Sprawozdanie – laboratorium 3 PSI

Przedmiot: Programowanie Sieciowe
Autorzy: Marek Dzięcioł, Jakub Mieczkowski, Tomasz Nejman-lider
Data sporządzenia: 05.12.2025
Wersja: 1.0

Treść zadania:

Z 1 Komunikacja UDP Napisz zestaw dwóch programów – klienta i serwera wysyłające datagramy UDP. Proszę napisać jedno zadanie w konfiguracji klient/serwer Python/C, a drugie w konfiguracji klient/serwer C/Python – do wyboru.

Z 1.2 (to zadanie realizowane jest jako ostatnie)
Klient ma za zadanie odczytać plik z dysku (proszę wygenerować plik z losowymi 10000B) i wysłać do serwera jego zawartość w paczkach po 100B. Serwer ma zrekonstruować cały plik i obliczyć jego hash. Jako dowód działania proszę m.in. porównać hash obliczony przez serwer z hashem obliczonym przez klienta (może to być wydrukowane w konsoli klienta/serwera, hashe muszą być identyczne). Należy zaimplementować prosty protokół niezawodnej transmisji, uwzględniający możliwość gubienia datagramów. Gubione pakiety muszą być wykrywane i retransmitowane aby serwer mógł odtworzyć cały plik. Należy uruchomić program w środowisku symulującym błędy gubienia pakietów.
(Informacja o tym, jak to zrobić znajduje się w skrypcie opisującym środowisko Dockera).

Opis rozwiązania problemu:

Utworzyliśmy dwa programy:

- klient udp w C
- serwer udp w Pythonie

Serwer nasłuchuje na danym porcie UDP (domyślnie 8888), używając funkcji recvfrom(). Po odebraniu datagramu, wysyła stałą odpowiedź z powrotem do klienta, używając adresu i portu nadawcy, uzyskanych z recvfrom().
Klient najpierw generuje pseudo-losowy ciąg 10000 znaków, nastepnie dzieli je na 100 pakietów po 100B. Następnie klient w pętli wysyła po jednym pakiecie. Jeżeli serwer nie odpowie zdefiniowanym czasie (timeout, u nas 200ms), pakiet uznawany jest z zgubiony i ponawia się jego wysłanie. Kiedy serwer odpowie, potwierdzając odebranie pakietu, klient przechodzi do następnego pakietu. Process kontynuuje do momentu wysłania wszystkich pakietów. Na końcu serwer składa pakiety w całość; serwer i klient liczą skrót całości danych.

W czasie, kiedy klient czeka (pierwsze 10 sekund, opisane poniżej), w drugim terminalu wpisano komendę:
ocker exec z38_client tc qdisc add dev eth0 root netem delay 1000ms 500ms loss 50%
Aby zasymulować problemy sieciowe.

Napotkane problemy

1. Klient nie rozpoznawał hosta. Rozwiązanie: w docker-compose.yaml dodano:

```
server:
        …
        networks:
                z38_network:
                        aliases:
                                - z38_server
…
```

2. Klient zbyt szybko wysyłał pakiety i nie było czasu na wprowadzenie zakłóceń. Rozwiązanie:

```
printf("Sleeping for 10 seconds...\n");
sleep(10);
```

3. Na początku datagramu, oprócz fragmentu danych zawierano identyfikator pakietu, jedna liczba całkowita kodowana jako 4 char'y (maska bitowa) i występował problem w wypisywaniu numeru datagramu. Rozwiązanie: odkodowanie 4 bajtów "charów" z powrotem jako int.

Opis konfiguracji

Serwer/środowisko: bigubu.ii.pw.edu.pl
Port domyślny: 8888
Sieć: z38_network
Adres sieci: 172.21.38.0/24, fd00:1032:ac21:38::/64
Nazwy kontenerów:

- Serwer: z38_server
- Klient: z38_client

Opis testowania i wyniki testów

Wycinek z wyjścia klienta:

```
z38_client  | Timeout occurred! Resending packet 0...
z38_client  | Sending Packet 0/100 (104 bytes)...
z38_client  | Packet content:
EIGENEIRGEGEGENIRRIINGGGGGGEEENRRRNGEIEIGGGNGEGGNRNIEGGGGGIEG
GGGGNIGENRGRGGGGGGGERRGGRGNGERRGREII
z38_client  | Timeout occurred! Resending packet 0...
z38_client  | Sending Packet 0/100 (104 bytes)...
```

