UDP vs TCP

Autor: Marcin Koczara IV FDS

STRESZCZENIE

W sieciach komputerowych używa się wielu protokołów. W pewnym sensie prawie każde działanie w sieci jest wykonywane w oparciu o taki czy inny protokół. Niektóre z nich działają na niższym poziomie modelu sieciowego OSI, inne na wyższym, a jeszcze inne pomiędzy nimi.[3]
W projekcie tym zostaną przedstawione główne protokołu warstwy transportowej, ich budowa, zastosowanie i sposób komunikowania się dwóch użytkowników za pomocą tych protokołów.

SPIS TREŚCI

1.WSTĘP

Warstwa transportowa jest pierwszą warstwą licząc od warstwy fizycznej mającą nadzór nad całością połączenia między stacja źródłową i stacją docelową. Ma ona za zadanie zagwarantowanie niezawodnego i przeźroczystego przekazu danych między stacjami końcowymi. W zależności od typu podsieci komunikacyjnej i jakości oferowanych przez nie usług protokoły transportowe mogą sterować przepływem wykorzystując podobnie jak w warstwie łącza danych mechanizmy okienkowe oraz realizować wykrywanie błędów bądź straconych pakietów gwarantując tym samym integralność przesyłanych wiadomości.
Jej dwa najważniejsze protokoły to TCP (Transmission Control Protocol) i UDP (User Datagram Protocol). TCP zapewnia usługi niezawodnie dostarczające dane, z wykrywaniem na obu krańcach błędów transmisji i ich korekcją. UDP udostępnia usługi dostarczające datagramy z małym narzutem, metodą bezpołączeniową.[6]

Protokół komunikacyjny to zbiór zasad i norm, których muszą przestrzegać komunikujące się ze sobą obiekty. Ponieważ protokoły mogą być dość skomplikowane, nadaje im się strukturę warstwowa. Według modelu OSI (*ang. Open Systems Interconnection*) wyróżniamy siedem takich warstw. Na samym spodzie występuje warstwa fizyczna, następnie idąc w górę kolejno warstwy łącza danych, sieciowa, transportowa, sesji, prezentacji i warstwa aplikacji. W każdej z wyżej wymienionych warstw zdefiniowane są osobne protokoły.

Tryb połączeniowy (*ang. connection oriented*) polega na ustanowieniu logicznego połączenia pomiędzy dwoma komunikującymi się ze sobą procesami. Aby nawiązać komunikacje trzeba najpierw nawiązać połączenie. Z obsługi połączeniowej korzysta się wtedy, gdy powstaje potrzeba przesyłania wielu komunikatów w dwu kierunkach. Najlepszym przykładem takiego połączenia jest telnet, a protokół połączeniowy to TCP.

Tryb bezpołączeniowy (lub obsługa datagramowa) jest przeciwstawieniem do trybu połączeniowego. W tym przypadku komunikaty przekazywane są zupełnie niezależnie. Typowym przykładem trybu bezpołączeniowego jest usługa poczty elektronicznej, a protokół bezpołączeniowy to UDP.[7]

2. PROTOKÓŁ UDP

Protokół UDP zapewnia podstawowy mechanizm wykorzystywany przez programy użytkowe przy przesyłaniu datagramów do innych programów użytkowych, a do rozróżniania poszczególnych programów wykonywanych na pojedynczej maszynie zapewnia on porty protokołów. Zatem oprócz wysyłanych danych, każdy komunikat UDP zawiera numer portu odbiorcy i numer portu nadawcy, dzięki czemu oprogramowanie UDP odbiorcy może dostarczyć komunikat do właściwego adresata oraz umożliwia wysyłanie odpowiedzi.

Do przesyłania komunikatów między maszynami UDP używa podstawowego protokołu IP i ma tę samą niepewną, bezpołączeniową semantykę dostarczania datagramów. Nie używa potwierdzeń w celu upewnienia się o dotarciu komunikatów, nie porządkuje przychodzących komunikatów i nie zapewnia mechanizmu sprzężenia zwrotnego do kontroli szybkości przesyłania danych między maszynami. Z tego powodu komunikaty UDP mogą być gubione, duplikowane lub przychodzić w innej kolejności niż były wysłane, a także mogą przychodzić szybciej niż odbiorca może je przetworzyć.

Program użytkowy korzystający z UDP bierze na siebie pełną odpowiedzialność za rozwiązywanie problemów niezawodności, w szczególności za gubienie komunikatów, ich duplikowanie, opóźnienia, dostarczanie w niewłaściwej kolejności i utratę łączności z adresatem. Dlatego też wiele programów opartych na UDP dobrze pracuje w środowisku lokalnym, ale zawodzą one w większych intersieciach TCP/IP.[2]

2.1 Datagram UDP

Każdy komunikat UDP nazywa się datagramem użytkownika. Datagram taki można podzielić na następujące dwie części: nagłówek UDP i obszar danych UDP.
Nagłówek składa się z czterech 16-bitowych pól, które określają port, z którego wysłano komunikat, port odbiorcy, długość komunikatu i sumę kontrolną UDP.

Bity	
0	16 31
Port UDP nadawcy	Port UDP odbiorcy
Długość komunikatu UDP	Suma kontrolna UDP
DANE ...	

Rys 2.1 Format pól w nagłówku datagramu UDP[9]

Pola port nadawcy i port odbiorcy zawierają 16-bitowe numery portów UDP używane do odnajdywania procesu oczekującego na dany datagram. Pole port nadawcy jest opcjonalne. Gdy jest używane, zawiera numer portu, do którego należy wysłać odpowiedź, w przeciwnym razie powinno zawierać zero. Pole długość zawiera wartość odpowiadającą liczbie oktetów datagramu UDP, wliczając nagłówek i dane użytkownika. Minimalna wartość pola długość wynosi osiem, czyli długość samego nagłówka. Suma kontrolna UDP jest opcjonalna i nie musi być używana: wartość zero w polu suma kontrolna oznacza, że nie obliczono sumy kontrolnej. Jednakże sumy kontrolne UDP stanowią jedyną gwarancję, że dane nie zostały uszkodzone i można je wykorzystać.[2]

2.2 Pseudonagłówek UDP

Suma kontrolna UDP dotyczy większej ilości informacji niż sam datagram UDP. W celu wyliczenia tej sumy UDP dodaje na początku datagramu pseudonagłówek, na końcu oktet zer, aby dopełnić datagram do wielokrotności 16 bitów, i wylicza sumę kontrolną tak uzyskanego obiektu. Oktety dopełniające i pseudonagłówek nie są przesyłane wraz z datagramem UDP, nie są też uwzględniane w polu długość. W celu wyliczenia sumy kontrolnej oprogramowanie umieszcza najpierw zero w polu suma kontrolna, a następnie tworzy 16-bitową sumę uzupełnień do jedności dla całego obiektu, w tym pseudonagłówka, nagłówka UDP i danych użytkownika.

