



**SAMODZIELNY ZAKŁAD SIECI KOMPUTEROWYCH
POLITECHNIKA ŁÓDZKA**

90-924 Łódź ul. Stefanowskiego 18/22
tel./fax. (42) 636 03 00
e-mail: szsk@zsku.p.lodz.pl

Michał Kustosik

**Implementacja protokołu IPv6
w środowisku MSK LODMAN**

praca dyplomowa magisterska

Promotor:

Dr inż. Stanisław Starzak

Dyplomant:

Michał Kustosik

nr albumu 81003

Łódź, luty 2001

Spis treści

1. Cel i zakres pracy	4
2. Wprowadzenie do protokołów sieciowych warstwy 3	5
3. Poziomy standaryzacji, edycje	10
4. Protokoły warstwy sieci	11
4.1. Protokół ARP i RARP	11
4.2. Protokół IPv4	13
4.2.1. Wstęp	13
4.2.2. Format nagłówka podstawowego IPv4	14
4.2.3. Schemat adresowania w IPv4	19
4.2.4. Zasady routingu	21
4.3. Protokół ICMPv4	23
4.4. Specyfikacja protokołu IPv6	25
4.4.1. Wstęp	25
4.4.2. Format nagłówka podstawowego IPv6	26
4.4.3. Nagłówki dodatkowe w protokole IPv6	29
4.4.4. Schemat adresowania w IPv6	36
4.5. Protokół ICMPv6	38
4.6. Porównanie wersji protokołów IPv4 i IPv6	40
5. Stan implementacji protokołu IPv6	42
5.1. Stan implementacji IPv6 w Polsce	42
5.2. Implementacje firmowe	44
5.2.1. Solaris 7	44
5.2.2. Cisco	45
5.2.3. Windows	45
5.3. Implementacje "public domain"	46
5.3.1. Linux	46
6. Projekt sieci testowej w środowisku MSK LODMAN	47
6.1. Założenia projektowe	47
6.2. Specyfikacja testów i kryteria wyników	48
6.2.1. Poprawność działania sieci testowej MSK LODMAN	48

6.2.2. Porównanie prędkości protokołów przy przesyłaniu dużych plików	50
6.2.3. Porównanie prędkości protokołów przy przesyłaniu wielu plików	53
6.2.4. Routing IPv6 w sieci 6BONE	56
6.2.5. Stabilność i poprawność działania oprogramowania oraz sprzętu	57
6.3. Konfiguracja sprzętu i oprogramowania	58
6.3.1. Sieć testowa IPv6 dla systemu Linux Debian 2.2	58
6.3.2. Tunel IPv6 do ogólnopolskiej sieci 6BONE	62
6.3.3. Sieć testowa IPv6 dla systemu Windows 2000	64
7. Wykonanie testów i interpretacja wyników	67
7.1. Poprawność działania sieci testowej MSK LODMAN	67
7.2. Porównanie prędkości protokołów IPv4 i IPv6	69
7.2.1. Przesyłanie dużych plików	69
7.2.2. Przesyłanie wielu plików	74
7.3. Routing IPv6 w sieci 6BONE	80
7.4. Stabilność i poprawność działania oprogramowania oraz sprzętu	81
8. Wnioski	82
9. Bibliografia	84

1. Cel i zakres pracy

Prace nad stworzeniem nowej wersji podstawowego protokołu internetowego, trwają już od 1992 roku. Podstawową definicję protokołu zakończono w grudniu 1995r. opublikowaniem między innymi dokumentów RFC. IPv6 nie wprowadza rewolucyjnych zmian w stosunku do swojego poprzednika. Mechanizmy dobrze działające w IPv4 nadal są używane. Zmiany będą polegać głównie na zwiększeniu zdolności adaptacji protokołu do nowych warunków pracy oraz zwiększeniu wydajności przetwarzania pakietów, co odciąży działanie routerów. Celem mojej pracy będzie opis i porównanie protokołów IP oraz przedstawienie podstawowych zagadnień związanych z nowym protokołem.

W pierwszej części pracy opisałem podstawowe zagadnienia dotyczące IP oraz TCP/IP, przedstawiłem warstwowy model sieci ISO-OSI, opisałem poszczególne jego warstwy oraz proces enkapsulacji i dekapulacji. W rozdziale 4.1. opisałem zasady działania ARP oraz RARP i jego związek z warstwą IP. Sam protokół IPv4 przedstawiony jest w rozdziale 4.2. oraz 4.3. Opisałem format nagłówka podstawowego, aby móc porównać go z nagłówkiem nowego protokołu (rozdział 4.4. i 4.5.). W kolejnej części mojej pracy opisałem stan implementacji IPv6 na różnych platformach, a rozdział 6 i 7 poświęcony jest testom związanym z wprowadzeniem IPv6.

2. Wprowadzenie do protokołów sieciowych warstwy 3 [2,3,13i]

Do nawiązania i podtrzymania komunikacji między komputerami niezbędny jest protokół, czyli zbiór zasad określających zachowanie się partnerów komunikacji. Zadania stawiane komunikującym się komputerom są tak złożone, że nie wystarcza pojedynczy protokół. Dla zrozumienia zależności między protokołami, a także dla ich praktycznej implementacji definiuje się modele protokołów. Jednym z najpopularniejszych modeli jest koncepcyjny model warstwowy ISO-OSI, który stworzony został przez Międzynarodową Organizację Normalizacji (ISO - International Standards Organization), a punktem odniesienia, do którego porównuje się inne rozwiązania jest wzorcowy model łączenia systemów otwartych (OSI - Open System Interconnect Reference Model).

Modele stosowane przez producentów oprogramowania dają się łatwo odwzorować na model ISO-OSI. Pomimo tego, że zagadnienia komunikacji są złożone, daje się jednak wyodrębnić pewne zadania na tyle niezależne, że mogą być rozwiązywane przez osobne pakiety oprogramowania lub układy sprzętowe, zwane w terminologii ISO-OSI obiektami. Klasę obiektów rozwiązujących dane zagadnienie nazywa się warstwą. Określenie warstwa wzięło się stąd, że klasy obiektów daje się uporządkować w pewną strukturę przypominającą stos. Warstwy wyższe zależą od niższych w tym sensie, że poprawne rozwiązanie przez warstwy niższe problemów postawionych tym warstwom umożliwia lub ułatwia poprawne skonstruowanie warstw wyższych.

Dana warstwa korzysta z usług świadczonych przez warstwę bezpośrednio niższą, a sama dostarcza usług dla warstwy bezpośrednio wyższej. Usługi te polegają na stwarzaniu wrażenia obiektowi z warstwy wyższej, że komunikuje się bezpośrednio ze swym odpowiednikiem na odległym komputerze. Na szczycie stosu znajdują się usługi "końcowe" świadczone bezpośrednio użytkownikom przez aplikacje sieciowe, na spodzie - sprzęt realizujący transmisję sygnałów niosących informacje. Aby wykonać swoje zadania, każda warstwa używa specyficznego protokołu (lub protokołów). Protokoły te wymagają wymiany pewnych informacji sterujących. Odbywa się to tak, że do każdego pakietu danych przychodzącego z warstwy wyższej, dana warstwa "dokleja" charakterystyczny dla siebie nagłówek z informacjami dla swej odpowiedniczki na zdalnym komputerze i przekazuje całość do warstwy niższej, gdzie proces ten się powtarza. Warstwy z zasady nie ingerują w treści przekazywane z warstw wyższych. Po odebraniu pakietu z warstwy niższej, dana warstwa interpretuje swój nagłówek i jeśli stwierdza, że pakiet należy przekazać wyżej, "odkleja" swój nagłówek. Dzięki temu warstwa wyższa dostaje pakiet w takiej samej postaci, w jakiej wysłała go jej odpowiedniczka z odległego komputera. Kolejne warstwy definiują różne funkcje

protokołów wymiany danych. Każda warstwa może zawierać kilka protokołów, z których każdy dostarcza usług zgodnych z funkcją danej warstwy. Można powiedzieć, że każdy protokół w danej warstwie zajmuje się komunikacją wyłącznie ze swoim odpowiednikiem na zdalnej maszynie, natomiast nie zajmuje się tym, jakie funkcje są spełniane w sąsiednich warstwach. Dlatego musi być jednoznacznie określony standard wymiany danych pomiędzy konkretnymi warstwami, ponieważ warstwa np. sesji musi kontaktować się i rozumieć protokoły pracujące w tej warstwie na innym komputerze. Dane w obrębie jednego komputera przekazywane są z górnej warstwy (aplikacji) do dolnej (warstwy fizycznej) a na komputerze docelowym dane odbierane są przez warstwę fizyczną i przekazywane do warstwy aplikacji, poprzez kolejne szczeble modelu OSI. Zaletą takiego sposobu wymiany informacji jest możliwość zmiany oprogramowania, lub samego standardu przesyłania danych jednej warstwy, bez ingerencji w sąsiednie.

warstwa aplikacji	aplikacje wykorzystujące sieć
warstwa prezentacji	określa strukturę danych przekazywanych pomiędzy aplikacjami
warstwa sesji	zarządza sesjami łączącymi aplikacje
warstwa transportowa	określa mechanizmy detekcji i korekcji błędów po obu stronach połączenia
warstwa sieciowa	zarządza połączeniami sieciowymi wykorzystywanymi przez wyższe warstwy
warstwa łącza	zapewnia niezawodne dostarczanie danych przez łącza fizyczne
warstwa fizyczna	określa fizyczne składniki nośników danych wykorzystywanych przez sieć

Rys. 2.1. Wzorcowy model OSI.

Zadaniem warstwy fizycznej jest fizyczne przekazywanie sygnałów z jednego komputera do drugiego, bezpośrednio z nim połączonego pewnym medium. W skład tej warstwy wchodzi nadajniki i odbiorniki, czyli karty sieciowe, modemy itp., media, a więc np. kable ethernetu, czy token-ring, sieć telefoniczna, eter radiowy itp.

Poniżej warstwy fizycznej znajduje się warstwa łącza. Jej zadaniem jest niezawodne dostarczanie informacji poprzez warstwę fizyczną. Tak więc obiekty tej warstwy potrafią się komunikować jedynie z analogicznymi obiektami w komputerach przyłączonych bezpośrednio do tego samego medium. Warstwa ta jest ściśle związana z warstwą fizyczną. Obiektami w tej warstwie są sterowniki kart sieciowych itp. Protokoły tej warstwy muszą uwzględniać fizyczną topologię połączeń i najczęstsze błędy pojawiające się w łączu fizycznym. Do warstwy tej może także należeć kompresja i szyfrowanie danych lub zlecenie tych czynności sprzętowi.

Zadaniem warstwy sieciowej jest kierowanie przepływem pakietów w sieci. Zapewnia ona, że pakiety przesyłane między komputerami nie łączącymi się bezpośrednio, będą podawane z komputera na komputer, aż osiągną adresata. Proces znajdowania drogi w sieci nazywa się trasowaniem lub routowaniem. Nie wymaga się, aby pakiety pomiędzy ustalonymi punktami poruszały się za każdym razem po tej samej drodze. W pewnych sytuacjach dopuszcza się gubienie pakietów przez tą warstwę (w przypadku przeciążenia routera, kiedy ten nie może przyjmować nowych pakietów).

Kolejną warstwą jest warstwa transportowa, której zadaniem jest zapewnienie niezawodnej komunikacji pomiędzy odległymi komputerami lub abstrakcyjnymi portami komunikacyjnymi w tych komputerach. Warstwa ta dba, by odbiorca dostawał pakiety w tej samej kolejności w jakiej nadawca je wysyła, a w razie, gdy przez określony czas brak pakietów od nadawcy, lub przychodzą uszkodzone, żąda ich retransmisji. Powyżej tej warstwy dane można traktować jako strumień.

Warstwa sesji kontroluje nawiązywanie i zrywanie połączenia przez aplikacje. W starszych implementacjach warstwa ta zapewniała dostęp do zdalnego komputera za pomocą sesji terminalowej. Do warstwy tej można zaliczyć funkcje API udostępniane programiście przez bibliotekę realizującą dostęp do sieci na poziomie powyżej warstwy transportowej, takie jak np. biblioteka strumieni i gniazdek BSD.

Kolejna warstwa - prezentacji - filtruje przepływające przez nią dane, zapewniając wzajemne dopasowanie sposobów reprezentacji danych, a w konsekwencji zrozumienie komunikujących się aplikacji. Rozwiązuje takie problemy, jak niezgodność reprezentacji liczb, znaków końca wiersza, liter narodowych itp., odpowiada także za kompresję i szyfrowanie danych.

Najwyżej położoną warstwą jest warstwa aplikacji, której zadanie polega na świadczeniu końcowych usług, takich jak zdalne udostępnianie plików, drukarek i innych zasobów, czy też sieciowe systemy baz danych.

Zestaw protokołów, taki jak TCP/IP, jest kombinacją różnych protokołów działających w kilku warstwach.

Historia zestawu protokołów TCP/IP sięga końca lat 50-tych, kiedy rząd amerykański utworzył organizację o nazwie ARPA (Advanced Research Projects Agency), której celem było rozwinięcie technik zapewniających stabilną, niezawodną i niezależną od sprzętu komunikację pomiędzy systemami komputerowymi. Rezultatem tych badań była utworzona w roku 1969 sieć pakietowa o nazwie ARPANET, która jest bezpośrednim prekursorem obecnej sieci Internet. TCP/IP jest wynikiem rozwoju ARPANET w latach 70-tych. Ponieważ poszczególne jego części nie powstały w żadnej konkretnej firmie, czy profesjonalnej organizacji, zdecydowano się utworzyć organizację charytatywną o nazwie IAB (Internet Architecture Board), która będzie zajmowała się koordynacją badań dotyczących protokołów TCP/IP oraz będzie wyznaczać kierunki rozwoju Internetu.

Protokoły sieciowe opracowane są zwykle z uwzględnieniem warstwowego modelu sieci ISO/OSI, w którym kolejne warstwy odpowiedzialne są za różne aspekty komunikacji.

Zestaw protokołów TCP/IP rozważa się zwykle jako system czterowarstwowy.

warstwa aplikacji	aplikacje wykorzystujące sieć, telnet, FTP, e-mail itd.
warstwa transportowa	implementacja takich protokołów jak TCP oraz UDP
warstwa sieci	obsługuje takie protokoły jak IP, ICMP, IGMP
warstwa łącza	dostarczanie danych przez łącza fizyczne (program obsługi urządzeń, karta interfejsu)

Rys. 2.2. Czterowarstwowy model protokołów TCP/IP.

Informacja wędruje z warstwy aplikacji do warstwy łącza, a na komputerze docelowym odwrotnie, z warstwy łącza poprzez kolejne, aż do warstwy aplikacji. Przechodząc przez kolejne warstwy, dodawane są nagłówki zgodnie z rysunkiem poniżej. Na komputerze docelowym nagłówki te są odczytywane przez kolejne warstwy, aż do ostatniej (aplikacji), gdzie dostajemy czystą informację, jaka została wysłana przez nadawcę. Proces dodawania nagłówków nazywa się enkapsulacją, a proces odwrotny dekapulacją.

			dane	warstwa aplikacji
		nagłówek warstwy aplikacji	dane	warstwa transportowa
	nagłówek warstwy transportowej	nagłówek warstwy aplikacji	dane	warstwa sieci
nagłówek warstwy sieci	nagłówek warstwy transportowej	nagłówek warstwy aplikacji	dane	warstwa łącza

Rys. 2.3. Proces enkapsulacji i dekapulacji.

Każda z warstw pokazanych na czterowarstwowym modelu protokołów TCP/IP (Rys. 2.2.) jest odpowiedzialna za coś innego.

Warstwa łącza, nazywana czasem warstwą łącza danych lub warstwą dostępu do sieci, zawiera zwykle programy obsługi urządzeń, wchodzące w skład systemu operacyjnego i odpowiadające im karty interfejsów sieciowych w komputerze. Definiują one sposób użycia sieci w celu wysłania lub odebrania datagramu IP. Warstwa ta obsługuje wszystkie sprzętowe aspekty związane z fizycznym dołączeniem komputera do kabla sieciowego. Warstwa dostępu do sieci musi więc znać cechy sieci fizycznej, do której dołączony jest system, aby móc wysłać pakiet IP zgodnie z obowiązującymi w danej sieci regułami.

Warstwa sieci leżąca powyżej warstwy łącza, czasem nazywana również warstwą internetu, obsługuje ruch pakietów w sieci, czyli jest odpowiedzialna za routing pakietów, adresowanie, fragmentację i defragmentację pakietów oraz przekazywanie pakietów do warstwy transportowej. W zestawie protokołów TCP/IP warstwę tę tworzą: IP (Internet Protocol), ICMP (Internet Control Message Protocol) oraz IGMP (Internet Group Management Protocol).

Warstwa transportowa leży powyżej warstwy sieci. Należą do niej dwa protokoły: TCP (Transmission Control Protocol - protokół kontroli transmisji) oraz UDP (User Datagram Protocol - protokół pakietów użytkownika), zapewniające przepływ danych pomiędzy dwoma hostami, obsługując znajdującą się nad nią warstwę aplikacji. Protokół TCP zapewnia niezawodny przepływ danych pomiędzy dwoma hostami. Realizuje on funkcje weryfikacji poprawności przesyłanych danych, kontrolę szybkości ich przesyłania, porządkowanie pakietów docierających w niewłaściwej kolejności oraz obsługi zdublowanych danych. Przed transmisją samych danych, hosty wymieniają komunikaty kontrolne i dopiero po pomyślnym zakończeniu, transmitowane są właściwe dane. Ponieważ protokół TCP warstwy transportowej zapewnia niezawodny przepływ danych, warstwa aplikacji nie musi już się tym zajmować.

Protokół UDP jest protokołem bezpołączeniowym, nie obsługuje takich mechanizmów jak TCP dotyczących kontroli transmisji danych. Przesyła on pomiędzy hostami pakiety zwane datagramami, nie gwarantując przy tym, że dotrą one do punktu przeznaczenia. Pakiety UDP mogą być dublowane, gubione lub przychodzić w innej kolejności. W tym przypadku, jeżeli żądany jest jakiś poziom niezawodności przesyłania danych, to musi go zapewnić warstwa aplikacji. Protokół UDP jest stosowany wszędzie tam, gdzie wydajność jest podstawowym kryterium.

Warstwa aplikacji obsługuje funkcje związane z określoną aplikacją, korzystającą z sieci. Istnieje wiele aplikacji TCP/IP, które dostępne są w prawie każdej implementacji. Najważniejsze z nich to:

- TELNET - pozwalający na zalogowanie się i pracę na zdalnym systemie,
- FTP (File Transfer Protocol) - protokół transmisji danych pomiędzy odległymi systemami,
- SMTP (Simple Mail Transfer Protocol) - protokół odpowiedzialny za przesyłanie poczty elektronicznej,
- SNMP (Simple Network Management Protocol) - protokół zarządzania siecią,
- NNTP (Network News Transfer Protocol) - protokół odpowiedzialny za obsługę grup dyskusyjnych,
- DNS (Domain Name System) - protokół używany do przekształcania nazw domenowych, czyli czytelnych dla użytkownika na adresy IP oraz odwrotnie,
- HTTP (Hyper Text Transfer Protocol) - protokół odpowiedzialny za przesyłanie stron WWW (World Wide Web).

3. Poziomy standaryzacji, edycje [5i, 9i]

Propagowaniem wiedzy o nowym protokole internetowym zajęło się IPv6 Forum. Jest to organizacja skupiająca około 40 firm (m.in. 3Com, Cisco, Microsoft, Nokia, Sprint, Ericsson), zainteresowanych promocją nowego standardu. IPv6 Forum ma ściśle współpracować z organizacją Internet Engineering Task Force, która dopracowała szczegóły techniczne IPv6. Organizacja zajmująca się zarządzaniem adresami IP - IANA (Internet Assigned Numbers Authority), w połowie lipca 1999r. poinformowała o rozpoczęciu światowego podziału pierwszej puli adresów IP zgodnych z IPv6.

Polska strona poświęcona nowemu protokołowi IPv6, to <http://www.6bone.pl/> , na której można znaleźć bieżące informacje o stanie sieci IPv6 w Polsce, jak również konfiguracje IPv6 w różnych środowiskach i wiele innych informacji.

Istnieje wiele dokumentów RFC traktujących o IPv6. Są to zarówno ogólne specyfikacje, jak i dokładne opisy adresowania pakietów, budowy poszczególnych nagłóweków, multicastingu itd. Dokumenty RFC ulegają ciągłym zmianom, a ich aktualne uzupełnienia można znaleźć na stronie:

<http://playground.sun.com/pub/ipng/html/specs/specifications.html>

4. Protokoły warstwy sieci

4.1. Protokół ARP i RARP [3]

ARP (Address Resolution Protocol) jest jednym z podstawowych protokołów w każdej implementacji TCP/IP, ale zwykle wykonuje swoje działania bez ingerencji aplikacji lub administratora systemu. Warstwa łącza danych nie używa adresu IP w celu dostarczenia ramki Ethernet do drugiego hosta, lecz 48-bitowego adresu sprzętowego. Program obsługi interfejsu nigdy nie sprawdza adresu IP przeznaczenia, który umieszczony jest w datagramie IP. Protokół ARP zapewnia dynamiczne mapowanie pomiędzy adresami IP a odpowiadającymi im adresami sprzętowymi. Protokół ten opracowany został dla sieci typu broadcast, w których wiele hostów i routerów dołączonych jest do jednej sieci. Jeżeli host chce nawiązać połączenie z innym hostem, ARP wysyła ramkę Ethernet nazywaną żądaniem ARP do każdego hosta pracującego w sieci. Takie działanie nazywane jest broadcast. Zapytanie takie, zawiera adres IP hosta przeznaczenia. Każdy host pracujący w sieci Ethernet odbiera zapytanie ARP i jeżeli adres IP zawarty w zapytaniu jest jego adresem IP wysyła do nadawcy żądania odpowiedź zawierającą swój adres sprzętowy. Teraz połączenie może zostać nawiązane. W celu przyśpieszenia działania ARP, na każdym z hostów utrzymywana jest pamięć podręczna ARP, w której przechowywane są bieżące mapowania adresów IP na adresy sprzętowe. 48-bitowy adres Ethernetowy zapisywany jest w postaci sześciu liczb szesnastkowych, oddzielonych dwukropkiem.

