Jak działa Internet – TCP i NAT

Witajcie w ostatnim artykule z serii „Jak działa Internet”! Zapoznaliśmy się w niej z różnymi zagadnieniami dotyczącymi Internetu – od najniższych warstw modelu OSI oraz adresowania IP i MAC po protokoły DNS i HTTP. Od czasu do czasu zdarza się jednak, że musimy przyjrzeć się warstwie „sklejającej” omówione dotąd mechanizmy. Dlatego w tym artykule poznamy podstawowe zagadnienia związane z protokołami TCP i UDP oraz zapoznamy się z procesem NAT.

WPROWADZENIE DO TCP ORAZ UDP

Często spotykamy się z nazwą TCP/IP i chociaż nazwa zawiera protokół TCP i IP, to odnosi się do wszystkich omawianych i wspomnianych do tej pory protokołów (HTTP, DNS, ARP) [1]. W tym artykule skupimy się na protokołach TCP oraz UDP, które są częścią warstwy transportowej (czwartej warstwy modelu OSI). To one decydują o sposobie nawiązywania połączenia oraz walidacji poprawności otrzymanych danych.

UDP

UDP (ang. *User Datagram Protocol*) jest protokołem znacznie prostszym niż TCP. Prostota ta jednak skutkuje ograniczonymi możliwościami. W ramach tego protokołu nie nawiązujemy połączenia ani nie otrzymujemy potwierdzenia dostarczenia pakietu. Korzystając z UDP, musimy liczyć się z tym, że niektóre pakiety mogą zostać utracone, a aplikacje muszą takie straty poprawnie obsłużyć. [5]

Zaletą UDP jest jednak jego „szybkość” w porównaniu do TCP – brak nawiązywania połączenia oraz brak mechanizmu potwierdzenia odbioru pakietów skutkuje mniejszą ilością przesyłanych danych. Dzięki temu jest on szeroko używany chociażby w strumieniowaniu wideo, grach online i czatów głosowych [6].

TCP

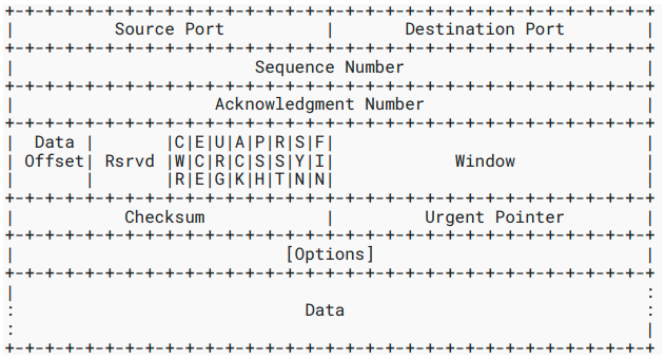
TCP jest niejako przeciwieństwem UDP zarówno co do narzutu danych, jak i funkcjonalności. Komunikacja w protokole TCP zaczyna się od nawiązania połączenia. Zarówno wysłane, jak i otrzymane pakiety są potwierdzane przez obie strony komunikacji. Wszystko to prowadzi do większej ilości wymienianych danych, ale za to mamy większe prawdopodobieństwo wykrycia błędów podczas ich przesyłu [7].

Ze względu na swoją charakterystykę protokół ten jest używany w większości aplikacji do komunikacji z serwisami oraz do transmisji plików. W tych przypadkach ważne jest, aby wiadomość dotarła kompletna bez błędów.

Zarówno protokół TCP, jak i UDP ma tzw. sumę kontrolną (ang. *checksum*). Suma ta zmniejsza prawdopodobieństwo zaakceptowania błędnego pakietu, ale mu nie zapobiega. Dużo lepszą integralność danych gwarantują dopiero protokoły wyższego poziomu jak np. TLS [16].

SZCZEGÓŁY TCP

Tematyki tego protokołu dotknęliśmy już przy okazji omawiania protokołu HTTP. HTTP w większości przypadków używa właśnie TCP do transmisji danych, dlatego ważne jest dla nas jego głębsze zrozumienie. Na Rysunku 1 przedstawiono strukturę pakietu TCP podaną w rfc9293 (specyfikacji protokołu) [3].