Pseudonagłówek jest umieszczony w celu zapewnienia, że datagram dotarł do właściwego adresata. Kluczem do zrozumienia pseudonagłówka jest stwierdzenie, że właściwy odbiorca jest wyznaczony przez wskazanie konkretnej maszyny i konkretnego portu odbiorcy. Nagłówek UDP określa jedynie numer portu protokołu. Dlatego w celu zweryfikowania odbiorcy, UDP na maszynie nadającej datagram wylicza sumę kontrolną, która obejmuje zarówno adres IP, jak i datagram UDP. Oprogramowanie UDP odbiorcy sprawdza sumę kontrolną nadawcy, używając adresu IP nadawcy uzyskując z nagłówka datagramu IP, w

którym wędrował komunikat UDP. Jeśli sumy się zgadzają, to datagram dotarł zarówno do właściwego komputera docelowego, jak i portu na tym komputerze.

Pseudonagłówek używany przy wyliczaniu sumy kontrolnej UDP składa się z 12 oktetów danych. Pola oznaczone adres ip nadawcy i adres ip odbiorcy zawierają adresy IP nadawcy i odbiorcy, które zostaną użyte przy wysyłaniu komunikatu UDP. Pole proto zawiera typ kodu protokołu IP (17 dla UDP), a pole oznaczone długość UDP zawiera długość datagramu UDP (bez pseudonagłówka). W celu sprawdzenia sumy kontrolnej odbiorca musi wyznaczyć te pola na podstawie nagłówka IP, zbudować z nich pseudonagłówek i obliczyć sumę kontrolną.

2.3 Multipleksowanie i demultipleksowanie

Teoretycznie wszelkie multipleksowanie i demultipleksowanie między oprogramowaniem UDP i programami użytkowymi jest realizowane przez mechanizm portów. W praktyce każdy program użytkowy musi uzyskać od systemu operacyjnego port protokołu i odpowiadający mu numer portu, zanim będzie mógł wysłać datagram UDP. Po przydzieleniu portu każdy datagram, który jest wysyłany przez program użytkowy przez ten port, będzie miał odpowiedni numer portu w polu port nadawcy udp. Przy przetwarzaniu danych wejściowych UDP przyjmuje datagramy nadchodzące od oprogramowania IP i demultipleksuje je w zależności od portu UDP odbiorcy.

W większości implementacji, gdy program użytkowy żąda od systemu przydzielenia określonego portu, system tworzy wewnętrzną kolejkę nadchodzących komunikatów. Często program może podać lub zmienić rozmiar tej kolejki. UDP, otrzymując datagram, sprawdza, czy numer portu odbiorcy odpowiada któremuś z obecnie używanych portów. Jeśli nie, to wysyła komunikat ICMP „port niedostępny” i porzuca datagram. Jeśli któryś z portów pasuje, to UDP umieszcza nowy datagram w kolejce portu, skąd program użytkowy może go pobrać. Oczywiście w przypadku przepełnienia kolejki występuje błąd i UDP porzuca datagram.[2]

2.4 Porty UDP

Jeśli np. komputer A chce uzyskać plik od komputera B, to musi wiedzieć, jakiego numeru portu używa program przesyłania plików komputera B. Istnieją dwa różne podejścia do problemu przydzielania numerów portów. Jedno z nich opiera się na centralnym autorytecie. Każdy zgadza się na przydzielenie numerów portów, według potrzeb, przez władzę centralną, która publikuje listy przydzielonych numerów. Następnie wszelkie oprogramowanie używa numerów określonych na takiej liście. To podejście jest czasami nazywane powszechnym przyporządkowaniem, a przydział portów powszechnie znanym przydziałem portów.

Drugie podejście do zagadnienia przydzielania portów wykorzystuje wiązania dynamiczne. W tym przypadku numery portów nie są znane globalnie. Kiedy program potrzebuje portu, zostaje on przydzielony przez oprogramowanie sieciowe. Aby ustalić bieżący przydział portów na danym komputerze, trzeba wysłać zapytanie typu „którego portu używa usługa przesyłania plików?”. Maszyna docelowa odpowiada, wysyłając numer odpowiedniego portu.[2]

3. PROTOKÓŁ TCP

Aplikacje, które wymagają od protokołu transportowego niezawodnego dostarczania danych, używają protokołu TCP. Protokół ten sprawdza, czy dane dokładnie dotarły do miejsca przeznaczenia i czy zrobił to we właściwej kolejności. TCP zapewnia niezawodność za pomocą mechanizmu zwanego pozytywne potwierdzenie z retransmisją – wysyła on dane ponownie, dopóki nie otrzyma informacji, że dane zostały poprawnie odebrane. Jednostka danych wymieniana między współpracującymi modułami TCP nosi nazwę segmentu. Każdy segment posiada sumę kontrolną, używaną przez odbierającego do sprawdzenia, czy dane nie zostały przekłamane. Jeśli dane są poprawne, odbierający wysyła do nadawcy pozytywne potwierdzenie. Gdy odebrane dane są niepoprawne, zostają zignorowane. Po określonym czasie nadający moduł TCP retransmituje dane, na które nie otrzymał potwierdzenia.[5]

3.1 Format segmentów TCP

Mianem segmentu określa się jednostkową porcję danych przesyłanych między oprogramowaniem TCP na dwu różnych maszynach. Segmentów używa się również do ustanawiania połączenia, do przesyłania danych, do potwierdzenia, do wysyłania propozycji okien oraz do zamykania połączeń.