Protokół ARP jest również wykorzystywany do poszukiwania swojego własnego adresu IP. Działanie takie jest zwykle wykonywane podczas ładowania systemu. Działanie takie nazywa się gratuitous ARP. Gratuitous ARP pozwala hostowi sprawdzić, czy jakiś host nie używa przypadkiem tego samego adresu IP. Host wysyła zapytanie o własny adres IP nie spodziewając się żadnej odpowiedzi. Jeżeli jednak odpowiedź taka nadejdzie oznacza to, że w sieci znajduje się drugi komputer o takim samym adresie IP, a na konsoli pojawi się komunikat o duplikacji adresów IP. Drugą funkcją wysyłania zapytań ARP podczas startu systemu jest informacja dla innych hostów, o adresie sprzętowym startującego hosta. Wysłana ramka ARP sprawia, że wszystkie inne hosty pracujące w sieci lokalnej, które przechowywały informację w pamięci podręcznej o starym adresie sprzętowym, zmieniają go na nowy.

Inną cechą ARP jest proxy ARP, które pozwala routerowi odpowiadać na zapytania ARP kierowane z jednej dołączonej do niego sieci do drugiej. Takie działanie sprawia, że host wysyłający zapytanie ARP traktuje router jako host docelowy, choć tak naprawdę host docelowy znajduje się po drugiej stronie routera. Router działa jako agent proxy dla hosta przeznaczenia, przekazując do niego pakiety. Układ taki sprawia, że fakt istnienia hosta przeznaczenia "za" routerem jest niezauważalny dla wszystkich hostów pracujących w sieci lokalnej.

RARP (Reverse Address Resolution Protocol) jest używany przez systemy bezdyskowe, w celu otrzymania swojego adresu IP w czasie ładowania systemu. Zapytanie RARP jest komunikatem typu broadcast, identyfikującym adres sprzętowy nadawcy i zawierającym pytanie skierowane do każdego z prośbą o odpowiedź zawierającą adres IP pytającego. Odpowiedź jest zwykle komunikatem typu unicast. Zapytania RARP są wysyłane jako ramki Ethernet z charakterystycznym polem typ ramki (0x8035). Oznacza to, że serwer RARP musi mieć możliwość wysyłania i odbierania ramek Ethernet tego właśnie typu. Ponieważ zapytania RARP wysyłane są jako komunikaty typu broadcast na poziomie sprzętowym, nie są więc one przekazywane przez routery. Aby umożliwić systemowi bezdyskowemu załadowanie się nawet wtedy, gdy nie działa host, na którym uruchomiony był serwer RARP, w jednej sieci uruchamia się zwykle kilka takich serwerów. W miarę wzrostu liczby serwerów w danej sieci rośnie również ruch, ponieważ każdy serwer RARP wysyła odpowiedź na każde zapytanie RARP, jakie pojawia się w sieci. Systemy bezdyskowe wysyłające te zapytania, wykorzystują tylko pierwszą odebraną odpowiedź RARP. Istnieje również możliwość, że dwa serwery RARP udzielą odpowiedzi w tym samym momencie, zwiększając prawdopodobieństwo wystąpienia kolizji.

4.2. Protokół IPv4 [1,2,3]

4.2.1. Wstęp

Protokół IP jest podstawowym protokołem w zestawie TCP/IP. Wszystkie dane z protokołów wyższych, lub niższych w hierarchii wykorzystują IP do przesyłania danych. IP świadczy usługi bezpołączeniowego przesyłania danych. Oznacza to, że nie ma żadnej gwarancji na pomyślne dostarczenie datagramu IP do punktu przeznaczenia. IP stara się jedynie dobrze przesłać datagramy. W razie jakichkolwiek błędów IP usuwa datagram i próbuje wysłać do źródła informację ICMP. Wszelkie mechanizmy niezawodności transmisji muszą być zapewniane przez wyższe warstwy, np. TCP. IP nie zachowuje żadnej informacji o datagramach, które zostały przesłane pomyślnie. Każdy datagram obsługiwany jest niezależnie od innych, co w praktyce oznacza, że mogą one docierać do punktu przeznaczenia w innej kolejności, niż zostały wysłane.

Zadaniem protokołu IP jest:

- definiowanie podstawowej jednostki przesyłanych danych używanej w intersieciach TCP/IP. W definicji tej określony jest dokładny format wszystkich danych przesyłanych przez intersieć TCP/IP;
- definiowanie operacji trasowania wykonywanej przez oprogramowanie IP, polegającej na wybieraniu trasy, którą będą przesyłane dane;
- oprócz precyzyjnego, formalnego wyspecyfikowania formatów danych i trasowania IP, definiowanie zbioru reguł, które służą do realizacji idei zawodnego przenoszenia pakietów. Reguły te opisują, w jaki sposób węzły i routery powinny przetwarzać pakiet, jak i kiedy powinny być generowane komunikaty o błędach oraz kiedy pakiety mogą być porzucane.

Protokół IP jest tak podstawową częścią konstrukcji intersieci TCP/IP, że często nazywa się TCP/IP techniką opartą na protokole IP.

4.2.2. Format nagłówka podstawowego IPv4

Moduły zajmujące się przesyłaniem ramek pomiędzy stacjami źródłową i docelową, wykorzystują przesyłany z każdą ramką nagłówek protokołu IP. Wiadomości zawarte w nagłówku IP, są również wykorzystywane przy dzieleniu i łączeniu pakietów danych podczas przesyłania. Poniżej przedstawiony jest schemat datagramu IP wersji 4:

0	4	8	16	32
WER	DŁ. NAGŁ.	TYP OBSŁUGI	DŁUGOŚĆ CAŁKOWITA	
IDENTYFIKACJA			ZNACZNIKI	PRZESUNIĘCIE FRAGMENTU
CZAS ŻYCIA (TTL)	PROTOKÓŁ		SUMA KONTROLNA	
ADRES IP NADAWCY				
ADRES IP ODBIORCY				
OPCJE IP (JEŚLI SĄ)			UZUPEŁNIENIE	
DANE				

Rys. 4.1. Format datagramu internetu IPv4.

Pole **WERSJA** jest pierwszym 4-bitowe polem w datagramie IP i zawiera informację o wersji protokołu IP, która była używana do tworzenia datagramu. Informacja ta jest wykorzystywana do sprawdzenia, że nadawca, odbiorca i wszystkie routery między nimi zgadzają się na format datagramu. Od całego oprogramowania IP wymaga się, aby przed rozpoczęciem przetwarzania datagramu sprawdziło pole wersji w celu upewnienia się, że jego format zgadza się ze spodziewanym.

Pole **DŁUGOŚĆ NAGŁÓWKA**, również 4-bitowe, zawiera długość nagłówka datagramu, mierzoną w 32-bitowych słowach. W nagłówku wszystkie pola mają tę samą długość, za wyjątkiem pól OPCJE i UZUPEŁNIENIE. Najczęściej spotykany nagłówek nie zawierający opcji i uzupełnienia ma 20 oktetów i pole długości nagłówka równa się 5.

Pole **TYP OBSŁUGI**, 8-bitowe, określa sposób, w jaki datagram powinien zostać obsłużony. Składa się ono z pięciu pól, które pokazane są na rys. 4.2.:

PIERWSZEŃSTWO	O	S	P	NIE UŻYWANE
---------------	---	---	---	-------------

Rys. 4.2. Podział 8-bitowego pola TYP OBSŁUGI.

Pierwsze 3 bity to pole pierwszeństwa (które obecnie jest ignorowane), 4 kolejne bity to TOS (ang. Type of Service), a kolejny 1 bit nie jest używany, a jego wartość musi być równa 0. 4-bitowe pole TOS przekazuje informacje o minimalizacji opóźnień, maksymalizacji szybkości przesyłania oraz o maksymalizacji poprawności i minimalizacji kosztów. Tylko jeden z tych czterech bitów może być włączony. W przypadku, gdy wszystkie 4 bity są 0, to stosowane są standardowe usługi. Rys 4.3. przedstawia tabelę z zalecanymi wartościami pola TOS dla różnych aplikacji.

APLIKACJA	Minimalizacja opóźnień	Maksymalizacja szybkości przesyłania	Maksymalizacja poprawności	Minimalizacja kosztów
Telene/Rlogin	1	0	0	0
FTP				
kontrola	1	0	0	0
dane	0	1	0	0
TFTP	1	0	0	0
SMTP				
faza komend	1	0	0	0
faza danych	0	1	0	0
DNS				
zapytanie UDP	1	0	0	0
zapytanie TCP	0	0	0	0
transmisja obszaru	0	1	0	0
ICMP				
błąd	0	0	0	0
zapytanie	0	0	0	0
dowolne IGP	0	0	1	0
SNMP	0	0	1	0
BOOTP	0	0	0	0
NNTP	0	0	0	1

Rys. 4.3. Zalecane wartości pola TYP USŁUGI dla różnych aplikacji.

Aplikacje interaktywne, takie jak Telnet i Rlogin, wymagają minimalnych opóźnień, ponieważ wykorzystywane są interaktywnie przez użytkowników do przesyłania niewielkiej liczby danych. Przesyłanie plików za pomocą FTP wymaga natomiast maksymalnej szybkości przesyłania. Maksymalna poprawność przesyłanych danych wymagana jest przez protokoły zarządzania siecią (SNMP) i routingu. Funkcja TOS nie jest obsługiwana przez używane obecnie implementacje TCP/IP, choć nowe wersje systemów, począwszy od 4.3BSD Reno ustawiają wartości tego pola.

Pole **DŁUGOŚĆ CAŁKOWITA** podaje długość datagramu IP w bajtach. Używając tego pola i pola długości nagłówka można wskazać miejsce, w którym rozpoczyna się część datagramu IP zawierająca dane, a także długość tych danych. Ponieważ jest to pole 16-bitowe, maksymalny rozmiar datagramu IP to 65535 bajtów.

Trzy kolejne pola nagłówka datagramu - **IDENTYFIKACJA**, **ZNACZNIKI** oraz **PRZESUNIĘCIE FRAGMENTU**, służą do kontroli procesów fragmentacji i składania datagramów.

Pole **IDENTYFIKACJA** zawiera liczbę całkowitą jednoznacznie identyfikującą datagram. Router, gdy fragmentuje datagram, kopiuje większość pól z nagłówka pierwotnego datagramu do fragmentów. Pole **IDENTYFIKACJA** musi zostać skopiowane. Służy ono przede wszystkim do umożliwienia docelowemu komputerowi zorientowania się, które z przychodzących fragmentów należą do którego datagramu. Gdy przychodzi fragment, komputer docelowy używa tego pola razem z adresem nadawcy do identyfikacji datagramu. Komputery wysyłające datagramy IP muszą generować po jednej wartości pola **IDENTYFIKACJA** dla każdego pojedynczego datagramu. Jedną z metod wykorzystuje licznik globalny, który zwiększany jest z każdym nowo tworzonym datagramem, a którego wartość jest przypisywana jako pole **IDENTYFIKACJA** datagramu.

Pole **ZNACZNIKI** jest kolejnym polem, którego rozmiar wynosi 3 bity. Dwa młodsze bity służą do kontroli fragmentacji. Zwykle programy użytkowe używające TCP/IP nie zajmują się sprawami związanymi z fragmentacją, gdyż zarówno fragmentację, jak i składanie obsługują automatyczne procedury, które znajdują się na niskim poziomie systemu operacyjnego i są niewidoczne dla użytkowników. Może jednak okazać się konieczne, do celów związanych z testowaniem i usuwaniem błędów działania, aby sprawdzić rozmiary datagramów, dla których zaszła fragmentacja. Pierwszy z bitów kontrolnych służy do takich testów określając, czy datagram może zostać podzielony na fragmenty. Nazywa się go bitem "nie fragmentuj". Ustawienie go na 1 oznacza, że datagram nie powinien być fragmentowany. Gdy router jest zmuszony podzielić datagram, który ma ustawiony bit "nie fragmentuj", porzuca go i odsyła do nadawcy komunikat o błędzie. Młodszy bit w polu **ZNACZNIKI** określa, czy fragment zawiera dane ze środka pierwotnego datagramu, czy z końca. Bit ten jest włączany dla każdego fragmentu zawierającego dane z datagramu, poza ostatnim fragmentem.

Pole **PRZESUNIĘCIE DLA FRAGMENTU** określa przesunięcie z pierwotnego datagramu dla danych przenoszonych za pomocą tego fragmentu. Aby zaoszczędzić miejsce w nagłówku jest ona mierzona w jednostkach 8-oktetowych, począwszy od przesunięcia w zero. W celu złożenia datagramu, komputer docelowy musi uzyskać wszystkie fragmenty zaczynające się od 0, aż do fragmentu o największym przesunięciu. Fragmenty nie muszą być dostarczane do komputera docelowego w kolejności, ponieważ router, który podzielił datagram i komputer docelowy, który próbuje datagram złożyć, nie porozumiewają się.

Pole **CZAS ŻYCIA** (TTL, ang. Time to Live) określa górną granicę liczby routerów, przez które może przejść datagram. Wartość ta określana jest przez wysyłającego (zwykle 32 lub 64) i zmniejszana o jeden przez każdy router, który obsługuje datagram. Gdy wartość w polu CZAS ŻYCIA równa jest 0, router porzuca datagram i wysyła do nadawcy komunikat o błędzie. Pomysł utrzymania zegara dla datagramów jest ważny, gdyż zapewnia, że datagramy nie mogą podróżować w sieci w nieskończoność, nawet gdy tablice tras staną się nieaktualne, a routery wyznaczają datagramom trasy w kółko.

Pole **PROTOKÓŁ** służy do określenia, który protokół wysokiego poziomu został użyty do utworzenia treści pola DANE datagramu. W rzeczywistości zawartość pola PROTOKÓŁ wyznacza format pola DANE.

Pole **SUMA KONTROLNA NAGŁÓWKA** służy do sprawdzenia sensowności zawartości nagłówka. Nie są w niej uwzględniane żadne dane, które następują po nagłówku. Protokoły ICMP, IGMP, UDP i TCP same wyliczają sumy kontrolne dla swoich nagłówków i danych. Wartość pola sumy kontrolnej jest ustawiana początkowo na 0. Następnie liczona jest dla nagłówka 16-bitowa suma z dopełnieniem (tzn. cały nagłówek traktowany jest jako sekwencja słów 16-bitowych). W polu sumy kontrolnej umieszczana jest wartość policzona w taki właśnie sposób. Kiedy datagram IP zostanie odebrany, suma kontrolna liczona jest po raz kolejny. Ponieważ suma policzona przez odbiorcę zawiera sumę kontrolną umieszczoną w polu przez wysyłającego, to jeśli nagłówek nie został po drodze zmodyfikowany, suma policzona przez odbiorcę zawiera same jedynki. Jeśli wynik jest inny, to IP odrzuca odebrany datagram. Nie jest przy tym tworzony żaden komunikat o błędzie. Wykrycie brakującego datagramu i powtórne jego przesłanie jest zadaniem wyższych warstw sieci.

Pola **ADRES IP NADAWCY** i **ADRES IP ODBIORCY** zawierają 32-bitowe adresy IP nadawcy datagramu i zamierzonego odbiorcy. Chociaż datagram może być przesyłany różnymi trasami, pola nadawcy i odbiorcy nigdy nie ulegają zmianie - określają one adres pierwotnego nadawcy i końcowego odbiorcy. Wyjątkiem jest tutaj sytuacja, gdy datagram zawiera wymienione dalej opcje wyznaczania trasy przez nadawcę. Każdy interfejs w sieci musi mieć unikalny adres internetowy (tzw. adres IP). Te 32-bitowe adresy są zapisywane w postaci czterech liczb dziesiętnych rozdzielonych kropkami. Każda liczba odpowiada jednemu bajtowi adresu. Taki zapis nazywany jest kropkowo-dziesiętnym.

Pole **OPCJE**, które następuje po adresie odbiorcy, nie występuje w każdym datagramie. Pierwotnym jego zastosowaniem było ułatwienie testowania i usuwania błędów. Jeżeli opcje znajdują się w datagramie, to ich długość jest zależna od tego, jakie opcje są wybrane. Niektóre z nich mają długość jednego oktetu - składają się z pojedynczego oktetu kodu opcji, inne mają zmienną długość. Opcje zajmują ciągły obszar bez specjalnych separatorów między nimi. Każda opcja składa się z kodu opcji długości jednego oktetu, po którym może pojawić się jeden oktet długości oraz ciąg oktetów danych tej opcji. Oktet kodu opcji podzielony jest na trzy pola (rys. 4.4).

KOPIUJ	KLASA OPCJI	NUMER OPCJI
--------	-------------	-------------

Rys. 4.4. Podział pola KOD OPCJI.

1-bitowy znacznik KOPIUJ ustawiony na 1 oznacza, że opcje powinny być kopiowane do wszystkich fragmentów. Gdy ustawiony jest na 0 oznacza, że opcje powinny być kopiowane tylko do pierwszego fragmentu, a nie do wszystkich. Bity pól KLASA OPCJI i NUMER OPCJI służą do określenia ogólnej klasy opcji oraz opcji w tej klasie. Pole OPCJI kończy się zawsze na granicy 32 bitów. Jeśli to konieczne, umieszczane są bity wypełnienia o wartości 0. Takie rozwiązanie sprawia, że nagłówek IP jest zawsze wielokrotnością 32 bitów.

4.2.3. Schemat adresowania w IPv4

O miejscu, do którego datagram zostanie dostarczony, decyduje adres internetowy (zwany również adresem IP). Każdy interfejs znajdujący się w sieci musi mieć unikalny adres IP, który w sieci IPv4 jest 32-bitowym numerem, zapisywanym w postaci czterech liczb dziesiętnych oddzielonych kropkami. Każda liczba odpowiada jednemu bajtowi adresu. Zapis taki nazywamy kropkowo-dziesiętnym. Przydziałem adresów dla sieci zajmuje się Internet Network Information Center, nazywane InterNIC. InterNIC przydziela adresy tylko dla danych sieci, natomiast przydziałem adresów do poszczególnych hostów zajmuje się administrator danego systemu.

Adresy IP podzielono na pięć typów adresów zwanych klasami, które oznaczono kolejnymi literami alfabetu. Założono, że będzie istniała potrzeba obsługi tylko kilku dużych sieci (działających w dużych firmach komputerowych i głównych uniwersytetach), średniej liczby sieci o średniej wielkości oraz wielu małych. Klasy są identyfikowane za pomocą kilku najstarszych bitów.

Klasa	Zakres
A	0.0.0.0 do 127.255.255.255
B	128.0.0.0 do 191.255.255.255
C	192.0.0.0 do 223.255.255.255
D	224.0.0.0 do 239.255.255.255
E	240.0.0.0 do 247.255.255.255

Rys. 4.5. Zakresy numerów dla różnych klas adresów IP.

Jeżeli pierwszym bitem adresu jest 0 - sieć należy do klasy A. Sieci klasy A może być nieco mniej niż 128, ale każda z nich może się składać z milionów hostów. W przypadku kiedy pierwsze bity to 1 0 - sieć należy do klasy B, których istnieją tysiące, a każda z nich może się składać z tysięcy hostów. W przypadku, kiedy pierwsze bity tworzą sekwencję 1 1 0 - sieć należy do klasy C. Sieci takich można tworzyć miliony, ale żadna z nich nie może składać się z więcej niż około 250 hostów. Adresy, w których pierwsze bity to 1 1 1 0 określają sieć klasy D. Klasa ta nie wskazuje pojedynczego hosta, lecz zestaw urządzeń, które wchodzi w skład grupy określanej jako multicast. Klasa, której adresy rozpoczynają się od 1 1 1 1 została zarezerwowana do wykorzystania w przyszłości. Taki sposób adresowania w sieciach IP nie przewidywał tak olbrzymiej ilości komputerów podłączonych do sieci. Okazało się, że klasa A jest zbyt duża, zaś klasa C zbyt mała. Nawet pomimo tego, że klasa B jest zbyt duża, była ona wykorzystywana z braku lepszych rozwiązań. Rozwiązaniem tego problemu jest

wprowadzenie tzw. maski podsieci. Są to 32-bitowe liczby zawierające bity jedynek w części będącej adresem sieci i bity zer dla części, która jest adresem hosta. Maski, podobnie jak adresy IP, zapisywane są za pomocą czterech liczb dziesiętnych oddzielonych kropkami. Maskowanie pozwala na ominięcie ograniczeń związanych z regułami klas. Za pomocą masek można zwiększyć przestrzeń adresową (tzw. tworzenie nadsieci) lub ją zmniejszyć (tzw. tworzenie podsieci). Np. maska 255.255.0.0 obejmuje adresy od xxx.xxx.0.0 do xxx.xxx.255.255 (czyli wielkość klasy B), podczas gdy maska 255.255.255.224 dotyczy przestrzeni adresowej np.: xxx.xxx.xxx.224 do xxx.xxx.xxx.255.

Istnieją trzy typy adresów IP:

- unicast - adres określający dany, konkretny host,
- broadcast - adres całej sieci w której pracuje dany host. Datagram wysłany na broadcast danej sieci zostanie odebrany przez każdy host znajdujący się w tej sieci. Datagramy typu broadcast nie są przekazywane dalej przez router,
- multicast - adres grupy hostów należących do tej samej grupy multicastowej. Pozwala to na zredukowanie zbędnego obciążenia hostów, które nie są zainteresowane daną aplikacją. Jeżeli stosowane są adresy typu multicast, to host musi specjalnie przyłączyć się do jednej z grup rozgłoszeniowych, aby dostawać odpowiednie datagramy.

4.2.4. Zasady routingu

Routing (trasowanie) w systemie z wymianą pakietów oznacza proces wyboru ścieżki, po której będą przesyłane pakiety. Router to komputer, który dokonuje tego wyboru. Idealne oprogramowanie trasujące powinno przy wyborze tras korzystać z takich informacji jak obciążenie sieci, długość datagramu, czy zawarty w nagłówku datagramu typ obsługi. Jednak przeważająca część oprogramowania trasującego wybiera trasy na podstawie ustalonych informacji o najkrótszych ścieżkach. Routowanie jest jedną z najważniejszych funkcji IP.

Możemy wyróżnić dwa rodzaje trasowania:

- dostarczanie bezpośrednio - przesyłanie datagramów od jednej maszyny do drugiej znajdującej się fizycznie w tej samej pojedynczej sieci - nadawca kapsułkuje datagram w fizycznej ramce, wiąże adres odbiorcy z adresem sprzętowym i wysyła utworzoną ramkę bezpośrednio do adresata;
- dostarczanie pośrednio - przesyłanie datagramów od jednej maszyny do drugiej znajdującej się w innej sieci, datagramy muszą więc przejść przez co najmniej jeden router. Ostatni router dostarcza datagram bezpośrednio do odbiorcy.

