego MAC (ang. *Media Access Control*) wybranego serwera. Serwer który odbierze pakiet będzie mógł odpowiedzieć klientowi bez pośrednictwa zarządcy. Wadą rozwiązania jest konieczność konfiguracji serwerów tak, aby nie wysyłały odpowiedzi ARP prowadzących do konfliktu adresów MAC/IP, nie każdy system operacyjny pozwala na taką konfigurację.

3.9.5. Techniki rozkładania obciążenia

Zakładając że wszystkie serwery działają poprawnie i żaden nie wymaga wykreślenia z puli sprawnych serwerów w tablicy zarządcy ruchu, zarządca potrafi rozdzielać ruch tak, aby równomiernie rozkładać obciążenie pomiędzy serwerami.

3.9.5.1. Algorytm karuzelowy (ang. *Round – Robin*)

Algorytm ten, traktuje wszystkie serwery równo, przypisując kolejne nowe, nadchodzące żądania do kolejnych serwerów.

3.9.5.2. Algorytm wagowo – karuzelowy

Algorytm ten działa podobnie do poprzedniego, potrafi jednakże traktować każdy serwer inaczej, w zależności np. od jego wydajności. Każdemu z serwerów mogą zostać przypisane wagi tak, aby serwery o większych wagach, były wykorzystywane częściej od tych o wagach o mniejszych wartościach.

3.9.5.3. Algorytm „Least – Connection”

Algorytm ten, jako następny serwer który powinien zostać wyznaczony do obsługi połączenia, wybiera ten z najmniejszą liczbą aktywnych połączeń. Na pierwszy rzut oka może on wyglądać obiecująco, tak jednak nie jest, ze względu na stan TIME_WAIT, w którym znajduje się każde zakończone już połączenie TCP przez okres ok. 2 minut.

3.9.5.4. Algorytm wagowy „Least – Connection”

Algorytm działa podobnie do poprzedniego, potrafi jednak obsługiwać serwery w zależności od przypisanych im wag, tak jak było to w przypadku algorytmu wagowo – karuzelowego.

3.10. Metody traktowania uszkodzeń w usługach www

Usługi świadczone przez serwery WWW zaczynają odgrywać w życiu każdego użytkownika Internetu coraz większą rolę. Operacje przez nie wykonywane nie ograniczają się już tylko do wyświetlania informacji w postaci statycznych stron. Coraz częściej mamy do czynienia z serwisami umożliwiającymi dokonywania zakupów, różnego rodzaju opłat oraz innych operacji o krytycznym znaczeniu, drogą elektroniczną.

W poważnych środowiskach w których dba się o niezawodność, coraz częściej stosowane są systemu klastrowe, w których po awarii jednego z elementów systemu, jego rolę może przejąć element zapasowy, odpowiednio skonfigurowany aby pełnił rolę podstawowej jednostki. Tego typu podejście do problemu awarii nie jest jednak doskonałe, ponieważ wszystkie żądania od użytkowników, których sesja została rozpoczęta zostają anulowane i żądania pochodzące od klientów powinny zostać przez nich ponowione tak, aby zapasowy serwer mógł je przetworzyć.

Idealnym rozwiązaniem byłoby zapewnienie transparentnej dla końcowego użytkownika obsługi uszkodzeń oraz zapewnienie ciągłości jego sesji. Tak jak w wielu rozwiązaniach, w których krytyczne jest zapewnienie ciągłości działania usługi, kluczową rolę odgrywa redundancja sprzętu i oprogramowania.

Rozwiązanie opiera się na zastosowaniu zapasowego serwera (lub kilku serwerów) pracujących równolegle z serwerem podstawowym oraz pośrednika pomiędzy użytkownikiem a serwerami – tzw. dystrybutora [7]. Dystrybutor kontroluje całą sesję użytkownika i na podstawie jej przebiegu określa docelowe serwery do których mają zostać skierowane żądania. Istotnym fundamentem na bazie którego opiera się idea, jest zastosowanie w warstwie transportowej protokołu stanowego jakim jest TCP. Przed rozpoczęciem właściwej pracy, dystrybutor tworzy kilka zapasowych sesji z obsługiwanymi przez siebie serwerami.

Gdy użytkownik rozpoczyna wysyłanie żądania do serwera, najpierw musi zostać utworzona sesja TCP. Użytkownik nie nawiązuje jednak połączenia bezpośrednio z serwerem WWW, a jedynie z dystrybutorem, który w tym czasie tworzy w swojej pamięci odpowiedni wpis identyfikujący sesję użytkownika na podstawie znaczników czasowych, adresu i portu źródłowego z którego została zainicjowana sesja.

Po zainicjowaniu sesji TCP, oprogramowanie działające po stronie użytkownika wysyła żądanie HTTP dotyczące informacji, które mają zostać zwrócone przez oprogramowanie serwera. Dystrybutor posiada własną strukturę w pamięci zawierającą informacje o położeniu dokumentów, ich rozmiarach, priorytetach oraz typie. Na podstawie tych informacji oraz informacji o żądaniu wysłanym przez klienta, dystrybutor może skierować zapytanie do odpowiedniego serwera. Po określeniu serwera, dystrybutor przystępuje do przekazywania pakietów od użytkownika do wybranego serwera modyfikując nagłówek IP oraz TCP [1] tak, aby komunikacja pomiędzy użytkownikiem a docelowym serwerem odbywała się przezroczyście, pomimo istniejącego pomiędzy dwoma końcami połączenia dystrybutora. W zaproponowanym rozwiązaniu, te funkcje dystrybutora zaimplementowane zostały jako moduł jądra w systemie Linux. Moduł operuje w warstwie znajdującej się pomiędzy interfejsem sieciowym a stosem TCP/IP tak aby dystrybutor mógł samodzielnie analizować pakiety i decydować o ich routingu [1].

