Protokoły transmisyjne stosu TCP/IP[1]

Adam Przybyłek
http://przybylek.wzr.pl
Uniwersytet Gdański, 2008

Wstęp

TCP/IP[2] to obecnie najpopularniejszy stos protokołów, umożliwiający łączenie heterogenicznych sieci o różnorodnych technologiach warstwy łącza danych oraz fizycznej. Podstawowe protokoły stosu TCP/IP opracowane zostały w latach 70.[3] na zamówienie Departamentu Obrony USA (ang. *U.S. Department of Defense* – DoD), w ramach projektu ARPANET.

Od momentu powstania TCP/IP jest stale rozwijany, udoskonalany i dostosowywany do nowych zastosowań. Obecnie pełni rolę standardu w komunikacji komputerowej we wszystkich rodzajach sieci, od domowych – łączących kilka komputerów, poprzez firmowe intranety, po korporacyjne ekstranety. O sukcesie rodziny protokołów TCP/IP zdecydowało nie tylko przyjęcie ich przez DoD jako podstawowych protokołów tzw. profilu sieci wojskowych, ale także ich otwarta specyfikacja (możliwość łączenia sprzętu różnych dostawców), elastyczność rozwiązań, oraz darmowe implementacje. Nowe zastosowania sieci komputerowych, takie jak telefonia internetowa, wideokonferencje, czy wideo na żądanie wymagają gwarantowanej jakości usług. Stos TCP/IP nie był projektowany z myślą o aplikacjach czasu rzeczywistego, jednak jego modularność umożliwiła dobudowanie szeregu nowych protokołów stosownie do potrzeb.

Przedmiotem niniejszego opracowania nie są wszystkie protokoły wchodzące w skład stosu TCP/IP, a jedynie te występujące w warstwie sieci oraz transportowej, czyli stanowiące szkielet komunikacji sieciowej. Informacje o pozostałych protokołach rodziny TCP/IP można odszukać w dokumentach RFC dostępnych za darmo w Internecie [www.rfc-editor.org].

---

[1] Rozdział VI książki pod red. Jacka Winiarskiego – *Sieci komputerowe w biznesie*, Wydawnictwo Uniwersytetu Gdańskiego, 2008 (http://www.wyd.strony.univ.gda.pl/szczegoly.php?id=475)
[2] Skrót pochodzi od nazw dwóch podstawowych protokołów tworzących stos: TCP oraz IP.
[3] Zatwierdzenie poszczególnych standardów nastąpiło w następnej dekadzie: UDP – 1980, TCP – 1981, IP – 1981, ICMP – 1981.

1. Protokół IP

1.1. Charakterystyka protokołu IP

Protokół IP funkcjonuje w warstwie trzeciej modelu ISO/OSI i należy do tzw. **protokołów routowalnych** (ang. *routed protocols*). Zadaniem tego typu protokołów jest przekazywanie między sieciami danych końcowego użytkownika. Na poziomie warstwy trzeciej[4] komputery użytkowników identyfikowane są za pomocą adresów logicznych. Struktura danych, na której operuje protokół IP, nazywana jest **datagramem**[5]. Datagram składa się z nagłówka i pola danych przenoszącego tzw. jednostkę danych (ang. *Protocol Data Unit* – PDU) protokołu wyższego poziomu. Do podstawowych elementów nagłówka należą adres IP nadawcy oraz odbiorcy datagramu.

Protokoł IP, na bazie połączonych sieci fizycznych o różnorodnych technologiach, tworzy jednolity system komunikacyjny dla całego Internetu. Mając na uwadze skalowalność, IP został zaprojektowany jako protokół bezpołączeniowy. Brak potrzeby zarządzania połączeniami upraszcza działanie routerów, a protokół IP czyni łatwiejszym w implementacji. Wybór modelu bezpołączeniowego oznacza brak możliwości sprawdzenia gotowości docelowego hosta do odebrania przesyłanych danych. Każdy datagram stanowi autonomiczną jednostkę, dla której router indywidualnie wybiera trasę do miejsca przeznaczenia. Ponadto IP nie posiada mechanizmów retransmisji w przypadku zagubienia lub uszkodzenia datagramów. Routery pośredniczące dokładają wszelkich starań (ang. *best-effort*), aby dostarczyć datagram do odbiorcy, ale w przypadku niesprzyjających okoliczności – usuwają go. Do niepomyślnych okoliczności można zaliczyć m.in. przeciążenie routerów, wygaśnięcie czasu życia pakietu, nieznajomość trasy do sieci docelowej. Protokół IP nie daje więc żadnej gwarancji, że wysłany datagram dotrze do adresata. Datagram może zostać odrzucony, powielony lub dostarczony z błędem, a protokół IP nie ma możliwości tego sprawdzić. Niezawodność dostarczenia danych musi być zatem zaimplementowana w protokołach wyższych warstw.

Do podstawowych funkcji protokołu IP należą:

– określenie struktury datagramu;
– zdefiniowanie hierarchicznego schematu adresacji, umożliwiającego przesyłanie datagramów między sieciami[6];
– po stronie źródłowej – umieszczanie segmentów pochodzących z warstwy transportowej w datagramach, a następnie przekazywanie datagramów do warstwy łącza danych;

[4] W niniejszym rozdziale warstwy liczone są względem modelu ISO-OSI.
[5] zamiennie można używać słowa pakiet
[6] W oparciu o protokoły warstwy łącza danych komunikacja możliwa jest jedynie w ramach jednej sieci.

– po stronie docelowej – odbieranie danych z warstwy łącza danych, odtwarzanie datagramów, a następnie przekazywanie zawartości datagramów do warstwy transportowej;
– dokonywanie w razie potrzeby fragmentacji datagramu oraz odtwarzanie z fragmentów oryginalnego datagramu.

Fragmentacja dokonywana jest w przypadku, gdy datagram IP nie może w całości „zmieścić się" w polu danych ramki warstwy niższej. Pojemność ramki to maksymalny rozmiar ładunku, jaki ramka może przenieść. Pojemność ta wyznacza tzw. **MTU** (ang. *Maximum Transmission Unit*) warstwy łącza danych. Przykładowo, MTU dla technologii Ethernet wynosi 1500 bajtów. Jeżeli datagram IP będzie większy niż MTU warstwy niższej, zostanie podzielony na fragmenty. Datagramy IP dzielone są na fragmenty w taki sposób, że każdy fragment staje się pełnoprawnym datagramem i podróżuje do miejsca przeznaczenia niezależnie od pozostałych. Poszczególne fragmenty, na drodze do miejsca przeznaczenia, mogą zostać podzielone na jeszcze mniejsze kawałki. Scalanie (ang. *reassembly*) fragmentów datagramu odbywa się dopiero w hoście docelowym.

1.2. Budowa datagramu IP

Strukturę datagramu IPv4 przedstawia rys. 1, a znaczenie poszczególnych pól opisano poniżej.

Rysunek 1. Struktura datagramu IP [RFC 791]

Wersja (ang. *version*) - 4 bity

Ponieważ różne wersje protokołu używają różnych formatów nagłówka, należy wiedzieć, którą z nich zastosować do interpretacji pozostałych pól. Obecnie powszechnie wykorzystywana jest wersja czwarta, choć coraz częściej wdraża się sieci działające w oparciu o wersję szóstą. Wersja piąta miała charakter eksperymentalny i nigdy nie weszła do powszechnego użytku.

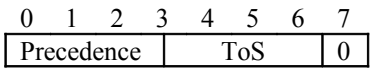
Długość nagłówka (ang. *Internet Header Length* – IHL) - 4 bity

Określa, ile 32-bitowych słów zajmuje nagłówek. Wartość minimalna dla tego pola to 5, ponieważ tyle linii zajmuje część stała nagłówka. Oznacza to, że na-

główek IP zajmuje co najmniej 20 bajtów (5×4 bajty). Wartości większe niż 5 oznaczają, że nagłówek zawiera pole *opcje*. Wartość maksymalna ograniczona jest przez rozmiar pola (4 bity), a więc wynosi 15 (2^4-1). Wynika stąd, że maksymalny rozmiar nagłówka to 60 bajtów (15×4 bajty).

Typ usługi (ang. *type of service*) - 8 bitów

Ogólnie pole to ma określać jakość usługi (ang. *Quality of Service* – QoS) żądanej przez warstwę aplikacji. Jego interpretacja wielokrotnie była modyfikowana. Początkowo każdy typ ruchu traktowany był jednakowo i obsługiwany zgodnie z koncepcją „w miarę możliwości" (ang. *best effort*). Dopiero zapotrzebowanie na usługi czasu rzeczywistego sprawiło, że pole *typ usługi* nabrało znaczenia. Obecnie routery mogą interpretować to pole na dwa sposoby: zgodnie z zaleceniami IANA (ang. *Internet Assigned Numbers Authority*, www.iana.org) z 18.10.2005 lub zgodnie ze specyfikacją RFC 2474. Pierwszy sposób definiuje strukturę tego pola w sposób przedstawiony na rys. 2.



Rysunek 2. Struktura pola typ usługi [RFC 1349]

Subpole *pierwszeństwo* (ang. *precedence*) powinno być uwzględniane podczas kolejkowania pakietów i określa ono następujące poziomy pilności:

- 000 – zwykły (ang. routine);
- 001 – priorytetowy (ang. priority);
- 010 – natychmiastowy (ang. immediate);
- 011 – błyskawiczny (ang. flash);
- 100 – ultra błyskawiczny (ang. flash override);
- 101 – datagram specjalny – CRITIC/ECP;
- 110 – datagram specjalny – Internetwork Control;
- 111 – datagram specjalny – Network Control.

Z kolei subpole *ToS* powinno być interpretowane w następujący sposób:

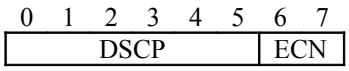
- 0000 – wartość domyślna;
- 0001 – minimalny koszt (ang. *monetary cost*);
- 0010 – maksymalna niezawodność (ang. *relibility*);
- 0100 – maksymalna przepustowość (ang. *throughput*);
- 1000 – minimalne opóźnienie (ang. *delay*);
- 1111 – maksymalne bezpieczeństwo (ang. *security*).

Zalecane wartość ToS dla przykładowych protokołów wyglądają następująco:

- Telnet – 1000;
- FTPdata – 0100;
- protokoły routingu wewnętrznego – 0010.

Drugi sposób interpretacji pola *ToS* został określony w specyfikacji RFC 2474 na potrzeb architektury DiffServ. Pole to podzielono na dwie części (rys. 3): 6-bitową DSCP oraz 2-bitową ECN. Subpole *ECN* (*Explicit Congestion Notification*), zdefiniowane w RFC 3168, służy do sygnalizacji przeciążenia wykorzystując następujące kody:

– 00 (not-ECN) – host końcowy nie wykorzystuje mechanizmu ECN;
– 01 lub 10 (ang. *ECN-capable Transport* – ECT) – host końcowy jest zdolny do odbierania informacji o przeciążeniach;
– 11 (ang. *Congestion Experienced* – CE) – powiadomienie o przeciążeniu.



Rysunek 3. Struktura pola *typ usługi* przystosowana do architektury DiffServ

Wartość zapisywana w subpolu DSCP (*Differentiated Services Code Point*) służy do klasyfikacji ruchu w celu zapewnienia zróżnicowanej obsługi. Istnieją następujące predefinowane kategorie DSCP:

– Best Effort (DSCP = 000000) – brak jakichkolwiek zapewnień o QoS;
– Class-Selector (DSCP = xyz000) – kategoria przeznaczona do zachowania kompatybilności wstecz z subpolem *pierwszeństwo*[7]; oznaczone w ten sposób datagramy powinny być obsłużone zgodnie z priorytetem pilności;
– Expedited Forwarding (DSCP = 101110) – charakteryzuje aplikacje najbardziej wrażliwe na opóźnienia, wymagające gwarantowanego pasma; wykorzystywana przez bramy VoIP do oznaczania datagramów zawierających głos;
– Assured Forwarding – definiuje 4 klasy (na bitach 0, 1, 2) oraz 3 poziomy odrzucenia datagramów (na bitach 3, 4) w sposób przedstawiony w tab. 1; ostatni bit jest wyzerowany[8]; umożliwia różnicowanie poziomu obsługi, mieszczące się w zakresie od "best effort" do poziomu zapewnianego przez klasę EF; dla każdej klasy można skonfigurować na routerze rozmiar bufora oraz dostępne pasmo na interfejsie.

Tab. 1. Wartości DSCP dla kategorii Assured Forwarding [RFC 2957]

Priorytet odrzucenia	klasa 1	klasa 2	klasa 3	klasa 4
Niski	001|01|0	010|01|0	011|01|0	100|01|0
Średni	001|10|0	010|10|0	011|10|0	100|10|0
Wysoki	001|11|0	010|11|0	011|11|0	100|11|0

Wartości pola DSCP, które nie zostały predefiniowane, mogą być interpretowane w sposób określony przez użytkownika.