<table>
<tr><td></td><td colspan="8">bity</td><td></td></tr>
<tr><td>słowa</td><td>0</td><td>4</td><td>8</td><td>12</td><td>16</td><td>20</td><td>24</td><td>28</td><td>31</td><td></td></tr>
<tr><td>1</td><td colspan="4">Port nadawcy</td><td colspan="4">Port odbiorcy</td><td rowspan="7">Nagłówek</td></tr>
<tr><td>2</td><td colspan="8">Numer porządkowy</td></tr>
<tr><td>3</td><td colspan="8">Numer potwierdzenia</td></tr>
<tr><td>4</td><td>Dł. nagł.</td><td colspan="2">Zarezerwowane</td><td>Bity kodu</td><td colspan="4">Okno</td></tr>
<tr><td>5</td><td colspan="5">Suma kontrolna</td><td colspan="3">Wskaźnik pilnych danych</td></tr>
<tr><td>6</td><td colspan="6">Opcje</td><td colspan="2">Uzupełnienie</td></tr>
<tr><td>7</td><td colspan="8">DANE ...</td><td></td></tr>
</table>

Rys 3.1 Format segmentu TCP z nagłówkiem[9]

Każdy segment jest podzielony na dwie części: nagłówek i dane. Pola port nadawcy i port odbiorcy zawierają numery portów TCP, które identyfikują programy użytkowe na końcach połączenia. Pole numer porządkowy wyznacza pozycję danych segmentu w strumieniu bajtów nadawcy. Pole numer potwierdzenia wyznacza numer oktetu, który nadawca spodziewa się otrzymać w następnej kolejności. Pole długość nagłówka zawiera liczbę całkowitą, która określa długość nagłówka segmentu mierzoną w wielokrotnościach 32 bitów. Jest ono konieczne, gdyż pole opcje ma zmienną długość w zależności od tego, jakie opcje zostały obrane. 6-bitowe pole nazwane zarezerwowane jest pozostawione do wykorzystania w przyszłości. Oprogramowanie TCP używa 6-bitowego pola bity kodu do wyznaczania przeznaczenia zawartości segmentu:

- URG – wskaźnik pilności jest istotny
- ACK – pole potwierdzenia jest istotne
- PSH – ten segment stanowi prośbę o wypchnięcie
- RST – skasuj połączenie
- SYN – zsynchronizuj numery porządkowe
- FIN – koniec strumienia bajtów u nadawcy[2]

3.2 Pseudonagłówek TCP

Pole suma kontrolna w nagłówku TCP zawiera liczbę całkowitą służącą do sprawdzania, czy dane i nagłówek nie zostały naruszone, a oblicza się ja podobnie jak przy UDP. Do segmentu jest dodawany pseudonagłówek, odpowiednia liczba zerowych bitów na końcu, aby segment stanowił wielokrotność 16 bitów i dla tak uzyskanego wyniku jest obliczana 16-bitowa suma kontrolna. TCP nie wlicza do długości segmentu ani tego pseudonagłówka, ani uzupełnienia długości segmentu; te twory nie są też przesyłane. Przy obliczaniu sumy kontrolnej dodatkowo zakłada się, że samo jej pole jest zerowe. Podobnie jak w przypadku innych sum kontrolnych TCP używa arytmetyki 16-bitowej i wykorzystuje uzupełnienie do jedności sumy obliczanej przy uzupełnianiu do jednego. U odbiorcy odbywają się te same obliczenia, aby sprawdzić, czy segment przybył bez uszkodzeń.

Pseudonagłówek jest używany w dokładnie tym samym celu, co w UDP. Umożliwia on odbiorcy sprawdzenie, czy segment dotarł do swojego rzeczywistego odbiorcy, którego opisuje zarówno adres IP węzła, jak i numer portu protokołu. Dla TCP ważne są oba adresy: nadawcy i odbiorcy, gdyż TCP używa obydwu do identyfikacji połączenia. Stąd, gdy przybywa datagram niosący segment TCP, IP musi przekazać do TCP oprócz samego segmentu także znajdujące się w nim adresy nadawcy i odbiorcy.

Pseudonagłówek używany przy wyliczaniu sumy kontrolnej składa się z 12 oktetów danych.
TCP u nadawcy przypisuje polu protokół wartość, którą będzie używał bazowy system dostarczania w swoim polu typu protokołu. W przypadku datagramów IP przenoszących segment TCP ta wartość to 6. Pole długość TCP wyznacza całkowitą długość segmentu TCP, która obejmuje również nagłówek TCP. U odbiorcy informacje używane w pseudonagłówku są wydobywane z datagramu IP, który przyniósł segment.[2]

3.3 Porty TCP

TCP umożliwia wielu działającym na jednej maszynie programom użytkowym jednoczesne komunikowanie się oraz rozdziela między programy użytkowe przybywające pakiety TCP. Podobnie jak protokół UDP, TCP używa numerów portów protokołu do identyfikacji w ramach maszyny końcowego odbiorcy. Każdy z tych portów ma przypisaną małą liczbę całkowitą, która jest używana do jego identyfikacji. Porty TCP są bardziej złożone, gdyż dany numer portu nie odpowiada bezpośrednio pojedynczemu obiektowi. TCP działa, wykorzystując połączenia, w których obiektami są obwody wirtualne, a nie poszczególne porty. Podstawowym pojęciem TCP jest pojęcie połączenia, a nie portu protokołowego. Połączenia są identyfikowane przez parę punktów końcowych. TCP definiuje punkt końcowy jako parę liczb całkowitych (węzeł, port), gdzie węzeł oznacza adres IP węzła, a port jest portem TCP w tym węźle. Dwa połączenia mogą równocześnie na tej samej maszynie używać tego samego portu TCP, gdyż przyporządkowuje on przychodzące komunikaty połączeniom, a nie portom protokołów – używa ono obu punktów końcowych do zidentyfikowania odpowiedniego połączenia.[2]