Po nawiązaniu połączenia TCP, żądania wysyłane do serwerów WWW podzielić można na 3 kategorie: zapytania statyczne, dynamiczne oraz sesje. Każda z nich wymaga innego podejścia, dlatego też zostaną opisane oddzielnie.

3.10.1. Żądania statyczne

W przypadku żądań dotyczących obiektów statycznych, takich jak dokumenty HTML, obrazy oraz inne pliki które pozostają niezmiennie w czasie, problem pojawiający się w momencie uszkodzenia serwera jest najprostszy do rozwiązania. W pierwszym kroku po wykryciu takiej sytuacji, dystrybutor wybiera zapasowy serwer, który przejmie rolę podstawowego oraz wiąże połączenie TCP użytkownika z nowym serwerem oraz jedną z utworzonych z nim wcześniej sesji. Po rekonfiguracji połączenia, dystrybutor sprawdza w swojej pamięci podręcznej informacje dotyczące nawiązywanego ponownie połączenia oraz na podstawie takich danych jak numery sekwencyjne pakietów TCP określa ile danych zostało przesłanych do użytkownika. Dzięki tym informacjom, do zapasowego serwera może zostać przesłane żądanie typu *range request* zdefiniowane w protokole HTTP 1.1, umożliwiające klientowi odebranie z serwera części pliku. Dzięki temu, po przejęciu przez serwer zapasowy roli podstawowego, dystrybutor może zażądać wysłania do użytkownika części, nie pobranej jeszcze przez niego żądanego dokumentu.

3.10.2. Żądania dynamiczne

Żądania dynamiczne, wysyłane przez użytkownika, powodują wygenerowanie przez serwer odpowiedzi zależnej od nadesłanych przez użytkownika parametrów. Z tego właśnie powodu nie można jednoznacznie określić rozmiaru plików jakie będą przesyłane do użytkownika, co więcej, często zdarza się że żądanie wysłane dwukrotnie z takimi samymi parametrami spowoduje wygenerowanie dwóch różnych odpowiedzi, taka sytuacja ma miejsce zazwyczaj gdy przekazywane dane pobierane są z bazy danych.

Podejście zaproponowane w tej sytuacji opiera się na przechwyceniu zapytanie przez dystrybutora, a następnie przechwyceniu odpowiedzi serwera. Wygenerowana odpowiedź, zwrócona zostanie do oprogramowania użytkownika dopiero wtedy, gdy serwer przetworzy całość żądania oraz prześle do dystrybutora pełną odpowiedź. Dopiero wtedy, po uzyskaniu

kompletnych danych, dystrybutor może przekazać je użytkownikowi. W ten sposób, jeśli serwer ulegnie uszkodzeniu w trakcie przetwarzania żądania i wysyłania odpowiedzi, żądanie może zostać ponowione na alternatywnym serwerze.

3.10.3. Żądanie sesyjne

Podczas sesji, użytkownik nie wysyła żądań pojedynczych stron statycznych czy też dynamicznych, jego zachowanie oraz możliwość interakcji jest kontrolowana przez oprogramowanie znajdujące się na serwerze (np. skrypt CGI), tworząc w ten sposób pewnego rodzaju maszynę stanową. Wszystkie akcje podejmowane przez użytkownika, uruchamiane są w jednej instancji do której tylko on ma dostęp i która dotyczy wyłącznie jego (jest unikalna na serwerze).

Tego typu usługi, wiążą najczęściej ze sobą trzy poziomy – końcowy poziom użytkownika, będący powiązany np. z jego przeglądarką, poziom na którym operuje serwer WWW oraz poziom bazy danych na podstawie których serwer WWW generuje odpowiedzi i odsyła je do użytkownika.

Podczas awarii środkowego ogniwa systemu jakim jest serwer WWW, użytkownik nie ma pewności czy jego działania i operacje zostały zatwierdzone w bazie danych. Odnawianie sesji w systemie o tej architekturze jest największym problemem z dotychczas przedstawionych. Wymagane są do tego specyficzne informacje takie jak dane o momencie w którym została rozpoczęta sesja, wszystkich stanach, parametrach oraz zakończeniu sesji.

Gdy sesja zostaje rozpoczęta, dystrybutor rozpoczyna przekazywanie danych jednocześnie do pary serwerów „bliźniaczych”. Para serwerów składa się z serwera głównego oraz zapasowego. Zapasowy, posiada dwa przypisane adresy IP – własny oraz adres IP serwera podstawowego, nadany jako alias jego interfejsu sieciowego (nie może jednak zostać on rozsyłany przez protokół ARP aby nie doprowadzić do konfliktu adresów IP [1]). Dzięki takiej konfiguracji, wszystkie pakiety mające trafić do serwera podstawowego, odebrane zostaną także przez serwer zapasowy. W wyniku takich operacji na dwóch serwerach utworzona zostaje sesja oraz jest ona w obu przypadkach zawsze w tym samym stanie.