[7] W rzeczywistości kompatybilność wstecz nigdy nie została zaimplementowana, a w dokumencie RFC-3260 została zniesiona.
[8] Wszystkie standaryzowane przez IANA wartość DSCP mają ostatni bit wyzerowany.

Całkowita długość (ang. *total length*) – 16 bitów

Łączna długość nagłówka oraz ładunku[9] (ang. *payload*) określona w bajtach. Z rozmiaru pola (16 bitów) wynika teoretyczna maksymalna długość datagramu 65535 ($65535 = 2^{16} - 1$) bajtów. Jednak specyfikacja protokołu IP wymaga, aby każdy host był w stanie zaakceptować datagram o rozmiarze 576 bajtów. Tym samym host nie może wysłać większego datagramu, jeżeli nie ma informacji od strony przeciwnej, że jest ona w stanie zaakceptować większy datagram. Minimalna długość datagramu wynosi 21 bajtów (20 bajtów nagłówka + 1 bajt danych).

Identyfikacja (ang. *identification*) – 16 bitów

Razem z adresami IP źródła, IP przeznaczenia oraz polem *protokół* służy do identyfikacji poszczególnych fragmentów tego samego datagramu. Po stronie źródłowej wykorzystywana w procesie dzielenia datagramu na fragmenty, a po stronie docelowej – w procesie scalania fragmentów w pierwotny datagram. Z chwilą dostarczenia do miejsca przeznaczenia pierwszego fragmentu uruchamiany jest stoper. Jeżeli w określonym czasie nie nadejdą kolejne fragmenty, to nieskompletowany datagram zostaje odrzucony. Pole *identyfikacja* ze względów technicznych[10] powinno być nadawane przez protokół warstwy wyższej.

Flagi (ang. *flags*) – 3 bity

Reprezentowane są przez trzy bity:

- bit 0 – nie używany, zawsze wyzerowany;
- bit 1 (ang. *don't fragment* – DF) – określa, czy można (wartość 0), czy też nie (wartość 1), fragmentować w razie potrzeby datagram;
- bit 2 (ang. *more fragments* – MF) – określa, czy dany fragment jest już ostatni (wartość 0), czy też są jeszcze kolejne fragmenty (wartość 1).

Przesunięcie fragmentu (ang. *fragment offset*) – 13 bitów

Określa w jednostkach 8-bajtowych pozycję pierwszego bajtu w polu danych bieżącego fragmentu, względem początku pola danych pierwotnego datagramu (niepodzielonego). Pole to umożliwia zatem hostowi docelowemu złożenie poszczególnych fragmentów w pierwotny datagram. Pierwszy fragment ma wartość przesunięcia ustawioną na zero.

Wszystkie fragmenty, z wyjątkiem ostatniego, muszą zawierać ładunek o wielokrotności 8 bajtów. Ponieważ maksymalny rozmiar pola danych datagramu przed fragmentacją wynosi 65515 (65515 = 65535-20), datagram może zostać podzielony na niewięcej niż 8190 fragmentów ($8190 = \lceil 65515/8 \rceil$).

[9] Ładunkiem datagramu nazywana jest zawartość jego pola danych.
[10] Przykładowo TCP dla retransmitowanego segmentu może ustalić taki sam identyfikator.

Wówczas każdy fragment będzie przenosił możliwie najmniejszą porcję danych.

Ponieważ nagłówek IP może wynosić 60 bajtów, a minimalny rozmiar pola danych w fragmencie to 8 bajtów, każda technologia warstwy drugiej współpracująca z IP, musi gwarantować możliwość przesłania datagramu 68-bajtowego.

Czas życia (ang. *time to live*) – 8 bitów

Licznik określający, ile sekund datagram może pozostawać w sieci. Ustawiany przez stację nadawczą na wartość z przedziału 1– 255 (wartość zalecana przez rekomendację wynosi 64), a następnie zmniejszany przynajmniej o 1, przy każdym przejściu datagramu przez router. Jeżeli datagram utknie wewnątrz routera w długiej kolejce i przetwarzany jest ponad sekundę, to pole TTL powinno być zmniejszone o tyle sekund, ile trwało jego przetworzenie. Jeżeli licznik osiągnie zero, datagram zostaje usunięty. Warstwa aplikacji powinna mieć możliwość ustawienia pola TTL.

Protokół (ang. *protocol*) – 8 bitów

Numer identyfikujący ładunek datagramu. Przykładowe wartości to:

- 1 – ICMP;
- 6 – TCP;
- 11 – UDP.

Suma kontrolna nagłówka (ang. *header checksum*) – 16 bitów

Służy do kontroli poprawności nagłówka[11]. Ponieważ wyliczana jest na podstawie pól nagłówka, w razie ich modyfikacji musi zostać zaktualizowana. Sprawdzana zarówno przez host docelowy, jak i przez routery pośredniczące. Jeżeli jest niepoprawna, datagram zostaje usunięty.

Adres źródła (ang. *source address*) – 32 bity

Adres IP nadawcy datagramu.

Adres docelowy (ang. *destination address*) – 32 bity

Adres IP adresata datagramu.

Opcje (ang. *options*) – zmienna długość

Pole umożliwiające określenie dodatkowych parametrów transmisji[12], np. wykorzystanie routingu źródłowego (ang. *source routing*), rejestrowanie przebytej

[11] Należy zwrócić uwagę, że ładunek IP nie jest chroniony sumą kontrolną.

[12] Pełna lista znajduje się pod adresem: http://www.iana.org/assignments/ip-parameters (24.03.2006).

trasy, przenoszenie znaczników czasu, określenie poziomu poufności datagramu. Jest polem opcjonalnym.

Uzupełnienie (ang. *padding*) – zmienna długość
Ewentualne dopełnienie zerami pola opcji do wielokrotność 32 bitów.

Dane (ang. *data*) – zmienna długość
Ładunek datagramu, czyli zaenkapsułowana jednostka danych protokołu wyższego poziomu (np. segment TCP, datagram UDP, pakiet ICMP).

2. Adresowanie logiczne

2.1. Budowa adresu IP i maski sieci

Aby Internet sprawiał złudzenie jednolitej sieci, wszystkie hosty muszą używać tego samego schematu adresowania. Do jednoznacznej identyfikacji hosta w Internecie służy adres IP. Obecnie komputery podłączone do Internetu wykorzystują adresowanie IPv4.

Adres IPv4 to 32-bitowa liczba binarna identyfikująca host. Ponieważ reprezentacja ta jest niewygodna dla ludzi, w praktyce wykorzystywana jest notacja dziesiętna z kropką (ang. *dotted-decimal notation*). W notacji tej adres IP reprezentowany jest przez cztery odseparowane kropką liczby dziesiętne. Dziedziną każdej z liczb jest zbiór liczb naturalnych z przedziału [0, 255].

Na poziomie logicznym w standardowym adresie IP wyróżniamy dwie części: identyfikator sieci oraz numer hosta. **Identyfikator sieci** adresuje sieć fizyczną, a **numer hosta** wskazuje konkretne urządzenie w ramach tej sieci. Oczywiste jest, że im więcej bitów adresu IP zostanie przeznaczonych na numer hosta, tym więcej hostów będzie można zaadresować w sieci fizycznej. Z kolei im więcej bitów przeznaczonych zostanie na identyfikator sieci, tym więcej sieci będzie można zaadresować.

Ze standardowym adresem IP skojarzona jest zawsze **maska sieci**, która fizycznie ma taką samą budowę jak adres IP. Maska sieci wskazuje, które bity z adresu IP identyfikują sieć, a które hosta. Bity o wartości 1 w masce oznaczają, że odpowiadające im bity w adresie IP należy interpretować jako identyfikator sieci. Natomiast wyzerowane bity w masce oznaczają, że odpowiadające im bity w adresie IP należy interpretować jako numer hosta. Innymi słowy, maska „zasłania" binarnymi jedynkami tą część adresu IP, w której zapisany jest identyfikator sieci.

2.2. Klasy adresów

W trakcie projektowanie schematu adresowania pojawił się problem, w jakich proporcjach podzielić adres IP na dwie części. Biorąc pod uwagę, że organiza-

cje mają różne wymagania co do liczby hostów w ramach sieci wewnętrznej, oraz przewidując liczbę sieci fizycznych, które będą wymagały unikalnych adresów, przestrzeń adresową podzielono na klasy A, B, C, D i E. Podział ten nazwano **adresowaniem klasowym**[13] (ang. *classful addressing*). Przynależność adresu do danej klasy determinowana jest przez wartość pierwszych bitów adresu, jak pokazano w tab. 2.

Tab. 2. Przyporządkowanie adresu IP do klasy

Klasa	**Pierwszy bajt adresu binarnie**								**Dziesiętnie**
A	0	X	X	X	X	X	X	X	0 – 127
B	1	0	X	X	X	X	X	X	128 – 191
C	1	1	0	X	X	X	X	X	192 – 223
D	1	1	1	0	X	X	X	X	224 – 239
E	1	1	1	1	X	X	X	X	240 – 255

Proporcja, w jakiej bity zostaną wykorzystane do zapisu każdej z części adresu, zależy od jego klasy (tab. 3). W klasie A identyfikator sieci zapisywany jest na pierwszym bajcie, w klasie B na pierwszych dwóch bajtach, a w klasie C na pierwszych trzech bajtach. Wynika z tego, że w klasie A pozostają trzy bajty na numery hostów, w klasie B – dwa bajty, a w klasie C – jeden bajt. Znając liczbę bitów przeznaczonych na poszczególne części adresu, łatwo wyliczyć, ile w każdej z klas można zaadresować sieci i ile hostów może każda z nich zawierać.

Tab. 3. Klasy adresów IPv4

1. bajt	**2. bajt**	**3. bajt**	**4. bajt**	**klasa**
Adres sieci	Numer hosta			A
Adres sieci		Numer hosta		B
Adres sieci			Numer hosta	C
Adres grupowy				D
Klasa zarezerwowana do badań				E

Należy od razu wspomnieć, że adres IP, który w części numer hosta ma binarnie same zera, nie identyfikuje żadnego hosta, a określa tylko **adres sieci**. Natomiast adres IP, który w części numer hosta ma binarnie same jedynki, to tzw. **adres rozgłoszeniowy** (ang. *broadcast*), oznaczający wszystkie hosty w danej sieci. Żadnego z tych adresów nie można przypisać do hosta.

Przy adresowaniu klasowym, każdy adres ma na sztywno przypisaną maskę sieci na podstawie klasy, do której należy:

– 255.0.0.0 dla klasy A;
– 255.255.0.0 dla klasy B;
– 255.255.255.0 dla klasy C.

Maska wynikająca z klasy adresu nazywana jest maską naturalną lub domyślną. Przy adresowaniu bezklasowym, omówionym w dalszej części opracowania, maska musi być podawana w postaci jawnej.

[13] W praktyce podział ten okazał się nieefektywny.

Maska służy routerowi do wydzielenia z adresu IP identyfikatora sieci. W tym celu router wykonuje operację bitowego AND na adresie IP i jego masce. Maska może być reprezentowana w alternatywnej postaci, jako tzw. prefiks adresu. Prefiks to liczba zapisywana po ukośniku, określająca, ile bitów adresu IP zajmuje identyfikator sieci. Zatem maska 255.255.0.0 w postaci prefiksowej będzie zapisana jako /16.

Adresy klasy D służą do komunikacji grupowej (ang. *multicast*), wykorzystywanej przykładowo na potrzeby konferencji. Przy komunikacji grupowej datagram dostarczany jest do zdefiniowanej grupy hostów. Pomimo że datagram kierowany jest do wielu odbiorców jednocześnie, host źródłowy wysyła tylko jedną kopię datagramu. Grupę mogą tworzyć hosty z różnych sieci, rozrzucone po całym Internecie, ale nasłuchujące na tym samym adresie. Host przyłącza się do grupy i odłącza za pomocą protokołu IGMP[14].

Adresy grupowe z zakresu od 224.0.0.0 do 224.0.0.255 zarezerwowane są dla protokołów działających w obrębie segmentu sieci lokalnej, np.: protokoły routingu wewnętrznego. W związku z tym datagramy o takich adresach nie powinny być przekazywane dalej przez routery szkieletowe Internetu. Oto przykładowe adresy grupowe wraz z ich znaczeniem:

– 224.0.0.9 – wszystkie routery z włączonym procesem routingu RIP2;
– 224.0.0.5 – wszystkie routery z włączonym procesem routingu OSPF;
– 224.0.0.6 – routery desygnowane z włączonym procesem routingu OSPF.

Organizacje w celu transmisji grupowej mogą wykorzystywać adresy z zakresu od 224.0.1.0 do 238.255.255.255, z drobnymi wyjątkami. Szczegółowy wykaz adresów grupowych wraz z przeznaczeniem znajduje się na stronach organizacji koordynującej funkcjonowanie Internetu – IANA.