3.4 Pasywne i aktywne otwarcie

W przeciwieństwie do UDP, TCP jest protokołem zorientowanym na połączenia i wymaga, aby obydwa punkty końcowe zgodziły się na współpracę. Zanim zostaną przesłane dane, programy użytkowe na obydwu końcach połączenia muszą porozumieć się. W tym celu jeden z programów użytkowych wykonuje funkcję pasywnego otwarcia, kontaktując się z systemem operacyjnym i zaznaczając, że akceptuje połączenie. W tym momencie system operacyjny przypisuje temu końcowi połączenia odpowiedni numer portu TCP. Program użytkowy na drugim końcu musi następnie używając funkcji aktywnego otwarcia, skontaktować się ze swoim systemem operacyjnym. Jeżeli ma zostać ustanowione połączenie, to te dwa modułu TCP muszą się skontaktować. Z chwilą utworzenia połączenia programy użytkowe mogą zacząć przekazywać dane. Moduły TCP na obu końcach wymieniają komunikaty, które gwarantują niezawodność dostarczania.[4]

3.5 Okna TCP

TCP widzi strumień jako ciąg oktetów lub bajtów, który dzieli na segmenty przeznaczone do transmisji. Zwykle każdy z takich segmentów podróżuje przez intersieć w pojedynczym datagramie IP. TCP, aby rozwiązać problem efektywnego przesyłania i kontrolowania przepływu używa specjalizowanego mechanizmu przesuwającego się okna. TCP umożliwia wysyłanie wielu segmentów zanim nadejdzie potwierdzenie, dzięki czemu zwiększa się przepływność, gdyż sieć przez cały czas pracuje. Wersja protokołu przesuwającego się oka w TCP rozwiązuje też problem kontroli przepływu między końcami, zezwalając odbiorcy na ograniczenie transmisji do momentu, kiedy jego bufory będą miały odpowiednią ilość wolnego miejsca.

Mechanizm przesuwających się okien działa w TCP na poziomie oktetów, a nie segmentów lub pakietów. Oktetom w strumieniu danych są nadawane numery porządkowe, a nadawca utrzymuje związane z każdym połączeniem trzy wskaźniki. Pierwszy wskaźnik wyznacza lewy skraj okna, oddzielający oktety, które już zostały wysłane i potwierdzone, od oktetów, które mają dopiero zostać potwierdzone. Drugi wskaźnik wyznacza prawy skraj okna i definiuje najdalszy oktet, który może zostać wysłany, zanim nadejdą potwierdzenia. Trzeci wskaźnik wyznacza granicę wewnętrzną okna, która oddziela oktety już wysłane od tych, które jeszcze nie zostały wysłane. Oprogramowanie protokołów wysyła bez żadnego oczekiwania wszystkie oktety z okna, więc granica wewnątrz okna porusza się szybko.

TCP umożliwia zmianę rozmiaru okna w czasie; każde potwierdzenie, które zawiera informacje o liczbie odebranych oktetów zawiera też propozycję rozmiaru okna, która wyznacza liczbę oktetów danych, które odbiorca jest gotów dodatkowo odebrać. W odpowiedzi na większą propozycję okna, nadawca zwiększa rozmiar swojego przesuwającego się okna i w dalszym ciągu wysyła oktety. W odpowiedzi na ogłoszenie zmniejszonego okna nadawca zmniejsza rozmiar swojego okna.

Zaletą używania okien o zmiennym rozmiarze jest taka, że umożliwia to zarówno kontrolę przepływu, jak i niezawodne przesyłanie. Jeżeli bufory odbiorcy zaczynają być wypełnione, to nie może on przyjmować większej liczby pakietów, wysyła więc propozycję mniejszego okna. W krańcowym przypadku odbiorca oferuje zerowe okno, co powoduje wstrzymanie przesyłania, aż do momentu gdy znajdzie się miejsce w buforze.[2]

3.6 Przekroczenie czasu i retransmisja

Jeden z najważniejszych i najbardziej złożonych pomysłów TCP jest zawarty w sposobie obsługi przekroczenia czasu i retransmisji. Jak inne niezawodne protokoły, TCP spodziewa się, że nadawca wysyła potwierdzenia zawsze, gdy przybywają nowe oktety strumienia danych. Za każdym razem gdy jest przesyłany segment, TCP uruchamia zegar i czeka na potwierdzenie. Gdy wyznaczony czas upłynie zanim zostanie uzyskane potwierdzenie dla danego z segmentu TCP retransmituje go.

TCP dostosowuje się do zmiennych czasów oczekiwania w intersieci, wykorzystując algorytm retransmisji z adaptacją. TCP bada wydolność wszystkich połączeń i określa pewną wartość czasów oczekiwania na potwierdzenie. W miarę jak następują zmiany w wydolności połączeń, TCP zmienia wartości tych czasów (tzn. adaptuje się do zmian).
Aby zebrać dane potrzebne algorytmowi z adaptacją, TCP zapisuje czas, w którym segment jest wysłany i czas, w którym przychodzi potwierdzenie dla danych z tego segmentu. Na podstawie tych dwóch wartości TCP oblicza czas, który nazywany jest jako próbka czasu podróży w obie strony albo jako próbka podróży w obie strony. Za każdym razem gdy TCP uzyskuje nową próbkę czasu poprawia wartość średniego czasu podróży przy danym połączeniu.[2]

3.7 Przeciążenia w sieci

Przeciążenie to sytuacja, w której powstały znaczące opóźnienia spowodowane natłokiem datagramów w co najmniej jednym punkcie wymiany pakietów. Gdy powstaje przeciążenie, opóźnienia wzrastają i np. ruter zaczyna kolejkować datagramy, aż do momentu, gdy będzie mógł wyznaczyć ich trasy. W najgorszym przypadku całkowita liczba datagramów przybywających do przeciążonego rutera rośnie, aż do zapełnienia jego pamięci, po czym następuje tracenie datagramów. W punktach końcowych zwykle nie są znane szczegóły tego gdzie i z jakiej przyczyny pojawiły się przeciążenia. Przeciążenie jest tam widziane jak zwiększone opóźnienie. Niestety większość protokołów transportowych używa czasów oczekiwania na retransmisję, na zwiększające się opóźnienia reagują one retransmitując datagramy. Retransmisje zaś powiększają przeciążenie zamiast je zmniejszać. Gdyby taka sytuacja była niekontrolowana, to zwiększony ruch, powodując zwiększone opóźnienia, prowadziłby do dalszego zwiększenia ruchu i tak dalej aż do momentu gdy sieć stałaby się bezużyteczna. Taka sytuacja znana jest jako zapaść z powodu przeciążenia.