Konfiguracja serwera zapasowego, musi jednak uwzględniać kilka ograniczeń:

- Serwer zapasowy, gdy pracuje w swoim normalnym trybie, nie wysyła odpowiedzi do użytkownika
- Serwer zapasowy potwierdza do podstawowego odebranie przez siebie i zapisanie zmian w sesji
- Zmiany w bazie danych dokonywane są wyłącznie przez serwer podstawowy
- Serwer zapasowy, dostaje jedynie potwierdzenia od bazy danych informujące o tym, że transakcja została wykonana poprawnie oraz wysyła do serwera podstawowego potwierdzenie zapisania przez siebie transakcji

Dopiero w tym momencie, serwer podstawowy ma pewność że transakcja została zapisana w bazie danych oraz że serwer zapasowy posiada wszystkie informacje na jej temat. Teraz może nastąpić odesłanie odpowiedzi do użytkownika. Odesłanie przez użytkownika potwierdzenia odebrania danych oraz zakończenia sesji powoduje zwolnienie zasobów z serwera zapasowego. Gdy w trakcie transakcji uszkodzeniu ulegnie serwer podstawowy,

serwer zapasowy w pełni aktywuje aliasowany przez siebie adres IP serwera podstawowego oraz przejmuje jego rolę w komunikacji z użytkownikiem oraz serwerem bazy danych. Takie podejście pozwala na transparentną współpracę z użytkownikiem końcowym a także serwerem bazy danych.

Analiza oraz implementacja tych rozwiązań jest czasochłonna, wymaga bowiem implementacji przekierowywania oraz analizowania pakietów przez dystrybutor, a także dodatkowego modułu w serwerach WWW zapasowym oraz podstawowym tak, aby wiedziały one w jaki sposób ze sobą współpracować. Zaproponowane rozwiązania, budzą jednak nadzieje swojej poprawnej pracy oraz wysokiej wydajności w obsługiwaniu żądań użytkowników wymagających dużej niezawodności całego sytemu.

3.11. LODES

Wykrywanie uszkodzeń w złożonych sieciach komputerowych jest skomplikowanym zadaniem. LODES (ang. *Large-internet observation and Diagnostic Expert System*) [19] to rozproszony system oparty na sztucznej inteligencji, który diagnozuje problemy w sieci w sposób automatyczny i/lub z pomocą człowieka.

3.11.1. Opis systemu

System składa się z agentów, czyli hostów, na których działają kopie aplikacji LODES. Każdy agent obsługuje jeden segment sieci. Segment sieci to taki jej obszar, którego ruch może być analizowany z jednego miejsca. Segmentem może być duża sieć oparta na tradycyjnych hubach ethernetowych, ponieważ stanowi ona pojedynczą magistralę, do której dostęp ma każdy host, czyli również agent LODES. W przypadku ethernetu przełączanego segmentem będzie sieć obejmowana przez przełącznik ethernetowy z portem monitorującym, do którego będzie wpięty agent LODES.
Każdy agent monitoruje sieć (głównie w sposób pasywny, analizując pakiety). Na podstawie tej analizy buduje swój model otoczenia. Np. korzystając z informacji zawartych w nagłówkach pakietów IP agent jest w stanie sprawdzić jaka adresacja jest używana w sieci. Monitorując pakiety UDP, których port docelowy jest ustawiony na wartość 53 agent może sprawdzić pod jakim adresem znajduje się serwer DNS. To samo dotyczy większości ustawień sieci. Jest to ciekawa właściwość systemu, ponieważ dzięki temu nie jest wymagana jego szczegółowa konfiguracja przez człowieka.
Monitorowanie obejmuje także działania aktywne. Jest to głównie odpytywanie przełączników i routerów z wykorzystaniem protokołu SNMP (ang. *Simple Network Management Protocol*) - np. w celu utworzenia statystyk ruchu sieciowego, lub wysyłanie diagnostycznych pakietów ICMP echo-request.
Agenty komunikują się ze sobą w przypadku, gdy do rozwiązania jest problem wykraczający poza segment, lub problem jednego segmentu objawia się w innym.
Jeśli podczas monitorowania zostanie wykryta nieprawidłowość w działaniu sieci (np. gwałtownie zmienią się charakterystyki ruchu, zostaną wykryte błędne pakiety itd.), lub użytkownik zgłosi defekt - agent przystępuje do diagnozy problemu.
W tym celu przeprowadza analizę modelu sieci, który utrzymuje w pamięci. Na jego podstawie formułuje hipotezy, które szereguje według współczynnika pewności. Współczynnik ten jest wyliczany na podstawie zmiennych takich jak: prawdopodobieństwo problemu (zmienna jest określana na podstawie wcześniejszych doświadczeń agenta), ważność problemu, koszt weryfikacji hipotezy, liczba warunków potrzebnych do jej spełnienia.

Każda hipoteza składa się z warunków oraz werdyktu. Warunki mogą być sprawdzane w modelu sieci, przez testowanie sieci lub przez analizę werdyktu innej hipotezy.
Uszeregowane hipotezy są kolejno weryfikowane aż do znalezienia hipotezy która staje się faktem.

3.11.2. Przykład działania

Aby lepiej zobrazować działanie systemu LODES prześledźmy jak zachowa się on w przypadku wystąpienia konkretnego problemu.