[14] więcej informacji w RFC 1112

2.3. Adresy specjalne

W przestrzeni adresowej IP zarezerwowano część adresów do zastosowań specjalnych (tab. 4). Adresów tych nie można przypisywać do hostów.

Tab. 4. Adresy zarezerwowane i ich znaczenie

Adres specjalny	**Reprezentacja**	**Znaczenie**
rozgłoszenie ograniczone (ang. *limited broadcast*)	255.255.255.255	Datagram wysłany pod ten adres przeznaczony jest dla wszystkich hostów z bieżącej sieci lokalnej.
rozgłoszenie skierowane (ang. *directed broadcast*)	_.255.255.255 _._.255.255 _._._.255	Datagram wysłany pod ten adres przeznaczony jest dla wszystkich hostów z pewnej określonej sieci. Sieć ta określona jest przez identyfikator zapisany w zależności od klasy na jednym, dwóch lub trzech pierwszych bajtach adresu.
ten host w tej sieci	0.0.0.0	Adres zastępczy wykorzystywany przez hosta nieposiadającego skonfigurowanego adresu, na potrzeby komunikacji z serwerem RARP, BOOTP lub DHCP.
adres sieci	_.0.0.0 _._.0.0 _._._.0	Oznacza sieć, a nie konkretnego hosta. Nie może wystąpić ani jako adres przeznaczenia, ani jako adres źródła w datagramie IP.
adres sprzężenia zwrotnego (ang. internal loopback)	127._._._	Dowolny adres z sieci 127.0.0.0. Oznacza lokalnego hosta, czyli bieżący komputer. Datagram wysłany pod taki adres nie opuszcza karty sieciowej. Wykorzystywany do komunikacji międzyprocesowej w ramach jednego hosta lub do celów diagnostycznych. Ponadto z adresem 127.0.0.1 skojarzona jest statycznie nazwa „localhost”.

2.4. Adresy prywatne

Każdy host musi posiadać unikalny lokalnie adres IP. Unikalność adresu oznacza, że system sieci, z których dany host jest osiągalny, nie zawiera innego hosta o tym samym adresie. Adresy unikalne w Internecie określa się jako globalnie unikalne lub publiczne. Unikalność adresów publicznych nadzorowana jest centralnie przez organizację ICANN (ang. *Internet Corporation for Assigned Names and Numbers*, www.icann.org). W celu otrzymania adresu publicznego należy zgłosić się do lokalnego dostawcy Internetu (ang. *Internet Service Provider* – ISP).

Ponieważ przestrzeń adresowa IPv4 składa się z ok. 4 mln adresów, wraz ze wzrostem liczby komputerów stało się jasne, że nie dla wszystkich wystarczy adresów publicznych. Jednak nie wszystkie komputery wymagają takich adresów. Dla części komputerów wystarczy, aby mogły nawiązać komunikację z komputerami osiągalnymi w Internecie, ale same nie muszą być widoczne dla innych. Dla takich komputerów zarezerwowano trzy bloki tzw. **adresów prywatnych**:

– 10.0.0.0 - 10.255.255.255,
– 172.16.0.0 - 172.31.255.255,

– 192.168.0.0 - 192.168.255.255.

Adresy prywatne nie są przypisane do żadnej organizacji i mogą być dowolnie wykorzystywane w sieciach wewnętrznych. Datagramy o prywatnym adresie źródła lub przeznaczenia mają taką własność, że nie są przesyłane przez Internet. Oznacza to, że wystarczy, aby hosty o adresach prywatnych były unikalne lokalnie – wewnątrz intranetu. Powtórzenie tych samych adresów prywatnych w sieciach dwóch różnych organizacji podłączonych do Internetu nie powoduje konfliktu. Aby hosty o adresach prywatnych miały dostęp do Internetu, należy zastosować translację adresów (ang. *Network Address Translation* – NAT). **NAT** polega na zamianie adresu prywatnego na publiczny w momencie, gdy datagram przekracza router łączący sieć wewnętrzną z Internetem. W szczególności zastosowanie adresów prywatnych wraz z mechanizmem translacji umożliwia podłączenie do Internetu całej sieci wewnętrznej przy wykorzystaniu tylko jednego adresu publicznego. Technika ta nazywana jest **maskaradą** lub ze względu na konieczność translacji portów **PAT** (ang. Port Address Translation). Adresy prywatne zaleca się także stosować w sieciach nie mających połączenia z Internetm.

3. Tworzenie podsieci i supersieci

3.1. Podsieci o masce stałej długości

Każdą sieć klasy A, B lub C można podzielić na mniejsze części logiczne nazywane **podsieciami** (ang. *subnets*). Dokonuje się tego poprzez pożyczenie określonej liczby bardziej znaczących bitów z tej części adresu IP, która przeznaczona jest na numer hosta. Na pożyczonych bitach adresuje się podsieci. Im więcej pożyczonych bitów, tym więcej podsieci można utworzyć i tym mniejsza przestrzeń adresowa w każdej z nich. Należy zatem racjonalnie dobierać liczbę pożyczanych bitów do konkretnych wymagań. Podstawowym celem wydzielenia podsieci jest utworzenie kilku mniejszych domen rozgłoszeniowych w miejsce jednej dużej. Wpływa to korzystnie na wydajność sieci przy znacznym ruchu rozgłoszeniowym. W przypadku wydzielenia podsieci można także precyzyjniej definiować politykę bezpieczeństwa. Ponadto w każdej podsieci można zastosować inną technologię warstwy łącza danych.

Do wskazania liczby pożyczanych bitów służy **maska podsieci**. Maskę podsieci tworzy się na bazie maski domyślnej. Ponieważ numer podsieci jest „przedłużeniem” identyfikatora sieci, tworząc maskę podsieci należy dodatkowo ustawić na wartość 1 te bity, gdzie w odpowiadających im bitach adresu IP zapisany jest numer podsieci. Podział na podsieci tworzy trzypoziomową hierarchię w schemacie adresowania IP. Rysunek 4 przedstawia przykładowy adres klasy B z wydzieloną podsiecią na całym trzecim bajcie; podsieć została utwo-

rzona poprzez pożyczenie 8 najbardziej znaczących bitów z części przeznaczonej na numer hosta.

255.255.255.0 ≡	11111111	.	11111111	.	11111111	.	00000000
153.19.122.1 ≡	10011001	.	00010011	.	01111010	.	00000001
	identyfikator sieci				nr podsieci		nr hosta

Rysunek 4. Przykładowy adres IP z wydzieloną podsiecią

Utworzenie podsieci jest decyzją lokalną; tylko router bezpośrednio połączony z podsieciami wie o ich istnieniu. Dla routerów z zewnątrz, utworzone podsieci sprawiają wrażenie pojedynczej sieci.

3.2. Adresowanie bezklasowe

W miarę rozwoju Internetu narastały problemy związane z skalowalnością. Najważniejsze z nich to:

1. Przepełnienie tablic tras (ang. *routing tables*) w routerach szkieletowych Internetu, uniemożliwiające ich efektywne przetwarzanie.
2. Wyczerpanie dostępnych sieci klasy A i B oraz marnotrawienie przestrzeni adresowej tych sieci.[15]

Rozwiązaniem doraźnym umożliwiającym dalsze, sprawne funkcjonowanie Internetu, było wprowadzenie **bezklasowego routingu międzydomenowego** (ang. *Classless Inter-Domain Routing* – CIDR) oraz zastosowanie w intranetach adresów prywatnych w połączeniu z techniką PAT.

CIDR definiuje bezklasowy schemat adresowania IP, umożliwiający bardziej efektywną, hierarchiczną alokację adresów. Przed wprowadzeniem adresowania bezklasowego[16] organizacje, występujące o przydzielenie adresów publicznych, otrzymywały całe sieci. CIDR umożliwia dystrybucję przestrzeni adresowej w sposób bardziej oszczędny, dopasowany indywidualnie do wymogów organizacji. Adresowanie bezklasowe polega na zaprzestaniu używania domyślnych masek do interpretacji adresów IP i zastąpieniu ich maskami jawnymi. Do nowego schematu adresowania dostosowane zostały protokoły routingu[17], tak aby razem z adresami sieci rozgłaszały także maski.

Początkowym zastosowaniem CIDR stało się łączenie przylegających jednocześnie fizycznie i logicznie sieci w **nadsieci** (ang. *supernet*). Łączenie polega na „skróceniu" maski sieci. Nadsieci wyznacza się w celu ograniczenia liczby tras (poprzez agregację) wymienianych w procesie routingu. Przy czym, im bardziej hierarchiczna struktura sieci, tym efektywniej można wykorzystać

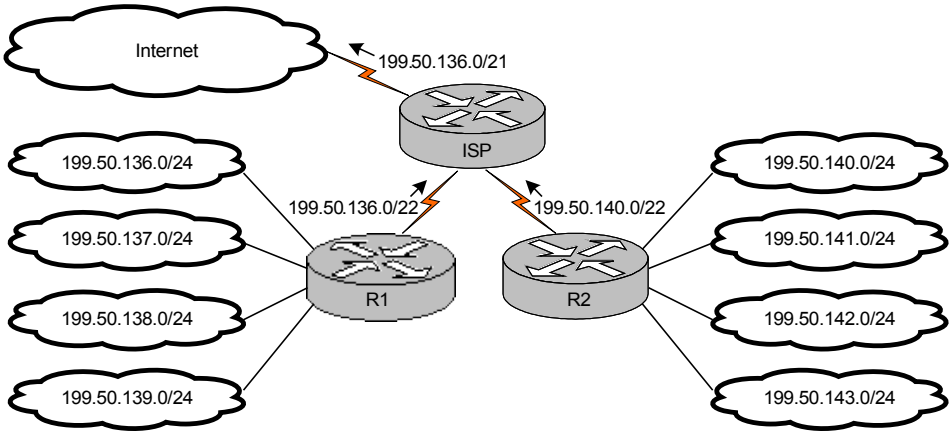
[15] Z jednej strony, duże organizacje zwykle potrzebują większe sieci (w sensie liczby hostów) niż dostępne w ramach klasy C (do 254 hostów). Natomiast z drugiej strony, żadko która organizacja wykorzysta całą przestrzeń adresową sieci klasy B (do 65534 hostów).

[16] CIDR opublikowany został w 1993 r., a wprowadzony na szeroką skalę w sieciach szkieletowych w 1997 r.

[17] Wprowadzono m.in. nowe wersje protokołów BGP (ang. *Border Gateway Protocol*), OSPF (ang. *Open Shortest Path First*) oraz RIP (ang. *Routing Information Protocol*).

CIDR i tym mniejsze tablice tras muszą utrzymywać routery. Obecnie, routery szkieletowe Internetu utrzymują przeciętnie 35 tyś wpisów.

W celu zobrazowania agregacji tras, adresy kolejnych sieci – od 199.50.136.0/24 do 199.50.143/24 – przydzielono do sieci fizycznych w sposób jak na rysunku 5.



Rysunek 5. Agregacja tras

Router R1, podczas wysyłania uaktualnień routingu do routera ISP, może zagregować cztery sieci lokalne i rozgłaszać je w postaci jednej nadsieci 199.50.136.0/22. Podobnie wystarczy, aby router R2 rozgłaszał nadsieć 199.50.140.0/22. Z kolei router ISP może dokonać dalszej agregacji i w kierunku Internetu rozgłaszać nadsieć 199.50.136.0/21. W przypadku adresowania klasowego router ISP musiałby rozgłaszać w kierunku Internetu informacje o ośmiu sieciach klasy C.

3.3. Maski podsieci o zmiennej długości

W większości przypadków stała długość maski dla wszystkich podsieci prowadzi do marnowania przestrzeni adresowej. Stąd zrodził się pomysł, aby dla każdej podsieci indywidualnie dobierać długość maski. Realizację pomysłu ułatwiło wprowadzenie bezklasowych protokołów routingu. Technika przydzielania masek podsieci o zmiennej długości (ang. *Variable Length Subnet Masks* – **VLSM**) umożliwia bardziej efektywne wykorzystanie przestrzeni adresowej, poprzez rozbicie jej na podsieci o wielkości dostosowanej do indywidualnych potrzeb organizacji.

4. Protokół ICMP

4.1. Charakterystyka protokołu ICMP

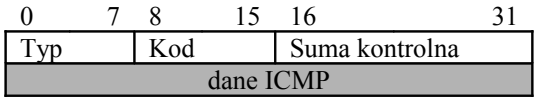
Protokół IP oferuje zawodną metodę dostarczania danych. Jeżeli w trakcie przesyłania datagramu wystąpi błąd, protokół IP nie ma możliwości, aby poinformować o tym stację nadawczą. Warstwa sieciowa posiada jednak mechanizm raportowania błędów IP – wykorzystuje do tego celu protokół ICMP (ang. *In-*

ternet Control Message Protocol). Urządzenie, które z powodu nieprawidłowości funkcjonowania systemu sieciowego musi usunąć datagram, może o przyczynie zaistniałego problemu powiadomić nadawcę datagramu. ICMP pełni także rolę pomocniczą w diagnozowaniu sieci, routingu, sterowaniu przepływem oraz zdobywaniu określonych informacji sieciowych.