W celu uniknięcia tej zapaści TCP musi zmniejszyć szybkość transmisji. Routery sprawdzają długość kolejek i używają np. metody tłumienia źródła w ICMP, która polega na poinformowaniu węzła, że pojawiło się przeciążenie, jednak protokoły transportowe mogą automatycznie pomagać w unikaniu przeciążenia przy zwiększeniu opóźnienia, zmniejszając szybkość transmisji. W celu uniknięcia przeciążenia standard TCP zaleca obecnie używanie dwóch metod: metody powolnego startu i metody wielokrotnego zmniejszania. Są one ze sobą powiązane i łatwe w implementacji. W celu kontrolowania przeciążenia TCP utrzymuje kolejne ograniczenie zwane ograniczeniem na okno przeciążeniowe lub oknem przeciążeniowym.

W stanie ustabilizowanym okno przeciążeniowe jest tej samej wielkości, co okno odbiorcy. Zmniejszanie okna przeciążeniowego zmniejsza ruch, ruch który TCP wprowadza do połączenia. W celu określenia rozmiaru okna przeciążeniowego TCP zakłada, że większość datagramów, które są gubione, ginie w związku z przeciążeniem. TCP redukuje okno

przeciążeniowe o połowę przy każdej utracie datagramu, przy powtarzających się stratach okno zmniejsza się wykładniczo. Jeżeli pojawia się podejrzenie, że powstało przeciążenie, TCP redukuje wykładniczo natężenie ruchu i częstość retransmisji. Jeżeli tracenie danych trwa, to TCP ogranicza transmisję do jednego datagramu i w dalszym ciągu podwaja czas opóźnienia przed retransmisją.

Do zwiększenia transmisji TCP wykorzystuje metodę zwaną powolnym startem. Uwalnia on intersieć od zalania dodatkowym ruchem tuż po zlikwidowaniu przeciążenia oraz przy nagłym pojawieniem się połączeń. TCP ustala okno przeciążeniowe na 1, wysyła początkowy segment i czeka. Gdy nadchodzi potwierdzenie, zwiększa wielkość okna na 2, wysyła 2 segmenty i czeka. Gdy przychodzą potwierdzenia dla tych 2 segmentów, każde z nich powoduje zwiększenie okna przeciążeniowego o 1, TCP może więc wysłać 4 segmenty. Potwierdzenia tych segmentów spowoduje zwiększenie okna przeciążeniowego do 8. Po czterech podróżach w obie strony TCP może wysłać 16 segmentów.

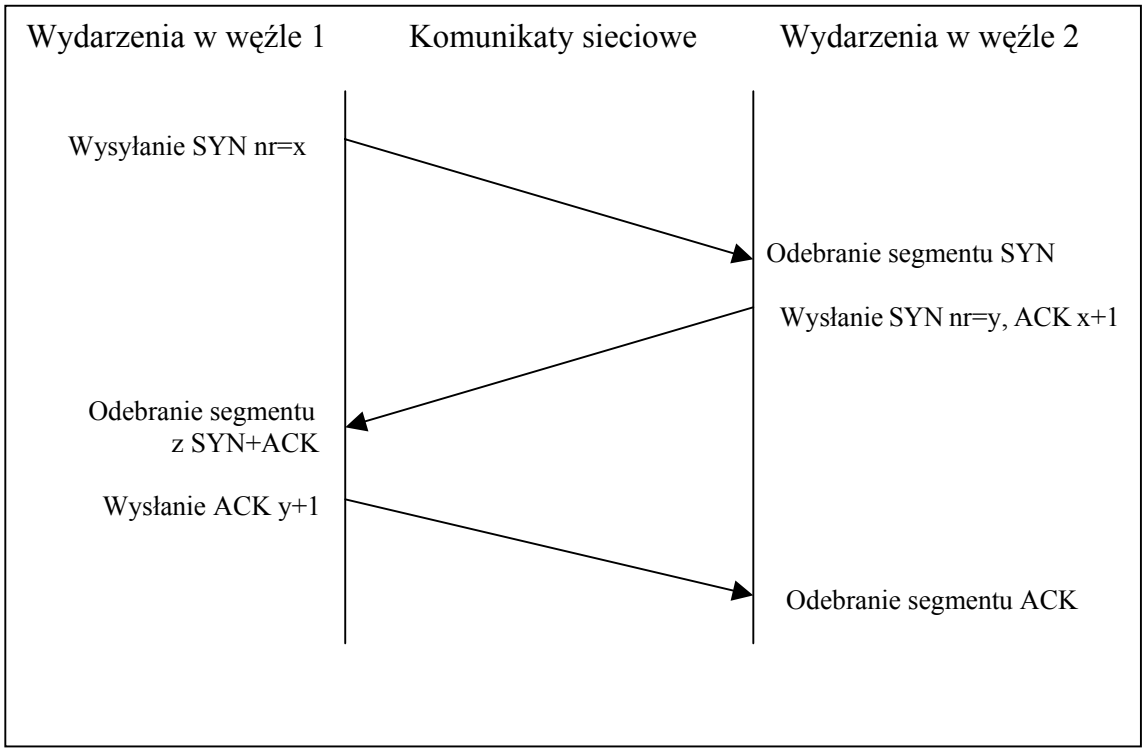
W celu uniknięcia zbyt szybkiego wzrostu rozmiaru okna i wywołania dodatkowego przeciążenia TCP dodaje jeszcze jedno ograniczenie. Kiedy okno przeciążeniowe osiąga połowę swojego pierwotnego rozmiaru sprzed przeciążenia, TCP wchodzi w stan unikania przeciążenia i zmniejsza szybkość zwiększenia okna. W stanie unikania przeciążenia okno zwiększane jest o 1 tylko wtedy, gdy wszystkie segmenty w oknie zostały potwierdzone.[2]

3.8 Otwieranie połączenia TCP

W celu ustanowienia połączenia TCP musi przesłać trzy komunikaty. Pierwszy segment w tej wymianie może zostać zidentyfikowany, gdyż zawiera on w polu kodu ustanowiony bit SYN. Drugi komunikat ma ustawiony bit SYN i ACK, co wskazuje on, że potwierdza pierwszy segment SYN oraz że jest to dalszy ciąg wymiany komunikatów. Końcowy krok to tylko potwierdzenie. Jest on używany jedynie do powiadomienia odbiorcy, że obydwie strony wiedzą, że połączenie zostało ustanowione.