Trasowanie IP oparte na tablicach tras polega na tym, że każdy datagram przechodzący przez router jest kierowany na odpowiedni interfejs za pomocą tablicy tras IP. Tablice takie, w celu minimalizacji objętości zawartych tam informacji, posiadają tylko informacje o najbliższym otoczeniu. W tablicach tras wystarczy przechowywać tylko adresy sieci, a nie pełne adresy IP poszczególnych komputerów. Datagramy, które docelowo mają trafić do danych sieci są kierowane na odpowiednie interfejsy. Router, który odebrał datagram, przeszukuje tablicę routingu w poszukiwaniu rekordu, który odpowiada adresowi przeznaczenia IP. Jeżeli taki adres zostanie znaleziony, wysyła pakiet do wskazanego routera będącego routerem następnego przejścia lub do bezpośrednio dołączonego interfejsu. Jeżeli nie ma wpisu traktującego bezpośrednio danego adresu IP, router przeszukuje tablicę routingu w poszukiwaniu rekordu, który odpowiada adresowi sieci. Jeżeli router nie znajdzie żadnej informacji o docelowej sieci, to datagram kierowany jest do routera domyślnego. Jeżeli żaden z opisanych kroków nie przyniesie rezultatu, datagram zostanie uznany za taki, którego nie można dostarczyć. Zawsze jako pierwsze przeszukiwane są adresy hostów, następnie rekordy z adresami sieci, a na samym końcu wykorzystywany jest rekord domyślnej trasy. Jeżeli router na podstawie swojej tablicy routingu stwierdzi, że otrzymany datagram ma być

wysłany na ten sam interfejs, wtedy do nadawcy wysyłany jest komunikat ICMP o przekierowaniu. Najczęściej polecenia o przekierowaniu wykorzystuje host, mający niewielką wiedzę o routowaniu, do udoskonalenia podczas pracy swojej tablicy routowania. Host może być uruchomiony tylko z domyślnymi pozycjami, a kiedy te okażą się błędne, host uaktualnia tablicę routingu zgodnie z informacjami o przekierowaniu, nadesłanymi przez domyślny router. Komunikaty ICMP o przekierowaniu są generowane tylko przez routery, a nie przez hosty, natomiast przeznaczone są tylko do wykorzystania przez hosty, a nie przez routery. Routowanie, które dotychczas opisałem, nazywa się routowaniem statycznym i dobre jest tylko w przypadku stosunkowo małych sieci, w których istnieje jeden punkt połączenia z innymi sieciami i nie ma routerów zapasowych. W większych sieciach używa się routowania dynamicznego, które polega na tym, że routery wymieniają informacje między sobą poprzez specjalny protokół routowania, informując się wzajemnie o tym, jakie sieci są obecnie dołączone do każdego z nich. Proces routera, który obsługuje protokół routowania, nazywany jest demonem routowania. Daemon taki zmienia tablicę routowania w sposób dynamiczny poprzez dodawanie i usuwanie tras, które ulegają zmianie podczas pracy systemu. Jeżeli daemon znajdzie kilka routerów obsługujących jeden punkt przeznaczenia, wybiera najlepszą trasę i dokonuje odpowiedniej modyfikacji tablicy routingu. Jeżeli daemon odkryje, że jakieś połączenie przestało działać, może usunąć związane z nim trasy lub dodać trasy alternatywne, co pozwoli ominąć problem. W sieci internet używa się obecnie wielu różnych protokołów routowania. Każdy system autonomiczny może stosować własny protokół routowania w celu wymiany informacji pomiędzy routerami pracującymi w tym systemie (IGP - Interior Gateway Protocol - wewnętrzny protokół routowania).

Najpopularniejszym protokołem routowania był RIP (Routing Information Protocol). Nowszym rozwiązaniem IGP jest protokół OSPF (Open Shortest Path First), który zastępuje RIP. Do komunikacji pomiędzy routerami pracującymi w różnych systemach autonomicznych służą między domenowe protokoły routowania (EGP - Exterior Gateway Protocols). Pierwszym takim protokołem był protokół o takiej samej nazwie - EGP. Nowszym rozwiązaniem jest BGP (Border Gateway Protocol).

4.3. Protokół ICMPv4 [3]

Protokół ICMP jest często rozpatrywany jako integralna część IP. Jest on częścią warstwy internetu i wykorzystuje datagramy IP do przesyłania informacji o błędach i innych zajściach, które wymagają kontroli. Komunikaty ICMP są zwykle wysyłane przez warstwę IP lub protokoły wyższych warstw (TCP, UDP). Niektóre błędy pochodzą od procesów użytkownika. Kiedy wysyłany jest komunikat o błędzie ICMP, znajduje się w nim nagłówek IP i pierwsze 8 bajtów datagramu IP, który spowodował wysłanie komunikatu ICMP. Pozwala to modułowi odbierającemu ICMP na skojarzenie komunikatu z konkretnym protokołem (TCP lub UDP) i z konkretnym procesem na podstawie portu TCP lub UDP, który jest zawarty w nagłówku TCP lub UDP. Komunikaty ICMP są przesyłane wewnątrz datagramów IP, a jego format zależy od typu komunikatu. Wspólny jest jedynie format 4 pierwszych bajtów:

0	8	16	32
TYP	KOD	SUMA KONTROLNA	
zawartość zależna od pola TYP i KOD			

Rys 4.6. Format komunikatu ICMP.

Najważniejsze komunikaty ICMP to:

- odpowiedź echo - program (np. ping) wysyła do hosta komunikat ICMP zawierający pytanie o echo, spodziewając się nadejścia odpowiedzi w postaci komunikatu ICMP;
- jeżeli miejsce przeznaczenia pakietu jest nieosiągalne, wtedy system, który to stwierdził, wysyła do nadawcy komunikat Destination Unreachable. Istnieje wiele typów tych komunikatów;

- stłumienie źródła jest błędem generowanym przez system (router lub host) w przypadku, kiedy otrzymuje on datagramy z prędkością przekraczającą możliwości ich przetwarzania;
- komunikaty o przekierowaniu są wysyłane w przypadku, kiedy router stwierdza, że datagram powinien być wysłany do innego routera. Po otrzymaniu takiego komunikatu nadawca powinien zmienić swoją tablicę routingu;
- komunikaty rozgłoszeniowe i ogłoszeniowe z prośbą o informację o aktualnym stanie sieci są przesyłane przez hosty, w celu uaktualniania tablicy routingu. Ponadto routery co jakiś czas rozgłaszają swoje komunikaty pozwalając, aby każdy nasłuchujący host mógł uaktualnić swoją tablicę routingu;
- komunikaty przekroczenia czasu są wysyłane do nadawcy w przypadku, kiedy pakiet przekroczył maksymalny czas przebywania w sieci. Pakiet taki jest wtedy usuwany przez router, który to wykrył, a do nadawcy wysyłany jest stosowny komunikat ICMP;

Protokół ICMP wersji 6 różni się nieco od protokołu ICMP w wersji 4. Protokół ICMPv6 jest opisany w dalszej części pracy.

4.4. Specyfikacja protokołu IPv6

4.4.1. Wstęp

IPv6 nazywa się IPng - New Generation, ale nie wprowadza on rewolucyjnych zmian. Wręcz przeciwnie - mechanizmy dobrze działające w IPv4, będą nadal używane, a zmiany będą polegać na zwiększeniu zdolności adaptacji protokołu do nowych, czasem jeszcze nie znanych, warunków pracy oraz zwiększeniu wydajności przetwarzania pakietów. W nowej wersji uproszczony został nagłówek pakietu IP, dzięki czemu, pomimo czterokrotnego wydłużenia rozmiaru każdego z adresów, nie zwiększył on zbyt swojej objętości. Wprowadzono pojęcie następnego nagłówka, który może zawierać opcje, informacje o autentyczności danych oraz pakiet TCP, czy UDP. Każdy nagłówek określa typ następnego poprzez pole NEXT HEADER.

Autorzy specyfikacji IPv6 wprowadzili szereg usprawnień w porównaniu do IPv4. Uproszczone zostały nagłówki pakietów, zmieniono sposób przetwarzania opcji, które mogą nieść informacje o wielkości pakietów o rozmiarach przekraczających 65535 bajtów; służyć do narzucania drogi pakietu; zawierać informacje o pakietach przesyłanych we fragmentach. Do nagłówka zostały dodane etykiety przepływu, które będą używane do wprowadzenia specjalnych reguł przetwarzania pakietów, np. przenoszących telekonferencje. Dodatkowo zawarto w definicji protokołu autentyfikację i szyfrowanie, co pozwoli ujednoczyć metody używane do kodowania danych przesyłanych w internecie. IPv4 potencjalnie umożliwia przydzielenie jednego adresu prawie każdemu mieszkańcowi Ziemi, ale nieunikniona nieefektywność, a często rozrzutność wykorzystania przestrzeni adresowej powoduje, że za kilka lat zacznie brakować adresów. Nowa, szósta wersja IP ma do wykorzystania 2^{128} adresów.

Prace nad stworzeniem nowej wersji podstawowego protokołu internetowego, trwają już od 1992 roku. Realizacją tego zadania zajął się zespół IETF (Internet Engineering Task Force - <http://www.ietf.cnri.reston.va.us>). Przede wszystkim stworzył on spójną definicję nowego protokołu, a potem dokładnie go przetestował. Prace nad podstawową definicją protokołu zostały zakończone w grudniu 1995 r. opublikowaniem m.in. dokumentów RFC. Wcześniej były używane protokoły eksperymentalne o różnych numerach i dlatego opuszczono w przejściu od IPv4 do IPv6 numer 5.

bity. Opcje i niektóre inne z ustalonych pól, które pojawiają się w nagłówku IPv4 zostały w IPv6 przeniesione do nagłówków dodatkowych, które zostały opisane w dalszej części tego rozdziału. W uogólnieniu zmiany w nagłówku datagramu odpowiadają zmianom w samym protokole.

Schemat nagłówka podstawowego przedstawia rys. 4.8.

0	4	12	16	32
WER	PRIORYTET	ETYKIETA PRZEPIYU		
DŁUGOŚĆ			NAST. NAGŁ.	LIMIT
ADRES NADAWCY				
ADRES ODBIORCY				

Rys. 4.8. Format datagramu IPv6.

Pole **WERSJA** jest pierwszym 4-bitowym polem w datagramie - (ang. version) i zawiera, podobnie jak w przypadku datagramu IPv4, informację o wersji protokołu IP, która była używana do tworzenia datagramu. Informacja ta jest wykorzystywana do sprawdzenia, że nadawca, odbiorca i wszystkie routery między nimi akceptują tę wersję datagramu IP.

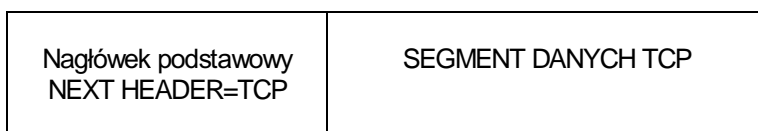
Pole **PRIORYTET** (ang. traffic class) (8-bitowe), określa poziom ważności pakietu. Najwyższe wartości mogą być stosowane do transmisji prowadzonych w czasie rzeczywistym.

Pole **ETYKIETA PRZEPIYU** (ang. flow label) (20-bitowe) jest podobne do poprzedniego pola. Informacja przeznaczona jest głównie dla routerów, przez jakie przesyłany jest pakiet. Pole to jest nadal w fazie eksperymentów. Hosty i routery, które nie wspierają tej funkcji powinny wstawiać zero, jeśli tworzą pakiet oraz ignorować to pole podczas otrzymywania pakietu do dalszej obróbki.

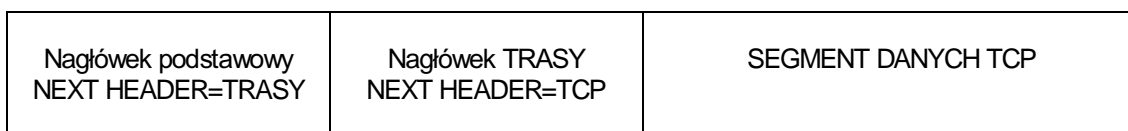
Pole **DŁUGOŚĆ** (ang. payload length) (16-bitowe) wskazuje długość obszaru danych (nie wliczając nagłówka).

Pole **NASTĘPNY NAGŁÓWEK** (ang. next header) (8-bitowe) określa typ następnego nagłówka, jeżeli istnieje. Nagłówek dodatkowy jest opcjonalny, może występować w danym pakiecie, lub nie. Nagłówek taki zawiera informacje dodatkowe, które są zawarte w oddzielnych nagłówkach, które umieszczane są za nagłówkiem podstawowym, a przed samymi danymi. Każdy z nagłówków podstawowych i dodatkowych zawiera pole **NASTĘPNY NAGŁÓWEK**, aby wskazać routerom pośrednim i odbiorcy końcowemu gdzie kończą się nagłówki a zaczynają się przesyłane dane. Wydobicie wszystkich informacji nagłówkowych z datagramu wymaga sekwencyjnego przeszukiwania nagłówków i oprogramowanie nie może przeszukiwać pakietu w poszukiwaniu odpowiedniego nagłówka dodatkowego, lecz musi przeglądać nagłówki dodatkowe po kolei. Routery pośrednie często nie potrzebują jednak przetwarzać wszystkich nagłówków dodatkowych, aby prawidłowo kierować pakiety. Nagłówki dodatkowe opisane zostały w dalszej części pracy.

a)



b)



Rys 4.9. Datagramy (a) z jednym nagłówkiem podstawowym, (b) z nagłówkiem podstawowym i jednym nagłówkiem dodatkowym - pole "NEXT HEADER" nagłówka podstawowego określa typ nagłówka następnego.

Pole **LIMIT** (ang. hop limit) (8-bitowe) określa maksymalną liczbę mijanych po drodze routerów. Pole to odpowiada polu CZAS ŻYCIA (TTL) w IPv4. Liczba w polu LIMIT jest zmniejszana o 1 przez każdy router, przez który przechodzi dany pakiet. Jeżeli wartość ta będzie równa 0, to pakiet jest porzucany.

Pole **ADRES NADAWCY** (ang. source address) (32-bitowe) zawiera 128-bitowy adres twórcy pakietu.

Pole **ADRES ODBIORCY** (ang. destination address) (32-bitowe) zawiera 128-bitowy adres odbiorcy pakietu, lub adres odbiorcy pośredniego, jeśli występuje znacznik w polu **NASTĘPNY NAGŁÓWEK** opisujący routing pakietu.

4.4.3. Nagłówki dodatkowe w protokole IPv6 [1,5,7,9,1i]

W protokole IPv6, aby ograniczyć rozmiar nagłówka podstawowego, wprowadzono nagłówki dodatkowe. Nagłówek dodatkowy może występować, ale nie musi. Zależy to od potrzeb przesyłania danego pakietu.

Wyróżniamy następujące nagłówki dodatkowe:

- Hop-by-Hop Options
- Routing
- Fragment Option
- Destination Options
- Authentication Header
- Encapsulating Security Payload

Jeżeli w tym samym pakiecie używanych jest 2 lub więcej nagłówków dodatkowych, wtedy preferowana jest następująca kolejność nagłówków dodatkowych:

- nagłówek podstawowy IPv6
- Hop-by-Hop Options
- Destination Options
- Routing
- Fragment Option
- Authentication Header
- Encapsulating Security Payload
- Destination Options
- upper-layer

Każdy nagłówek dodatkowy powinien pojawiać się w pakiecie najwyżej raz, za wyjątkiem nagłówka Destination Option, który może pojawić się najwyżej dwa razy - po raz pierwszy przed nagłówkiem Routing, a po raz drugi przed nagłówkiem upper-layer.

Nagłówek dodatkowy OPCJI [1,1i]

Dwa z obecnie zdefiniowanych nagłówków dodatkowych - Hop-by-Hop i Destination Option - używają zmiennych w formacie type-length-value (TLV) w polu Opcje, które ma następujący format:

Typ opcji	Długość opcji	Specyfikacja opcji
-----------	---------------	--------------------

Rys. 4.10. Format pola Opcje stosowanego w nagłówkach Hop-by-Hop oraz Destination Option.

Typ opcji - 8-bitowy identyfikator typu opcji. Pierwsze 2 bity tego pola zawierają informację, jaką ma podjąć oprogramowanie, jeżeli węzeł IPv6 nie rozpoznaje typu opcji. Pole to jest zakodowane w sposób, który pozwala na określenie akcji, którą ma wykonać węzeł nie obsługujący tego typu opcji:

00 - zignoruj opcje i kontynuuj czytanie nagłówka

01 - odrzuć pakiet

10 - odrzuć pakiet i nie zważając na to, czy adres docelowy był typu multicasting wyślij ICMP Parameter Problem, Code 2 do nadawcy pakietu mówiący o tym, że router nie rozpoznaje typu opcji.

11 - odrzuć pakiet i jeżeli tylko adres docelowy nie był typu multicasting, wyślij ICMP Parameter Problem, Code 2 do nadawcy pakietu mówiący o tym, że router nie rozpoznaje typu opcji.

Kolejny bit, również niezależny od typu opcji określa, czy dane dla tego typu opcji mogą być zmienione w trakcie przesyłania pakietu. Wartość równa jedności umożliwia ich zmianę. Pozostałe 5 bitów określa właściwy typ opcji.

Długość opcji - 8-bitowe pole o wartości długości pola *specyfikacja opcji* (w oktetach).

Nagłówek dodatkowy "Hop-by-Hop" [1i]

Nagłówek dodatkowy Hop-by-Hop jest nagłówkiem, który jest przetwarzany przez każdy router, przez który przechodzą dane pakiety. Nagłówek ten jest identyfikowany przez wartość 0 w polu NAGŁÓWKA DODATKOWEGO i ma następujący format:

Następny nagłówek	Dł. nagłówka	
Opcje		

Rys. 4.11. Format nagłówka dodatkowego Hop-by-Hop.

Następny nagłówek - 8-bitowe pole identyfikujące następny nagłówek dodatkowy występujący bezpośrednio po nagłówku Hop-by-Hop.

Długość nagłówka - 8-bitowe pole opisujące długość nagłówka Hop-by-Hop w jednostkach 8-oktetów (nie zawiera pierwszych ośmiu oktetów)

Opcje - to zmiennej długości pole zawierające właściwe opcje zakodowane w formacie TLV.

Nagłówek Hop-by-Hop Options pozwala na umieszczenie specjalnych opcji, które muszą być sprawdzone przez wszystkie węzły pośredniczące w przekazywaniu pakietu. Umożliwia to zarządzanie lub śledzenie pracy routerów.

Nagłówek dodatkowy "Destination Options" [1i]

Nagłówek Destination Options zawiera dodatkowe informacje przeznaczone tylko dla węzła docelowego. Jego budowa jest w zasadzie identyczna jak nagłówka Hop-by-Hop Options. Nie dopuszcza on jednak, aby adres odbiorcy był typu multicast.

Nagłówek dodatkowy "Routing Options" [7,1i]

Routing Options jest używany przez protokół IPv6 w celu wymuszenia drogi pakietu do hostu docelowego. Nagłówek ten może zawierać jeden lub więcej adresów pośrednich routerów, przez które musi przejść pakiet. Nagłówek Routing Options jest identyfikowany przez wartość o numerze 43 w polu Next Header. Format nagłówka Routing Options przedstawiony jest na poniższym rysunku:

Następny nagłówek	Dł. nagłówka	Typ Routingu	Segments Left
type-specific data			

Rys. 4.12. Format nagłówka dodatkowego Routing Options.

Następny Nagłówek oraz Długość Nagłówka mają znaczenie identyczne jak w poprzednich nagłówkach i wskazują na typ następnego nagłówka oraz wyznaczają długość nagłówka Routing Options.

Typ Routingu to 8-bitowe pole określające szczegółowy wariant routingu.

Segments Left - to 8-bitowe pole określające liczbę routerów, przez które powinien jeszcze przejść dany pakiet przed dostarczeniem go do miejsca docelowego.

type-specific data - zmiennej długości pole formatu określonego przez pole Typ Routingu. Długość tego pola jest taka, aby całość nagłówka była wielokrotnością ośmiu oktetów.

Jeżeli podczas procesu otrzymywania pakietu, router rozpozna nagłówek Routing Options z nieznaną wartością Typ Routingu, wtedy działanie routera zależy od zawartości pola Segments Left. Jeżeli pole Segments Left ma wartość zero, wtedy router ignoruje nagłówek Routing Options i odczytuje następny nagłówek dodatkowy, którego typ jest zidentyfikowany przez wartość pola Następny Nagłówek. Jeżeli pole Segments Left ma wartość różną od zera, wtedy router odrzuca pakiet i wysyła komunikat ICMP Parameter Problem, Code 0 do adresata danego pakietu, mówiący o nieznanym Typie Routingu. Jeżeli po przeanalizowaniu nagłówka dodatkowego Routing Options router rozpozna, że pakiet ma być przesłany do sieci, której MTU jest mniejsze niż rozmiar pakietu, wtedy pakiet jest porzucany oraz wysyłany jest komunikat ICMP Packet Too Big do adresata danego pakietu.

Typ 0 nagłówka Routing Options ma następujący format:

Następny nagłówek	Dł. nagłówka	Typ Routingu = 0	Segments Left
Zarezerwowane			
Adres 1			
Adres 2			
[...]			
Adres n			

Rys. 4.13. Format nagłówka dodatkowego Routing Options z polem Typ Routingu = 0

Pierwsze cztery pola opisane zostały przy opisywaniu ogólnej postaci nagłówka Routing Options, a pole Adres 1 - n zawiera 128 bitowe adresy routerów pośrednich.

Nagłówek podstawowy "Fragment Option" [1,5]

Nagłówek Fragment Option jest używany przez protokół IPv6 w celu przesłania pakietów większych niż MTU sieci. Zasadniczą różnicą w porównaniu do IPv4 jest to, że za fragmentację w IPv6 odpowiedzialny jest nadawca, a nie router pośredni, który stoi przed siecią o małym MTU. Przed rozpoczęciem wysyłania nadawca musi wykonać procedurę określania MTU ścieżki, dzięki której może poznać minimalne MTU wzdłuż ścieżki do odbiorcy. Następnie, zanim datagram zostanie wysłany, nadawca fragmentuje go tak, że każdy fragment jest mniejszy od MTU ścieżki. Dzięki temu fragmentacja zachodzi na końcach połączenia i nie musi być wykonywana w pośrednich routerach.

Nagłówek Fragment Option jest identyfikowany przez wartość 44 w polu NextHeader i posiada następujący format:

NextHeader	Reserved	Fragment Offset	Res	M
Identification				

Rys. 4.14. Format nagłówka dodatkowego Fragment Option.