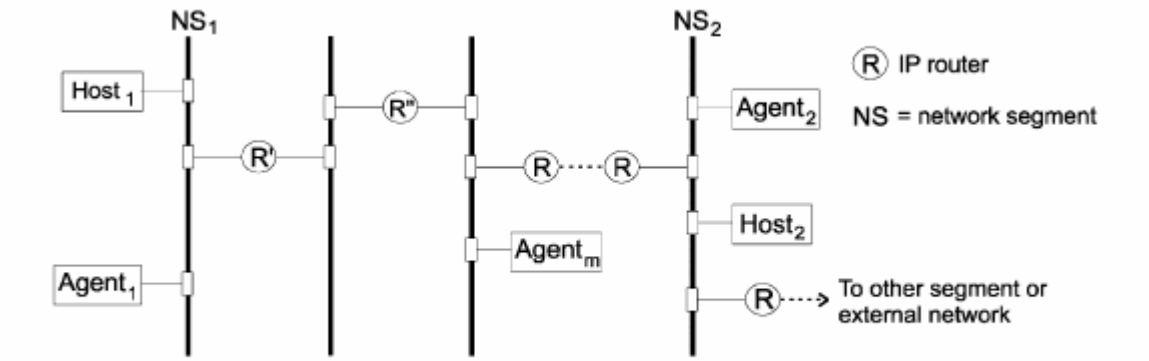

Fig. 1. An example of ISR problem.

Powyższy diagram przedstawia sieć komputerową. NS1 i NS2 to segmenty obsługiwane odpowiednia przez agenty: Agent1 i Agent2. R to routery, które służą do zapewnienia łączność między segmentami.
Użytkownik Hosta1 zgłasza do systemu defekt, który polega na tym, że nie może nawiązać połączenia telnet z Hostem2.
Rzeczywistą przyczyną defektu jest błędna konfiguracja Hosta2 polegająca na tym, że posiada on niekompletną tablicę routingu, w której brak jest trasy do Hosta1. Poniżej przedstawiamy jak do tego wniosku dochodzi system.
Po otrzymaniu zgłoszenia Agent1 rozpoczyna diagnozę problemu. Ponieważ to może nie być problem lokalnego segmentu, Agent1 wysyła rozkaz diagnostyczny do Agenta2.
Agent1 nie może wskazać objawów problemu innych niż ten, że Host2 nie odpowiada na pakiety testowe. Dlatego Agent1 zaczyna podejrzewać, że problem może być nie związany z jego segmentem. Wszystkie hipotezy, które może sformułować cechuje niski współczynnik pewności. Agent1 może formułować hipotezy związane z problemami z innymi segmentami, jednak te hipotezy są odrzucane, ze względu na to, że mogą być zweryfikowane przez inne agenty niższym kosztem.
Agent2 także nie widzi objawów problemu; Host2 odpowiada na wszystkie pakiety diagnostyczne wysyłane do niego przez Agenta2. Jednak otrzymanie sygnału o defekcie od Agenta1 powoduje, że Agent2 wysnuwa kilka hipotez. Dwie z nich mają wysoki współczynnik pewności: hipoteza H1 – problem protokołu warstwy czwartej lub aplikacji, oraz hipoteza H2 – intensywny ruch (np. chwilowe przeciążenie) na odcinku łączącym Host1 z Hostem2, które może prowadzić do tymczasowego zaniku komunikacji między tymi hostami. Aby zweryfikować H2 Agent2 przeszukuje swoje lokalne archiwum statystyk ruchu i nie znajduje w nim symptomów przeciążenia sieci, odrzuca więc tą hipotezę. Aby sprawdzić H1 Agent2 próbuje nawiązać połączenie telnet z Hostem2. Jako, że połączenie udaje się pozostaje przypuszczenie, że połączenie telnet nie jest możliwe tylko wtedy, kiedy uczestniczą w nim Host1 i Host2. Jednak weryfikacja tej hipotezy jest kosztowna, ze względu na to, że musi w niej uczestniczyć człowiek, dlatego jest tymczasowo wstrzymywana.
Ponieważ oba agenty mają tylko hipotezy o niskim współczynniku pewności przechodzą do fazy współpracy. Wymieniają ze sobą hipotezy, jednak nie powoduje to zwiększenia

współczynika pewności żadnej z nich. Zaczynają rozpatrywać hipotezy o niższym priotytecie związane z problemami w warstwie IP. Agent1 rozpatruje hipotezę No-Routing-Data-In-Local-Host (brak danych o routingu w konfiguracji hosta) i ponieważ część warunków tej hipotezy jest spełnionych zwiększa jej współczynnik pewności. Wysyła tą hipotezę do Agenta2, który przystępuje do jej weryfikacji. W tym celu agenty wykonują wspólnie operację Sync-Ping-Observation (synchronicza obserwacja komunikacji diagnostycznej ICMP). Operacja polega na tym, że Agent1 wysyła pewną liczbę pakietów ICMP echo-request do Hosta2, zaś Agent2 obserwuje te pakiety oraz odpowiedzi na nie. Ponieważ Host2 nie odpowiada, może to zwiększyć pewność hipotez H4 (uszkodzenie interfejsu hosta) oraz H5 (host działa w trybie pojedynczego użytkownika). Jednak te hipotezy są odrzucane, ze względu na poprzednie testy. Zostaje tylko hipoteza No-Routing-Data-In-Local-Host, której współczynnik pewności jest zwiększany i osiąga wartość największą spośród hipotez pozostałych. Hipoteza ta wygrywa i problem zostaje poprawnie zdiagnozowany.