Rysunek 1. Schemat pakietu według rfc9293

Widzimy wiele pól w pakiecie, a o najważniejszych z nich opowiemy w tym artykule [8]:

Port źródłowy i docelowy (source port, destination port) – to nic innego jak liczby, które pozwalają identyfikować komunikację między różnymi aplikacjami w ramach jednego hosta. Możemy to sobie wyobrazić na przykładzie przeglądarki internetowej z wieloma otwartymi zakładkami, na których jest wczytana ta sama strona internetowa. Ponieważ strona internetowa jest pewną aplikacją, to wszystkie pakiety TCP wysłane przez te zakładki będą miały ten sam port docelowy. Jednak różne zakładki mogą poruszać się po różnych podstronach witryny, żądając innych danych, dlatego każde takie połączenie do serwera będzie mieć inny port źródłowy. Dzięki temu serwer będzie mógł odróżnić połączenia od siebie i wysyłać odpowiednie dane, używając odpowiedniego połączenia.

Numer sekwencyjny i numer potwierdzenia (sequence number i acknowledgement number) – liczby pozwalające stronom komunikacji na wzajemne informowanie się o wysłanych i odebranych danych, co umożliwia retransmisję zagubionych pakietów.

Data offset – informuje nas, gdzie w pakiecie znajdują się dane przeznaczone dla aplikacji. Wartość ta jest zmienna ze względu na pole Options, które ma zmienną długość.

Flagi (CWR, ECE, URG, ACK, RST, PSH, SYN, FIN) – bity określające przeznaczenie danego pakietu. Na przykład flaga SYN służy do nawiązania połączenia (synchronizacji), a PSH (push) do przesyłania danych.

Okno (ang. *Window*) – Informuje o wielkości bufora danych zarezerwowanego do komunikacji między klientem a serwerem. Jego użycie w praktyce zobaczymy w dalszej części artykułu.

Checksum – suma kontrolna pozwalająca na weryfikację poprawności przesłanych danych.

Urgent Pointer – obecnie jest głównie nieużywany. Nie będzie omawiany w tym artykule.

Options – dodatkowe opcje połączenia, które mogą być wspierane przez strony komunikacji. Pole może również zawierać dodatkowe dane przesyłane razem z pakietem, jak np. te omawiane w sekcji SACK.

Data – dane wysyłane do aplikacji w pakiecie TCP.

Spójrzmy na pakiety TCP w akcji. Wyślijmy żądanie HTTP do dowolnego serwisu (w tym przypadku http://www.google.com) i podejrzyjmy pakiety w aplikacji Wireshark.

Uruchommy więc Wiresharka, a do wysłania żądania wykorzystamy narzędzie cURL, jak przedstawiono to w Listingu 1.

Listing 1. Wykorzystanie narzędzia cURL do wysłania żądania

```
curl http://www.google.com
```

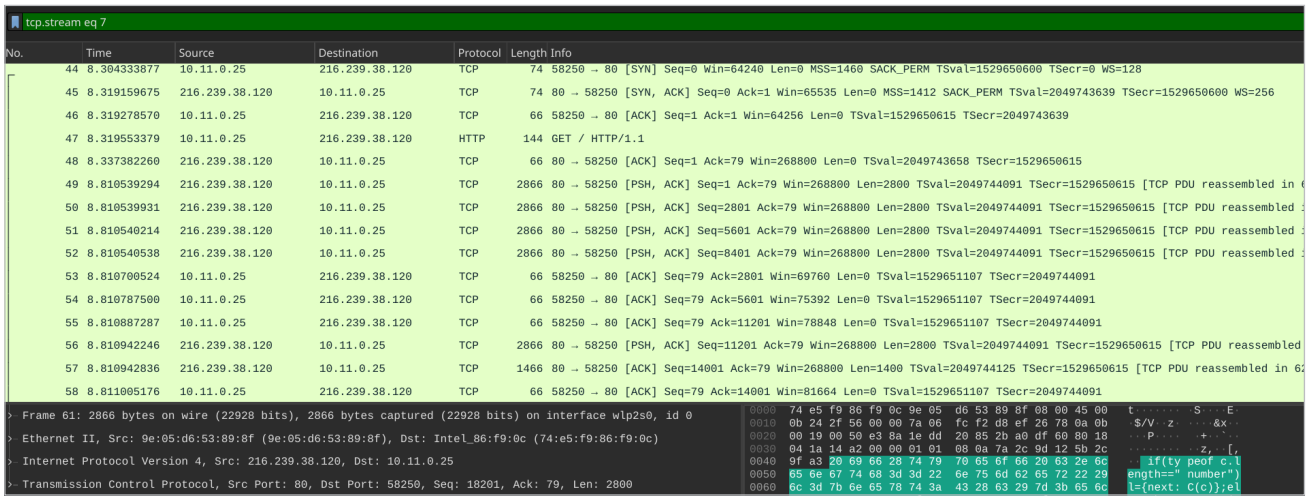
W oknie Wiresharka pojawią się nam kolejne pakiety TCP. Po ich odfiltrowaniu powinniśmy ujrzeć okno podobne do tego z Rysunku 2. Na jego podstawie przeanalizujemy przebieg komunikacji TCP.