Informacje ICMP kodowane są za pomocą zbioru predefiniowanych komunikatów. Każdy komunikat ma własny format. Struktura pierwszych 4 bajtów jest wspólna (rysunek 6):

– typ (ang. *type*) – określa typ komunikatu;
– kod (ang. *code*) – precyzuje typ komunikatu;
– suma kontrolna (ang. *checksum*) – obliczana z całego komunikatu ICMP.

Dalsze pola zależą od kodu komunikatu.



Rysunek 6. Format ogólnego komunikatu ICMP [RFC 792]

W celu przesłania przez sieć, komunikaty ICMP przenoszone są w datagramach IP. W przypadku niemożliwości dostarczenia datagramu, zawierającego komunikat ICMP, nie jest generowany nowy komunikat ICMP[18]. Zapobiega to powstaniu powodzi komunikatów ICMP w przypadku awarii sieci.

4.2. Wybrane komunikaty ICMP

Komunikaty ICMP można podzielić na dwie grupy: błędy i zapytania. Typowe komunikaty[19], należące do każdej z grup, przedstawiono w tablicy 5 i 6. Komunikaty zapytań (ang. *query messages*) służą do pozyskiwania informacji i występują w postaci par zapytanie–odpowiedź.

Tablica 5. Komunikaty zapytań [RFC 792]

Typ	Kod	Opis
0	0	echo reply
8	0	echo request
9	0	router advertisement
10	0	router selection
13	0	timestamp request
14	0	timestamp reply
17	0	address mask request
18	0	address mask reply

Komunikaty błędów (ang. *error messages*) generowane są w sytuacji niemożności dostarczenia datagramu. Z datagramu, który musi zostać odrzucony, pobierany jest adres IP źródła. Adres ten używany jest jako adres przeznaczenia

[18] Inne sytuacje, w których komunikaty ICMP nie są generowane opisane zostały w RFC 1812 sekcja 4.3.2.7.
[19] Zbiór wszystkich komunikatów dostępny jest pod adresem http://www.iana.org/assignments/icmp-parameters.

dla datagramu zawierającego stosowny komunikat ICMP. Komunikat błędu, oprócz podania przyczyny, powinien zawierać także tak dużo informacji z pierwotnego datagramu, jak to możliwe[20], pod warunkiem że sam zmieści się w datagramie nie przekraczającym 576 bajtów.

Tablica 6. Komunikaty błędów [RFC 792]

Typ	Kod	Opis
3		destination unreachable
	0	network unreachable
	1	host unreachable
	2	protocol unreachable
	3	port unreachable
	4	Fragmentation needed but don't-fragment bit set
	5	source route failed
	6	-
	7	destination host unknown
	8	-
	9	destination network administratively prohibited
	10	destination host administratively prohibited
	11	network unreachable for type of service
	12	host unreachable for type of service
	13	communication administratively prohibited by filtering
	14	host precedence violation
	15	precedence cutoff in effect
4	0	source quench
5		redirect
	0	redirect for network
	1	redirect for host
	2	redirect for type of service and network
	3	redirect for type of service and host
11		time exceeded
	0	time-to-live exceeded during transit
	1	fragment reassembly time exceeded
12		parameter problem
	0	pointer indicates the error
	1	required option missing
	2	bad length

Znaczenie najczęściej wykorzystywanych komunikatów jest następujące:

- **Network Unreachable** – generowany przez router, jeżeli w tablicy tras nie istnieje trasa do sieci docelowej, do której przeznaczony jest datagram;
- **Host Unreachable** – generowany przez router, jeżeli host docelowy znajduje się w bezpośrednio podłączonej sieci i jest niedostępny (nie odpowiada na zapytanie ARP);
- **Protocol Unreachable** – generowany, jeżeli protokół warstwy transportowej hosta docelowego jest niekompatybilny z protokołem transportowym hosta źródłowego;
- **Port Unreachable** – generowany, jeżeli port docelowy jest zamknięty;
- **Fragmentation needed and DF flag Set** – generowany jeżeli router musi wykonać fragmentację, a ustawiona jest flaga DF (don't fragment);

[20] Od wczesnych implementacji ICMP wymagano, aby zawierały jedynie nagłówek i osiem pierwszych bajtów z pola danych datagramu, który spowodował błąd.

- **Destination Host Unknown** – generowany, jeżeli router na poziomie sygnalizacji w warstwie łącza danych jest w stanie rozpoznać, że po drugiej stronie łącza nie jest podłączony żaden host;
- **Redirect for Host** – generowany przez router, aby powiadomić hosta o istnieniu lepszej trasy;
- **Time-To-Live Exceeded during Transit** – generowany przez router, jeżeli czas życia przetwarzanego datagramu zmalał do zera;
- **Fragment Reassembly Time Exceeded** – generowany przez host docelowy, jeżeli w określonym przedziale czasu nie otrzyma wszystkich fragmentów pakietu;
- **Source Quench** – generowany w przypadku przepełnienia bufora i niemożliwości przetworzenia nadsyłanych pakietów; stanowi prośbę o spowolnienie transmisji;
- **Source Route Failed** – generowany, jeżeli router nie może przesłać datagramu zgodnie z trasą routingu źródłowego;
- **Communication with Destination Network is Administratively Prohibited** – generowany przez router ze skonfigurowaną listą kontroli dostępu, blokującą datagramy przeznaczone do wskazanej sieci docelowej;
- **Communication with Destination Host is Administratively Prohibited** – generowany przez router ze skonfigurowaną listą kontroli dostępu, blokującą datagramy przeznaczone do wskazanego hosta docelowego;
- **Network Unreachable for Type of Service** – generowany przez router, jeżeli trasa do wskazanej sieci docelowej nie spełnia wymogów QoS wymaganych przez pole typ usługi;
- **Echo Request/Echo Reply** – para komunikatów wykorzystywana w celu weryfikacji łączności między dwoma hostami; lokalny host w celu sprawdzenia łączności z hostem zdalnym wysyła do niego komunikat *echo request*; jeżeli zdalny host w odpowiedzi odeśle komunikat *echo reply*, to uważany jest za osiągalny;
- **Timestamp Request/Timestamp Reply**[21] – lokalny host w celu uzyskania od zdalnego hosta bieżącego czasu wysyła do niego komunikat *timestamp request*; zdalny host powinien odpowiedzieć komunikatem *timestamp reply*, zawierającym czas w postaci liczby milisekund, które upłynęły od północy, według CUT (ang. *Coordinated Universal Time*).

[21] Bardziej wiarygodną metodą uzyskania czasu jest protokół NTP (Network Time Protocol) działający w warstwie aplikacji.

5. Warstwa transportowa

5.1. Charakterystyka warstwy transportowej

Protokół IP implementuje zawodną usługą przesyłania danych. Oznacza to, że nie posiada mechanizmu retransmisji na wypadek, gdy dane zostaną utracone lub przesłane niepoprawnie. Ponadto IP nie porządkuje nadchodzących danych, jeżeli przyjdą w innej kolejności niż zostały wysłane. Zadania te może realizować warstwa transportowa. Struktura danych, na której operuje warstwa transportowa, to segment.

Warstwa transportowa oferuje dwa sposoby komunikacji: **połączeniowy** – za pomocą protokołu TCP (ang. *Transmission Control Protocol*) oraz **bezpołączeniowy** – za pomocą protokołu UDP (ang. *User Datagram Protocol*). Komunikacja połączeniowa wymaga zestawienia połączenia przed rozpoczęciem przesyłania segmentów. Po zakończeniu komunikacji połączenie należy zwolnić. Typowe protokoły warstwy aplikacji, działające w oparciu o usługi połączeniowe warstwy transportowej, to: Telnet, SSH, FTP, HTTP, SMTP, POP3. Z kolei w przypadku komunikacji bezpołączeniowej segment może być wysłany bez żadnych czynności wstępnych. W szczególności odbiorca nie musi wyrażać zgody na odebranie segmentu. Komunikacja bezpołączeniowa wykorzystywana jest zwykle przez protokoły: TFTP, SNMP, DHCP, DNS, NFS, RTP.

Zadania warstwy transportowej realizowane są wyłącznie na komunikujących się hostach końcowych. Z punktu widzenia warstwy transportowej cała infrastruktura sieciowa między jednym hostem a drugim to jednolity system.

5.2. Cechy protokołu TCP

Protokół TCP odseparowuje warstwę aplikacji od zawodnych usług warstwy sieciowej. Jak już wspomniano, protokół TCP jest połączeniowy (ang. *connection-oriented*). Jednak ze względu na to, że cała intersieć traktowana jest jako jedno łącze komunikacyjne, połączenie to ma charakter wirtualny i realizowane jest tylko w oparciu o hosty końcowe. W ramach wirtualnego połączenia wykonywane są następujące funkcje: sterowanie przepływem (ang. flow control), kontola przeciążenia (ang. congestion control) oraz gwarancja niezawodności.

TCP oferuje wyłącznie komunikację unicast, innymi słowy wszystkie połączenia są typu punkt–punkt (ang. *point-to-point*). Wykorzystywany przez TCP kanał komunikacyjny jest dwukierunkowy i umożliwia komunikację full-duplex[22].

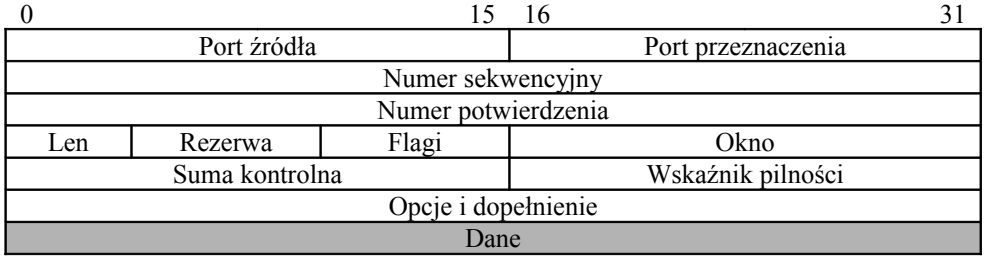
TCP udostępnia interfejs strumieniowy dla aplikacji, ale sam działa w trybie pakietowym. Oznacza to, że od protokołu warstwy aplikacji odbiera strumień bajtów i umieszcza go w segmentach.

[22] Full-duplex oznacza możliwość równoczesnej komunikacji w obu kierunkach.

TCP wykorzystuje technikę „jazdy na barana” (ang. *piggy-backing*). Oznacza to, że informacje takie, jak potwierdzenia, czy rozmiar okna, nie są przesyłane w postaci specjalnych segmentów sterujących. Są wtrącane w nagłówki zwykłych segmentów z danymi i wysyłane „przy okazji”.

5.3. Budowa segmentu TCP

Strukturę segmentu TCP przedstawia rysunek 7, a znaczenie poszczególnych pól opisano poniżej.



Rysunek 7. Struktura segmentu TCP [RFC 793]

Port źródła (ang. *source port*) – 16 bitów

Punkt dostępowy do warstwy aplikacji po stronie nadawcy.

Port przeznaczenia (ang. *destination port*) – 16 bitów

Punkt dostępowy do warstwy aplikacji po stronie odbiorcy. Wskazuje dla jakiej aplikacji przeznaczony jest ładunek przenoszony w segmencie.

Numer sekwencyjny (ang. *sequence number*) – 32 bity

Numer pierwszego bajtu danych enkapsułowanych w segmencie. Określa pozycję danych przenoszonych w bieżącym segmencie względem pierwotnego strumienia bajtów generowanego przez warstwę aplikacji. Stronie odbiorczej umożliwia odtworzenie pierwotnego strumienia. Jeżeli dodatkowo ustawiony jest bit SYN, to pole to zawiera tzw. inicjujący numer sekwencyjny, od którego rozpoczyna się numerację bajtów w połączeniu.

Numer potwierdzenia (ang. *acknowledgment number*) – 32 bity

Numer sekwencyjny następnego oczekiwanego od strony przeciwnej segmentu. Dodatkowo informuje, że wszystkie bajty o numerach mniejszych od zawartego w tym polu zostały odebrane poprawnie.

Długość nagłówka (ang. *header length*) – 4 bity

Liczba słów 32-bitowych, z których składa się nagłówek. Umożliwia wyliczenie miejsca, w którym rozpoczyna się pole danych.

Rezerwa (ang. *reserved*) – 6 bitów

W początkowej wersji pozostawione do przyszłych zastosowań i wypełnione zerami. W nowszych implementacjach na trzech ostatnich bitach rezerwy wydzielono pole jawnej informacji o przeciążeniach[23] (ECN – ang. Explicit Congestion Notification). Pole ECN składa się z trzech znaczników:

– NS (ang. Nonce Sum);
– ECE (ang. ECN-Echo);
– CWR (ang. Congestion Window Reduced).