z38_client | Packet content:
EIGENEIRGEGEGENIRRIINGGGGGGEEENRRRNGEIEIGGGNGEGGNRNIEGGGGGIEG
GGGGNIGENRGRGGGGGGGGERRGGRGNGERRGREII
z38_client | Received ACK: 0
z38_client | Sending Packet 1/100 (104 bytes)...
z38_client | Packet content:
GIGINGNGGRIGEGGEGNINIGGEGNGNGEGINGRNINNGIGGNGNRRINIGRNGEGGENNI
GGEGEIGIRINGGGNEGIINNINNREGENEGEGGG
z38_client | Timeout occurred! Resending packet 1...
z38_client | Sending Packet 1/100 (104 bytes)...
z38_client | Packet content:
GIGINGNGGRIGEGGEGNINIGGEGNGNGEGINGRNINNGIGGNGNRRINIGRNGEGGENNI
GGEGEIGIRINGGGNEGIINNINNREGENEGEGGG
z38_client | Received ACK: 0
z38_client | Sending Packet 1/100 (104 bytes)...
z38_client | Packet content:
GIGINGNGGRIGEGGEGNINIGGEGNGNGEGINGRNINNGIGGNGNRRINIGRNGEGGENNI
GGEGEIGIRINGGGNEGIINNINNREGENEGEGGG
z38_client | Received ACK: 0
z38_client | Sending Packet 1/100 (104 bytes)...
z38_client | Packet content:
GIGINGNGGRIGEGGEGNINIGGEGNGNGEGINGRNINNGIGGNGNRRINIGRNGEGGENNI
GGEGEIGIRINGGGNEGIINNINNREGENEGEGGG
z38_client | Timeout occurred! Resending packet 1...
z38_client | Sending Packet 1/100 (104 bytes)...
z38_client | Packet content:
GIGINGNGGRIGEGGEGNINIGGEGNGNGEGINGRNINNGIGGNGNRRINIGRNGEGGENNI
GGEGEIGIRINGGGNEGIINNINNREGENEGEGGG
z38_client | Received ACK: 0
z38_client | Sending Packet 1/100 (104 bytes)...
z38_client | Packet content:
GIGINGNGGRIGEGGEGNINIGGEGNGNGEGINGRNINNGIGGNGNRRINIGRNGEGGENNI
GGEGEIGIRINGGGNEGIINNINNREGENEGEGGG
z38_client | Received ACK: 0
z38_client | Sending Packet 1/100 (104 bytes)...
z38_client | Packet content:
GIGINGNGGRIGEGGEGNINIGGEGNGNGEGINGRNINNGIGGNGNRRINIGRNGEGGENNI
GGEGEIGIRINGGGNEGIINNINNREGENEGEGGG
z38_client | Received ACK: 0
z38_client | Sending Packet 1/100 (104 bytes)...
z38_client | Packet content:
GIGINGNGGRIGEGGEGNINIGGEGNGNGEGINGRNINNGIGGNGNRRINIGRNGEGGENNI
GGEGEIGIRINGGGNEGIINNINNREGENEGEGGG
z38_client | Timeout occurred! Resending packet 1...
z38_client | Sending Packet 1/100 (104 bytes)...
z38_client | Packet content:
GIGINGNGGRIGEGGEGNINIGGEGNGNGEGINGRNINNGIGGNGNRRINIGRNGEGGENNI
GGEGEIGIRINGGGNEGIINNINNREGENEGEGGG

z38_client | Received ACK: 1
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Received ACK: 1
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Received ACK: 1
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Received ACK: 1
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Timeout occurred! Resending packet 2...
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Received ACK: 1
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Received ACK: 1
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Received ACK: 1
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Received ACK: 1
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Received ACK: 1
z38_client | Sending Packet 2/100 (104 bytes)...

z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Timeout occurred! Resending packet 2...
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Received ACK: 1
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Timeout occurred! Resending packet 2...
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Timeout occurred! Resending packet 2...
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Timeout occurred! Resending packet 2...
z38_client | Sending Packet 2/100 (104 bytes)...
z38_client | Packet content:
GIEENGGIIIRRGGNRGGRNGNINIRNNGIGGRGEGGIGNGRNGIGIINIGEERINRGIRRNGIRGGGRGRRGNGIGNGNGIEGIGENNIGGERGEG
z38_client | Received ACK: 2
z38_client | Sending Packet 3/100 (104 bytes)...
z38_client | Packet content:
IGRRREGRERNGIIGGGGRNRGEREENEGGEEGNNRNGRIRIINGERGIRGGEGGEIGGNGGRGNGGIREGINNGGREIINIEENEGEEGRGIIIGG
z38_client | Timeout occurred! Resending packet 3...
z38_client | Sending Packet 3/100 (104 bytes)...
z38_client | Packet content:
IGRRREGRERNGIIGGGGRNRGEREENEGGEEGNNRNGRIRIINGERGIRGGEGGEIGGNGGRGNGGIREGINNGGREIINIEENEGEEGRGIIIGG
z38_client | Received ACK: 2
z38_client | Sending Packet 3/100 (104 bytes)...
z38_client | Packet content:
IGRRREGRERNGIIGGGGRNRGEREENEGGEEGNNRNGRIRIINGERGIRGGEGGEIGGNGGRGNGGIREGINNGGREIINIEENEGEEGRGIIIGG
z38_client | Timeout occurred! Resending packet 3...
z38_client | Sending Packet 3/100 (104 bytes)...
z38_client | Packet content:
IGRRREGRERNGIIGGGGRNRGEREENEGGEEGNNRNGRIRIINGERGIRGGEGGEIGGNGGRGNGGIREGINNGGREIINIEENEGEEGRGIIIGG