3.11.3. Wnioski

Niewątpliwą zaletą systemu LODES jest to, że konfigurując nowy host w sieci komputerowej, lub dokonując zmian w konfiguracji wystarczy przeprowadzić jeden test. Test polega na sprawdzeniu, czy host ma łączność z systemem LODES. Większość innych błędów będzie natychmiast zauważona przez system, więc nie powstaną defekty. Jeśli błąd nie zostanie zauważony automatycznie, wystarczy zgłosić jego symptomy do systemu aby uzyskać wskazówkę jak usunąć defekt.
Wszystkie agenty są identyczne pod względem programowym, różnią się tylko nieznacznie konfiguracją (np. adresem IP). Dzięki temu system zachowuje się prawidłowo nawet po wystąpieniu uszkodzenia na trasie komunikacji między agentami, ponieważ każdy z nich jest w pełni autonomiczny.

3.12. Wielowarstwowe tolerowanie uszkodzeń w szkieletowych sieciach teletransmisyjnych

Dzisiejsze teletransmisyjne sieci szkieletowe są z reguły budowane na bazie trzech warstw. Są to od najniższej:

1. Warstwa optyczna WDM (ang. *Wave Division Multiplexing*), która wykorzystuje jako medium światłowody.
2. Warstwa SDH (ang. *Synchronous Digital Hierarchy*).
3. Warstwa ATM (ang. *Asynchronous Transfer Mode*)

Aktualnym trendem jest scalanie wielu usług w jedną sieć, w celu obniżenia kosztów. I tak na warstwę ATM nakłada się równolegle warstwy IP, telefonii, telewizji i inne.
Uszkodzenia mogą występować w każdej z warstw, jednak powstałe błędy powodują najbardziej odczuwalne defekty, gdy uszkodzenie wystąpi w jednej z trzech najniższych. Defekty są poważne, ponieważ przestają działać wszystkie usługi powyżej warstwy ATM. Uszkodzenie które w klasycznych, rozdzielnych sieciach blokowało jedną usługę, w sieciach zintegrowanych zatrzymuje wszystkie. Dlatego rozważania w tym rozdziale ograniczymy do trzech najniższych warstw.

Każda warstwa posiada mechanizmy tolerowania uszkodzeń, dzięki którym dostępność jest utrzymana na pewnym poziomie. Dla różnych warstw przedstawia się ona następująco [5]:

Warstwa	Dostępność [%]
ATM VC	99,6
ATM VP	99,6
SDH VC-12	99,7
SDH VC-4	99,9
WDM	99,99

Dostępność wszystkich trzech warstw można zwiększyć przez odpowiednią strategię. Polega ona na tym, że łączy się mechanizmy tolerowania i traktowania uszkodzeń poszczególnych warstw w jeden współpracujący system. Można wyróżnić następujące strategie [6]:

1. Od dołu do góry (ang. *bottom-up*).
2. Od góry do dołu (ang. *up-bottom*).
3. Diagnostyczna.

Strategia „od dołu do góry" polega na próbie naprawy uszkodzenia przez warstwę pierwszą. W przypadku, gdy ta warstwa nie poradzi sobie z uszkodzeniem w określonym czasie, zadanie przejmuje warstwa druga, a później trzecia. I tak np. uszkodzenie światłowodu (przecięcie kabla) „zauważa" warstwa WDM. Jest to uszkodzenie nienaprawialne w tej warstwie, tak więc odpowiedzialność za nie przechodzi na wyższe warstwy. Te maskują uszkodzenie przez wybranie alternatywnej ścieżki w grafie połączeń sieci.
Strategia „od góry do dołu" zaczyna diagnozowanie i naprawianie uszkodzenia od warstwy najwyższej. Jeśli problem nie znika, to zajmują się nim kolejne, niższe warstwy.
Strategia diagnostyczna polega na rozpoczęciu usuwania awarii w warstwie, która jest odpowiedzialna za najczęstsze uszkodzenia. Jest to strategia adaptacyjna i warstwa początkowa jest ustalana przez algorytm indywidualnie dla każdej sieci. Dodatkowo, kolejność warstw może się zmieniać w czasie, jeśli model uszkodzeń nie będzie stały.
Wymienione strategie należą do klasy strategii sekwencyjnych. Innym podejściem jest traktowanie uszkodzeń w wielu warstwach jednocześnie. Jest to sposób na najszybsze usunięcie problemu powstałego na skutek uszkodzenia, jednak jest też najbardziej kosztowny. Procedury naprawy są uruchamiane we wszystkich warstwach jednocześnie, choć uszkodzenie najczęściej występuje tylko w jednej z nich.