NextHeader - 8-bitowe pole identyfikujące następny nagłówek występujący po nagłówku Fragment Option.

Reserved - 8-bitowe pole zarezerwowane do wykorzystania w przyszłości, wypełniane zerami przez nadawcę i ignorowane przez odbiorcę.

Fragment Offset - 13-bitowe pole określające przesunięcie danych w stosunku do oryginalnego pakietu.

Res - 2-bitowe pole zarezerwowane do wykorzystania w przyszłości, wypełniane zerami przez nadawcę i ignorowane przez odbiorcę.

M - 1 bitowe pole określające czy są jeszcze kolejne części pofragmentowanego pakietu. Wartość "1" oznacza, że są dalsze części pakietu, wartość "0" - nie ma kolejnych części pakietu.

Identification - 32 bitowe pole jednoznacznie oznaczające podzielone pakiety.

Nagłówek dodatkowy "Authentication Header" [9]

Nagłówek dodatkowy Authentication Header to mechanizm uwierzytelniania oraz sprawdzania integralności danych. Ogólna postać nagłówka Authentication Header wygląda następująco:

Next Header	Payload Len	Reserved
Security Parameter Index		
Sequence Number		
Authentication Data		

Rys. 4.15. Format nagłówka dodatkowego Authentication Header.

Next Header - 8-bitowe pole identyfikujące następny nagłówek występujący po nagłówku Authentication Header.

Payload Len - 8 bitowe pole określające długość nagłówka Authentication Header.

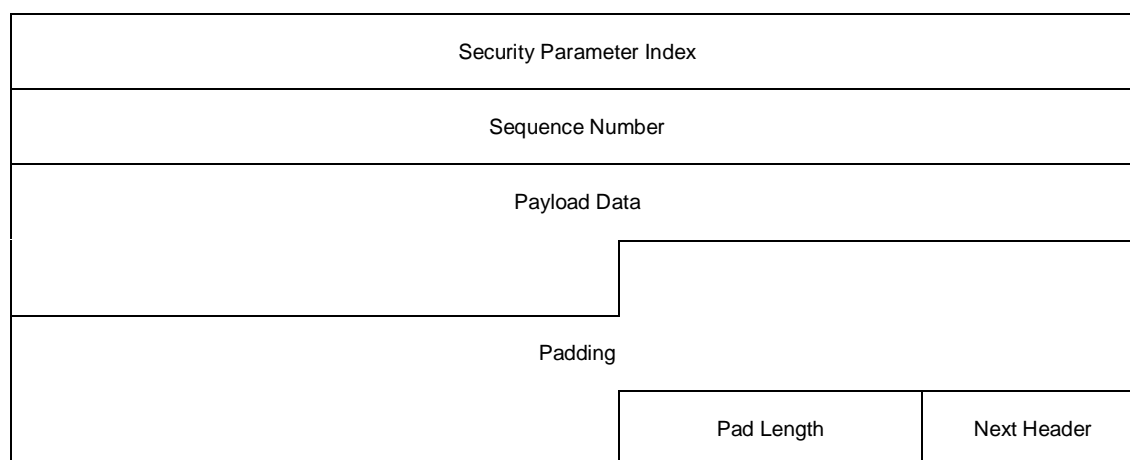
Security Parameter Index - 32 bitowe pole służące wraz z adresem odbiorcy pakietu do jednoznacznego określenia Security Association.

Sequence Number - 32 bitowe pole zapewniające integralność przesyłanych danych. Jego wartość jest synchronizowana pomiędzy nadawcą a odbiorcą pakietu w czasie tworzenia Security Association a następnie cyklicznie zwiększana.

Authentication Data - pole o zmiennej długości zawierające dane podlegające uwierzytelnieniu.

Nagłówek dodatkowy "Encapsulating Security Payload" (ESP) [9]

Nagłówek dodatkowy Encapsulating Security Payload (ESP) zapewnia poufność danych. Ze względu na specyficzną funkcję tego nagłówka jego format jest nieco inny od typowych nagłówków dodatkowych IPv6.



Rys. 4.16. Format nagłówka dodatkowego Encapsulating Security Payload (ESP).

Pola Security Parameter Index oraz Sequence Number pełnią taką samą funkcję jak w nagłówku dodatkowym Authentication Header omawianym wcześniej.

Payload Data - zmiennej długości, w którym przechowywane są przesyłane dane.

Padding - zmiennej długości, uzupełnia pozostałą część nagłówka do pełnego słowa.

Pad Length - 8-bitowe pole określające długość dopełnienia.

Next Header - 8-bitowe pole określające następny nagłówek.

4.4.4. Schemat adresowania w IPv6 [6,8,6i]

Rozmiar przestrzeni adresowej w IPv6 uległ znacznemu zwiększeniu. Każdy adres w IPv6 zajmuje 4 razy więcej niż adres w IPv4. Długie adresy rozwiązują problem niewystarczającej przestrzeni, choć zmuszają administratorów sieci do ich wprowadzania i operowania. Przykładowy 128-bitowy adres używany w protokole IPv6 możemy zapisać w następujący sposób:

2034:0:0:0:0:240C:437C

Adres podzielony jest na kawałki po 16 bitów i zapisywany szesnastkowo przedzielany dwukropkami. Obowiązuje zasada, że zera nieznaczące można opuścić, a serie pól z samymi zerami można zastąpić przez podwójny dwukropek. Nasz adres przykładowy wyglądałby więc 2034::240C:437C. Po przejściu na protokół IPv6 komputer posiadający adres IP wersji 4 może go nadal używać jako adresu IP wersji 6. Tak więc adres komputera 212.51.209.177 będzie w wersji IPv6 zapisany jako ::212.51.209.177 (96 bitów 0, ostatnie 32 bity zapisane jako 4 liczby dziesiętne po 8 bitów każda).

W tabeli poniżej przedstawiony jest proponowany podział adresów IPv6:

Prefiks (binarnie)	typ adresu	część przestrzeni adresowej
0000 0000	zarezerwowane (kompatybilność z IPv4)	1/256
0000 0001	zarezerwowane	1/256
0000 001	adresy NSAP	1/128
0000 010	adresy IPX	1/128
0000 011	zarezerwowane	1/128
0000 100	zarezerwowane	1/128
0000 101	zarezerwowane	1/128
0000 110	zarezerwowane	1/128
0000 111	zarezerwowane	1/128
0001	zarezerwowane	1/16
001	zarezerwowane	1/8
010	adresy jednostkowe przypisywane przez usługodawców sieciowych	1/8
011	zarezerwowane	1/8
100	zarezerwowane na adresy geograficzne	1/8
101	zarezerwowane	1/8
110	zarezerwowane	1/8
1110	zarezerwowane	1/16
1111 0	zarezerwowane	1/32
1111 10	zarezerwowane	1/64
1111 110	zarezerwowane	1/128
1111 1110	dostępne dla użytku miejscowego	1/256
1111 1111	używane do rozsyłania grupowego	1/256

Rys. 4.17. Proponowany podział adresów IPv6 na typy.

Z tabeli widać, że większa część przestrzeni adresowej została zarezerwowana do użytku w przyszłości. Pomimo tego, że prefix 0000 0000 jest oznaczony jako zarezerwowany, to jego mała część jest wykorzystywana do kodowania adresów IPv4.

Przy przechodzeniu z IPv4 do IPv6 potrzebne jest kodowanie, ponieważ można przejść na nowszy protokół, zanim zostanie przyznany odpowiedni adres IPv6, ale jednocześnie komputer pracujący już pod IPv6 może potrzebować komunikować się z komputerem, który działa jeszcze jedynie z IPv4.

Mechanizm przejścia IPv6 (ang. IPv6 transition mechanisms [TRAN]) zawiera technikę dla hostów i routerów do dynamicznego tunelowania pakietów IPv6 przez sieć IPv4.

Adresy IPv4 maszyn, które wspierają IPv6 są zapisywane następująco:

80 bitów	16	32 bity
0000...	...0000	0000 ADRES IPv4

a nazwa tych adresów to "IPv4-compatible IPv6 address".

Adresy hostów i routerów, które nie wspierają IPv6, znane są jako adresy IPv6 mapowane do IPv4 (ang. IPv4 mapped IPv6 address), a ich budowa wygląda następująco:

80 bitów	16	32 bity
0000...	...0000	FFFF ADRES IPv4

W związku z dużą długością adresu trudno go będzie zapamiętać, ale zaletą 128 bitów jest możliwość łatwiejszej organizacji hierarchicznej przydziału adresów i preadresowania w razie zmiany usługodawcy (zmienia się tylko kilkadziesiąt bitów prefiksu, reszta, może zostać bez zmian). Podobnie jak IPv4, również IPv6 wiąże adresy z poszczególnymi podłączeniami sieciowymi, a nie komputerami. Przypisywanie adresów jest podobne do przypisywania w IPv4 - router IPv6 posiada dwa lub więcej adresów, a węzeł IPv6 z pojedynczym przyłączeniem do sieci posiada jeden adres. Oprócz możliwości jednoczesnego wiązania wielu adresów z jednym połączeniem sieciowym IPv6 rozszerza oraz ujednolica adresy docelowe wg jednej z trzech kategorii:

- adres jednostkowy (ang. unicast address) - adres odbiorcy określa pojedynczy komputer (węzeł lub router), a datagram powinien być przesyłany do odbiorcy najkrótszą drogą.
- adres rejonowy (ang. anycast address) - odbiorcą jest zbiór komputerów, które mają ten sam prefiks (są np. podłączone do tej samej sieci fizycznej), datagram jest wtedy przesyłany do grupy najkrótszą drogą, a następnie powinien zostać dostarczony do dokładnie jednego jej członka (np. do najbliższego)
- adres grupowy (ang. multicast address) - odbiorcą jest zbiór komputerów, które mają ten sam prefiks, a kopia datagramu jest przesyłana do każdego z członków grupy przy użyciu sprzętowego rozsyłania grupowego, lub rozgłaszania.

4.5. Protokół ICMPv6 [4,10,11,9i]

Protokół ICMP wersji 6 nie różni się wiele od swojego poprzednika wersji 4. Przede wszystkim zdefiniowano tu nowe reguły przetwarzania komunikatów oraz określania nadawcy, co wynika z innych w stosunku do IPv4 typów adresów. Komunikaty o błędach zostały wyraźnie oddzielone od komunikatów informacyjnych. Format 4 pierwszych bajtów komunikatu ICMP jest identyczny jak w wersji ICMP4.

Uogólniając wyróżniamy następujące główne typy komunikatów:

TYP	rodzaj komunikatu
1	cel nieosiągalny (Destination Unreachable)
2	pakiet za duży (Packet Too Big)
3	czas przekroczony (Time Exceeded)
4	błąd parametru (Parameter Problem)

Rys. 4.18. Komunikaty błędów ICMPv6.

Poniżej przedstawione są główne rodzaje komunikatów informacyjnych:

TYP	rodzaj komunikatu
128	echo żądanie (Echo Request)
129	echo odpowiedź (Echo Reply)

Rys. 4.19. Komunikaty informacyjne ICMPv6.

Nowym, w porównaniu do poprzedniej wersji ICMP jest komunikat błędu pakiet za duży (Packet Too Big), który wysyłany jest przez router, który stwierdził, że MTU sieci, do której ma być posłany dalej dany pakiet jest mniejsze od wielkości samego pakietu.

W porównaniu do poprzedniej wersji ICMP, zwiększono jeszcze wielkość wysyłanej części datagramu, powodującej generowanie komunikatu ICMP, która dołączona jest przez niektóre typy komunikatów. Nowa wersja pozwala na dołączenie takiej liczby danych oryginalnego pakietu, aby komunikat ICMP nie przekroczył wartości MTU sieci.

4.6. Porównanie wersji protokołów IPv4 i IPv6 [1,2,3]

Nowy protokół IPv6, zwany również IP New Generation (IPng), zachowuje wiele cech IPv4, które przyczyniły się do sukcesu tego protokołu. W rzeczywistości konstruktorzy IPv6 scharakteryzowali ten protokół jako IPv4 z kilkoma modyfikacjami. Przykładowo IPv6 zapewnia w dalszym ciągu dostarczanie datagramów bez połączenia, umożliwia nadawcy ustalanie rozmiaru datagramu oraz wymaga od niego, aby określał maksymalną liczbę etapów, które datagram może przebyć zanim zostanie usunięty. Poza tym IPv6 zachowuje większość cech związanych z fragmentacją pakietów i wyznaczaniem tras według nadawcy. Mimo wielu koncepcyjnych podobieństw większość szczegółów protokołu IPv6 jest zmieniona.

Podsumowując Protokół Internetu wersja szósta (IPv6) zwany również IP New Generation (IPng) zawiera następujące zmiany w stosunku do IPv4:

- zwiększona przestrzeń adresowa z 32 do 128 bitów - nowe rozmiary adresów to najbardziej znacząca zmiana w stosunku do poprzedniej wersji protokołu IP. IPv6 powiększa cztery razy rozmiar adresu IP z 32 bitów do 128 bitów. Przestrzeń adresowa IPv6 jest tak duża, że nie powinna być wyczerpana w przewidywanej przyszłości;
- elastyczny format nagłówka - IPv6 używa całkowicie nowego formatu datagramu, niekompatybilnego z poprzednim. Przeciwnie, niż w przypadku IPv4, w którym nagłówek datagramu ma ustalony format (wszystkie pola za wyjątkiem opcji zajmują ustaloną liczbę oktetów i pozycję), IPv6 używa zbioru opcjonalnych nagłówków. Nagłówki dodatkowe pojawiają się, jeśli zachodzi taka potrzeba. Pozwala to w wielu przypadkach na zmniejszenie długości nagłówka, gdy pewne jego opcje nie są używane;
- wsparcie dla rezerwowania zasobów - IPv6 zastępuje specyfikowanie typu obsługi w IPv4 mechanizmem, który umożliwia wcześniejsze rezerwowanie zasobów sieciowych. W szczególności nowy mechanizm stanowi podstawę programów użytkowych służących do przekazywania dźwięków oraz obrazów w czasie rzeczywistym (wideo konferencje), które wymagają gwarantowanej przepustowości oraz stałego opóźnienia. Wsparcie dla rezerwowania zasobów umożliwia nam większą kontrolę przepływu pakietów oraz ich sterowania;

- zapewnienie możliwości rozszerzania protokołu - najbardziej znaczącą zmianą w IPv6 jest odejście od idei protokołu, który w pełni specyfikuje wszystkie szczegóły, co do możliwości dodawania kolejnych funkcji. Możliwość rozszerzania to cecha, która pozwoli IETF dostosowywać protokół do zmian w sprzęcie sieciowym oraz świecie programów użytkowych. Możliwość rozszerzania protokołów powoduje ewentualne dodawanie kolejnych opcji bez wykonywania rewolucji;
- poprawione opcje - podobnie jak w IPv4, IPv6 pozwala, aby datagramy zawierały opcjonalne informacje sterujące. IPv6 zawiera nowe opcje, które dają dodatkowe możliwości niedostępne w IPv4;
- opcje szyfrowania pakietów - wersja protokołu IPv6 zawiera dodatkowe opcje, które pozwalają na szyfrowanie pakietów. Poprawia to bezpieczeństwo datagramów przesyłanych za pomocą nowego protokołu, co powoduje zwiększenie bezpieczeństwa całej sieci.

Różnice są również w protokołach ICMP, choć ogólny format nagłówka ICMP pozostał niezmienny. Najważniejsze zmiany w stosunku do poprzedniej wersji ICMP to:

- uporządkowanie poszczególnych typów komunikatów poprzez podzielenie ich na dwie grupy: komunikaty o błędach (o numerach od 0 do 127) oraz komunikaty informacyjne (od 128 do 255). W poprzedniej wersji protokołu podział taki nie istniał,
- w wersji 6 protokołu ICMP wprowadzono komunikat błędu pakiet za duży, wysyłany przez router, który stwierdził, że MTU sieci do której ma być wysłany pakiet jest za małe. Błąd ten został wprowadzony, ponieważ w odróżnieniu od wersji 4 protokołu IP za fragmentacje w IPv6 odpowiedzialny jest nadawca, a nie router pośredni, który stoi przed siecią o małym MTU,
- zwiększono wielkość wysyłanej części datagramu powodującej generowanie komunikatu ICMP, która jest dołączana przez niektóre typy komunikatów. W nowej wersji protokołu można dołączyć taką ilość danych, aby komunikat ICMP nie przekroczył MTU ścieżki.

5. Stan implementacji protokołu IPv6

5.1. Stan implementacji IPv6 w Polsce [9i]

Polska strona poświęcona IPv6 to <http://www.6bone.pl/>. 6BONE jest testową siecią IPv6, do której od 1997 roku podłączane są węzły na terenie Polski. 6BONE wykorzystuje do komunikacji protokół IPv6. Sieć ta jest oparta w większości o połączenia tunelujące IPv6 w protokole IPv4 i służy głównie do badania zachowania IPv6 w różnych warunkach. Pierwszy router 6BONE w Polsce z routingiem statycznym został uruchomiony w kwietniu 1997r. w PDi w Toruniu. W 1998r. Szczecin i ICM uruchomiły pierwsze połączenia BGP z zagranicą z SICS w Szwecji. ICM uzyskał w maju 1999r. status pTLA (pseudo Top Level Aggregation) w sieci testowej 6BONE i rozdaje chętnym adresy z sieci 3ffe:8010::/28. Ośrodek szczeciński ma połączenia BGP z ATT (NL), MERIT, 6COM (INR, TELEBIT DK w planach) i razem z ICM tworzą ośrodek pTLA. Szczecin przydziela podsieci 3ffe:8014::/32.

W chwili pisania pracy do testowej sieci 6BONE w Polsce podłączone są 103 węzły w 41 miastach. Działające obecnie serwisy to http, pop3, telnet, ssh, ident, ftp, irc, smtp, jednak nie wszystkie serwery wspierają wszystkie te serwisy. Sieć oparta na IPv6 jest siecią testową, a wdrożenie nowego protokołu we wszystkich ośrodkach komputerowych zajmie jeszcze z pewnością wiele czasu. Podłączenie się do 6BONE, to przede wszystkim okazja do poznania wielu aspektów działania sieci opartych o protokół IP, wykorzystania w praktyce routingu dynamicznego opartego o BGP itp.

Najbliższe plany dotyczące IPv6, to między innymi zestawianie połączeń IPv6/ATM przy użyciu routera Cisco albo karty ATM, uzyskanie adresów nietestowych, uporządkowanie routingu BGP, podłączenie do 6REN za pośrednictwem POL-34 oraz wiele innych.

Podłączenie się do sieci 6BONE nie wiąże się z żadnymi kosztami. Poprosić o tunele mogą zarówno instytucje komercyjne jak i niezarobkowe. Aby skonfigurować własny tunel w sieci 6bone należy dysponować komputerem mającym stały adres IP, mogącym pełnić funkcje routera IPv6. Potrzebna jest również podstawowa implementacja i możliwość tunelowania IPv6 w IPv4. Po instalacji oprogramowania, należy skontaktować się z administratorem najbliższego routera IPv6 i poprosić o skonfigurowanie tunelu po jego stronie oraz odpowiednio skonfigurować tunel w swoim komputerze.

Aktualny spis połączeń 6bone-gw jest dostępny na stronie <http://6bone-gw.6bone.pl/stats/>, a przykłady konfiguracji na stronie <http://www.6bone.pl/konfiguracja.html>.

Dobrze jest również zapisać się na listy dyskusyjne mówiące o IPv6:

- 6bone-pl@sunsite.icm.edu.pl - lista zajmująca się 6BONE w Polsce,
- 6bone@isi.edu - lista zajmująca się 6BONE na całym świecie.

Warto również śledzić kanał [#ipv6](#) oraz [#ipv6.pl](#) na IRC, gdzie codziennie pojawiają się informacje i dyskusje na temat IPv6 w Polsce oraz na całym świecie.

5.2. Implementacje firmowe

5.2.1. Solaris 7 [2i]

Implementacja IPv6 dla Solarisa jest jeszcze w fazie prób i projektów. Obecnie prototyp IPv6 jest dostępny dla wersji Solaris 7. Na wcześniejszych wersjach systemu protokół IPv6 nie będzie działał. W celu instalacji IPv6 na Solaris 7 najlepiej jest ściągnąć z internetu odpowiednie oprogramowanie (*IPv6 Prototype Version 1.1*) dostępne na stronach Firmy Solaris i zainstalować je zgodnie z załączoną instrukcją lub opisem na stronach.

Maksymalna ilość miejsca potrzebna do implementacji IPv6 na tym systemie, to 23 Mb wolej przestrzeni na dysku. Obecnie dostępne są następujące komendy i aplikacje pracujące pod IPv6:

- *finger server i client,*
- *getent,*
- *inetd,*
- *telnet server i client,*
- *ftp sever i client, tftp server i client,*
- *rdate sever i client,*
- *sendmail, mconnect,*
- *rlogin server i client,*
- *rsh server i client, rcp,*
- *rdist.*

Serwisy pracujące pod IPv6:

- *DNS server (bind), DNS client-side lookup,*
- *etc/inet/ipnodes support,*
- *name service switch support (nsswitch.conf),*
- *NIS/NIS+ support for IPv6 addresses.*

Poniżej znajduje się spis programów administracyjnych obecnie dostępnych pod IPv6:

- *netstat support for IPv6 information,*
- *snoop program,*
- *ping program,*
- *route program,*
- *ifconfig program,*
- *nslookup,*
- *traceroute IPv6 extensions.*

5.2.2. Cisco [8i,12i]

Implementacja protokołu IPv6 w środowisku CISCO jest obecnie w fazie rozwoju. Pomimo tego wersja eksperymentalne nowego protokołu jest dostępna na wiele platform Cisco. Firma ostrzega, że nie jest to jeszcze oficjalna i ostateczna wersja implementacji, ponieważ sam standard protokołu IPv6 wciąż ulega modyfikacjom.

Obecnie IPv6 jest wspierany na takie platformy jak: C1000, C1600, C2500, C36x0, C4x00 i C7x00 i obsługuje większość cech nowego protokołu, między innymi routing statyczny, filtracja pakietów, tunelowanie, ICMPv6, translacje IPv4 do IPv6. Są również dostępne podstawowe aplikacje oraz komendy administracyjne.

Należy zwrócić uwagę, że firma Cisco brała udział w wielu konferencjach i pracach związanych nad stworzeniem standardu protokołu IPv6, a serwer <http://www.ipv6.com/> pracuje właśnie pod kontrolą routera tej firmy. Została wydana książka "*Internetworking IPv6 With Cisco Routers*" opisująca dokładnie protokół IPv6 oraz stan implementacji w systemach Cisco (<http://www.ipv6.com/>).