Flagi (ang. *flags*) – 6 bitów

Pole to składa się z sześciu flag sprawujących funkcje kontrolne:

1. **URG** (ang. *urgent*) – oznacza segment z pilnymi danymi[24] oraz wskazuje na ważność pola wskaźnik pilności. Segment z pilnymi danymi powinien mieć pierwszeństwo przed innymi segmentami. W szczególności host docelowy nie powinien umieszczać takiego segmentu w kolejce wejściowej, tylko przetworzyć go natychmiast. Z kolei aplikacja, dla której przeznaczony jest segment z pilnymi danymi, może otrzymać sygnał w celu zawieszenia dotychczasowych działań i natychmiastowego przetworzenia pilnych danych.
2. **ACK** (ang. *acknowledge*) – oznacza, że segment jest jednocześnie potwierdzeniem. Wskazuje na ważność pola *numer potwierdzania*.
3. **PSH** (ang. *push*) – oznacza żądanie warstwy aplikacji o natychmiastowe opróżnienie buforu i wysłanie segmentu. Funkcja wykorzystywana zwykle przez aplikacje interakcyjne, np. Telnet. Jeżeli flaga ta nie jest ustawiona, to dane przekazywane z warstwy aplikacji, ze względów efektywności wykorzystania pasma, mogą być buforowane w celu przesłania większej porcji w jednym segmencie.
4. **RST** (ang. *reset*) – informacja o konieczność zresetowania połączenia. Zwykle ustawiana w sytuacjach awaryjnych.
5. **SYN** (ang. *synchronize*) – flaga wykorzystywana w trakcie nawiązywania połączenia. Wskazuje, że w polu numer sekwencyjny umieszczony jest inicjujący numer sekwencyjny.
6. **FIN** (ang. *finalize*) – flaga wykorzystywana w trakcie kończenia połączenia. Wskazuje, że nadawca nie zamierza wysyłać więcej danych i ma zamiar zakończyć połączenie. Nadawca wciąż jednak pozostaje gotowy do odbierania danych.

[23] Wymaga współpracy z polem ECN w nagłówku IP.
[24] Przykładowo podczas sesji Telnet pilne dane generowane są w momencie naciśnięcia kombinacji Ctrl-C.

Okno (ang. *window*) – 16 bitów

Ogłoszenie liczby bajtów, które host jest gotów odebrać w ramach bieżącego połączenia, począwszy od bajtu zapisanego w polu numer sekwencyjny potwierdzenia. Wartość ta ustalana jest dynamicznie. Zerowa wartość oznacza, że z wysłaniem kolejnych bajtów strona przeciwna powinna się wstrzymać.

Suma kontrolna (ang. *checksum*) – 16 bitów

Wielkość obliczona na podstawie: nagłówka TCP, ładunku TCP, długość segmentu TCP oraz pseudonagłówka. Pseudonagłówek tworzony jest na podstawie adresów źródła i przeznaczenia oraz pola *protokół* z nagłówka IP.

Wskaźnik pilności (ang. *urgent pointer*) – 16 bitów

Wskazuje koniec pilnych danych. Zawiera numer bajtu (liczony względem numeru sekwencyjnego), następującego po pilnych danych.

Opcje (ang. *options*) – zmienna długość

Pole służące do przesłania dodatkowych informacji. Najczęściej wykorzystywane opcje to:

1. **Maksymalny rozmiar segmentu** (ang. *Maximum Segment Size* – MSS) – informacja przesyłana w trakcie nawiązywania połączenia w celu ogłoszenia maksymalnego rozmiaru segmentu, jaki host jest w stanie przyjąć.
2. **Potwierdzenie negatywne** (ang. *Negative Acknowledgment* – NAK) – standardowe mechanizmy TCP umożliwiają potwierdzenie ciągłego strumienia poprawnie odebranych bajtów. Problem pojawia się, jeżeli utracony zostanie pojedynczy segment, po którym nastąpi seria segmentów poprawnie odebranych. Wówczas wykorzystując standardowe mechanizmy TCP, nie ma możliwości potwierdzenia wszystkich poprawnych segmentów i zażądania retransmisji tylko tego utraconego. NAK umożliwia wskazanie konkretnego segmentu, który powinien być retransmitowany. Żądanie retransmisji NAK wysyłane jest, jak tylko pojawi się luka między oczekiwanym numerem sekwencyjnym a numerem sekwencyjnym odebranego segmentu. Należy nadmienić, że taka luka wcale nie musi oznaczać, że segment został utracony. Oczekiwany segment może być po prostu opóźniony.
3. **Dozwolone potwierdzenia selektywne** (SACK-permitted) – informacja przesyłana w trakcie nawiązywania połączenia w celu ogłoszenia możliwości stosowania selektywnych potwierdzeń.
4. **Potwierdzenie selektywne** (ang. *Selective Acknowledgement* – SACK) – potwierdzenie selektywne poprawnie odebranych segmentów. Poprawnie odebrane bajty strumienia przedstawiane są w postaci przedziałów.

5. **Znacznik czasu** (ang. *timestamp*) – służy do oszacowania czasu podróży segmentu w obie strony (ang. *Round-Trip Time* – RTT).
6. **Duże okno**[25] (ang. *big window*) – wskazuje, że 16 bitów z pola *okno* należy interpretować jako mniej znaczącą część liczby, będącej rzeczywistym rozmiarem okna. Pozostałe 14 bitów bardziej znaczącej części tej liczby znajduje się w polu *opcji*.
7. **Skalowanie okna** (ang. *window scale*) – służy do przesłania współczynnika, przez który należy pomnożyć wartość zapisaną w polu *okno*, aby uzyskać rzeczywistą wartość okna. Rozwiązanie alternatywne w stosunku do opcji *duże okno*.

Dopełnienie (ang. *padding*) – zmienna długość
Uzupełnienie zerami pola opcji do wielokrotności 32 bitów.

Dane (ang. *data*) – zmienna długość
Ładunek segmentu, czyli zaenkapsułowane dane protokołu wyższego poziomu.

5.4. Rozmiar segmentu TCP

Na trasie między dwoma hostami mogą występować sieci o różnych MTU. Minimalna wartość MTU na trasie nazywana jest **PMTU** (ang. *path MTU*). Znajomość PMTU jest istotna, ponieważ dzięki tej informacji host lokalny wie, jak duży segment TCP może utworzyć, aby bez fragmentacji dotarł do miejsca przeznaczenia. Wybór zbyt dużego rozmiaru segmentu wymusza fragmentację, która wymaga dodatkowego przetwarzania, a więc wydłuża czas transmisji. Z kolei przy aplikacjach takich jak FTP wybór zbyt małego rozmiaru segmentu wprowadza większe narzuty transmisji związane z przesłaniem danych sterujących.

Proces ustalenia PMTU może być zrealizowany w następujący sposób. Początkowa wartość PMTU ustawiana jest na wartość MTU sieci lokalnej. Do zdalnego hosta wysyłany jest datagram z ustawioną flagą DF (ang. *don't fragment*) oraz maksymalnym rozmiarem pola danych, wynikającym z PMTU. Jeżeli datagram będzie zbyt duży, aby przejść bez fragmentacji przez któryś z odcinków sieci, zostanie wysłany komunikat błędu „fragmentation needed and DF set". Komunikat ten zawiera dodatkowo MTU sieci, na której zatrzymany został pierwotny datagram. Nadawca musi wówczas stosownie zmniejszyć PMTU i ponownie wysłać datagram. Proces ten powtarzany jest tak długo, aż datagram dotrze do miejsca przeznaczenia. Wówczas znana jest prawdziwa wartość PMTU.

[25] Poprawia znacznie efektywność protokołu w przypadku łącza o dużej przepustowości oraz dużym opóźnieniu, np. łącze satelitarne.

Protokół IP jest bezpołączeniowy, a więc nie przechowuje informacji, jak duże datagramy może wysyłać do poszczególnych hostów docelowych. Wystarczy jednak, aby informacje te pamiętał protokół TCP, a do protokołu IP przekazywał dane w odpowiednich porcjach. W tym celu przy nawiązywaniu połączenia, każda ze stron powinna ogłosić **MSS** (ang. *Maximum Segment Size*). MSS to maksymalna liczba bajtów w polu danych segmentu, którą host jest w stanie odebrać, przy założeniu, że nagłówki IP oraz TCP wynoszą po 20 bajtów. Wartość ta obliczana jest w następujący sposób:

```
MSS := MTU - 20 - 20
```

Jeżeli któraś ze stron nie ogłosi MSS, to domyślnie przyjmuje się, że ogłosiła wartość 536 (576 – 20 – 20). Wartość ta wynika z specyfikacji IP, która wymaga, aby każdy host był w stanie zaakceptować datagram o rozmiarze do 576 bajtów.

Na podstawie ogłoszonych wartości MSS lub procesu odkrywania PMTU, każda ze stron wylicza **efektywny MSS**. Efektywny MSS wyznacza maksymalny rozmiar pola danych wysyłanego segmentu. Jeżeli host ma możliwość ustalenia PMTU to efektywny MSS wylicza w następujący sposób:

```
hlen := rozmiarNagłówkaIP + rozmiarNagłówkaTCP
if (istnieje możliwość ustalenia PMTU) then
  efektywny_MSS := PMTU - nagłówki
else
  efektywny_MSS := min( MTU, odebranyMMS + 40 ) - hlen
```

Zmienne *rozmiarNagłówkaIP* oraz *rozmiarNagłówkaTCP* ustalane są dynamicznie w zależności od rzeczywistych danych, które mają być wysłane.

5.5. Sterowanie przepływem TCP

Podczas nawiązywania połączenia TCP rezerwowana jest pamięć na bufor nadawczy oraz odbiorczy. Bufor nadawczy przechowuje segmenty oczekujące na wysłanie. Z kolei bufor odbiorczy służy do składowania napływających z sieci segmentów, zanim zostaną przetworzone i przekazane do warstwy aplikacji. Jeżeli host zasypywany jest dużą ilością segmentów (np. serwer, który otrzymuje równocześnie dane z wielu źródeł), może nie nadążać z ich obsługą. W przypadku zapełnienia buforu odbiorczego, wszystkie przychodzące segmenty zostaną odrzucone. Podobnie wygląda sytuacja, kiedy urządzenia międzysieciowe nie mogą pomieścić nadchodzących danych – muszą je usuwać. Wówczas host nadawczy nie otrzyma potwierdzenia poprawnego dostarczenia segmentów i po pewnym czasie dokona retransmisji, zgodnie z algorytmem zapobiegania przeciążeniom. Aby jednak nie dopuścić do takiej sytuacji wykorzystuje się mechanizm zapobiegający przepełnieniu bufora odbiorczego – sterowanie przepływem.

Mechanizm **sterowania przepływem** umożliwia odbiorcy ograniczenie ilości wysyłanych do niego danych. Realizowane jest to w następujący sposób.

Strona odbiorcza relacjonuje do strony nadawczej ilość wolnego miejsca w buforze odbiorczym. Informacja ta zapisana jest w polu *okno*[26] i wskazuje, ile bajtów danych można do danego hosta wysłać przed otrzymaniem kolejnego zezwolenia. Zezwoleniem będzie oczywiście segment, z niezerowym rozmiarem okna, potwierdzający odebrane do tej pory dane. Segment z zerowym rozmiarem okna wysyłany jest w przypadku zapełnienia bufora i oznacza, że należy wstrzymać wysyłanie segmentów.

5.6. Kontrola przeciążenia TCP

Wartość okna ogłaszana przez zdalnego hosta (ang. *advertised window*) informuje jedynie o ilości danych, które jest gotów odebrać. Nie uwzględnia natomiast możliwości przesłania tych danych przez urządzenia pośredniczące. W celu uzależnienia ilości wysyłanych danych od aktualnej przepustowości sieci, TCP dostarcza mechanizm **kontroli przeciążenia**. Ilość danych, jaką host może wysłać nie powodując przepełnienia buforów w routerach pośredniczących, to **okno przeciążenia** (ang. *congestion window*). Dla ułatwienia dalszych zapisów, wzorując się na specyfikacji TCP, wykorzystywane będą następujące oznaczenia:

- *cwnd* – rozmiar okna przeciążenia;
- *rwnd* – rozmiar okna ogłoszony od przeciwnej strony w nagłówku TCP;
- *MSS* – maksymalny rozmiar segmentu ustalony w trakcie nawiązywania połączenia;
- *ssthresh* – próg powolnego startu (ang. slow start threshold);
- *min(x,y)* – funkcja zwracająca wartość minimalną z przekazanych argumentów;
- *max(x,y)* – funkcja zwracająca wartość maksymalną z przekazanych argumentów.