4. Podsumowanie i wnioski

W naszej pracy przedstawiliśmy część rozwiązań stosowanych w sieciach komputerowych w celu zwiększenia niezawodności poprzez tolerowanie uszkodzeń. Można zastanawiać się na poziomem niezawodności jaki jest akceptowalny w tej dziedzinie. Wydaje się, niezawodność na poziomie 99% jest wartością wysoką. Jednak w skali roku, oznaczałoby to prawie 4 dni przerwy w działaniu sieci. Idąc dalej, można założyć że wartość 99,999% jest wystarczająca. Po przeliczeniu, okazuje się jednak, że oznacza to prawie 9 godzin awarii w przeciągu roku. Dla niektórych zastosowań jest to wartość zadowalająca, jednak w przypadku tak krytycznych zastosowań jak telemedycyna jest niedopuszczalną. Motywuje to naukowców do ciągłej pracy nad zwiększaniem dostępności sieci i systemów komputerowych.

Istotnym problemem występującym w dzisiejszych sieciach komputerowych są kolizje. Jedną z technik umożliwiającą uniknięcie tego zjawiska jest protokół STP. Działa on poniżej warstwy protokołu IP, jest więc od niego niezależny przez co nie jest ograniczony tylko do niego. Opracowany został i ustandaryzowany przez niezależną organizację jaką jest IEEE oraz implementowany jest w wielu urządzeniach sieciowych. Pozwala on na poprawną pracę sieci o skomplikowanych topologiach, wyznaczając optymalne ścieżki łączące kolejne urządzenia sieciowe oraz udostępnia możliwość szybkiego i automatycznego wyznaczenia ścieżki alternatywnej, powiadamiając o niej wszystkie urządzenia sieciowe. Takie rozwiązanie pozwala nie tylko tolerować, ale również zapobiegać ewentualnym uszkodzeniom. Rozwiązanie to polega na kontrolowaniu redundancji, która w tym przypadku dotyczy ścieżek łączących urządzenia sieciowe. Zapasowe ścieżki początkowo wykluczane są z użycia, dopiero gdy zajdzie potrzeba ich wykorzystania, węzeł zarządzający uaktywnia je informując o tym pozostałe węzły.

Dzisiejsze usługi świadczone poprzez sieci komputerowe takie jak telemedycyna czy też operacje finansowe wymagają maksymalnej niezawodności oraz jakości usług, co oznacza konieczność zapewnienia określonej wydajności połączenia sieciowego. Takie możliwości zapewnia MPLS, który jest zorientowanym połączeniowo protokołem potrafiącym zapewnić żądaną przez aplikację jakość usługi nawet w przypadku awarii ścieżki po której odbywa się przesyłanie ramek z danymi. Jeśli nie jest możliwe zapewnienie określonej jakości usługi, urządzenia sieciowe potrafią zestawić połączenie o jakości najbardziej zbliżonej do wymaganych parametrów. Wykrycie awarii w tradycyjnych sieciach IP może trwać nawet kilka minut, w sieciach MPLS jest to sprawa priorytetowa, dlatego też typowe wartości czasowe dla takich sytuacji są rzędu dziesiątek milisekund. Umożliwia to błyskawiczną reakcję oraz przywrócenie sieci jej poprawnego działania.

Najpopularniejszy w dzisiejszym świecie Internetu protokół IP posiada wiele niedoskonałości, które stara się zniwelować protokół wyższej warstwy jakim jest TCP. Algorytmy retransmisji oraz wykrywania awarii połączenia pomiędzy dwoma węzłami, które są w nim stosowane, pozwalają zapewnić skuteczną wymianę informacji, jednak nie w przypadku usług wymagających szybkich czasów reakcji na błędy oraz wysokiej jakości. Jest jednak protokołem bardzo popularnym i potrafi zapewnić zwykłym użytkownikom akceptowalny poziom jakości.

Projektanci aplikacji zastanawiają się również w jaki sposób zapewnić jakość oraz niezawodność swoich usług na poziomie aplikacji oraz sesji. Jedną z przedstawionych przez nas metod jest wprowadzenie klastra serwerów usług sieciowych (takich jak serwisy WWW) oraz systemu zarządzającego klastrem. Pierwsze rozwiązanie, polegające na wysyłaniu żądań

użytkowników do całego klastra oraz odsyłaniu odpowiedzi tylko od jednego, aktywnego w danej chwili systemu umożliwia zapamiętanie i odtworzenie stanu sesji w jakim znajduje się połączenie z użytkownikiem. Kolejne rozwiązanie jakim jest LVS, jest podejściem bardziej uniwersalnym i może dotyczyć dowolnej usługi sieciowej. Dodatkowe zastosowanie takich rozwiązań jak Heartbeat oraz rozproszone systemy plików zwiększają poziom bezpieczeństwa całego systemu. Kolejnym atutem rozwiązania LVS jest fakt, że klaster serwerów świadczących określony rodzaj usługi, może pracować pod kontrolą dowolnego systemu operacyjnego na dowolnej platformie sprzętowej.