5.2.3. Windows [3i,4i,7i]

Wdrożenie IPv6 w środowisku Windows jest jeszcze w fazie prób i testów. Istnieją wersje eksperymentalne implementacji protokołu dla Windows 2000 Server i Windows 2000 Advanced Server. Windows 2000 Professional wspiera IPv6 jedynie w wersji angielskiej, a na stronach firmowych można znaleźć informacje o tym, że Microsoft nie ma planów napisania oprogramowania wspierającego nowy protokół w innych wersjach językowych tego systemu. Szczątkowe implementacje IPv6 dostępne są również dla Windows NT wersji 4.0 lub 5.0. Windows 2000 wersja beta nie wspiera nowego protokołu.

Dostępne aplikacje i narzędzia współpracujące z IPv6 są różne w zależności od zastosowanej wersji systemu. Są to głównie komendy administracyjne takie jak *ping6*, *tracert6*, *ipv6* (główne narzędzie do ręcznego konfigurowania IPv6), *6to4cfg* (używane do automatycznej konfiguracji IPv6 over IPv4) i inne, jak również klient ftp, telnet, serwer ftpd. *Microsoft Internet Explorer* wspiera IPv6 w wyżej wymienionych wersjach systemu Windows 2000. Na stronach firmy Microsoft znajdują się informacje, że nowy protokół będzie w pełni zaimplementowany w kolejnych wersjach tego systemu.

5.3. Implementacje "public domain"

5.3.1. Linux [5i]

Obsługa IPv6 jest dostępna w systemach Linux, które pracują na jądrach nowszych od 2.1.8. Aby uruchomić protokół IPv6, należy zainstalować podstawowe pakiety wspierające IPv6 oraz dokonać odpowiednich zmian w konfiguracji. Dokładniejszy opis znajduje się na stronach polskiego 6bone: <http://www.6bone.pl/>

Dla dystrybucji RedHat pakiety te dostępne są również jako RPMy pod adresem <http://www.pld.org.pl/>

Obecnie dostępne są następujące aplikacje wspierające IPv6 pod Linux-em:

btunnel, *BitchX*, *apache* (serwer web oraz proxy serwer), *bind*, *boa* (web serwer), *cfingerd* (serwer dla usługi finger), *cvs* (wsparcie IPv6 dla pservera), *ellipse* (anonimowy serwer FTP), *exim* (agent transferu poczty (MTA)), *fetchmail* (program do ściągania poczty z serwerów POP3, IMAP itp.), *ffingerd* (serwer fingera), *ftpd-BSD* (serwer ftp pochodzący z OpenBSD), *heimdal* (dystrybucja kerberos 5), *inet6-apps* (inetd, ping, ftp, ftpd), *inn* (serwer USENET), *ircii* (klient usługi IRC), *ircd* (serwer usługi IRC), *irssi* (klient usługi IRC), *leafnode* (serwer USENET), *lftp* (wspiera IPv6 zarówno po ftp jak i http), *libpcap*, *lsof* (program do śledzenia procesów w systemie), *lynx*, *mpg123*, *mrt* (narzędzie do routingu dynamicznego), *ncftp*, *openssh* (wersja free SSH), *qmail* (agent transferu poczty (MTA)), *qpopper*, *pftp* (program do przesyłania plików - wykorzystuje własny protokół), *postfix* (agent transferu poczty (MTA)), *ppp* (daemon pppd), *rlinead* (następca inetd'a), *rsync*, *sendmail* (agent transferu poczty (MTA)), *solidpop3d* (daemon POP3), *squid*, *ssh*, *strace*, *tcpd* (konkurencja dla tcp_wrappers), *tcpdump*, *tcp_wrappers*, *telnet*, *tin* (czytnik wiadomości USENET), *wget*, *whois*, *wu-ftpd*, *zebra* (narzędzie do routingu dynamicznego), *zmailer* (agent transferu poczty (MTA)).

W przygotowaniu do pracy pod protokołem IPv6 są *gated-ipv6*, *python* oraz *apache-perl*.

6. Projekt sieci testowej w środowisku MSK LODMAN

6.1. Założenia projektowe

Do dyspozycji mam dwa komputery PC klasy Pentium oraz switch firmy Centre COM 8324 SL. Testy będą polegały na porówniu wydajności protokołu IPv6 oraz IPv4 poprzez przesyłanie danych, wykorzystując do tego zarówno system operacyjny Linux jak i Windows. Przetestuję poprawność działania i możliwości konfiguracji platform Windows i Linux, budując wewnętrzną sieć testową IPv6 oraz podłączając ją do ogólnopolskiej sieci 6BONE. Podczas testów będę obserwował zachowanie się aplikacji współpracujących z nowym protokołem, jak również zachowanie się systemów i testowanych przeze mnie urządzeń.

Komputery PC to Pentium K6 o procesorach 333 MHz, każdy po 64 MB pamięci RAM. Dyski twarde w każdym z nich to QUANTUM FIREBALL EX6.4. Prędkość dysków twardych sprawdzona Linuxową aplikacją *hdparm* wynosi 12,57 MB/sec dla każdego z komputerów:

```
ipv6_pc-3:~# hdparm -t -T /dev/hda
```

```
/dev/hda:
```

```
Timing buffer-cache reads: 128 MB in 3.52 seconds=36.36 MB/sec
```

```
Timing buffered disk reads: 64 MB in 5.09 seconds=12.57 MB/sec
```

```
ipv6_pc-3:~#
```

oraz dla drugiego komputera:

```
ipv6_pc-2:~# hdparm -t -T /dev/hda
```

```
/dev/hda:
```

```
Timing buffer-cache reads: 128 MB in 3.47 seconds=36.89 MB/sec
```

```
Timing buffered disk reads: 64 MB in 5.09 seconds=12.57 MB/sec
```

```
ipv6_pc-2:~#
```

Przy zastosowaniu kart sieciowych prędkości 10 MB/sec przepustowość nie będzie ograniczona przez szybkość pracy dysków.

Ewentualne błędy w działaniu sprzętu i oprogramowania, lub niestabilne zachowanie się systemów operacyjnych, będzie uwzględnione w dalszej części pracy.

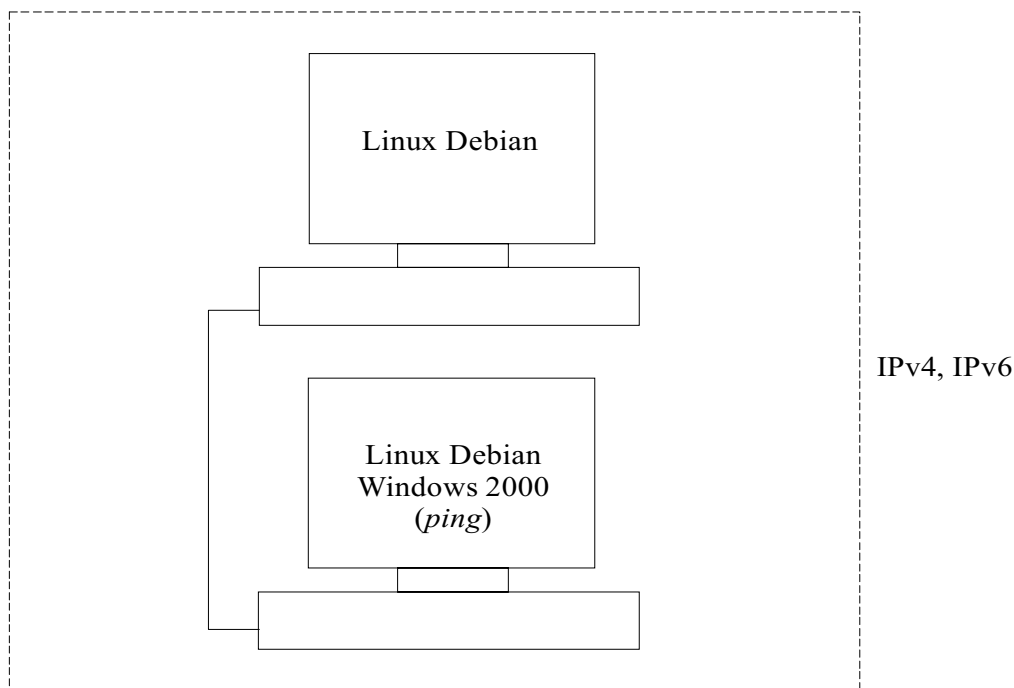
6.2. Specyfikacja testów i kryteria wyników

6.2.1. Poprawność działania sieci testowej MSK LODMAN

Test będzie miał na celu zbadanie poprawności działania i jakości routingu w sieci testowej MSK LODMAN.

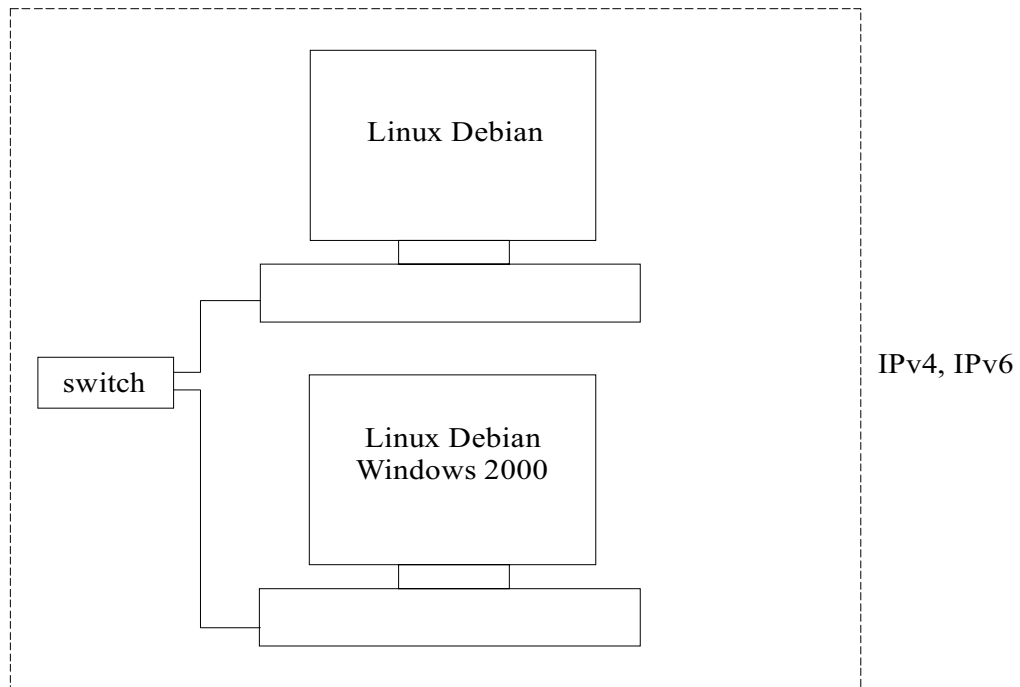
Procedura testowa

Komendą *ping* zbadam czasy przebycia pakietów między komputerami działającymi wewnątrz sieci testowej MSK LODMAN (Rys. 6.1.). Testy wykonam dla protokołu IPv4, a później dla IPv6, wykorzystując system Linux oraz Windows.



Rys. 6.1. Badanie czasów powrotów pakietów ICMP przy wykorzystaniu połączenia bezpośredniego.

Następnie komputery połączę ze sobą za pomocą switcha (Rys. 6.2.). Zbadam w ten sposób wpływ switcha na czasy powrotów pakietów kontrolnych ICMP.



Rys. 6.2. Badanie czasów powrotów pakietów ICMP przy wykorzystaniu switcha.

Akceptacja wyników

Oczekuję, że wyniki średnich czasów powrotów pakietów ICMP dla przypadków wykorzystania protokołu IPv6, jak i IPv4 oraz platform Linux i Windows będą poniżej 1ms.

Oczekuję również, że liczba straconych pakietów w wykonywanych testach będzie 0. Oczekuję też, że zainstalowanie switcha nie wpłynie na wynik doświadczenia więcej niż w 0,5%.

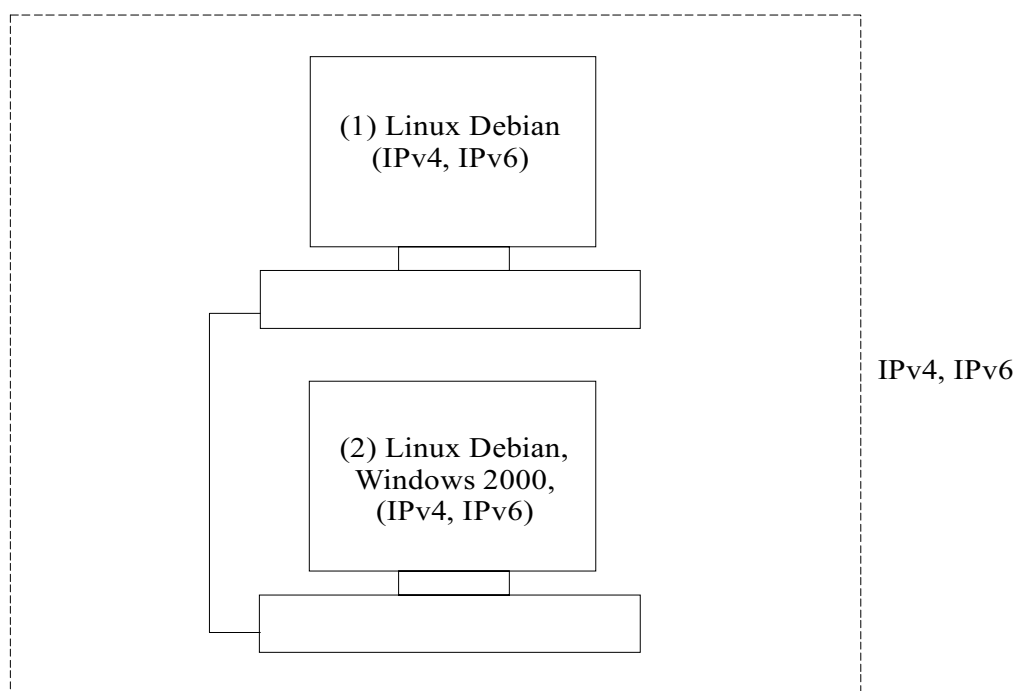
6.2.2. Porównanie prędkości protokołów przy przesyłaniu dużych plików

Test będzie polegał na porównaniu prędkości przesyłania plików o wielkościach 10Mb, 25Mb, 50Mb, 100Mb, 150Mb, 200Mb, 250Mb, 300Mb, 350Mb oraz 400Mb przy użyciu protokołów IPv4 i IPv6 oraz przy wykorzystaniu platformy Windows i Linux. Sprawdzę również, jak zainstalowany przez mnie switch wpłynie na różnice w prędkości przesyłania danych.

Procedura testowa

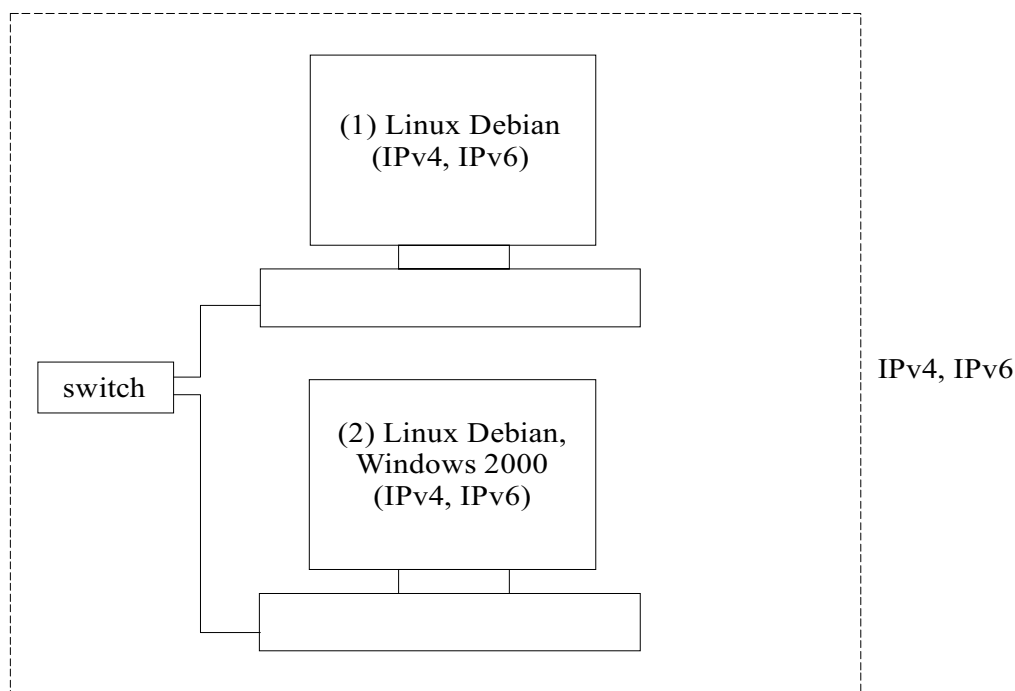
Test będzie przebiegał w dwóch etapach:

1) W stworzonej przeze mnie sieci takiej, jak na rysunku poniżej, porównam czas oraz prędkości przesyłu tej samej wielkości danych, wykorzystując do tego najpierw protokół IPv4, a potem IPv6. Jako serwer *ftpd* pracuje komputer (1) i system Linux Debian, a jako stacja robocza komputer (2) i system Linux Debian oraz Windows 2000.



Rys. 6.3. Wykorzystanie połączenia bezpośredniego komputerów dla testów przesyłu dużych plików.

2) Takie same testy przesyłania plików zostaną wykonane dla sytuacji, gdy pomiędzy komputerami zainstalowany będzie switch. Sprawdzę w ten sposób, czy dodatkowe urządzenie w postaci switcha wprowadza zmiany prędkości przy przesyłaniu danych, podczas wykorzystywania IPv4 jak i IPv6.



Rys. 6.4 Wykorzystanie switcha dla testów przesyłu dużych plików.

Eliminacja błędów

Aby sprawdzić, czy wydajność komputerów nie wpływa na wyniki doświadczenia, dla każdego z testów uruchomię aplikację *top* dla Linux lub *Task Manager* dla Windows, aby sprawdzić jaki procent czasu procesora zajmują klient *ftp* oraz serwer *ftpd*.

Jeżeli podczas uruchomianych testów okaże się, że przynajmniej jeden z systemów obciążony jest więcej niż w 75% stwierdzę, że wydajność komputerów jest niewystarczająca do wykonania testu.

Jeżeli komputery będą obciążone w mniej niż 75% będę mógł stwierdzić, że wydajność komputerów nie wpływa na prędkość przesyłu danych.

W celu minimalizacji błędów wynikających z ewentualnych innych zakłóceń, każdy test powtórzę pięciokrotnie, a w tabelach podam średnie czasy oraz średnie prędkości przesyłu.

Akceptacja wyników

Oczekuję, że czasy i prędkość przesyłania plików przy wykorzystaniu protokołu IPv6 nie będą różniły się więcej niż 5% w porównaniu z protokołem IPv4.

Oczekuję również, że dla różnych platform wyniki testów nie będą różniły się więcej niż 5%.

Dla przypadku, gdy pomiędzy komputerami zainstalowany zostanie switch oczekuję, że czasy i prędkości przesyłu danych nie będą się różniły więcej niż 0,5%.

6.2.3. Porównanie prędkości protokołów przy przesyłaniu wielu plików

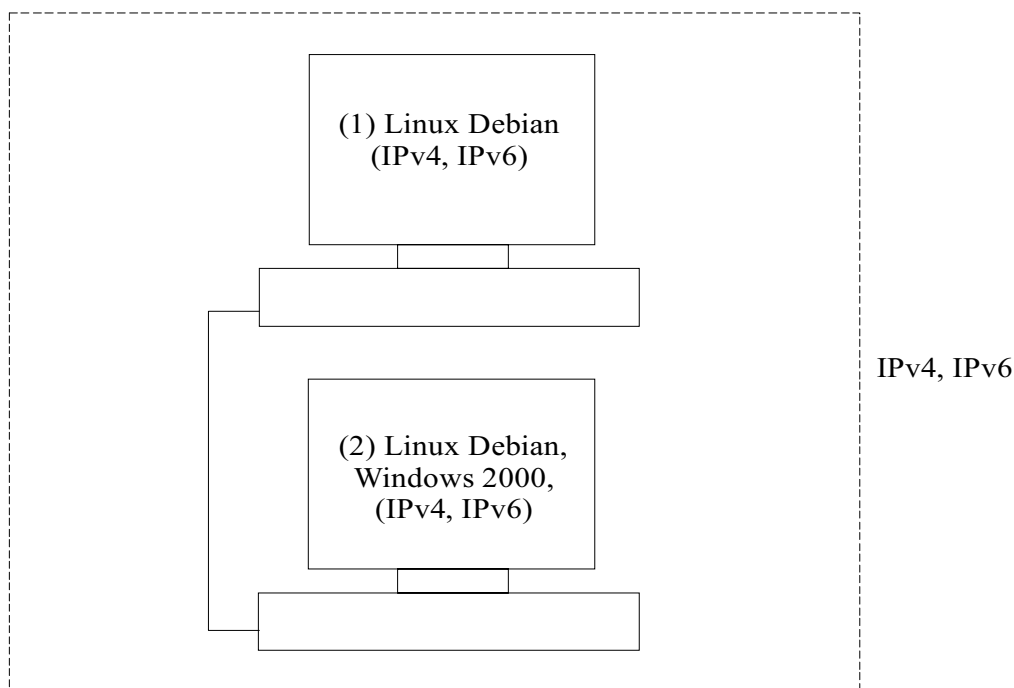
Test będzie polegał na porównaniu prędkości przesyłania dużej ilości małych plików przy użyciu protokołów IPv4 i IPv6 oraz przy wykorzystaniu platformy Windows i Linux. Do tego celu wygeneruję 1000 plików o wielkości kolejno 10b, 100b, 1Kb, 10Kb, 20Kb, 40Kb, 60Kb, 80Kb, 100Kb, 120Kb, 140Kb, 160Kb, 180Kb oraz 200Kb i prześlę je pomiędzy komputerami, notując czas oraz średnią prędkość przesyłu. Sprawdzę również, jak zainstalowany przeze mnie switch wpłynie na różnice w prędkości przesyłania danych.