Maksymalna liczba bajtów danych, które host może wysłać przed otrzymaniem kolejnego zezwolenia wynosi *min(rcwd, cwnd)*.

Dwa podstawowe algorytmy wykorzystywane do ustalania wartości okna przeciążenia to powolny start i unikanie przeciążeń. Opcjonalnie TCP może także implementować algorytmy szybkiej retransmisji oraz szybkiego przywracania[27].

Powolny start (ang. *slow start*) polega na transmisji początkowych bajtów danych niewielkimi porcjami w celu wypróbowania przepustowość sieci. Niewielka porcja danych zwykle oznacza wstępne ustawienie *cwnd* na wartość *MSS*, która domyślnie wynosi 536 bajtów. Z kolei wartość zmiennej *ssthresh* w momencie nawiązywania połączenia ustawiana jest na stosunkowo wysoką war-

[26] Ze względu na adaptacyjne dostosowywanie wartości okna do bieżącego stanu bufora, okno to nazywane jest przesuwnym (ang. *sliding window*).

[27] Sposób działania algorytmu szybkiej retransmisji oraz szybkiego przywracania został opisany w sekcji 3.2 dokumentu RFC 2581.

tość, np. 65535, lub na wartość *rwnd*. W momencie, gdy wysłana porcja danych zostanie potwierdzona, wartość zmiennej *cwnd* jest podwajana. Wykładnicze zwiększanie warości *cwnd* kończy się, jeżeli *cwnd* > *ssthresh*. Dalsze zwiększanie *cwnd* ma charakter liniowy i określone jest przez algorytm unikania przeciążenia.

Unikanie przeciążenia (ang. *congestion avoidance*) opiera się na założeniu, że utrata segmentu spowodowana jest przeciążeniem sieci. Segment uznaje się za utracony, jeżeli w ustalonym czasie nie dotrze jego potwierdzenie lub jeżeli dotrze duplikat potwierdzenia poprzedniego. Po każdorazowym otrzymaniu potwierdzenia wartość okna przeciążenia zwiększana jest według następującej formuły:

$$cwnd := cwnd(1 + MSS^2/cwnd^2)$$

Natomiast w razie utraty segmentu wykonywany jest poniższy algorytm:

```
ssthresh := max[ 2×MSS , 0.5×min(cwnd, rwnd) ]
cwnd := MMS
if (cwnd < ssthresh) then przejście do algorytmu powolnego startu
```

W momencie zapełnienia bufora odbiorczego, kolejne nadchodzące pakiety zostaną odrzucone. Wówczas host nadawczy nie otrzyma potwierdzenia i stwierdzi utratę segmentu. Na podstawie utraty segmentu wywnioskuje, że w sieci wystąpił zator, i rozpocznie procedurę unikania przeciążenia. Po rozładowaniu zatoru wszystkie utracone segmenty zostaną retransmitowane. Mechanizm ten nie zakłóca funkcjonowania aplikacji odpornych na opóźnienia (np. poczta elektroniczna), ale zupełnie nie nadaje się do przenoszenia danych aplikacji interaktywnych. W aplikacjach interaktywnych opóźnienia związane z retransmisją segmentów są niedopuszczalne. Istnieje zatem potrzeba rozpoczęcia procesu rozładowywania zatorów, zanim bufory routerów zostaną przepełnione i zanim pakiety będą odrzucane. Mechanizm wczesnej detekcji zatorów i informowania o nich hostów końcowych nazywany jest ogólnie **aktywnym zarządzaniem kolejką** (ang. *active queue management* – AQM). AQM umożliwia routerom powiadomienie końcowych hostów o powstaniu zatorów za pomocą znacznika CE umieszczonego w nagłówku IP. Host końcowy powinien wówczas rozpocząć procedurę unikania przeciążeń w taki sam sposób jak przy utracie segmentu. Ponadto przy wysłaniu potwierdzenia powinien ustawić flagę ECE w segmencie TCP. Potwierdzenie z ustawioną flagą ECE powinno uaktywnić procedurę unikania przeciążenia w taki sam sposób jak przy żądaniu retransmisji. Zastosowanie się do flagi ECE sygnalizowane jest flagą CWR ustawioną w segmencie wysyłanym w przeciwnym kierunku.

5.7. Gwarancja niezawodności transmisji TCP

Gwarancja niezawodności dostarczania danych w podstawowej wersji TCP osiągana jest przez mechanizm pozytywnych potwierdzeń (ang. *positive ack-*

nowledgment) i retransmisji. Każdy bajt ze strumienia danych generowanego przez aplikację jest numerowany. Protokół TCP dzieli strumień danych generowany przez aplikację na fragmenty, które następnie pakuje w segmenty. Aby strona odbiorcza wiedziała, w jakiej kolejności składać poszczególne segmenty w celu odtworzenia pierwotnego strumienia, nagłówek każdego z nich opatrywany jest numerem sekwencyjnym. Pole *numer sekwencyjny* informuje jaką pozycję w pierwotnym strumieniu zajmował pierwszy bajtu z pola danych bieżącego segmentu.

W momencie wysłania segmentu jego kopia wędruje do kolejki retransmisji oraz uruchamiany jest stoper. Jeżeli segment dotrze do miejsca przeznaczenia i będzie zawierał poprawną sumę kontrolną, strona odbiorcza będzie zobowiązana, aby go potwierdzić. Po odebraniu potwierdzenia, segment usuwany jest z kolejki retransmisji. Jeżeli wysłany segment nie zostanie potwierdzony w przedziale czasu na to przewidzianym (ang. *retransmission timeout* – RTO), wysyłany jest ponownie. RTO ma charakter adaptacyjny, uzależniony od ważonej średniej ruchomej czasu podróży segmentu w obie strony (ang. *Smoothed Round Trip Time* – SRTT) oraz zmienności RTT (ang. *Round Trip Time Variation* – RTTVAR). Przy pierwszym pomiarze RTT ustawiane są następujące zmienne:

```
SRTT := RTT
RTTVAR := 0.5×RTT
RTO := SRTT + 4×RTTVAR
```

Natomiast przy kolejnych pomiarach wykonywany jest poniższy algorytm:

```
RTTVAR := (1-β)×RTTVAR + β×|SRTT - RTT|
SRTT := (1-α)×SRTT + α×RTT
RTO := SRTT + 4×RTTVAR
```

gdzie $\alpha=1/8$, $\beta=1/4$.

Może się zdarzyć, że segment zostanie odebrany poprawnie i potwierdzony, a mimo wszystko stacja nadawcza wykona retransmisję, np. z powodu nie otrzymania potwierdzenia na czas. W takiej sytuacji stacja docelowa, na podstawie numeru sekwencyjnego, ma możliwość odrzucenia duplikatu.

Najprostszy sposób **potwierdzeń pozytywnych** – realizowany przez protokoły typu „**stop and wait**”[28] – polega na potwierdzaniu każdego segmentu osobno. Kolejny segment w tym samym kierunku nie może zostać wysłany, dopóki poprzedni nie zostanie potwierdzony przez drugą stronę. Potwierdzenie polega na wysłaniu segmentu w przeciwnym kierunku[29], z polem *numer potwierdzenia* ustawionym na numer kolejnego oczekiwanego bajtu danych. Numer oczekiwanego bajtu można wyliczyć, dodając do numeru sekwencyjnego ostatnio odebranego segmentu rozmiar jego pola danych. Do wysłania potwier-

[28] Nazwa wzięła się stąd, że po wysłaniu segmentu należy się wstrzymać z wysłaniem kolejnego i poczekać, aż przyjdzie potwierdzenie.
[29] Od strony potwierdzającej do strony, która wygenerowała potwierdzany segment.

dzenia zwykle nie jest tworzony osobny, pusty segment tylko wykorzystywany jest „najbliższy" segment z danymi przeznaczonymi dla drugiej strony. Wyjątek stanowi sytuacja, gdy strona potwierdzająca nie ma żadnych danych do wysłania.

Może się jednak zdarzyć, że segment nie dotrze do miejsca przeznaczenia na czas lub dotrze uszkodzony. Uszkodzenie danych wykrywane jest na podstawie sumy kontrolnej zamieszczonej w nagłówku segmentu. W takiej sytuacji generowane jest żądanie retransmisji, polegające na umieszczeniu w polu *numer potwierdzenia* numeru sekwencyjnego brakującego segmentu.

Efektywniejszą formę potwierdzeń pozytywnych oferują **protokoły okienkowe**, do których należy zatwierdzony w 1981r. TCP. Protokoły okienkowe umożliwiają stronie nadawczej wysłanie więcej niż jednego segmentu, bez każdorazowego czekiwania na potwierdzenie. Z kolei strona odbiorcza może wysłać potwierdzenie natychmiast po odebraniu segmentu lub ze względów efektywnościowych może chwilę poczekać w celu łącznego potwierdzenia kilku segmentów jednocześnie.

Potwierdzenie takie nazywane skumulowanym, polega na zapisaniu w polu *numer potwierdzenia*, numeru o jeden większego niż numer ostatniego bajtu ładunku z poprawnie odebranego segmentu. Wymóg ciągłości jest ważny, ponieważ pole *numer potwierdzenia* interpretowane jest w ten sposób, że wszystkie bajty danych o numerach mniejszych zostały przesłane poprawnie.

W podejściu kumulatywnym segmenty poprawnie odebrane nie mogą zostać potwierdzone dopóty, dopóki nie zostaną poprawnie przesłane wszystkie ich poprzedniki (te o mniejszych numerach sekwencyjnych). Ponadto na podstawie prośby o retransmisję nie można wywnioskować, czy utracie uległ tylko jeden segment (ten, którego numer sekwencyjny zapisany jest w polu *numer potwierdzenia*), czy więcej.

Jeżeli stacja źródłowa jednorazowo retransmitować będzie tylko jeden segment – ten jawnie wskazywany jako utracony, to w przypadku utraty wielu kolejnych segmentów znacznie spadnie wydajność protokołu. Natomiast jeżeli stacja źródłowa zastosuje strategię agresywną i retransmitować będzie wszystkie segmenty, począwszy od numeru przekazanego w prośbie o retransmisję, to w przypadku utraty jednego segmentu, wszystkie pozostałe będą retransmitowane niepotrzebnie.

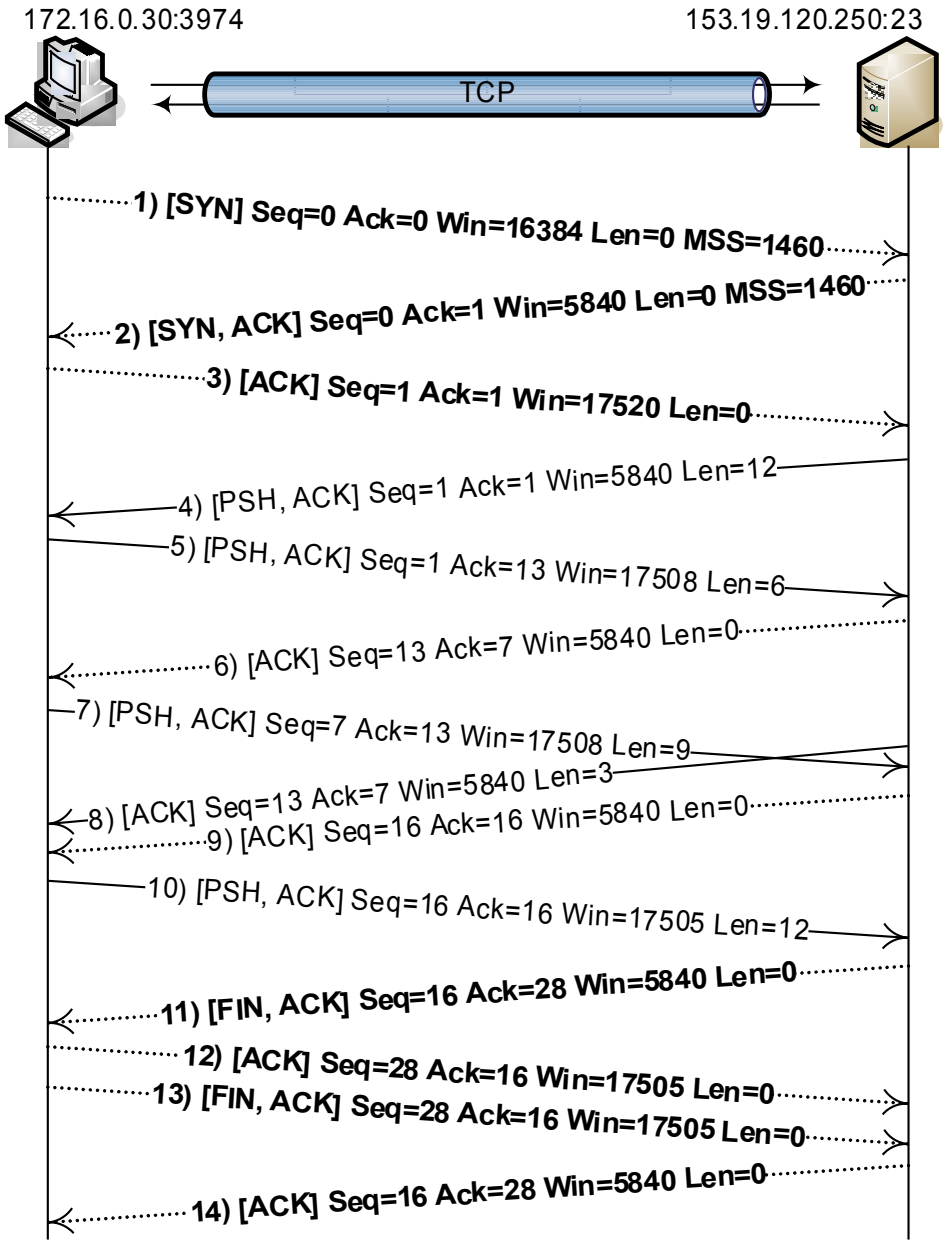
W 1996 roku, aby usprawnić potwierdzenia kumulatywne, wprowadzono do TCP opcjonalny mechanizm **potwierdzeń selektywnych**[30] (ang. *Selective Acknowledgment* – SACK). SACK umożliwia w jednym potwierdzeniu przesłanie numerów poprawnie odebranych segmentów w postaci kilku przedziałów.