Opisane w naszej pracy rozwiązania cechują różne typy redundancji. W większości metod duplikowane są kompletne komputery (np. VRRP/CARP, systemy klastrowe). Inne metody zakładają redundancję interfejsów sieciowych hosta (DRS), nadmiarowość łącz sieciowych (STP, MPLS, rozdział 3.5) lub też całych sieci transmisyjnych (DRS).
Zapewnienie tolerowania uszkodzeń jest bardziej skomplikowane, kiedy redundantne elementy są wykorzystywane podczas normalnej pracy systemu do równoważenia obciążenia (systemy klastrowe, LVS, rozdział 3.5)
Redundacja występuje także w warstwie oprogramowania. Stosuje się implementowanie funkcji jednych elementów przez inne (np. host pełniący rolę routera w sieci DRS).
Kiedy elementy posiadają stan wewnętrzny, oprogramowanie musi uwzględniać sychronizację stanów (rozdziały 3.6, 3.10)

Najważniejszą cechą oprogramowania jest ponowanie nad redundancją sprzętową. Istnieją dwa podejścia do zarządzania redundancją. Pierwsze polega na wyróżnieniu jednego elementu systemu, wstawiając go w centrum i implementując na nim specjalizowane funkcje. Przykładem takiego elementu jest dystrybutor w systemach klastrowych (np. LVS). Centralny element musi cechować się znacznie większą niezawodnością niż pozostałe elementy, ponieważ jego uszkodzenie powoduje niedostępność całego systemu. Niezawodność można zwiększać przez duplikowanie centralnego elementu (z wykorzystaniem technologii Heartbeat) co wraz z kolejnymi warstwami redundancji pociąga za sobą skomplikowanie i zhierarchizowanie systemu.
Tam gdzie jest to możliwe preferuje się płaski model redundancji oraz brak wyróżnionego elementu centralnego. W miejsce specjalizacji elementów wprowadza się uniwersalność i autonomiczność (np. LODES, DRS, MPLS).
Ważną funkcją systemów tolerujących uszkodzeń jest ich wykrywanie. Istnieje wiele algorytmów, które je realizują tą funkcję (np. LODES, STP, protokół TCP). Istnieją także sposoby na zapewnienie tolerowania uszkodzeń, które nie wymagają ich wykrywania (rozdział 3.5).

Uszkodzenia, które występują w niższych warstwach modelu logicznego sieci są niwelowane przez protokoły warstw wyższych oraz inne rozwiązania takie jak redundancja sprzętu i oprogramowania, leżące na zewnątrz płaszczyzny na której powstaje problem.
Sieci komputerowe odgrywają coraz większą rolę w naszym życiu, dlatego też projektanci sprzętu oraz oprogramowania wprowadzają coraz więcej technik poprawiających ich niezawodność, których część przedstawiona została w tym opracowaniu.

5. Literatura

[1] C. Hunt, „TCP/IP Administracja sieci – drugie wydanie”, Read Me, 1997.
[2] R. Hinden, „Virtual Router Redundancy Protocol“, http://www.ietf.org/internet-drafts/draft-ietf-vrrp-spec-v2-09.txt, Aug. 2003.
[3] Ryan McBride, „CARP”, http://marc.theaimsgroup.com/?l=openbsd-misc&m=106642790513590&w=2, October 2003.
[4] A. Chowdhury, O. Frieder, E. Burger, D. Grossman, K. Makki, “Dynamic Routing System DRS : fault tolerance in network routing”, Computer Networks #31, 1999, p. 89–99.
[5] P. Demeester et al., „PANEL - Protection across network layers“, NOC ‘97, European Conference on Networks and Optical Communications, Antwerp, Belgium, 1997.
[6] P. Demeester, M. Gryseels, K. Van Doorselaere, A. Autenrieth, C. Brianza, G. Signorelli, R. Clemente, M. Ravera, „Resilience in a multi-layer network”, IEEE Communications Magazine, vol. 37, no. 8, Aug. 1999, p. 70-76.
[7] Mon-Yen Luo, Chu-Sing Yang, “Enabling fault resilience for web services”, Computer Communications, Vol. 25/3, pp. 198-209, Feb. 2002.
[8] “Media Access Control (MAC) Bridges”, ANSI/IEEE Std 802.1D, 1998 Edition.
[9] L. Buytenhek, “Linux Bridge Documentation”, http://sourceforge.net/projects/bridge/, 2001.
[10] E. Rosen, A. Viswanathan, R. Callon, “Multiprotocol Label Switching Architecture”, http://www.ietf.org/rfc/rfc3031.txt, 2001.
[11] J.B. Postel „Transmission Control Protocol“ RFC793, 1981.
[12] V. Jacobson, M.J. Karels “Congestion Avoidance and Control”, Computer Communication Review, tom 18, nr 3, strony 314-329 1988.
[13] P. Karn, C. Partridge „Improving Round-Trip Time Estimates in Reliable Transport Protocols”, Computer Communication Review, tom 17, nr 5, strony 2 – 7, 1987.
[14] F. Le Faucheur, et al, “MPLS Support of Differentiated Services,” RFC3270, maj 2002.
[15] S. Blake, et al, “An Architecture for Differentiated Services,” RFC2475, grudzień 1998.
[16] Y. Rekhter, T. Li, „A Border Gateway Protocol 4 (BGP-4)”, RFC 1771, http://www.rfc-editor.org/rfc/rfc1771.txt, March, 1995.
[17] CISCO, „Border Gateway Protocol”, http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/bgp.pdf.
[18] B. Hubert el al., “Linux Advanced Routing & Traffic Control HOWTO”, http://www.lartc.org/howto/, chapter 4.2.
[19] T. Sugawara, “A multi-agent monitoring and diagnostic system for TCP/IP-based network and its coordination”, Knowledge-Based Systems, Elsevier Science, Vol: 14, Issue: 7, November 1, 2001.
[20] D. Hartmeier, „Design and Performance of the OpenBSD Stateful Packet Filter (pf)”, http://www.benzedrine.cx/pf-paper.html.
[21] M. Zobniow, „Markcb - balansowanie polaczen wychodzacych z LAN”, http://markcb.zobniow.net/.
[22] L. Raggi, „Lbalance patch”, http://lists.netfilter.org/pipermail/netfilter-devel/2003-June/011743.html, June, 2003.
[23] Rainfinity, “RainConnect”, http://www.rainfinity.com/products/ds_rainconnect_isa.pdf.
[24] Traceroute Project, http://www.traceroute.org/.
[25] T.I.P. Osso, http://www.osso.sys.net.pl/.
[26] Cyberbajt Reseter, http://cyberbajt.pl/reseter/.
[27] M. Krasnyansky, M. Yevmenkin, „Universal TUN/TAP driver”, http://vtun.sourceforge.net/tun/.