Procedura testowa

Test będzie przebiegał w dwóch etapach:

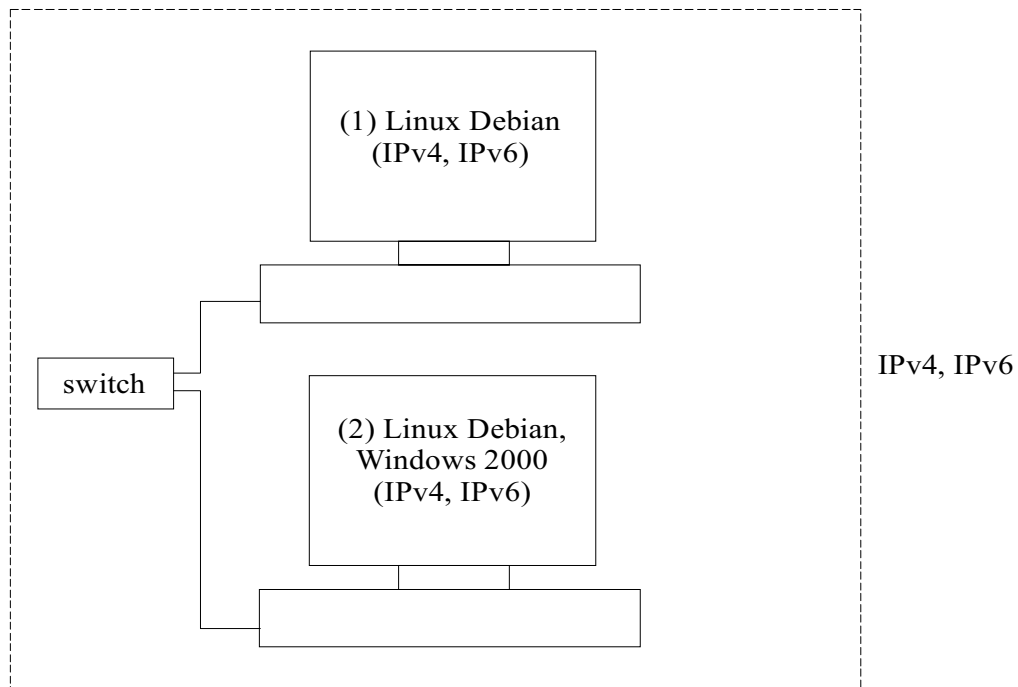
1)

W stworzonej przeze mnie sieci takiej, jak na rysunku poniżej, porównam czasy oraz średni transfer przesyłu wielu plików tej samej wielkości, wykorzystując do tego najpierw protokół IPv4, a potem IPv6. Jako serwer *ftpd* pracuje komputer (1) i system Linux Debian, a jako stacja robocza komputer (2) i system Linux Debian oraz Windows 2000.



Rys. 6.5. Wykorzystanie połączenia bezpośredniego dla testów przesyłu wielu plików.

2) Takie same testy przesyłania plików zostaną wykonane dla sytuacji, gdy pomiędzy komputerami zainstalowany będzie switch. Sprawdzę w ten sposób, czy dodatkowe urządzenie w postaci switcha wprowadza zmiany prędkości przy przesyłaniu dużej ilości plików.



Rys. 6.6. Wykorzystanie switcha dla testów przesyłu wielu plików.

Eliminacja błędów

Aby sprawdzić, czy wydajność komputerów nie wpływa na wyniki doświadczenia, dla każdego z testów uruchomię aplikację *top* dla Linux lub *Task Manager* dla Windows, aby sprawdzić, jaki procent czasu procesora zajmują klient *ftp* oraz serwer *ftpd*.

Jeżeli podczas uruchomianych testów okaże się, że przynajmniej jeden z systemów obciążony jest więcej niż w 75% stwierdzę, że wydajność komputerów jest niewystarczająca do wykonania testu.

Jeżeli komputery będą obciążone w mniej niż 75%, będę mógł stwierdzić, że wydajność komputerów nie wpływa na prędkość przesyłu danych.

W celu minimalizacji błędów wynikających z ewentualnych innych zakłóceń, każdy test powtórzę pięciokrotnie, a w tabelach podam średnie czasy oraz średnie prędkości przesyłu.

Akceptacja wyników

Oczekuję, że czasy i prędkość przesyłania plików przy wykorzystaniu protokołu IPv6 nie będą różniły się więcej niż 5% w porównaniu z protokołem IPv4.

Oczekuję również, że dla różnych platform wyniki testów nie będą różniły się więcej niż 5%.

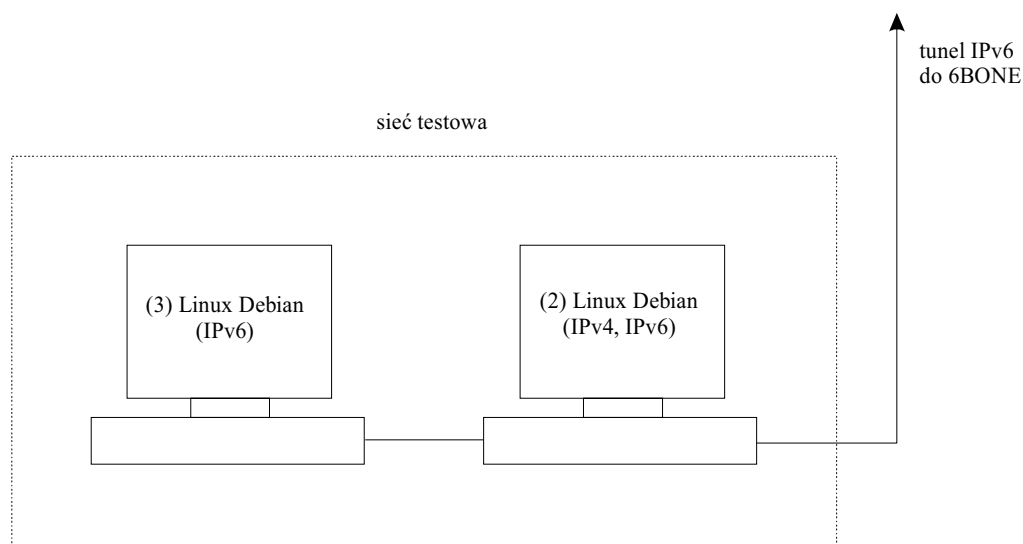
Dla przypadku, gdy pomiędzy komputerami zainstalowany zostanie switch oczekuję, że czasy i prędkości przesyłu danych nie będą się różniły więcej niż 0,5%.

6.2.4. Routing IPv6 w sieci 6BONE

Test będzie miał na celu zbadanie poprawności działania routingu w sieci 6BONE.

Procedura testowa

Komendą *tracroute* zbadam trasę oraz porównam czasy przebycia pakietów z komputera działającego w sieci testowej MSK LODMAN do serwera 6bone-gw.icm.edu.pl, wykorzystując do tego protokół IPv6 oraz IPv4. Testy wykonam dla platform Windows i Linux. Do tego celu wykorzystam sieć testową skonfigurowaną do współpracy z nowym protokołem oraz skonfigurowany tunel do sieci 6BONE. Każdy test wykonam dziesięciokrotnie, podając wyniki oraz wartość średnią w tabeli.



Rys. 6.7. Sieć testowa połączona tunelem IPv6 do polskiego 6BONE.

Akceptacja wyników

Droga pakietów do serwerów w ogólnopolskiej sieci 6BONE powinna przebiegać przez komputer (2), który pełni rolę routera, a następnie do głównego serwera 6bone-gw.icm.edu.pl.

Oczekuję, że czasy przebycia pakietów z komputera (1) do głównego serwera 6bone-gw.icm.edu.pl nie będą różniły się więcej niż 5% przy wykorzystaniu protokołów IPv6 i IPv4.

Oczekuję również, że czasy te nie będą różniły się więcej niż 5% przy wykorzystaniu różnych platform.

6.2.5. Stabilność i poprawność działania oprogramowania oraz sprzętu

Procedura testowa

Przez cały czas wykonywania testów będę obserwował zachowanie się i stabilność działania systemów operacyjnych oraz poszczególnych aplikacji i zainstalowanych serwisów. Badaniu poddam również zachowanie się switcha, obserwując diody kontrolne na płycie czołowej i informując o ewentualnych „dziwnych” zachowaniach. Według informacji uzyskanych od kolegów z list dyskusyjnych poświęconych IPv6, zarówno polskich jak i ogólnoswiatowych, może występować „migotanie” wszystkich diod przypominających wysyłanie pakietów broadcast-owych pod IPv4.

Akceptacja wyników

Jeżeli podczas wykonywania testów aplikacje i serwisy będą zachowywały się poprawnie, a cały system będzie działał stabilnie stwierdzę, że zainstalowanie i skonfigurowanie nowego protokołu oraz nowych aplikacji wypierających ten protokół nie wpłynęło negatywnie na stabilność i poprawność działania systemu.

6.3. Konfiguracja sprzętu i oprogramowania

Na komputerach PC zainstalowany został system operacyjny Linux Debian oraz na jednym z komputerów, równolegle na drugiej partycji, system Windows 2000 Professional w angielskiej wersji językowej.

W celu stworzenia tunelu do sieci 6BONE wystąpiłem o przyznanie puli adresów testowych do ICM. ICM jest regionalnym koordynatorem pNLA (pseudo Next Level Aggregation) i zajmuje się przydziałem adresów IPv6. Do mojej dyspozycji otrzymałem 48-bitową pulę adresów 3ffe:8010:81::/48 oraz adresy IPv6 i IPv4 drugiego końca tunelu w ICM, potrzebne do dalszych testów. Dla moich potrzeb z puli otrzymanych adresów wydzieliłem sieć 3ffe:8010:81::1:0/124. Komputerom PC przydzieliłem adresy 3ffe:8010:81::1:2/124 (*ipv6_pc-2*) oraz 3ffe:8010:81::1:3/124 (*ipv6_pc-3*).

MTU sieci testowej zarówno dla protokołu IPv4, jak i IPv6 wynosi 1500 bajtów.

6.3.1. Sieć testowa IPv6 dla systemu Linux Debian 2.2 [4,9i,10i]

Na dwóch komputerach PC zainstalowany został system Linux Debian 2.2 z wersją jądra 2.2.16. Aby jądro wspierało IPv6, należało podczas jego kompilacji uaktywnić następujące opcje:

```
Code maturity level options:
  [*] Prompt for development and/or incomplete code/drivers
Networking options:
  [*] IP: tunneling
  [*] The IPv6 protocol (EXPERIMENTAL)
  [*] IPv6: enable EUI-64 token format (NEW)
  [*] IPv6: disable provider based addresses (NEW)
```

W celu przygotowania systemu do konfiguracji z nowym protokołem, należało dokonać pewnych zmian w plikach konfiguracyjnych systemu.

Zgodnie z dokumentacją RFC 2292 należało sprawdzić i ewentualnie dodać następujące definicje protokołów do pliku */etc/protocols*.

```

hopopt      0    # hop-by-hop options for ipv6
ipv6        41    # ipv6
ipv6-route  43    # routing header for ipv6
ipv6-frag   44    # fragment header for ipv6
esp         50    # encapsulating security payload for ipv6
ah          51    # authentication header for ipv6
ipv6-icmp   58    # icmp for ipv6
ipv6-nonxt  59    # no next header for ipv6
ipv6-opts   60    # destination options for ipv6

```

Do pliku `/etc/hosts` należało dodać odpowiednie linijki definiujące interfejs loopback oraz podstawowe adresy multicastowe. Dodałem również adresy komputerów działających w naszej testowej sieci lokalnej:

```

::1          ipv6-localhost ipv6-loopback
fe00::0      ipv6-localnet
ff00::0      ipv6-mcastprefix
ff02::1      ipv6-allnodes
ff02::2      ipv6-allrouters
ff02::3      ipv6-allhosts
3ffe:8010:81::1:1  ipv6_cisco
3ffe:8010:81::1:2  ipv6_pc-2
3ffe:8010:81::1:3  ipv6_pc-3

```

Podstawowym oprogramowaniem do konfiguracji interfejsów, które należało zainstalować, a które będzie używane w dalszej części konfiguracji, jest `iproute2` dostępny w internecie, np. pod adresem: <ftp://ftp.icm.edu.pl/vol/rzm0/linux-iproute/>. Dodatkowo zainstalowane zostały `lftp-2.1.10` - jako klient ftp, `ftpd-BSDv0.3.2` - jako serwer ftpd, `xinetd-2.1.8.8p2`, `tcpdump-2000-04-30`. Wszystkie powyższe pakiety oraz wiele innych, dostępne są na stronach Petera Bieringera (<http://www.bieringer.de/linux/IPv6/>).

Konfiguracja systemu Linux z użyciem skryptów zawartych w pakiecie `iproute2` polega na odpowiednim użyciu komendy `ip`. Wykorzystując przestrzeń adresową przeznaczoną dla sieci testowej, skonfigurowałem interfejs `eth0`, nadając mu adres `3ffe:8010:81::1:2/124`:

```
ipv6_pc-2:~# ip addr add 3ffe:8010:81::1:2/124 dev eth0
ipv6_pc-2:~#
```

Poprawność działania interfejsu sprawdziłem komendą *ping*:

```
ipv6_pc-2:~# ping6 3ffe:8010:81::1:2
PING 3ffe:8010:81::1:2(ipv6_pc-2) 56 data bytes
64 bytes from ipv6_pc-2: icmp_seq=0 time=0.1 ms
64 bytes from ipv6_pc-2: icmp_seq=1 time=0.1 ms
64 bytes from ipv6_pc-2: icmp_seq=2 time=0.1 ms

--- 3ffe:8010:81::1:2 ping6 statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.1 ms
ipv6_pc-2:~#
```

Komputer został poprawnie skonfigurowany do współpracy z protokołem IPv6. Analogicznie takie same kroki wykonałem na drugim komputerze. Po fizycznym połączeniu komputerów ze sobą, sprawdziłem czy komputery te widzą się wzajemnie w stworzonej przeze mnie sieci:

```
ipv6_pc-2:~# ping6 3ffe:8010:81::1:3
PING 3ffe:8010:81::1:3(ipv6_pc-3) 56 data bytes
64 bytes from ipv6_pc-3: icmp_seq=0 time=2.5 ms
64 bytes from ipv6_pc-3: icmp_seq=1 time=0.8 ms
64 bytes from ipv6_pc-3: icmp_seq=2 time=0.8 ms

--- 3ffe:8010:81::1:3 ping6 statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.7/1.2/2.5 ms
ipv6_pc-2:~#
```

Poprawność konfiguracji potwierdził program *tcpdump* uruchomiony na jednym z komputerów:

```
18:56:18.971459 ipv6_pc-2 > ipv6_pc-3: icmp6: echo request
18:56:18.971596 ipv6_pc-3 > ipv6_pc-2: icmp6: echo reply
18:56:19.965732 ipv6_pc-2 > ipv6_pc-3: icmp6: echo request
18:56:19.965836 ipv6_pc-3 > ipv6_pc-2: icmp6: echo reply
18:56:20.965715 ipv6_pc-2 > ipv6_pc-3: icmp6: echo request
18:56:20.965827 ipv6_pc-3 > ipv6_pc-2: icmp6: echo reply
```

Analogiczne wyniki otrzymałem uruchamiając na komputerze *ipv6_pc-3* ping do *ipv6_pc-2*:

```
ipv6_pc-3:~# ping6 3ffe:8010:81::1:2
PING 3ffe:8010:81::1:2(ipv6_pc-2) 56 data bytes
64 bytes from ipv6_pc-2: icmp_seq=0 time=0.2 ms
64 bytes from ipv6_pc-2: icmp_seq=1 time=0.1 ms
64 bytes from ipv6_pc-2: icmp_seq=2 time=0.1 ms

--- 3ffe:8010:81::1:2 ping6 statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.2 ms
ipv6_pc-3:~#
```

Wynik działania aplikacji *tcpdump*:

```
18:59:16.393129 ipv6_pc-3 > ipv6_pc-2: icmp6: echo request
18:59:16.393196 ipv6_pc-2 > ipv6_pc-3: icmp6: echo reply
18:59:17.391092 ipv6_pc-3 > ipv6_pc-2: icmp6: echo request
18:59:17.391147 ipv6_pc-2 > ipv6_pc-3: icmp6: echo reply
18:59:18.391077 ipv6_pc-3 > ipv6_pc-2: icmp6: echo request
18:59:18.391130 ipv6_pc-2 > ipv6_pc-3: icmp6: echo reply
```

6.3.2. Tunel IPv6 do ogólnopolskiej sieci 6BONE [4,9i]

Tunel do sieci 6BONE zostanie uruchomiony na komputerze *ipv6_pc-2*. Komendą *ip* skonfigurowałem tunel do ICM, podając jako punkt początkowy adres IPv4 komputera *ipv6_pc-2* (212.51.192.66), a jako adres końcowy główny węzeł IPv6 w ICM (6bone-gw.6bone.pl - 193.219.28.246). Nazwa tunelu po naszej stronie to *MAN*. Następnie uaktywniłem tunel *MAN*:

```
ipv6_pc-2:~# ip tunnel add man mode dit local 212.51.192.66
remote 193.219.28.246
ipv6_pc-2:~# ip link set man up
ipv6_pc-2:~#
```

Utworzony został nowy interfejs *MAN*, któremu przydzieliłem adres 3ffe:8010:81::2 (adres IPv6 mojego końca tunelu oraz końca tunelu w ICM został narzucony przez ICM):

```
ipv6_pc-2:~# ip addr add 3ffe:8010:81::2/126 dev man
ipv6_pc-2:~#
```

Poprawność skonfigurowania lokalnego interfejsu potwierdza komenda *ping*:

```
ipv6_pc-2:~# ping6 3ffe:8010:81::2
PING 3ffe:8010:81::2 56 data bytes
64 bytes from ipv6_pc-2: icmp_seq=0 time=0.1 ms
64 bytes from ipv6_pc-2: icmp_seq=1 time=0.1 ms
64 bytes from ipv6_pc-2: icmp_seq=2 time=0.1 ms

--- 3ffe:8010:81::2 ping6 statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.1 ms
ipv6_pc-2:~#
```

Poprawność skonfigurowania tunelu potwierdziła komenda *ping* wykonana do węzła IPv6 w ICM:

```
ipv6_pc-2:~# ping6 3ffe:8010:81::1
PING 3ffe:8010:81::1 (6bone-gw.6bone.pl) 56 data bytes
64 bytes from 6bone-gw.6bone.pl: icmp_seq=0 time=10.1 ms
64 bytes from 6bone-gw.6bone.pl: icmp_seq=1 time=12.5 ms
64 bytes from 6bone-gw.6bone.pl: icmp_seq=2 time=9.1 ms

--- 3ffe:8010:81::1 ping6 statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 9.1/10.6/12.5 ms
ipv6_pc-2:~#
```

Kolejnym etapem konfiguracji było dodanie do tablicy routingu domyślnej trasy prowadzącej do drugiego końca tunelu, do ICM:

```
ipv6_pc-2:~# ip route add 3ffe::/16 dev man
ipv6_pc-2:~#
```

Ostatnim etapem konfiguracji komputera *ipv6_pc-2* było uaktywnienie routingu IPv6:

```
ipv6_pc-2:~# cat 1 >/proc/sys/net/ipv6/conf/all/forwarding
ipv6_pc-2:~#
```

Aby poprawnie skonfigurować routing na komputerze *ipv6_pc-3*, działającym w testowej sieci lokalnej, należało jeszcze do jego tablicy routingu dodać domyślną trasę wskazującą na komputer *ipv6_pc-2*:

```
ipv6_pc-3:~# ip route add 3ffe::/16 via 3ffe:8010:81::1:2 dev
eth0
ipv6_pc-3:~#
```

Systemy Linux na obu komputerach zostały skonfigurowane do pracy z protokołem IPv6, a na jednym z komputerów skonfigurowany został tunel do ogólnopolskiej sieci 6BONE.

6.3.3. Sieć testowa IPv6 dla systemu Windows 2000 [7i,11i]

Na stronach www firmy Microsoft znajdują się informacje o dostępności nowego protokołu dla różnych wersji systemu Windows. Aby móc zainstalować pakiety, które umożliwiają konfigurację IPv6, należało na poprawnie zainstalowanym systemie zainstalować Service Pack 1. Następnie ze stron Microsoftu należało ściągnąć i zainstalować odpowiednie oprogramowanie, pozwalające wspierać nowy protokół. W skład pakietu wchodzi następujące aplikacje:

- *ipv6.exe* - konfiguracja ipv6, wyświetlanie informacji o skonfigurowanych interfejsach, konfiguracja routingu,
- *ping6.exe* - odpowiednik *ping.exe* dla IPv4,
- *tracert6.exe* - odpowiednik *tracert.exe* dla IPv4,
- *ttcp.exe* - używany do wysyłania pakietów TCP lub UDP pomiędzy dwoma komputerami,
- *6to4cfg.exe* - oprogramowanie do automatycznej konfiguracji IPv6 over IPv4,
- *ipsec6.exe* - konfiguracja reguł bezpieczeństwa dla IPv6,
- *checkv4.exe* - program skanujący kod źródłowy danej aplikacji i wskazujący zmiany konieczne do wykonania celem współpracy z IPv6.

Ponadto pakiet zawierający wsparcie dla IPv6 podmienia pewne biblioteki systemowe oraz między innymi aplikacje *ftp*. Po zainstalowaniu Service Pack 1 oraz pakietu umożliwiającego wsparcie dla IPv6, należało skonfigurować sieć IPv4, co jest niezbędne do rozpoczęcia pracy z IPv6. Po skonfigurowaniu IPv4, system został przygotowany do konfiguracji nowego protokołu, która opisana została poniżej.

