Programowanie Sieciowe

Laboratorium 1.1 + 1.2 + MTU

Sprawozdanie Zespołu 51

03.12.2025

Maciej Bogusławski 331362
Hubert Kaczyński 331386
Bartosz Żelazko 331457

1. Opis zaimplementowanych programów

Podczas realizacji zadania 1.1 sporządzone zostały 4 programy – klient i serwer połączenia UDP w języku Python oraz w języku C. Kod źródłowy został wykonany kierując się podstawowym kodem zamieszczonym przez prowadzącego przedmiot na serwerze studia. Kluczowe dla działania programów było zaprojektowanie przesyłanego datagramu. Przyjęta została konstrukcja, w której pierwsze dwa bajty zarezerwowane są na 16 bitową liczbę binarną, która informuje, ile par klucz wartość nastąpi dalej w datagramie. Każda para składa się z wielkiej litery alfabetu (char) o wielkości bajtu oraz z binarnej liczby 16 bitowej – para ma więc wielkość 3 bajtów. Kolejne pary zawierają kolejne litery alfabetu i kolejne liczby, zaczynając od litery „A" i liczby „0000000000000000", przy czym litery powtarzają się cyklicznie przy przekroczeniu długości alfabetu. Tym samym dopuszczalną długością datagramu jest liczba podzielna przez 3 z resztą 2.

Serwer C (c_server.c) i Python (p_server.py) po uruchomieniu wyczekują na wiadomość na ustalonym porcie. Po otrzymaniu wiadomości sprawdzają, czy jej długość zgadza się z tą zakodowaną w dwóch pierwszych bajtach. Jeśli jest poprawna, to wysyła odpowiedź „VALID", jeśli nie to „INVALID". Serwery wyprowadzają na stdout adres klienta przysyłającego datagram o maksymalnej długości 65535 bajtów. Działanie serwerów należy zatrzymać ręcznie.
Przykładowe wywołanie klienta Python:
python3 -u p_server.py --host 0.0.0.0 --port 8000
Serwer C przyjmuje argument z numerem używanego portu oraz opcjonalny argument przyjmujący wartość 1 – program wypisuje treść datagramu lub 0 – program nie wypisuje treści. Trzecim argumentem jest adres hosta.
Przykładowe wywołanie serwera C:
./c_server 8000 0 0.0.0.0

Klient C (c_client.c i c_client.h) przyjmuje następujące wymagane argumenty wywołania:
- Adres, na który ma zostać wysłany datagram,
- Port, który ma zostać wykorzystany,
- Ilość datagramów do wysłania
- Opóźnienie w ms między wysłanymi datagramami,
- Binarny argument, który wskazuje czy ma zostać włączona fragmentacja pakietów IP (wartość 0), czy ma zostać wyłączona (wartość 1),
- Listę długości (w bajtach) wysyłanych datagramów (liczby oddzielone spacją). Podczas wysyłania klient przechodzi cyklicznie przez listę aż wyśle tyle datagramów ile zlecił wcześniejszy argument.

Maksymalna długość datagramu została ustawiona na 65535 bajtów (maksymalna wartość zmiennej uint16_t). Klient informuje o błędach sygnalizowanych przez funkcje systemowe i wykonanych czynnościach w terminalu. Po wysłaniu klient czeka na odpowiedź od serwera, informując w terminalu o wyniku.
Przykładowe wywołanie:
./c_client 0.0.0.0 8000 9 100 0 5

Klient Python (p_client.py) przyjmuje następujące argumenty wywołania (każdy ma też wartość domyślną):
- Adres, na który ma zostać wysłany datagram,
- Port, który ma zostać wykorzystany,

- Listę długości (w bajtach) wysyłanych datagramów (liczby oddzielone spacją). Wysłane zostanie tyle datagramów ile długości otrzymał program.

Maksymalna długość datagramu została ustawiona na 65535 bajtów. Klient informuje o błędach sygnalizowanych przez funkcje systemowe i wykonanych czynnościach w terminalu. Po wysłaniu klient czeka na odpowiedź od serwera, informując w terminalu o wyniku.
Przykładowe wywołanie:
python3 -u p_client.py --host 0.0.0.0 --port 8000 --sizes 32 64 128 512

Do wykonania polecenia 1.1 zostały przygotowane odpowiednie pliki dockerowe, które odpowiadają za uruchomienie serwerów i klientów (z podaną jedną długością datagramu) w każdej konfiguracji między językami Python i C.

Do wykonania polecenia 1.2 wystarczyło zmienić argument wywołania klientów (lista długości) tak aby wysyłały one coraz większe datagramy w zaplanowanym eksperymencie.

2. Prezentacja działania programów i wnioski

Na potrzeby zadania 1.1 każda konfiguracja języków została przetestowana w formie komunikacji klient-serwer. Dla przygotowanego w środowisku Docker eksperymentu zostało również wykonane polecenie 1.2. Wyniki powstałe w terminalu zostały sczytane z użyciem asciinema i umieszczone w plikach zad_1_client.cast, zad_1_server.cast, zad_1_mtu_client.cast, zad_1_mtu_server.cast. Za każdym razem komunikacja odbyła się pomyślnie – błędne datagramy były wyłapywane, a klient otrzymywał odpowiedź od serwera o pomyślnym wysłaniu dobrych datagramów.

Podczas wykonywania zadania 1.2 klientowi C została podana następująca lista długości datagramów: 65501, 65507, 65510. Udało się wysłać datagram o długości 65507 bajtów, ale wysłanie kolejnej długości wg formuły 3n + 2 czyli 65510 zakończyło się niepowodzeniem, co oznacza, że maksymalna długość wysyłanego datagramu to 65507 bajtów. Zgadza się to z oczekiwanym wynikiem, ponieważ z maksymalnej wielkości pakietu 65535 bajtów należy odjąć 20 bajtów zarezerwowanych na nagłówek IP oraz 8 bajtów zarezerwowanych na nagłówek UDP, co pozostawia właśnie 65507 bajtów na przesyłany datagram.

3. Zaobserwowanie wpływu MTU

Na potrzebę dodatkowego zadania został edytowany klient C, który po podaniu odpowiedniego argumentu wywołania (wyjaśnione wyżej) wyłączał fragmentację pakietów IP. Powinno to blokować przed wysłaniem pakiety o wielkości przekraczającej MTU. Tak dokładnie się stało i przykładowo dla ustawionego MTU = 1500 pakiet o zadanej wielkości 1472 (+ 28 bajtów nagłówka) został wysłany, natomiast już pakiet o zadanej wielkości 1475 (+ 28 bajtów nagłówka) został zablokowany.