[28] O. P. Damani, P. E. Chung, Y. Huang, C. Kintala, Y. Wang, “ONE-IP: techniques for hosting a service on a cluster of machines”, Computer Networks and ISDN Systems, Vol: 29, Issue: 8-13, September, 1997.
[29] D. Anderson, T. Yang, V. Holmedahl and O. H. Ibarra, ``SWEB: Towards a Scalable World Wide Web Server on Multicomputers'', http://www.cs.ucsb.edu/Research/rapid_sweb/SWEB.html, IPPS'96, April, 1996.
[30] S. L. Garfinkel, ``The Wizard of Netscape'', WebServer Magazine, July/August 1996, pp. 58-64.
[31] C. Yoshikawam, B. Chun, P. Eastham, A. Vahdat, T. Anderson, and D. Culler, ``Using Smart Clients to Build Scalable Services'', USENIX'97, Jan. 1997.
[32] T. Brisco, ``DNS Support for Load Balancing'', Network Working Group, RFC 1794, http://andrew2.andrew.cmu.edu/rfc/rfc1794.html.
[33] C. R. Attanasio, and S. E. Smith, ``A Virtual Multiprocessor Implemented by an Encapsulated Cluster of Loosely Coupled Computers'', IBM Research Report RC18442, 1992.
[34] D. Dias, W. Kish, R. Mukherjee, and R. Tewari, ``A Scalable and Highly Available Server'', COMPCON 1996, pp. 85-92, 1996.
[35] C. Gage, “IBM Network Dispatcher Version 3.0: Scalability, Availability and Load-balancing for TCP/IP applications”, http://www-900.ibm.com/cn/support/library/sw/download/nd30whitepaper.pdf, June, 2000.
[36] ``Cisco Local Director'', http://www.cisco.com/warp/public/751/lodir/index.html.
[37] E. Anderson, D. Patterson, and E. Brewer, ``The Magicrouter, an Application of Fast Packet Interposing'', http://www.cs.berkeley.edu/~eanders/magicrouter/osdi96-mr-submission.ps, OSDI, 1996.
[38] R. Russel et al., Netfilter: firewalling, NAT and packet mangling for linux 2.4, http://www.netfilter.org/documentation/index.html.
[39] B. Conoboy et al., ”IPF-HOWTO”, http://www.obfuscation.org/ipf/ipf-howto.pdf.
[40] G. Palmer, A. Nash, “FreeBSD Handbook”, http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls.html, chapter 10.8.
[41] H. Welte, “Summary of the netfilter developer workshop 2003”, http://www.netfilter.org/~laforge/nf-workshop-2003-summary/, chapter 2.4.
[42] H. Welte, “The future of Linux packet filtering”, http://www.ukuug.org/events/linux2003/papers/welte.pdf.
[43] S. Lerena, “High Availability in Linux Firewalls using VRRP” http://www.gnusec.com/resource/docs/HAFirewallLinux-VRRP.pdf.
[44] H. Welte, „How to replicate the fire - HA for netfilter based firewalls”, http://gnumonks.org/cgi-bin/cvsweb.cgi/presentation/netfilter-failover-ols2002/netfilter-failover-ols2002.tex.
[45] H. Welte, „Conntrack state synchronization development”, http://lists.netfilter.org/pipermail/netfilter-devel/2003-October/012804.html.
[46] Z. James, “Packet filter state synchronization: pfsyncd”, http://www.greyhats.org/openbsd/openbsd.html#pfsyncd.
[47] CISCO, “NAT Stateful Failover of Network Address Translation”, http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t13/ftsnat.pdf.
[48] W. Zhang „Linux Virtual Server for Scalable Network Services”, National Laboratory for Parallel & Distributed Processing Changsha, Hunan 410073 China, http://www.linuxvirtualserver.org
[49] L. Marovsky-Bree “Heartbeat overwiev”, Ottawa Linux Symposium 2003, http://www.linux-ha.org/heartbeat/talks/LMB-OLS2003-tutorial.pdf

[50] B. Cahill “OpenGFS Overwiev” 2003,
http://opengfs.sourceforge.net/cvsmirror/opengfs/docs/OpenGFS_2003.pdf
[51] P. J. Braam, G. Matzigkeit „The InterSync & InterMezzo“
https://projects.clusterfs.com/intermezzo/HowTo
[52] P. J. Braam “The Coda Distributed File System”,
http://www.coda.cs.cmu.edu/ljpaper/lj.html
[53] W. Simpson „RFC 1853 - IP in IP Tunneling”, październik 1995
http://www.faqs.org/rfcs/rfc1853.html