z38_client | Timeout occurred! Resending packet 3...
z38_client | Sending Packet 3/100 (104 bytes)...
z38_client | Packet content:
IGRRREGRERNGIIGGGGRNRGEREENEGGEEGNNRNGRIRIINGERGIRGGEGGEIGGN
GGRGNGGIREGINNGGREIINIEENEGEEGRGIIIGG
z38_client | Received ACK: 2
z38_client | Sending Packet 3/100 (104 bytes)...
z38_client | Packet content:
IGRRREGRERNGIIGGGGRNRGEREENEGGEEGNNRNGRIRIINGERGIRGGEGGEIGGN
GGRGNGGIREGINNGGREIINIEENEGEEGRGIIIGG

Serwer odpowiada na niektóre pakiety po ustalonym czasie odpowiedzi (timeout) wysyłając ack, co klient odbiera, jednak nie ma to znaczenia, gdyż pakiet zostaje ponownie wysłany wcześniej. Z powodu ponownego wysłania i opóźnienia serwer odpowiada ack kilka razy na ten sam fragment danych (ten sam fragment nie oznacza tej samej wiadomości).

z38_server | Packet recived # 4
z38_server | Packet recived # 4
z38_server | Packet recived # 3
z38_server | Packet recived # 4
z38_server | Packet recived # 4
z38_server | Packet recived # 3
z38_server | Packet recived # 5
z38_server | Packet recived # 4
z38_server | Packet recived # 5
z38_server | Packet recived # 5
z38_server | Packet recived # 5
z38_server | Packet recived # 6
z38_server | Packet recived # 5
z38_server | Packet recived # 5
z38_server | Packet recived # 5
z38_server | Packet recived # 6
z38_server | Packet recived # 6
z38_server | Packet recived # 7
z38_server | Packet recived # 7
z38_server | Packet recived # 7
z38_server | Packet recived # 7
z38_server | Packet recived # 6
z38_server | Packet recived # 7
z38_server | Packet recived # 7
z38_server | Packet recived # 7
z38_server | Packet recived # 8
z38_server | Packet recived # 8
z38_server | Packet recived # 8
z38_server | Packet recived # 8
z38_server | Packet recived # 8
z38_server | Packet recived # 8
z38_server | Packet recived # 9

z38_server | Packet recived # 8
z38_server | Packet recived # 9
z38_server | Packet recived # 9
z38_server | Packet recived # 9
z38_server | Packet recived # 9
z38_server | Packet recived # 10
z38_server | Packet recived # 10
z38_server | Packet recived # 10

Na koniec klient oblicza skrót:
z38_client | b7e9bb430b7bf92b5782598ece674dc05e9a53359e3b13296d359bb00623c9fc
z38_client | Done
z38_client exited with code 0

Serwer też oblicza skrót:
z38_server | Packet recived # 98
z38_server | Packet recived # 99
z38_server | SHA-256:
b7e9bb430b7bf92b5782598ece674dc05e9a53359e3b13296d359bb00623c9fc
z38_server exited with code 0

Ostatni pakiet odebrany na indeks 99, gdyż zaczynaliśmy indeksowanie od 0, tj. pierwszy pakiet to #0.

Wnioski końcowe

Udało się nam uruchomić dwa komunikujące się programy. Udało się, za pomocą docker-compose uruchomić dwa kontenery. Poprawnie zasymulowano problemy z siecią, komendą:
docker exec z38_client tc qdisc add dev eth0 root netem delay 1000ms 500ms loss 50%.
Klient, tak jak założono, wysyła pakiety ponownie, jeśli nastąpi timeout. Serwer i klient obliczyły skrót danych, który na szczęście był taki sam.