[30] Specyfikację SACK zawiera dokument RFC-2018. W 2000 roku opublikowano rozszerzenie tego dokumentu w postaci RFC-2883. RFC-2883 opisuje sposób obsługi duplikatów potwierdzeń.

Na tej podstawie stacja źródłowa może wywnioskować, które segmenty zostały utracone i wykonać ich retransmisję.

5.8. Nawiązywanie i kończenie połączenia TCP

Protokoły połączeniowe przed wymianą danych muszą zestawić połączenie, a po zakończeniu wymiany muszą połączenie zwolnić. Przebieg sesji TCP przedstawia rysunek 8.



Rysunek 8. Sesja TCP
Linią ciągłą między klientem a serwerem oznaczono segmenty z danymi, natomiast linią przerywaną – puste segmenty (o zerowym rozmiarze pola danych). Wytłuszczoną czcionką oznaczono segmenty służące do zestawienia oraz zwolnienia połączenia.

Klient wysyła żądanie pod adres 153.19.120.250, do procesu nasłuchującego na porcie 23. Serwer wykonuje żądanie, a odpowiedź odsyła na adres 172.16.0.30, na port 3974. Proces ustanowienia połączenia przebiega trzyetapowo (ang. *three-way handshake*):

1. Inicjalizacja połączenia przez klienta – klient wysyła do serwera segment z prośbą o nawiązanie połączenia (flaga SYN). Informuje także, że jego po-

czątkowy numer sekwencyjny, od którego rozpocznie numerowanie przesyłanych danych, wynosi zero[31] (Seq=0). Ponadto ogłasza wartość okna – 16384 bajtów, oraz maksymalny rozmiar segmentu – 1460 bajtów.

2. Akceptacja serwera (jeżeli zgadza się na połączenie) – serwer wysyła segment ze zgodą na nawiązanie połączenia (flaga SYN). Ustala swój numer sekwencyjny na zero[32] (Seq=0) oraz podobnie jak klient informuje o wartości okna i maksymalnym rozmiarze segmentu. Przy okazji serwer potwierdza (flaga ACK), że odebrał segment o numerze 0 i oczekuje na segment o numerze jeden (Ack=1).
3. Potwierdzenie klienta – klient wysyła segment o numerze jeden[33] (Seq=1) z potwierdzeniem odebrania zgody (flaga ACK). Informuje także serwer, że oczekuje na segment o numerze jeden (Ack=1). W tym momencie połączenie jest nawiązane.

Segmenty od czwartego do dziesiątego przenoszą dane warstwy aplikacji. Warto zwrócić uwagę, że segmenty siódmy i ósmy transmitowane są równolegle w przeciwnych kierunkach, co jest możliwe, ponieważ połączenie jest typu full-duplex.

Ostatnie cztery segmenty reprezentują proces zwalniania połączenia. Strona chcąca zakończyć połączenie (w naszym przypadku serwer) wysyła segment informujący, że nie ma już więcej danych do wysłania (flaga FIN). Strona przeciwna potwierdza odebranie informacji i połączenie w jednym kierunku zostaje zamknięte. Strona, która nie zgłosiła chęci zamknięcia połączenia może nadal wysyłać segmenty lub, jeśli jest gotowa, również poinformować o chęci zakończenia połączenia. Potwierdzenie tej chęci przez drugą stronę oznacza rozłączenie.

5.9. Charakterystyka, budowa i zastosowanie protokołu UDP

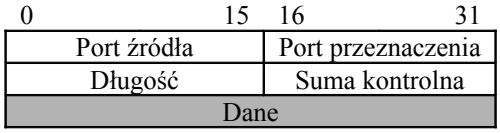
W komunikacji sieciowej występują sytuacje, w których wykorzystanie zaawansowanych i kosztownych[34] mechanizmów TCP jest nieuzasadnione lub niemożliwe. Niemożliwe – jeżeli komunikacja ma się odbywać w trybie rozsiewczym (ang. broadcast) lub grupowym (ang. multicast). Nieuzasadnione – jeżeli chwilowa zawodność w dostarczaniu danych nie wpłynie negatywnie na funkcjonowanie aplikacji. Przykładowo przy rozsyłaniu uaktualnień routingu nie jest wymagana gwarancja niezawodności. Jeżeli uaktualnienie nie zostanie

[31] Numer bezwzględny wynosił 3647252804, ale dla uproszczenia analizy wykorzystano numery względne.
[32] Również jest to numer względny, bezwzględny wynosił 2533126467.
[33] W normalnej sytuacji, gdy segment nie zawiera żadnych danych z warstwy aplikacji, jego następnik ma ten sam numer sekwencyjny. Segmenty z fazy nawiązywania połączenia są wyjątkiem – każdy traktowany jest tak, jakby przenosił jeden bajt danych w polu *dane*.
[34] W sensie narzutów związanych z przesyłaniem informacji sterujących.

odebrane poprawnie, routery będą funkcjonować dalej. Mogą wprawdzie mieć chwilowo nieaktualną tablicę tras, ale tylko do czasu kolejnego rozgłoszenia[35]. Innym przykładem aplikacji, które tolerują sporadyczną utratę segmentów, są multimedia czasu rzeczywistego. Strumień danych, taki jak głos, czy obraz, musi być odtwarzany natychmiast, choćby miał być zniekształcony. Stąd retransmitowane segmenty, ze względu na wprowadzone opóźnienia, byłyby bezużyteczne.

Istnieją także zastosowania, w których mechanizm retransmisji implementowany jest na poziomie aplikacji. Przykładowo brak odpowiedzi na pytanie do serwera DNS o odwzorowanie nazwy domenowej, można interpretować jako utratę segmentu i wysłać segment ponownie. W takiej sytuacji efektywniejsze jest ewentualne powtórzenie segmentu z zapytaniem, niż angażowanie mechanizmów związanych z nawiązaniem połączenia i zarządzaniem sesją.

Do realizacji zadań, które nie wymagają zaawansowanych usług TCP, zaprojektowano protokół UDP. Protokół ten oferuje warstwie aplikacji interfejs pakietowy. UDP jest protokołem bezpołączeniowym, dzięki czemu ma niewielki nagłówek (rysunek 9), który może być szybko przetwarzany.



Rysunek 9. Struktura segmentu UDP [768]

Znaczenie pól *port źródła*, *port przeznaczenia* oraz *dane* jest takie samo jak w przypadku protokołu TCP. Pole *długość* zawiera rozmiar całego segmentu (nagłówek + dane). Suma kontrolna obliczana jest na podstawie całego segmentu oraz dodatkowo uwzględnia następujące pola z nagłówka IP: *adres źródła*, *adres przeznaczenia*, *protokół*. Binarne wypełnienie jedynkami pola *suma kontrolna* oznacza, że dla przesyłanych danych wykrycie błędu nie ma większego znaczenia – np. w przypadku transmisji mowy w czasie rzeczywistym.

Prostota nagłówka UDP uniemożliwia sterowanie przepływem, odtwarzanie kolejności segmentów, korzystanie z potwierdzeń oraz gwarancję niezawodności. Realizacja tych zadań w razie potrzeby musi być wykonana przez warstwę aplikacji.

5.10. Model komunikacji klient–serwer

Komunikacja sieciowa opiera się na modelu interakcji klient–serwer. Klient i serwer komunikują się używając rozumianych przez obie strony poleceń – protokołu warstwy aplikacji. Do przesyłania tych poleceń wykorzystywana jest warstwa transportowa. **Klient** to program inicjujący połączenie i żądający od

[35] Przykładowo protokół RIP domyślnie wysyła uaktualnienia co 30 sekund.

serwera usługi. **Serwer**[36] to program będący dostawcą usług. Przykładowo, serwer WWW będzie udostępniał strony internetowe, serwer SMTP będzie umożliwiał wysyłanie e-maili, serwer DNS będzie odwzorowywał nazwy domenowe na adresy IP, itd. Cechą wspólną wyżej wymienionych serwerów jest pasywne oczekiwanie na żądanie, wykonanie żądania i wysyłanie odpowiedzi.

Klient używa adresu IP do określenia komputera z uruchomionym serwerem. Jednak na jednym komputerze może współbieżnie pracować wiele serwerów, każdy oferujący inną usługę. Aby klient mógł wskazać, z którym serwerem chce się połączyć, warstwa transportowa przypisuje każdej aktywnej usłudze unikalny numer portu. Numer ten wiązany jest z procesem serwera w trakcie jego uruchamiania. Mówi się wówczas, że serwer nasłuchuje zgłoszeń na zadanym porcie. Klient wskazuje, do którego serwera ma zostać dostarczony ładunek przenoszony w segmencie, poprzez numer zapisany w polu *port przeznaczenia*. Ponieważ po stronie klienta może być uruchomionych klika procesów komunikujących się z różnymi serwerami, istnieje potrzeba przypisania również każdemu z nich unikalnego numeru. Numer ten zwykle generowany jest automatycznie przez system operacyjny w momencie uruchamiania procesu klienta. Następnie klient ogłasza ten numer do serwera za pomocą pola *port źródła* wysyłanego segmentu. Po odebraniu segmentu, serwer odczytuje numer i wykorzystuje go jako port przeznaczenia przy wysyłaniu odpowiedzi. Numery portów zostały pogrupowane następująco:

– od 0 do 1023 – dobrze znane porty (ang. *well known ports*);
– od 1024 do 49151 – porty zarejestrowane;
– od 49152 do 65535 – porty prywatne.

Dobrze znane porty zdefiniowane są przez IANA[37] jako domyślne dla standardowych usług sieciowych. Przykładowe usługi standardowe oraz porty z nimi związane to:

– FTP-data – 20;
– FTP-control – 21;
– SSH – 22;
– TELNET – 23;
– SMTP – 25;
– DNS – 53;
– TFTP – 69;
– WWW – 80;
– POP3 – 110.

[36] Należy zwrócić uwagę, że w niniejszym rozdziale terminu serwer używa się wyłącznie w odniesieniu do programu. Natomiast potocznie terminem serwer określa się komputera o wysokich osiągach przeznaczony do świadczenia usług sieciowych.
[37] http://www.iana.org/assignments/port-numbers

Porty zarejestrowane zarezerwowane są przez organizacje dostarczające aplikacje sieciowe na rynek oprogramowania. Pisząc własny serwer należy wybrać numer z zakresu portów prywatnych.

6. Zadania z rozwiązaniami

Przykład 1. *Określić klasę adresu 153.19.121.30.*

Rozpisując 153 binarnie, otrzymujemy 10011001, co pasuje do wzorca dla klasy B, czyli 10XXXXXX (X oznacza dowolną cyfrę binarną). Znając zakresy przedziałów dla poszczególnych klas, można od razu stwierdzić, że adres 153.19.121.30 należy do klasy B, ponieważ wartość pierwszego bajta mieści się w przedziale [128,191].

Przykład 2. *Obliczyć ile jest sieci w klasie C i ile hostów można zaadresować w każdej z nich.*

Identyfikator dowolnej sieci z klasy C zapisany jest na 24 bitach, z których pierwsze trzy muszą mieć wartość 110. Pozostaje zatem 21 bitów do adresowania sieci. Wynika z tego, że takich sieci może być 2^{21}. Z kolei do numerowania hostów przeznaczonych jest 8 bitów (ostatni bajt). Należy jednak uwzględnić, że numer składający się binarnie z samych zer lub z samych jedynek nie może być wykorzystany do adresowania hostów. Wynika z tego, że w każdej sieci klasy C, można zaadresować 2^8–2 hosty.