Zwykle oprogramowanie TCP na jednej maszynie czeka biernie na dalszy ciąg wymiany, a oprogramowanie TCP na innej ją inicjuje. Wymiana ta jest tak opracowana, aby działała nawet wtedy, gdy obydwie maszyny próbują naraz zainicjować połączenie. Wobec tego połączenie może zostać ustanowione z dowolnego końca albo z obydwu równocześnie. Z chwilą ustanowienia połączenia dane mogą równie dobrze przepływać w obydwu kierunkach.

Ta trójetapowa wymiana jest zarówno konieczna, jak i wystarczająca do uzyskania poprawnej synchronizacji między obydwoma końcami połączenia. TCP jest zbudowane na bazie usługi niepewnego dostarczania pakietów. Komunikaty mogą więc być gubione, przychodzić z opóźnieniem, być duplikowane oraz dostarczane nie w kolejności. Wobec tego protokół ten musi używać mechanizmu przekraczania czasów oraz retransmitować zagubione komunikaty. Kłopoty się pojawiają, gdy podczas ustanawiania połączenia przybywa pierwotna prośba i prośba retransmitowana albo jeżeli retransmisje prośby zostają opóźnione i rozpoczynają się, gdy połączenie po ustanowieniu i wykorzystaniu zostaje zamknięte. Trzykrotna wymiana rozwiązuje te problemy.[1]

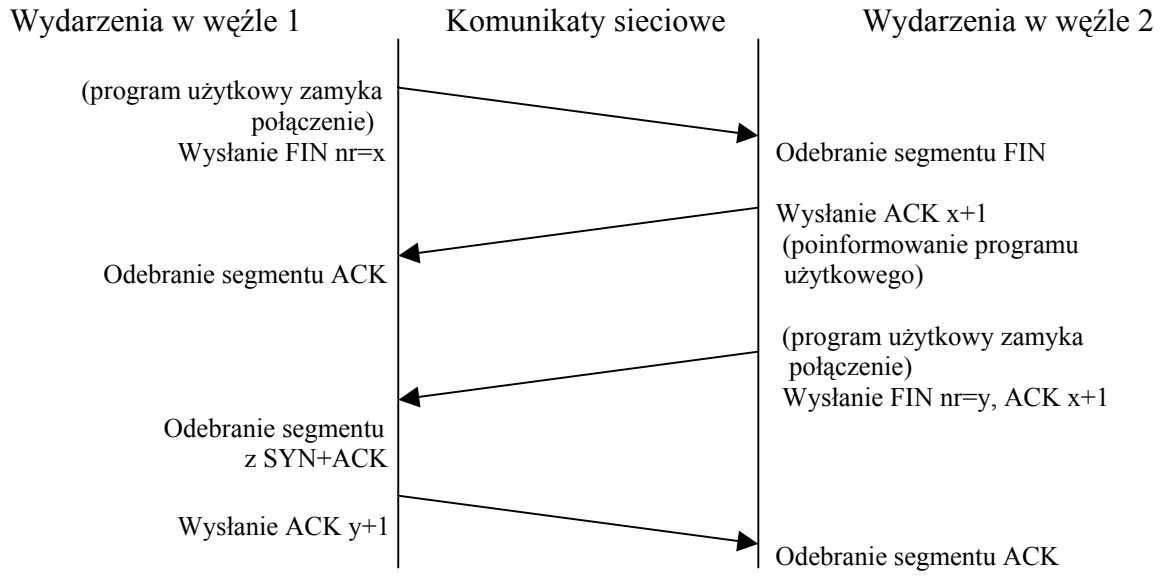


Rys 3.2 Trójetapowa wymiana komunikatów przy ustanawianiu połączenia[2]

3.9 Zamykanie połączenia TCP

Dwa programy używające TCP do komunikowania się mogą za pomocą operacji łagodnego zamykania zakończyć swoją komunikację. Wewnętrznie TCP przy zamykaniu połączenia używa zmodyfikowanej trójstopniowej wymiany komunikatów. Gdy program użytkowy mówi TCP, że nie ma już więcej danych do wysłania, TCP zamyka połączenie w jednym kierunku. W celu zamknięcia swojej połowy połączenia nadające TCP kończy przesyłać pozostające jeszcze dane i czeka, aż odbiorca je potwierdzi, a następnie wysyła segment z ustawionym bitem FIN. TCP odbiorcy potwierdza segment FIN i informuje program użytkownika na swoim końcu, że więcej danych już nie będzie.

Z chwilą zamknięcia połączenia w danym kierunku TCP odmawia przyjmowania kolejnych danych stamtąd napływających. W tym czasie dane mogą dalej płynąć w odwrotną stronę, dopóki odpowiedni nadawca jej nie zamknie. Nawet po zamknięciu przeciwnej strony połączenia potwierdzenia nadal będą napływały do nadawcy. Gdy obydwa kierunki ulegają zamknięciu, oprogramowanie TCP na każdym z końców kasuje zapisy dotyczące tego połączenia.