Najpierw należało uruchomić możliwość konfiguracji IPv6, poprzez odpowiednie użycie komendy *net.exe*:

```
C:\>net start tcpip6
```

```
The Microsoft IPv6 Protocol Driver service was started  
successfully.
```

```
C:\>
```

Spowodowało to utworzenie podstawowych interfejsów, w tym interfejsu *Loopback Pseudo-Interface*. Jego działanie można potwierdzić komendą *ping6*:

```
C:\>ping6 -n 3 ::1

Pinging ::1 with 32 bytes of data:

Reply from ::1: bytes=32 time<1ms
Reply from ::1: bytes=32 time<1ms
Reply from ::1: bytes=32 time<1ms

C:\>
```

Cała konfiguracja nowego protokołu sprowadzała się do odpowiedniego użycia komendy *ipv6*, którą na początku skonfigurowałem odpowiednio interfejs nr. 4, nadając mu, zgodnie z wcześniejszymi ustaleniami, adres *3ffe:8010:81::1:3* :

```
C:\>ipv6 adu 4/3ffe:8010:81::1:3
C:\>
```

Użycie komendy *ipv6 if [if#]* pozwoliło na skontrolowanie poprawności skonfigurowania interfejsu:

```
C:\>ipv6 if 4
Interface 4 (site 1): Local Area Connection
  uses Neighbor Discovery
  link-level address: 00-10-4b-b4-92-3a
    preferred address 3ffe:8010:81::1:3, infinite/infinite
    preferred address fe80::210:4bff:feb4:923a, infinite/
infinite
    multicast address ff02::1, 1 refs, not reportable
    multicast address ff02::1:ffb4:923a, 1 refs, last reporter
    multicast address ff02::1:ff01:3, 1 refs, last reporter
  link MTU 1500 (true link MTU 1500)
  current hop limit 128
  reachable time 21000ms (base 30000ms)
  retransmission interval 1000ms
  DAD transmits 1
C:\>
```

Interfejsowi nr. 4 przyznany został adres 3ffe:8010:81::1:3. Komenda *ping6* potwierdziła poprawność skonfigurowania interfejsu:

```
C:\>ping6 -n 3 3ffe:8010:81::1:3

Pinging 3ffe:8010:81::1:3 with 32 bytes of data:

Reply from 3ffe:8010:81::1:3: bytes=32 time<1ms
Reply from 3ffe:8010:81::1:3: bytes=32 time<1ms
Reply from 3ffe:8010:81::1:3: bytes=32 time<1ms

C:\>
```

Następnie skonfigurowałem odpowiednio routing do naszej wewnętrznej sieci testowej:

```
C:\>ipv6 rtu 3ffe:8010:81::1:0/124 4
C:\>
```

Wszystkie pakiety z adresów 3ffe:8010:81::1:0 z maską podsieci 124 kierowane będą na interfejs nr. 4:

```
C:\>ipv6 rt
3ffe:8010:81::1:0/124 -> 4 pref 0 (lifetime infinite)
::/96 -> 2 pref 0 (lifetime infinite)
C:\>
```

Komendą *ping6* sprawdziliśmy poprawność skonfigurowania sieci, próbując wysłać pakiety do komputera 3ffe:8010:81::1:2:

```
C:\>ping6 -n 3 3ffe:8010:81::1:2

Pinging 3ffe:8010:81::1:2 with 32 bytes of data:

Reply from 3ffe:8010:81::1:2: bytes=32 time<1ms
Reply from 3ffe:8010:81::1:2: bytes=32 time<1ms
Reply from 3ffe:8010:81::1:2: bytes=32 time<1ms

C:\>
```

Poprawność działania komendy potwierdziło prawidłowe skonfigurowanie interfejsu systemu Windows 2000 Professional do współpracy z nowym protokołem.

7. Wykonanie testów i interpretacja wyników

7.1. Poprawność działania sieci testowej MSK LODMAN

Wykonanie testu

Zgodnie z procedurą testową opisaną w rozdziale 6.2.1., komendą *ping* wykonaną na jednym z komputerów działających w sieci testowej sprawdziłem, jakie są czasy powrotów pakietów ICMP od drugiego komputera. Testy wykonałem dla protokołu IPv6 oraz IPv4, zarówno dla systemu Linux jak i Windows. Wyniki testu przy bezpośrednim połączeniu komputerów przedstawia poniższa tabela:

użyty system i protokół	liczba wysłanych pakietów	liczba straconych pakietów	procent straconych pakietów	min czas powrotu pakietów ICMP	max czas powrotu pakietów ICMP	średni czas powrotu pakietów ICMP
Linux IPv6	1000	0	0%	0,5 ms	2,1 ms	0,5 ms
Linux IPv4	1000	0	0%	0,3 ms	3,1 ms	0,5 ms
Windows IPv6	1000	0	0%	0,5 ms	2 ms	0,5 ms
Windows IPv4	1000	0	0%	0,3 ms	2,9 ms	0,5 ms

Rys. 7.1. Wyniki komendy ping dla połączenia bezpośredniego komputerów.

Następnie takie same testy wykonałem dla przypadku, gdy pomiędzy komputerami podłączony był switch. Wyniki doświadczenia przedstawia poniższa tabela:

użyty system i protokół	liczba wysłanych pakietów	liczba straconych pakietów	procent straconych pakietów	min czas powrotu pakietów ICMP	max czas powrotu pakietów ICMP	średni czas powrotu pakietów ICMP
Linux IPv6	1000	0	0%	0,5 ms	2,7 ms	0,5 ms
Linux IPv4	1000	0	0%	0,3 ms	2,5 ms	0,5 ms
Windows IPv6	1000	0	0%	0,5 ms	2,3 ms	0,5 ms
Windows IPv4	1000	0	0%	0,3 ms	3,1 ms	0,5 ms

Rys. 7.2. Wyniki komendy ping dla połączenia bezpośredniego komputerów.

Interpretacja wyników - wnioski

Zgodnie z przypuszczeniami (rozdział 6.2.1.), czasy otrzymane komendą *ping*, dla wersji protokołów IPv6 i IPv4 oraz dla systemów Linux i Windows, wynoszą poniżej 1ms. Procent oraz liczba straconych pakietów wynosi 0.

Wyniki testów potwierdzają poprawne skonfigurowanie i działanie sieci testowej stworzonej w MSK LODMAN, zarówno dla systemu Linux, jak i Windows.

Zainstalowanie switcha pomiędzy komputerami nie wpłynęło na jakość połączenia.

7.2. Porównanie prędkości protokołów IPv4 i IPv6

7.2.1. Przesyłanie dużych plików

Błędy pomiaru

Zgodnie z założeniami z rozdziału 6.2.2., aby sprawdzić, czy wydajność komputerów jest wystarczająca, uruchamiałem testy, jednocześnie monitorując programem *top* pod Linux oraz *Task Manager* pod Windows procent zajętości czasu procesora. Okazało się, że system Linux pracujący jako serwer był wykorzystywany w 15-25%, a system Linux pracujący jako klient w 20-30%. System Windows 2000 Professional zajmował 35-45% czasu procesora. Wyniki te były wystarczające, aby stwierdzić, że ograniczenie prędkości przesyłu danych nie jest związane z szybkością komputerów.

Wykonanie testu

1)

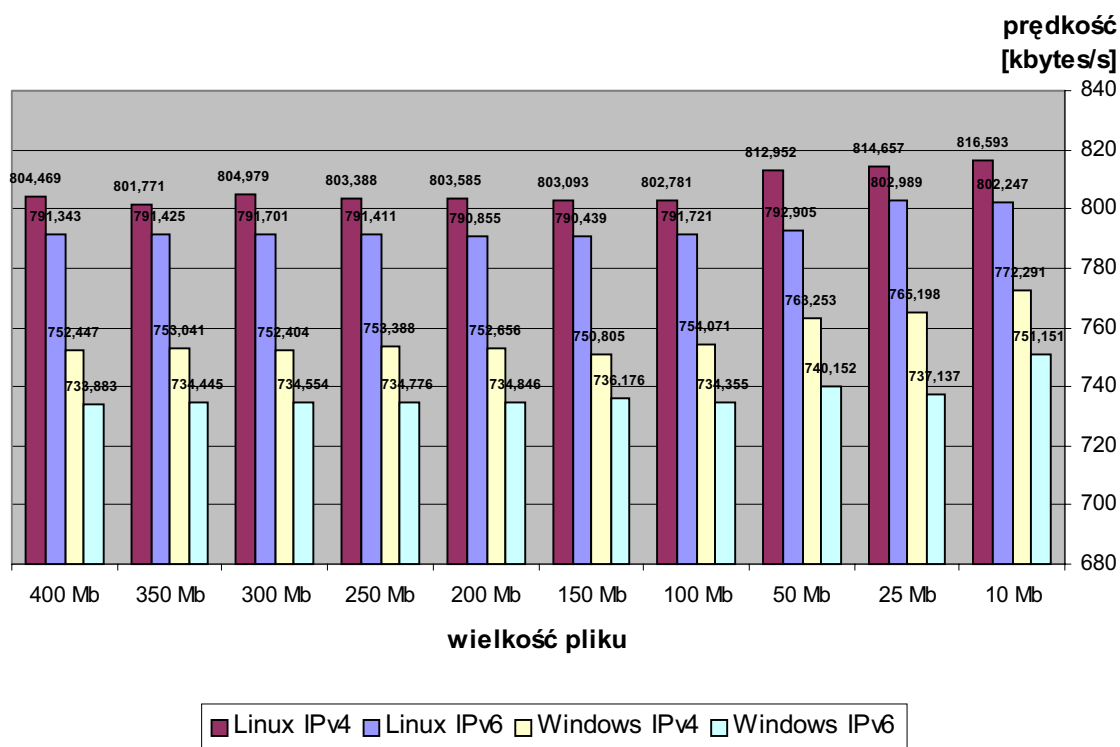
W pierwszej części testu karty sieciowe komputerów PC połączyłem bezpośrednio ze sobą za pomocą skrętki z przeplotem. Na obu komputerach uruchomiłem system Linux Debian, uprzednio skonfigurowany do współpracy z protokołem IPv4 oraz IPv6. Na jednym uruchomiłem serwer *ftpd*, a na drugim klientem *lftp*, wykorzystując protokół IPv6, przesyłałem na dysk lokalny kolejno jeden plik wielkości 10Mb, 25Mb, 50Mb, 100Mb, 150Mb, 200Mb, 250Mb, 300Mb, 350Mb oraz 400Mb. Analogiczne doświadczenie wykonałem, wykorzystując do testów protokół IPv4. Następnie jeden z komputerów uruchomiłem pod kontrolą systemu Windows 2000 Professional. Wykorzystując klienta *ftp* wykonałem takie same testy.

Wyniki średniego czasu oraz średniej prędkości przesyłania danych przy bezpośrednim połączeniu komputerów przedstawiają poniższe table (rys. 7.3. i rys. 7.5.) oraz wykresy (rys. 7.4. i 7.6):

wielkość pliku	Linux IPv6	Linux IPv4	Windows IPv6	Windows IPv4
400 Mb	791,343 kb/s	804,469 kb/s	733,883 kb/s	752,447 kb/s
350 Mb	791,425 kb/s	801,771 kb/s	734,445 kb/s	753,041 kb/s
300 Mb	791,701 kb/s	804,979 kb/s	734,554 kb/s	752,404 kb/s
250 Mb	791,411 kb/s	803,388 kb/s	734,776 kb/s	753,388 kb/s
200 Mb	790,855 kb/s	803,585 kb/s	734,846 kb/s	752,656 kb/s
150 Mb	790,439 kb/s	803,093 kb/s	736,176 kb/s	750,805 kb/s
100 Mb	791,721 kb/s	802,781 kb/s	734,355 kb/s	754,071 kb/s
50 Mb	792,905 kb/s	812,952 kb/s	740,152 kb/s	763,253 kb/s
25 Mb	802,989 kb/s	814,657 kb/s	737,137 kb/s	765,198 kb/s
10 Mb	802,247 kb/s	816,593 kb/s	751,151 kb/s	772,291 kb/s

Rys. 7.3. Średnia prędkość przesyłu plików dla połączenia bezpośredniego komputerów.

Wykres z powyższej tabeli:

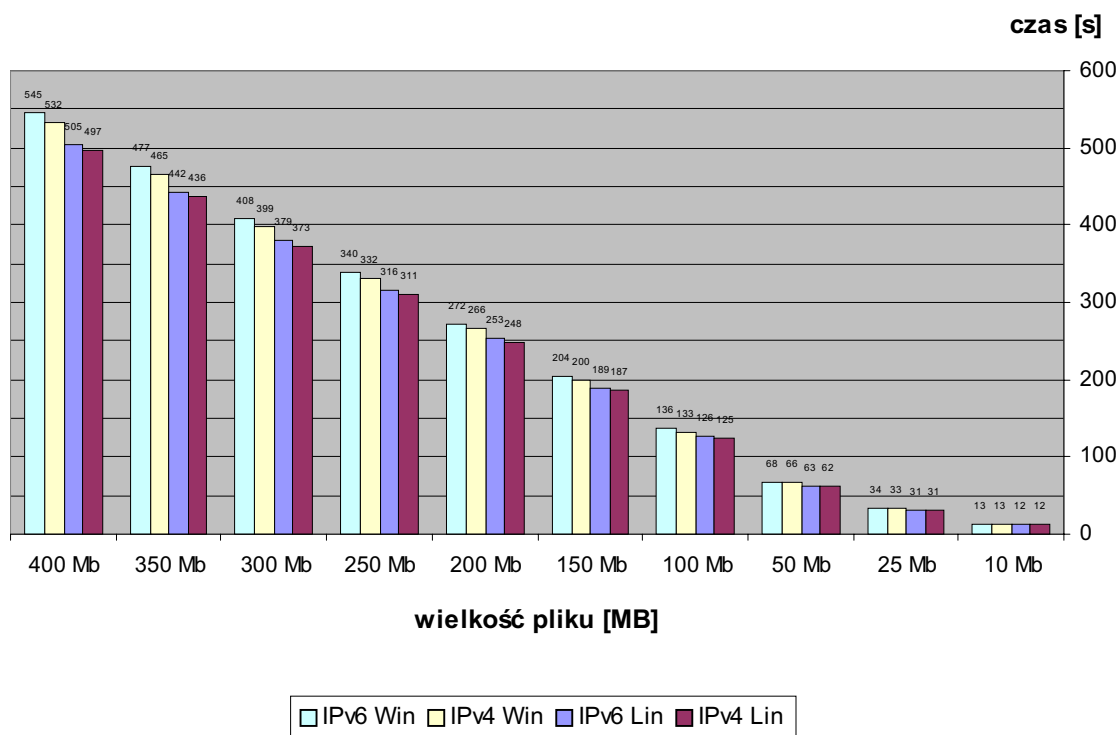


Rys. 7.4. Zmiany średniej prędkości przesyłu danych w zależności od wielkości pliku i rodzaju systemu.

wielkość pliku	Linux IPv6	Linux IPv4	Windows IPv6	Windows IPv4
400 Mb	505 sec	497 sec	545 sec	532 sec
350 Mb	442 sec	436 sec	477 sec	465 sec
300 Mb	379 sec	373 sec	408 sec	399 sec
250 Mb	316 sec	311 sec	340 sec	332 sec
200 Mb	253 sec	248 sec	272 sec	266 sec
150 Mb	189 sec	187 sec	204 sec	200 sec
100 Mb	126 sec	125 sec	136 sec	133 sec
50 Mb	63 sec	62 sec	68 sec	66 sec
25 Mb	31 sec	31 sec	34 sec	33 sec
10 Mb	12 sec	12 sec	13 sec	13 sec

Rys. 7.5. Średni czas przesyłu plików dla połączenia bezpośredniego komputerów.

Wykres z powyższej tabeli:



Rys. 7.6. Zmiany średniego czasu przesyłu danych w zależności od wielkości pliku i rodzaju systemu.

2)

Druga część doświadczenia polegała na sprawdzeniu różnic w prędkościach oraz czasach przesyłania plików po zainstalowaniu switcha pomiędzy komputerami. Wykonałem identyczne testy jak poprzednio, przesyłając pliki takiej samej wielkości. Wyniki testów przedstawione są w tabelach:

wielkość pliku	Linux IPv6	Linux IPv4	Windows IPv6	Windows IPv4
400 Mb	791,331 kb/s	804,553 kb/s	734,103 kb/s	752,421 kb/s
350 Mb	791,134 kb/s	801,527 kb/s	734,026 kb/s	753,211 kb/s
300 Mb	791,743 kb/s	805,032 kb/s	734,412 kb/s	752,342 kb/s
250 Mb	791,947 kb/s	803,202 kb/s	734,861 kb/s	753,528 kb/s
200 Mb	790,954 kb/s	802,755 kb/s	735,018 kb/s	752,043 kb/s
150 Mb	790,143 kb/s	803,238 kb/s	736,354 kb/s	749,543 kb/s
100 Mb	792,122 kb/s	802,631 kb/s	734,123 kb/s	753,854 kb/s
50 Mb	792,941 kb/s	813,122 kb/s	740,553 kb/s	763,533 kb/s
25 Mb	803,045 kb/s	814,473 kb/s	737,021 kb/s	764,976 kb/s
10 Mb	802,253 kb/s	816,345 kb/s	751,795 kb/s	772,043 kb/s

Rys. 7.7. Średnia prędkość przesyłu plików określonej długości dla połączenia ze switchem.

wielkość pliku	Linux IPv6	Linux IPv4	Windows IPv6	Windows IPv4
400 Mb	505 sec	497 sec	545 sec	532 sec
350 Mb	442 sec	436 sec	477 sec	465 sec
300 Mb	379 sec	373 sec	408 sec	399 sec
250 Mb	316 sec	311 sec	340 sec	332 sec
200 Mb	253 sec	248 sec	272 sec	266 sec
150 Mb	189 sec	187 sec	204 sec	200 sec
100 Mb	126 sec	125 sec	136 sec	133 sec
50 Mb	63 sec	62 sec	68 sec	66 sec
25 Mb	31 sec	31 sec	34 sec	33 sec
10 Mb	12 sec	12 sec	13 sec	13 sec

Rys. 7.8. Średni czas przesyłu plików określonej długości dla połączenia ze switchem.

W poniższej tabeli (rys. 7.9.) przedstawiłem różnice w prędkości przesyłania danych pomiędzy tymi samymi systemami i protokołami IPv4 i IPv6 oraz różnice pomiędzy systemami przy wykorzystaniu tego samego protokołu:

wielkość przesyłanego pliku	wzrost prędkości przy porównaniu Linux IPv4 do Linux IPv6 [%]	wzrost prędkości przy porównaniu Windows IPv4 do Windows IPv6 [%]	wzrost prędkości przy porównaniu Linux IPv6 do Windows IPv6 [%]	wzrost prędkości przy porównaniu Linux IPv4 do Windows IPv4 [%]
400 Mb	1,63	2,47	7,26	6,47
350 Mb	1,29	2,47	7,20	6,08
300 Mb	1,65	2,37	7,22	6,53
250 Mb	1,49	2,47	7,16	6,22
200 Mb	1,58	2,37	7,08	6,34
150 Mb	1,58	1,95	6,86	6,51
100 Mb	1,38	2,61	7,25	6,07
50 Mb	2,47	3,03	6,65	6,11
25 Mb	1,43	3,67	8,20	6,07
10 Mb	1,76	2,74	6,37	5,43
średnio	1,63	2,61	7,13	6,18

Rys. 7.9. Porównanie prędkości protokołów IPv4 i IPv6 w systemach Linux i Windows liczone w %.

Interpretacja wyników - wnioski

Okazało się, zgodne z przypuszczeniami z rozdziału 6.2.2, że wyniki otrzymane dla bezpośredniego połączenia komputerów (rys. 7.3. i 7.5.) i wyniki dla połączenia z wykorzystaniem switcha (rys. 7.7. i 7.8.) różnią się od siebie nie więcej niż 0,5%. Można więc przyjąć, że switch nie wprowadzał opóźnień przy przesyłaniu plików w sieci testowej.

Dla systemu Linux różnice prędkości przy przesyłaniu danych podczas używania protokołów IPv6 i IPv4 różnią się o 1,63%, a więc o mniej niż przewidywane w założeniach 5% (rys. 7.9. kolumna 1). Analogicznie to samo można powiedzieć o przypadku przesyłania danych dla systemu Windows, gdzie średnia różnica pomiędzy IPv6 i IPv4 wynosi 2,61% (rys. 7.9. kolumna 2).

Jednak podczas przesyłania danych protokołem IPv6 w przypadku porównania systemów Windows i Linux widać, że różnice w prędkości wynoszą średnio 7,13% na niekorzyść systemu Windows (rys. 7.9. kolumna 3), co jest większe od zakładanych przez mnie 5%. Analogicznie taka sama sytuacja występuje przy porównaniu systemów Windows i Linux dla protokołu IPv4, gdzie średnia różnica w prędkościach wynosi 6,18% (rys. 7.7. kolumna 4). Różnice te mogą być spowodowane innym sposobem implementacji protokołu IP w tych systemach.

7.2.2. Przesyłanie wielu plików

Błędy pomiaru

Zgodnie z założeniami testu (rozdział 6.2.3), aby sprawdzić, czy wydajność komputerów jest wystarczająca, uruchomiłem test, jednocześnie monitorując programem *top* pod Linux oraz *Task Manager* pod Windows procent zajętości czasu procesora. Okazało się, że system Linux pracujący jako serwer *ftpd* był wykorzystywany w 30-40%. Procent wykorzystania czasu procesora komputera, pracującego jako klient *ftp*, przedstawia poniższa tabela:

wielkość pliku	zużycie procesora	
	system Linux	system Windows
10b	90%	95%
100b	80 - 90%	95%
1kb	75 - 85%	80 - 90%
10kb	60 - 70%	65 - 75%
20kb	30 - 40%	45 - 55%
40kb - 200kb	25 - 30%	40 - 50%

Rys. 7.10. Procent zużycia procesora.

Okazało się, że dla plików poniżej 10kb system Linux pracujący jako klient, był zajęty w 75-90%, a system Windows 2000 w 80-95%. Wyniki te wskazywały na to, że podczas przesyłania 1000 plików wielkości do 10kb, aplikacje *ftp* zajmują znacznie więcej czasu procesora, niż podczas przesyłania takiej samej ilości plików większych. Wykonanie miarodajnych testów dla plików długości poniżej 10kb było niemożliwe ze względu na niewystarczającą wydajność komputerów. Dlatego wszystkie testy oraz wyniki w tabelach, przedstawione będą dla plików większych i równych 10kb.