Nawiązywanie połączenia TCP

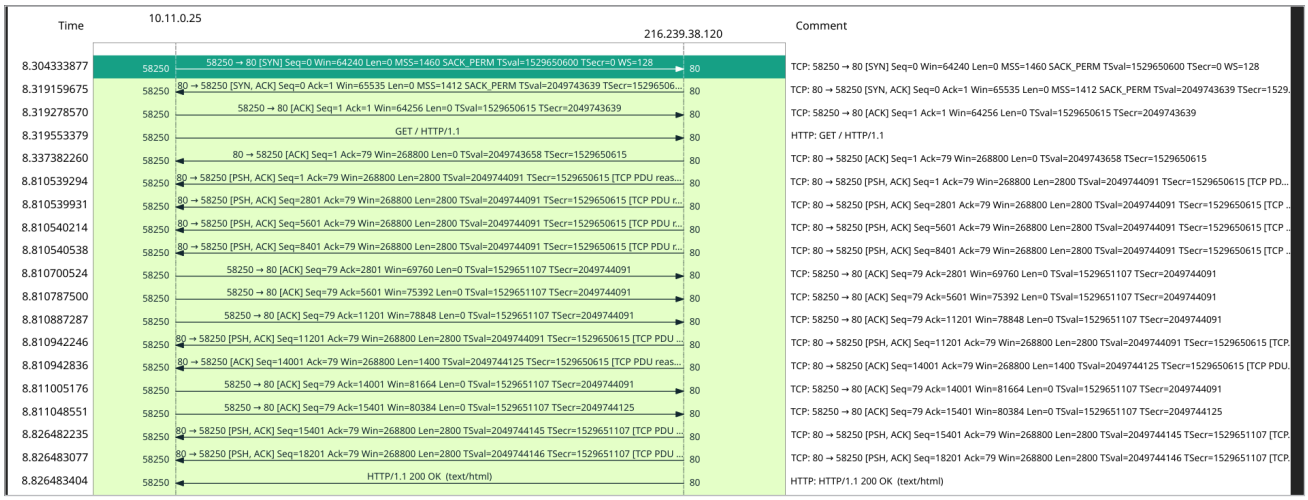
Na początku komunikacji następuje proces nawiązywania połączenia. Często określa się go jako *three-way handshake*, ponieważ składa się z trzech wymienianych pakietów (klient → serwer, serwer → klient, klient → serwer). Warto wspomnieć, że jeden z tych pakietów wynika z połączenia pakietów SYN i ACK, które zobaczymy za chwilę. W związku z tym poprawne jest również nawiązywanie połączenia w czterech krokach.

Protokół TCP przewiduje także sytuację, w której obie strony jednocześnie inicjują nawiązanie połączenia. Chociaż jest to rzadkie zjawisko, warto o nim pamiętać, analizując mniej typowe scenariusze komunikacji TCP [9].

Korzystając z opcji Flow Graph (dostępnej w Statistics → Flow Graph), możemy lepiej zwizualizować przebieg komunikacji, co pokazuje Rysunek 3.



Rysunek 2. Podgląd komunikacji TCP w aplikacji Wireshark



Rysunek 3. Komunikacja TCP w widoku Flow Graph

PAIR PROGRAMMING – CZY TO DZIAŁA? · STREFA CVE

INDEX: 285358

www.programistamag.pl

Magazyn programistów i liderów zespołów IT

programista

2/2025 (117)

Cena 32.90 zł (w tym VAT 8%)

CZY „CZYSTY KOD" WYTRZYMAŁ PRÓBĘ CZASU?

ISSN 2084-9400 02

JAK DZIAŁA INTERNET - TCP I NAT

WŁASNY [illegible] NA VPS

MEGA65 - WSKRZESZONY NASTĘPCA WIELKIEJ LEGENDY

[illegible]ENA AUTOMATYZACJA [illegible]EN DLA ASP.NET

NOWY NUMER JUŻ W EMPIKACH