Rys 3.3 Zmodyfikowana trójstopniowa wymiana komunikatów używana do zamykania połączeń[2]

Różnica między trójstopniową wymianą używaną do ustanowienia połączenia a tą wykorzystywaną do zamykania pojawia się wtedy, kiedy maszyna odbiera pierwszy segment FIN. Zamiast natychmiast generować drugi segment FIN oprogramowanie TCP wysyła potwierdzenie, a następnie informuje program użytkowy o propozycji zakończenia. Informowanie o tym fakcie programu użytkowego oraz otrzymywanie odpowiedzi może zająć znacząco dużo czasu. Potwierdzenie służy do zabezpieczenia przed retransmisją pierwotnego segmentu FIN podczas tego oczekiwania. Gdy program użytkowy poinstruuje TCP, żeby zamknęło połączenie, TCP wysyła drugi segment FIN, po czym pierwotny węzeł odpowiada trzecim komunikatem ACK.[2]

3.10 Własności usługi niezawodnego dostarczania

TCP organizuje dwukierunkową współpracę między warstwą IP, a warstwami wyższymi, uwzględniając przy tym wszystkie aspekty priorytetów i bezpieczeństwa. Musi prawidłowo obsłużyć niespodziewane zakończenie aplikacji, do której właśnie wędruje datagram, musi również bezpiecznie izolować warstwy wyższe – w szczególności aplikacje użytkownika – od skutków awarii w warstwie protokołu IP. Scentralizowanie wszystkich tych aspektów w jednej warstwie umożliwia znaczną oszczędność nakładów na projektowanie oprogramowania.

Interfejs między programami użytkowymi a usługami niezawodnego dostarczania TCP można scharakteryzować za pomocą następujących własności:

- Przesyłanie strumieniami – gdy dwa programy użytkowe (procesy użytkownika) przesyłają duże ilości danych, myślimy o tych danych jako o strumieniu bitów, który jest podzielony na 8-bitowe oktety nieformalnie nazywane bajtami. Usługa

dostarczania strumieniami na maszynie docelowej to przekazanie odbiorcy dokładnie tego samego ciągu oktetów, który na swojej maszynie przekazał nadawca.

- Łączenie w obwód wirtualny – przed rozpoczęciem przesyłania, programy użytkowe zarówno nadawcy, jak i odbiorcy muszą się porozumieć z własnymi systemami operacyjnymi, informując je o potrzebie przesyłania za pomocą strumienia. Moduły oprogramowania protokołów w obu systemach porozumiewają się, wysyłając przez intersieć odpowiednie komunikaty sprawdzające czy transfer został autoryzowany oraz czy obydwie strony znajdują się w stanie gotowości. Po ustaleniu tych szczegółów moduły protokołów informują programy użytkowe, że połączenie zostało ustanowione i że można rozpocząć przesyłanie. W trakcie komunikacji oprogramowanie protokołów w dalszym ciągu wymienia informacje, które potwierdzają poprawność otrzymywanych danych. Jakiekolwiek zakłócenie komunikacji (np. z powodu awarii osprzętu sieciowego na ścieżce pomiędzy nadawcą a odbiorcą) zostaje wykryte przez obie maszyny i powoduje poinformowanie odpowiednich programów użytkowych. Tego typu połączenie określa się mianem obwodu wirtualnego, bo chociaż programy użytkowe widzą to połączenie tak, jakby istniał tutaj specjalny obwód elektroniczny, to niezawodność jest złudzeniem wywołanym przez usługę dostarczania strumieniami.
- Przesyłanie z użyciem buforów – programy użytkowe przesyłają strumienie danych obwodami wirtualnymi dzięki przekazywaniu kolejnych oktetów danych do oprogramowania protokołów. Program użytkowy przesyła dane w porcjach, które uważa za wygodne, a które mogą być nawet wielkości jednego oktetu. U odbiorcy oprogramowanie protokołów dostarcza oktety ze strumienia danych w takim samym porządku, w jakim zostały wysłane, udostępniając je odbierającemu programowi użytkowemu, po otrzymaniu i sprawdzeniu. Oprogramowanie protokołów ma swobodę przy dzieleniu strumienia na pakiety i może to robić niezależnie od tego, jak strumień jest dzielony przez program użytkowy. W celu zwiększenia efektywności przesyłania i zminimalizowania ruchu w sieci przyjmuje się zwykle strategię czekania, aż uzbiera się tyle danych ze strumienia, żeby wypełnić datagram o rozsądnej wielkości zanim się go wyśle do intersieci. Dzięki temu, nawet jeśli program użytkowy generuje strumień po jednym bajcie naraz, przesyłanie danych przez intersieć może być dosyć efektywne. Podobnie jeśli program użytkowy wygeneruje bardzo duży blok danych, to oprogramowanie protokołów może podzielić go do transmisji na mniejsze porcje. Dla tych programów użytkowych, w których dane powinny zostać dostarczone, nawet jeżeli bufor nie został wypełniony, udostępnia się mechanizm "wypychania" (ang. *Push*), którego programy użytkowe używają do wymuszania przesyłania. Po stronie nadawcy operacja wypchnięcia wymusza na oprogramowaniu protokołów przesłanie wszystkich tych danych, które zostały dotąd wygenerowane, bez czekania na wypełnienie bufora. Po stronie odbiorcy zaś powoduje, że TCP udostępnia dane programowi użytkownika beż opóźnienia.
- Brak strukturalizacji strumienia – usługa przesyłania za pomocą strumieni TCP nie uwzględnia strukturalizacji strumienia danych. Programy użytkowe wykorzystujące usługi przesyłania za pomocą strumieni muszą interpretować zawartość strumienia i jeszcze przed rozpoczęciem połączenia zgadzać się na format strumienia.
- Połączenie w pełni dwukierunkowe – połączenia zapewniane przez usługę przesyłania za pomocą strumieni TCP są połączeniami w pełni dwukierunkowymi (ang. *Duplex*). Z punktu widzenia programu użytkowego połączenie w pełni dwukierunkowe składa się z dwu niezależnych strumieni danych płynących w przeciwnych kierunkach bez żadnej jawnej interakcji między nimi. Usługa przesyłania za pomocą strumieni pozwala procesowi użytkownika na zatrzymanie przepływu w jednym z kierunków bez zakłócania przepływu w drugim, co czyni połączenie połączeniem połowicznie

dwukierunkowym (ang. *Half duplex*). Przy połączeniu w pełni dwukierunkowym oprogramowanie protokołów obsługujących może wysyłać nadawcy informacje kontrolne związane z jednym strumieniem w datagramach niosących dane w kierunku przeciwnym, co jest jego zaletą. Powoduje to zmniejszenie ruchu w sieci.