Wykonanie testu

Test składał się z dwóch etapów:

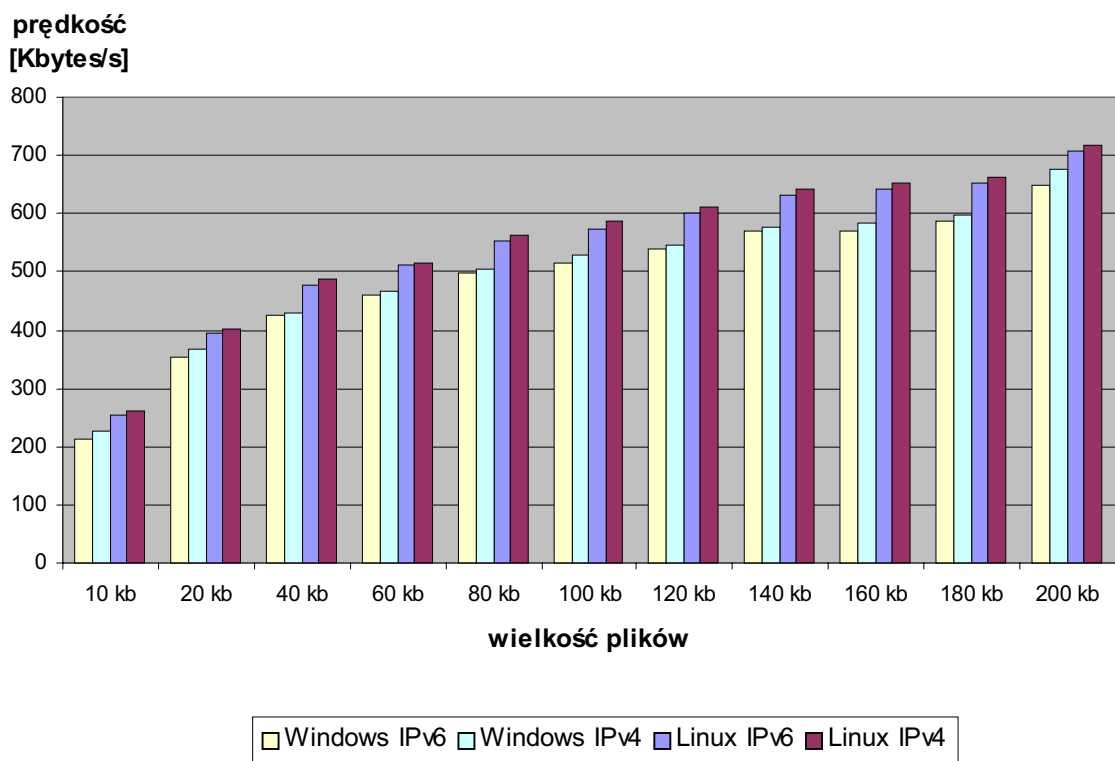
1)

W pierwszej części testu karty sieciowe komputerów PC połączyłem bezpośrednio ze sobą za pomocą skrętki z przeplotem. Na obu komputerach uruchomiłem system Linux Debian, uprzednio skonfigurowany do współpracy z protokołem IPv4 oraz IPv6. Na jednym uruchomiłem serwer *ftpd*, a na drugim klientem *lftp*, wykorzystując protokół

IPv6, przesyłałem na dysk lokalny 1000 plików wielkości kolejno 10kb, 20kb, 40kb, 60kb, 80kb, 100kb, 120kb, 140kb, 160kb, 180kb oraz 200kb, notując wyniki w poniższych tabelach. Analogiczne doświadczenie wykonałem wykorzystując do testów protokół IPv4. Następnie jeden z komputerów uruchomiłem pod kontrolą systemu Windows 2000 Professional. Wykorzystując klienta *ftp* wykonałem takie same testy:

wielkość pliku	Prędkość przesyłania plików [kb/sec] dla danego systemu i protokołu			
	Linux IPv6	Linux IPv4	Windows IPv6	Windows IPv4
10 kb	254,002	259,254	214,563	225,434
20 kb	395,157	400,233	354,477	366,591
40 kb	475,709	486,438	425,343	430,122
60 kb	512,063	515,094	460,095	465,986
80 kb	553,384	562,98	497,291	504,543
100 kb	574,581	585,459	514,509	528,439
120 kb	599,317	611,034	537,912	545,55
140 kb	631,958	642,097	569,091	578,429
160 kb	642,457	651,434	571,366	582,187
180 kb	652,78	663,435	585,675	597,477
200 kb	705,83	717,409	650,341	675,409

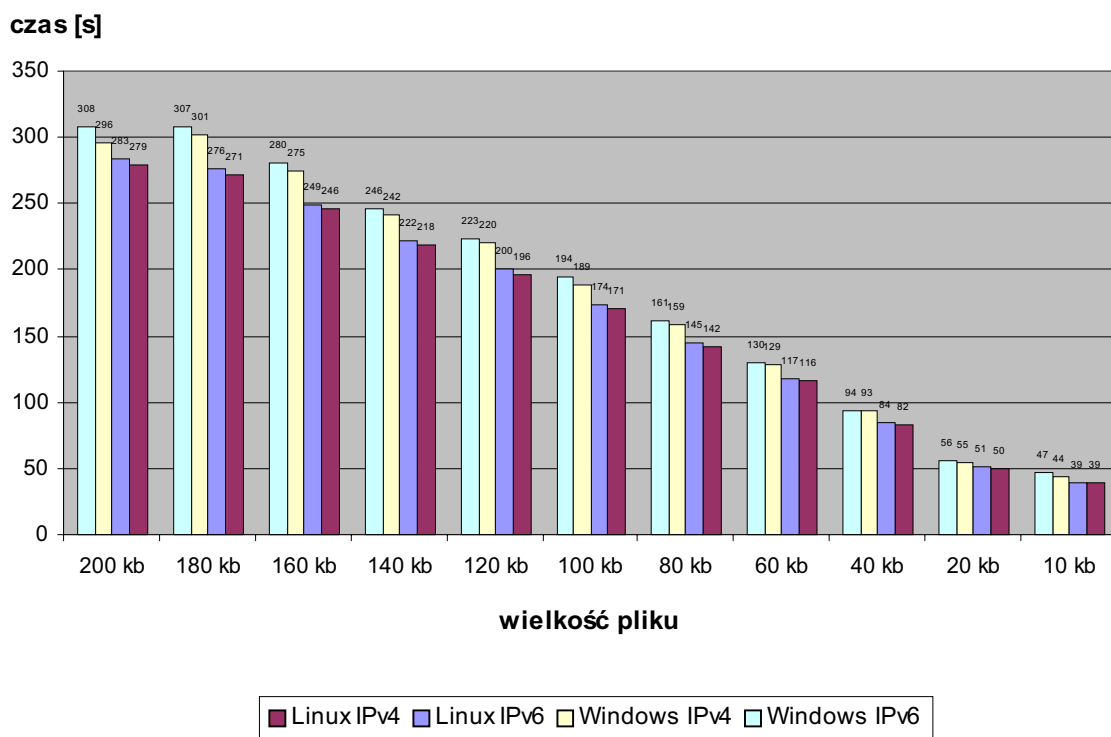
Rys. 7.11. Średnia prędkość przesyłania 1000 plików [kb/sec] przy bezpośrednim połączeniu komputerów.



Rys. 7.12. Zmiany prędkości przy przesyłaniu 1000 plików różnej długości.

wielkość pliku	Czas przesyłania plików [sec] dla danego systemu i protokołu			
	Linux IPv6	Linux IPv4	Windows IPv6	Windows IPv4
10 kb	39	39	47	44
20 kb	51	50	56	55
40 kb	84	82	94	93
60 kb	117	116	130	129
80 kb	145	142	161	159
100 kb	174	171	194	189
120 kb	200	196	223	220
140 kb	222	218	246	242
160 kb	249	246	280	275
180 kb	276	271	307	301
200 kb	283	279	308	296

Rys. 7.13. Średni czas przesyłania 1000 plików [sec] przy bezpośrednim połączeniu komputerów.



Rys. 7.14. Zmiany prędkości przy przesyłaniu 1000 plików różnej długości.

2)

Druga część testów polegała na przesyłaniu 1000 plików pomiędzy komputerami połączonymi ze sobą za pomocą switcha. Wyniki doświadczenia przedstawiają poniższe tabele:

wielkość pliku	Prędkość przesyłania plików [kb/sec] dla danego systemu i protokołu			
	Linux IPv6	Linux IPv4	Windows IPv6	Windows IPv4
10 kb	254,122	259,439	214,872	225,254
20 kb	395,342	400,983	354,275	366,619
40 kb	475,54	486,49	425,197	429,984
60 kb	511,956	514,793	459,965	466,054
80 kb	553,432	562,591	497,451	504,49
100 kb	574,609	585,128	514,128	528,519
120 kb	598,997	610,845	537,384	545,312
140 kb	632,132	642,843	568,764	578,987
160 kb	642,772	651,912	571,501	582,328
180 kb	652,718	663,128	585,592	597,511
200 kb	705,132	717,726	650,41	675,204

Rys. 7.15. Średnia prędkość przesyłania 1000 plików [kb/sec] przy połączeniu przez switch.

wielkość pliku	Czas przesyłania plików [sec] dla danego systemu i protokołu			
	Linux IPv6	Linux IPv4	Windows IPv6	Windows IPv4
10 kb	39	39	47	44
20 kb	51	50	56	55
40 kb	84	82	94	93
60 kb	117	117	130	129
80 kb	145	142	161	159
100 kb	174	171	195	189
120 kb	200	196	223	220
140 kb	221	218	246	242
160 kb	249	245	280	275
180 kb	276	271	307	301
200 kb	284	279	307	296

Rys. 7.16. Średni czas przesyłania 1000 plików [sec] przy połączeniu przez switch.

Porównanie prędkości protokołów oraz systemów Windows i Linux dla bezpośredniego połączenia komputerów przedstawione jest w poniższej tabeli:

wielkość przesyłanego pliku	wzrost prędkości przy porównaniu Linux IPv4 do Linux IPv6 [%]	wzrost prędkości przy porównaniu Windows IPv4 do Windows IPv6 [%]	wzrost prędkości przy porównaniu Linux IPv6 do Windows IPv6 [%]	wzrost prędkości przy porównaniu Linux IPv4 do Windows IPv4 [%]
10 kb	2,03	4,82	15,53	13,05
20 kb	1,27	3,30	10,29	8,41
40 kb	2,21	1,11	10,59	11,58
60 kb	0,59	1,26	10,15	9,53
80 kb	1,70	1,44	10,14	10,38
100 kb	1,86	2,64	10,45	9,74
120 kb	1,92	1,40	10,25	10,72
140 kb	1,58	1,61	9,95	9,92
160 kb	1,38	1,86	11,07	10,63
180 kb	1,61	1,98	10,28	9,94
200 kb	1,61	3,71	7,86	5,85
średnio	1,66	2,20	9,98	9,47

Rys. 7.17. Porównanie prędkości protokołów i systemów dla bezpośredniego połączenia komputerów.

Interpretacja wyników - wnioski

W wyniku wykonania testów okazało się, zgodne z moimi założeniami (rozdział 6.2.3.), że wyniki otrzymane dla bezpośredniego połączenia komputerów (rys. 7.11. i 7.13.), jak i wyniki dla połączenia z użyciem switcha (rys. 7.15. i 7.16.) różnią się od siebie w granicach błędu 0,5%. Można przyjąć, że switch nie wprowadzał opóźnień przy przesyłaniu 1000 plików różnej długości wewnątrz sieci testowej.

Z testów wynika, że protokół IPv4 jest szybszy o 1,66% dla systemu Linux (rys. 7.17. kolumna 1) oraz o 2,20% dla systemu Windows (rys. 7.17. kolumna 2), co mieści się w założonych przeze mnie granicach 5%.

Porównując systemy Windows i Linux działające pod tym samym protokołem, widać zmianę prędkości dla IPv6 o 9,98% na niekorzyść Windows oraz dla IPv4 o 9,47% również na niekorzyść w.w. systemu.

Z wykresów na rys. 7.12. widać, że prędkość przesyłu 1000 plików jest mniejsza przy plikach małych, a przy plikach rzędu 200 kb, prędkość ta zbliża się już do prędkości przesyłu dużych plików (rozdział 7.2.1.)

Niestety, nie mogłem wykonać testów przesyłania plików długości mniejszych niż 10kb z powodu małej wydajności komputerów. Podczas przesyłania plików protokołem *ftp*, dla każdego z nich musi nastąpić jego otwarcie i zamknięcie oraz sama transmisja danych. Pliki przesyłane przez sieć dzielone są na mniejsze części, których wielkość wraz z nagłówkami nie przekracza MTU sieci. W moich testach MTU sieci wynosiło 1500 bajtów. Dla plików o wielkości mniejszej niż 10kb okazało się, że liczba informacji przetwarzanych przez komputer w związku z procesem otwarcia i zamknięcia pliku w stosunku do jego wielkości jest na tyle duża, że aplikacja *ftp* zarówno dla systemu Windows, jak i Linux zajmowała ponad 75% czasu procesora.

7.3. Routing IPv6 w sieci 6BONE

Wykonanie testu

Zgodnie z założeniami testu z rozdziału 6.2.4., z komputera działającego wewnątrz sieci testowej, wykonałem komendę *traceroute6* do głównego serwera 6BONE w Polsce. Aby uniknąć ewentualnych błędów wynikających z zakłóceń, test wykonałem dziesięciokrotnie, podając wyniki w poniższej tabeli. Test wykonałem zarówno dla systemu Windows, jak i Linux oraz dla obu testowanych protokołów.

nr. testu	czasy powrotów pakietów [ms], użyty system i protokół			
	Linux IPv6 [ms]	Linux IPv4 [ms]	Windows IPv6 [ms]	Windows IPv4 [ms]
1	6,255	5,597	6,032	6,437
2	6,043	5,022	5,932	6,092
3	6,272	6,538	6,873	5,933
4	5,755	5,691	5,332	5,833
5	7,492	6,312	6,322	5,903
6	5,436	6,178	6,093	5,546
7	5,814	5,071	5,954	7,453
8	6,073	6,285	4,987	5,045
9	6,617	6,044	7,032	6,419
10	5,347	7,645	6,651	6,455
średnia:	6,110	6,038	6,121	6,112

Rys 7.18. Czasy powrotu pakietów do sieci 6BONE dla IPv4 i IPv6 dla systemów Linux i Windows.

Różnice w średnich czasach powrotów pakietów dla różnych systemów oraz protokołów przedstawia poniższa tabela:

	różnice czasów przy porównaniu Linux IPv4 do Linux IPv6 [%]	różnice czasów przy porównaniu Windows IPv4 do Windows IPv6 [%]	różnice czasów przy porównaniu Linux IPv6 do Windows IPv6 [%]	różnice czasów przy porównaniu Linux IPv4 do Windows IPv4 [%]
średnia:	1,180	0,150	0,170	1,199

Rys. 7.19. Porównanie czasów powrotu pakietów do sieci 6BONE.

Interpretacja wyników - wnioski

Zgodnie z oczekiwaniami średnie czasy powrotu pakietów z głównego serwera sieci 6BONE nie różniły się więcej niż 5% dla przypadku wykorzystania obu testowanych systemów oraz obu protokołów.

Routing z komputera znajdującego się wewnątrz sieci testowej odbywał się przez router mający skonfigurowany tunel do sieci 6BONE:

```
ipv6_pc-3:~# traceroute6 6bone-gw.icm.edu.pl
traceroute6 to 6bone-gw.icm.edu.pl (3ffe:8010:81::1) from
3ffe:8010:81::3, 30 hops max, 16 byte packets
 1 ipv6_pc-2 (3ffe:8010:81::1:2) 0.534ms 0.437ms 0.401ms
 2 6bone-gw.icm.edu.pl (3ffe:8010:81::1) 6.434ms 6.227ms 5.239ms
ipv6_pc-3:~#
```

7.4. Stabilność i poprawność działania oprogramowania oraz sprzętu

Wykonanie testu

Przez cały czas wykonywania poprzednich doświadczeń badałem zachowanie się sprzętu i oprogramowania skonfigurowanego do obsługi IPv6. Obserwowałem zachowanie się używanego przeze mnie switcha, jak i zainstalowanych aplikacji oraz serwisów IPv6.

Wnioski

Zarówno system Windows, jak i Linux, podczas wykonywanych przeze mnie testów, zachowywały się stabilnie. Żaden z systemów nie zawiesił się, nie zaobserwowałem innego dziwnego zachowania się komputerów, które pracowały bez przerwy przez około 2 miesiące. Zainstalowany przeze mnie switch firmy Centre COM działał poprawnie.

8. Wnioski

Wszystkie testy przedstawione w pracy przeprowadziłem zgodnie z założeniami projektu. Zbadałem praktyczne aspekty wykorzystania nowego protokołu i jego współpracy z obecnie obowiązującą wersją protokołu IP. W testach potwierdziłem poprawność konfiguracji routingu wewnątrz naszej sieci testowej, która została połączona z ogólnopolską siecią 6BONE poprzez stworzony do ICM tunel IPv6. Okazuje się, że IPv6 jest wolniejszy nie więcej niż 3% w porównaniu do swojego poprzednika.

Badania implementacji IPv6 dla dwóch podstawowych systemów operacyjnych wykazały wolniejsze nie więcej niż 10% działanie protokołu IP dla systemu Windows 2000 w porównaniu z systemem Linux. Może to być spowodowane lepszym sposobem implementacji samego protokołu IP dla systemu Linux, lub sposobem implementacji niższej lub wyższej warstwy w modelu protokołów TCP/IP.

Żaden z systemów operacyjnych oraz żadna z używanych przeze mnie aplikacji nie „zawiesiła się”, nie zaobserwowałem innego dziwnego zachowania się komputerów. Zainstalowany przeze mnie switch firmy Centre COM działał poprawnie. Switch nie wprowadzał również zmian prędkości przesyłania danych pomiędzy komputerami, w porównaniu z bezpośrednim połączeniem komputerów.

Niestety, nie mogłem wykonać testu przesyłania 1000 plików długości mniejszej niż 10kb, z powodu małej wydajności komputerów. Okazało się, że aplikacja *ftp* zarówno dla systemu Windows, jak i Linux, zajmowała ponad 75% czasu procesora. Dlatego testy wykonywane były dla plików powyżej 10kb.

Przeprowadzone przeze mnie testy nie wyczerpują tematu związanego z protokołem IPv6. Interesujące byłyby następujące aspekty:

- zbadanie poprawności i porównanie (dla protokołów IPv6 i IPv4) prędkości działania routerów sprzętowych firm 3COM, Intel, Cisco i innych oraz porównanie możliwości ich konfiguracji,
- zbadanie poprawności oraz porównanie prędkości działania aplikacji i serwisów internetowych, takich jak np: przeglądarki i serwery www, ssh, telenet, dla obu protokołów i różnych platform (np. Linux, Windows, Solaris),
- porównanie prędkości działania różnych serwerów i klientów *ftp* dla IPv6,
- uruchomienie i zbadanie poprawności działania routingu dynamicznego BGP4+, np. na mrt, gated lub zebra,
- skonfigurowanie i testy poprawności działania serwerów DNS i rDNS.

Wprowadzenie nowego protokołu wiąże się z uaktualnieniem wszystkich systemów operacyjnych oraz potrzebnych aplikacji, działających zarówno na stacjach roboczych, jak i na serwerach oraz routerach pośrednich. Standard protokołu IPv6 nadal rozwija się, a dokumenty RFC opisujące nowy protokół wciąż ulegają aktualizacjom. Na stronach www poświęconych sieci 6BONE można znaleźć informacje, że wdrażanie nowego protokołu zacznie się za około 5 lat. Aby móc przejść bezkolizyjnie z IPv4 na IPv6, stara wersja protokołu pozostanie w działaniu jeszcze przez jakiś czas. Obecnie istnieją tylko wersje beta systemów operacyjnych oraz niektórych aplikacji wspierających IPv6. W chwili pisania pracy nowy protokół wykorzystywany jest jedynie w celach testowych. W większych ośrodkach informatycznych stosuje się praktykę tworzenia tunelu IPv6 do głównego ośrodka 6BONE w Polsce, celem udostępniania adresów testowych mniejszym instytucjom, firmom oraz użytkownikom prywatnym.

9. Bibliografia

Literatura

1. "TCP/IP - Administracja sieci" Wydanie drugie, Craig Hunt, Wydawnictwo RM, Warszawa 1998,
2. "Sieci Komputerowe TCP/IP, Tom 1 - zasady, protokoły i architektura", Douglas E. Comer, Wydawnictwa Naukowo Techniczne, Warszawa 1998,
3. "Biblia TCP/IP, Tom 1 - protokoły", W. Richard Stevens, Wydawnictwo RM, Warszawa 1998,
4. Nimir Leśniewski - Praca dyplomowa inżynierska "Protokół Internetu IPv6"

Dokumenty RFC

5. "Internet Protocol, Version 6 (IPv6) Specification" S. Deering, R. Hinden (RFC 2460)
6. "IP Version 6 Addressing Architecture" R. Hinden, S. Deering (RFC 2373)
7. "IPv6 Router Alert Option" C. Partridge, A. Jackson (RFC 2711)
8. "IPv6 Jumbograms" D. Borman, S. Deering, R. Hinden (RFC 2675)
9. "IP Encapsulating Security Payload (ESP)" S. Kent, R. Atkinson (RFC 2406)
10. A. Conta, S. Deering, Internet Control Message Protocol (ICMPv6) for the Intern Protocol Version 6 (IPv6), RFC 2463, December 1998.
11. A. Conta, S. Deering, Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6), Internet Draft, <draft-ietf-ipngwg-icmp-v3-00.txt> Internet Control , June 1999.

Publikacje Internetowe:

- 1i. IP Next Generation Overviewln, R. Hinden, <http://playground.sun.com/pub/ipng/html/INET-IPng-Paper.html>,
- 2i. IPv6 for Solaris, <http://playground.sun.com/pub/solaris2-ipv6/html/solaris2-ipv6.html>
- 3i. Microsoft Research IPv6 Implementation (MSRIPv6), <http://research.microsoft.com/msripv6/>

- 4i. SDR and RAT with IPv6 Support, <http://research.microsoft.com/msripv6/mbone.htm>
- 5i. Polish(ed) Linux Distribution, <http://www.pld.org.pl/>
- 6i. IPng Current Specifications, <http://playground.sun.com/pub/ipng/html/specs/specifications.html>
- 7i. Microsoft IPv6 Technology Preview for Windows 2000
<http://msdn.microsoft.com/downloads/sdks/platform/tpipv6.asp>
- 8i. Cisco Connection Online by Cisco Systems, Inc.
<http://www.cisco.com/IPv6/>
- 9i. 6BONE w Polsce - <http://www.6bone.pl/>
- 10i. Peter Bieringer's Linux-Section: IPv6 - <http://www.bieringer.de/linux/IPv6/>
- 11i. Windows 2000 Service Pack 1 - <http://www.microsoft.com/windows2000/downloads/recommended/sp1/default.asp>
- 12i. Cisco IOS Software Solutions: IPv6 - <http://www.cisco.com/warp/public/732/ipv6/index.html>
- 13i. mgr inż. Bartosz Lis "Wykład nt. Sieci Komputerowe" Instytut Informatyki Politechniki Łódzkiej - <http://stud-ics.p.lodz.pl/staff/bartoszl/edu/sieci-SPA-w/sieci.html>