Przykład 3. *Zakładając adresowanie klasowe wyznaczyć identyfikator sieci dla adresu 153.19.121.30.*

Rozpisując adres 153.19.121.30 oraz jego domyślną maskę binarnie, a następnie wykonując operację bitowego AND, otrzymujemy:

```
          10011001.00010011.01111001.00011110
AND       11111111.11111111.00000000.00000000
          10011001.00010011.00000000.00000000
```

Zatem adres sieci to 153.19.0.0.

Przykład 4. *Wyznaczyć adres sieci z adresu 153.19.121.30/20 (adresowanie bezklasowe).*

Rozpisując adres 153.19.121.30 oraz jego maskę binarnie, a następnie wykonując operację bitowego AND, otrzymujemy:

```
          10011001.00010011.01111001.00011110
AND       11111111.11111111.11110000.00000000
          10011001.00010011.01110000.00000000
```

Zatem adres sieci to 153.19.112.0.

Przykład 5. *Ile minimalnie bitów należy pozostawić w części „numer hosta”, tworząc podsieci do adresowania łączy punkt-punkt.*

Ponieważ w podsieci służącej do adresowania łącza punkt–punkt potrzebne będą tylko dwa adresy, wystarczy pozostawić 2 bity w części „numer hosta". Na 2 bitach można utworzyć 4 adresy, z czego dwa będą użyteczne do adresowania hostów, jeden będzie oznaczał rozgłoszenie, a jeden – adres podsieci.

Przykład 6. *W ramach sieci 153.19.0.0 wydzielić 6 podsieci o maksymalnej przestrzeni adresowej oraz rozpisać adresowanie w każdej z nich. Podać maskę po wydzieleniu podsieci oraz liczbę możliwych do zaadresowania hostów w każdej podsieci.*

Do utworzenia 6 podsieci o maksymalnej przestrzeni adresowej należy „pożyczyć" 3 bity[38]. Ponieważ identyfikator sieci oraz numer podsieci zajmują łącznie 19 bitów maska wygląda następująco:

11111111.11111111.111|00000.00000000

lub w systemie dziesiętnym–kropkowym 255.255.224.0. Do numerowania hostów w każdej podsieci pozostaje 13 bitów. Zatem możliwa liczba hostów w każdej podsieci wynosi $2^{13}–2 = 8190$.

Poniżej rozpisano kolejne etapy adresowania dla podsieci numer 5.

a) wyznaczenie adresu podsieci numer 5:

– ponieważ na dwóch pierwszych bajtach zapisany jest adres sieci pozostanie on niezmienny podczas tworzenia podsieci – 153.19._._
– zapisujemy liczbę 5 w systemie binarnym – 101,
– umieszczamy numer podsieci (101) na trzech bardziej znaczących bitach trzeciego bajtu - 153.19.[101|□□□□□].[□□□□□□□□],
– w celu uzyskania adresu podsieci zerujemy bity w tej części adresu, która przechowuje numer hosta: 153.19.[101|00000].[00000000] ≡ 153.19.160.0

b) wyznaczenie zakresu możliwych adresów dla hostów:

– bierzemy adres podsieci i w części hosta wpisujemy numer hosta,
– dla pierwszego hosta otrzymujemy: 153.19.[101|00000].[00000001] ≡ 153.19.160.1,
– dla drugiego hosta otrzymujemy: 153.19.[101|00000].[00000010] ≡ 153.19.160.2, itd.,
– dla ostatniego hosta otrzymujemy: 153.19.[101|11111].[11111110] ≡ 153.19.191.254

c) wyznaczenie adresu rozgłoszeniowego w podsieci:

– bierzemy adres sieci i w części hosta wpisujemy same jedynki: 153.19.[101|11111].[11111111] ≡ 153.19.191.255

Wykonując analogiczne operacje dla pozostałych podsieci otrzymujemy wyniki jak w tablicy 7.

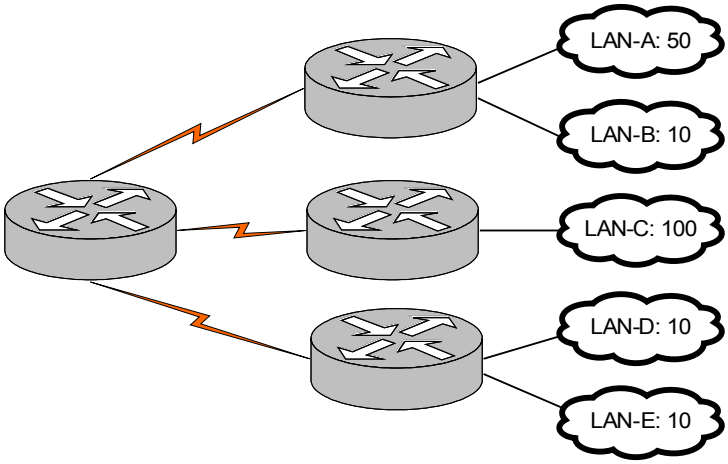
[38] Na n bitach można utworzyć 2^n różnych podsieci.

Tablica 7. Adresowanie w podsieciach dla adresu 153.19.0.0/19

numer podsieci		adres podsieci		zakres adresów hostów
0	[000]	153.19.[000|00000].0	153.19.0.0	153.19.0.1 – 153.19.31.254
1	[001]	153.19.[001|00000].0	153.19.32.0	153.19.32.1 – 153.19.63.254
2	[010]	153.19.[010|00000].0	153.19.64.0	153.19.64.1 – 153.19.95.254
3	[011]	153.19.[011|00000].0	153.19.96.0	153.19.96.1 – 153.19.127.254
4	[100]	153.19.[100|00000].0	153.19.128.0	153.19.128.1 – 153.19.159.254
5	[101]	153.19.[101|00000].0	153.19.160.0	153.19.160.1 – 153.19.191.254
6	[110]	153.19.[110|00000].0	153.19.192.0	153.19.192.1 – 153.19.223.254
7	[111]	153.19.[111|00000].0	153.19.224.0	153.19.224.1 – 153.19.255.254

uwaga: starsze routery mogą nieobsługiwać podsieci zerowej i ostatniej

Przykład 7. *Dysponując adresem sieci 195.10.52.0, utworzyć podsieci w ten sposób, aby zaadresować cały intranet o topologii przedstawionej na rysunku 10. W chmurach reprezentujących sieci Ethernet podano przewidywaną liczbę hostów. Zastosować technikę VLSM.*



Rysunek 10. **Topologia intranetu**

Algorytm rozwiązujący zadanie polega na tworzeniu podsieci w kolejności malejącej przestrzeni adresowej. Jeżeli w pewnym momencie okaże się, że kolejnej podsieci nie można utworzyć w taki sposób, aby zawierała wymaganą liczbę adresów, zadanie jest niewykonalne. Rozwiązując zadanie należy pamiętać, że w każdej sieci Ethernet trzeba uwzględnić dodatkowy adres na interfejs routera oraz potrzebne będą 3 podsieci na adresowanie interfejsów szeregowych między routerami.

Przestrzeń adresowa LAN-C wymaga 7 bitów w części hosta. Należy wydzielić zatem dwie podsieci:

– 195.10.52.[0|0000000] ≡ 195.10.52.0/25;
– 195.10.52.[1|0000000] ≡ 195.10.128.0/25.

Pierwsza z tych podsieci zostanie zarezerwowana na LAN-C, a druga zostanie przeznaczona do dalszego podziału. W kolejnym kroku potrzebna jest podsieć o 51 adresach, czyli należy zostawić 6 bitów w części hosta. W tym celu „pożyczony” zostanie kolejny bit z adresu 195.10.128.0/25, w wyniku czego powstaną dwie podsieci:

– 195.10.52.[10|000000].0 ≡ 195.10.128.0/26;
– 195.10.52.[11|000000].0 ≡ 195.10.192.0/26.

Podobnie jak poprzednio pierwszą podsieć należy zarezerwować – tym razem na LAN-A, a drugą przeznaczyć do dalszego podziału. W kolejnej iteracji należy utworzyć 3 podsieci po 11 adresów każda. Zatem w części hosta należy pozostawić 4 bity, a na adresowanie podsieci przeznaczyć kolejne 2 bity:

– 195.10.52.[1100|0000].0 ≡ 195.10.192.0/28;
– 195.10.52.[1101|0000].0 ≡ 195.10.208.0/28;
– 195.10.52.[1110|0000].0 ≡ 195.10.224.0/28;
– 195.10.52.[1111|0000].0 ≡ 195.10.240.0/28.

Pierwsze trzy podsieci rezerwujemy, odpowiednio, na LAN-B, LAN-D i LAN-E, a ostatnią przeznaczamy do dalszego podziału. Do zaadresowania pozostały już tylko połączenia punkt–punkt między routerami. Mając na uwadze, że w części hosta należy pozostawić co najmniej 2 bity oraz potrzebne są 3 podsieci, pożyczamy kolejne 2 bity:

– 195.10.52.[111100|00].0 ≡ 195.10.240.0/30;
– 195.10.52.[111101|00].0 ≡ 195.10.244.0/30;
– 195.10.52.[111110|00].0 ≡ 195.10.248.0/30;
– 195.10.52.[111111|00].0 ≡ 195.10.252.0/30.

Pierwsze trzy podsieci wykorzystujemy do adresowania łączy szeregowych między routerami, a ostatnia pozostaje niewykorzystana. Zbiorcze zestawienie podsieci zawiera tablica 8.

Tablica 8. Zestawienie podsieci

<table>
<tr><td>195.10.52.0/25
(126 adresów)</td><td colspan="4">LAN-C</td></tr>
<tr><td rowspan="8">195.10.52.128/25
(126 adresów)</td><td>195.10.52.128/26
(62 adresy)</td><td colspan="3">LAN-A</td></tr>
<tr><td rowspan="7">195.10.52.192/26
(62 adresy)</td><td>195.10.52.192/28
(14 adresów)</td><td colspan="2">LAN-B</td></tr>
<tr><td>195.10.52.208/28
(14 adresów)</td><td colspan="2">LAN-D</td></tr>
<tr><td>195.10.52.224/28
(14 adresów)</td><td colspan="2">LAN-E</td></tr>
<tr><td rowspan="4">195.10.52.240/28
(14 adresów)</td><td>195.10.52.240/30
(2 adresy)</td><td>p2p</td></tr>
<tr><td>195.10.52.244/30
(2 adresy)</td><td>p2p</td></tr>
<tr><td>195.10.52.248/30
(2 adresy)</td><td>p2p</td></tr>
<tr><td>195.10.52.252/30
(2 adresy)</td><td>wolne</td></tr>
</table>

Literatura

1. Tanenbaum A.S. (2003), Computer Networks, Prentice Hall
2. Comer D.E. (2001), Sieci komputerowe i intersieci, WNT, Warszawa
3. RFC 791 (1981), Internet Protocol
4. RFC 1122 (1989), Requirements for Internet Hosts - Communication Layers
5. RFC 1812 (1995), Requirements for IP Version 4 Routers
6. RFC 1918 (1996), Address Allocation for Private Internets
7. RFC 1517 (1993), Applicability Statement for the Implementation of CIDR
8. RFC 1518 (1993), An Architecture for IP Address Allocation with CIDR
9. RFC 1519 (1993), CIDR: An Address Assignment and Aggregation Strategy
10. RFC 1520 (1993), Exchanging Routing Information Across Provider Boundaries in the CIDR Environment
11. RFC 792 (1981), Internet Control Message Protocol
12. RFC 768 (1980), User Datagram Protocol
13. RFC 793 (1981), Transmission Control Protocol
14. RFC 1122 (1989), Requirements for Internet Hosts - Communication Layers
15. RFC 1106 (1989), TCP Big Window and Nak Options
16. RFC 1323 (1992), TCP Extensions for High Performance
17. RFC 1072 (1988), TCP extensions for long-delay paths
18. RFC 2018 (1996), TCP Selective Acknowledgement Options
19. RFC 2883 (2000), An Extension to the Selective Acknowledgement (SACK) Option for TCP
20. RFC 3168 (2001), The Addition of Explicit Congestion Notification (ECN) to IP
21. RFC 3540 (2003), Robust Explicit Congestion Notification (ECN) Signaling with Nonces
22. RFC 2001 (1997), TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms
23. RFC 2581 (1999), TCP Congestion Control
24. RFC 2309 (1998), Recommendations on Queue Management and Congestion Avoidance in the Internet
25. RFC 2414 (1998), Increasing TCP's Initial Window
26. RFC 2988 (2000), Computing TCP's Retransmission Timer
27. RFC 879 (1983), The TCP Maximum Segment Size and Related Topics
28. RFC 1063 (1998), Path MTU Discovery
29. Armitage G. (2002), Quality of Service in IP Networks, New Riders, USA
30. Vegesna S. (2001), IP Quality of Service, Cisco Press, Indianapolis