Rozpatrując TCP z punktu widzenia funkcjonalności można potraktować jego pracę jako ustanowienie kanału wirtualnego realizującego komunikację między "końcówkami" – tak wygląda to z punktu widzenia programu użytkownika.[8]

3.11 Usługa zapewniana przez TCP programom użytkowym

Z punktu widzenia programu użytkowego usługa oferowana przez TCP ma siedem głównych cech:

- Zorientowanie na połączenie – TCP zapewnia usługę zorientowaną połączeniowo, w której program użytkowy musi najpierw poprosić o połączenie do odbiorcy, aby następnie używać go do przesyłania danych.
- Komunikacja punk-do-punktu – każde połączenie TCP ma dokładnie dwa końce.
- Pełna niezawodność – TCP gwarantuje, że dane wysyłane połączeniem będą dostarczone dokładnie tak, jak były wysłane, bez żadnych braków czy dostarczania nie w kolejności.
- Komunikacja w pełni dwukierunkowa – połączenie TCP pozwala, żeby dane przepływały w każdym kierunku, a każdy program może wysyłać dane w dowolnym momencie. TCP umożliwia buforowanie wychodzących i przychodzących danych z obu kierunków, co pozwala programom na wysyłanie danych i dalsze wykonywanie obliczeń, podczas gdy te informacje są przesyłane.
- Interfejs strumieniowy – oprogramowanie TCP zapewnia interfejs strumieniowy, w którym program wysyła połączeniem ciągłą sekwencję oktetów. Oznacza to, że TCP nie udostępnia pojęcia rekordu oraz nie gwarantuje, że dane zostaną dostarczone do programu odbierającego w kawałkach tego samego rozmiaru co wysyłane przez program nadający.
- Niezawodne tworzenie połączenia – protokół TCP wymaga, aby dwa programy tworzące połączenie, musiały się na nie zgodzić; duplikaty pakietów używanych w poprzednich połączeniach nie będą wyglądały jak prawidłowe odpowiedzi ani też w inny sposób nie będą zakłócać nowego połączenia.
- Łagodne kończenie połączenia – program użytkowy może otworzyć połączenie, wysłać dowolną ilość danych, a następnie poprosić, aby połączenie zostało zamknięte. Protokół TCP gwarantuje niezawodne dostarczenie wszystkich danych przed zamknięciem połączenia.[4]

4. ZAKOŃCZNIE

Większość systemów komputerowych umożliwia jednoczesne wykonywanie wielu programów użytkowych. Protokół UDP służy do rozróżniania procesów pracujących na danym komputerze, umożliwiając nadawcy i odbiorcy dodanie do każdego komunikatu UDP dwóch 16-bitowych liczb nazywanych numerami portów. Numery portów identyfikują nadawcę i odbiorcę. Niektóre numery portów UDP, nazywane powszechnie znanymi, są przydzielone na stałe i honorowane w obrębie całego internetu, inne są dostępne dla dowolnych programów użytkowych.

UDP nie zmienia zbytnio semantyki IP. Dostarcza jedynie programom użytkowym możliwość komunikowania się za pośrednictwem zawodnego bezpołączeniowego mechanizmu dostarczania pakietów. Komunikaty UDP mogą ginąć, być duplikowane, opóźniane lub dostarczane w innej kolejności niż były wysyłane. Program użytkowy korzystający z UDP musi sam radzić sobie z tymi problemami.[2]

Protokół UDP służy temu samemu zadaniu co TCP, dysponuje jednak mniejszą liczbą funkcji. Zarówno pakiety TCP, jak i UDP są przenoszone w pakietach IP, ale jedyną funkcją zapewnienia niezawodności, jaką dysponuje UDP jest ponowne rozsyłanie pakietów, które nie dotarły do adresu docelowego. Główną zaletą UDP jest znacznie większa prędkość w najprostszych połączeniach sieciowych, takich jak wysyłanie strony internetowej do komputera-klienta. Ponieważ UDP nie posiada wielu funkcji sprawdzania i usuwania błędów, powinien być używany jedynie wtedy, jeśli nie ma dla nas większego znaczenia, że dane czasami zagubią się między punktami, lub gdy oprogramowanie jest w stanie samo przeprowadzić kontrolę i usunąć błędy.[3]

Protokół TCP opisuje najważniejszą usługę oferowaną przez intersieć – dostarczanie pewnymi strumieniami. TCP zapewnia w pełni dwukierunkowe połączenie między dwoma maszynami, co umożliwia efektywną wymianę dużych ilości danych. TCP używa protokołu przesuwającego się okna, dzięki czemu sieć może być efektywnie wykorzystana. Protokół ma niewiele założeń i jest na tyle elastyczny, że może działać w wielu systemach.

TCP realizuje kontrolę przepływu, umożliwiając nadawcy proponowanie, ile danych może on przyjąć. Dzięki możliwości specyfikowania pilności danych komunikaty mogą być przesyłane poza pasmem transmisyjnym, a dzięki mechanizmowi wypychania – przenoszenie danych można wymusić.

Obecny standard TCP określa wykładnicze wydłużanie czasów oczekiwania na retransmisję oraz algorytmy unikania przeciążenia, takie jak powolny start, wielokrotne zmniejszanie oraz stopniowe zwiększanie. Dodatkowo TCP używa algorytmów heurystycznych służących do zapobiegania przesyłaniu małych pakietów.[2]

LITERATURA

[1]. Stevens W. Richard „UNIX: programowanie usług sieciowych”, WNT, Warszawa 2002
[2]. Douglas Comer „Sieci komputerowe: intersieci”, WNT, Warszawa 2000
[3]. Bruce Hallberg „Sieci komputerowe, kurs podstawowy”, ”Edition2000”, Kraków 2001
[4]. Douglas E. Comer „Sieci komputerowe TCP/IP. Zasady, protokoły i architektura”, WNT, Warszawa 1997
[5]. Craig Hunt „TCP/IP – administracja sieci”, Oficyna Wydawnicza Read Me, Warszawa 1996
[6]. Nowicki K. Woźniak J. „Sieci LAN, MAN, WAN – protokoły komunikacyjne” Wydawnictwo Postępu Technicznego, Kraków 1998
[7]. http://rainbow.mimuw.edu.pl/SO/Linux/index08.html
[8]. http://main.amu.edu.pl/~psi/informatyka/tcpip/
[9]. http://www.net.pagina.pl/5transportowa.htm