Programowanie sieciowe

Autorzy: Adam Czupryński, Szymon Makuch, Michał Sadlej

Data: 18.11.2024r.

Komunikacja UDP

Celem zadania było napisanie zestawu dwóch programoów - klienta i serwera wysyłające datagramy UDP.

Rozwiązanie

Program klienta wysyła kolejne datagramy o przyrastającej wielkości bajtów. Datagramy posiadają ustaloną formę danych: pierwsze dwa bajty datagramu zawierają informację o jego długości, a kolejne bajty powtarzające się litery A-Z.

```
def generateDatagram(size: int) -> bytes:
    message = bytes([(size & 0xFF00) >> 8, size & 0x00FF])

    for i in range(size - 2):
        message += bytes([ord('A') + (i % 26)])

    return message
```

Program serwera weryfikowuje odebrany datagram i odsyła w odpowiedzi potwierdzenie. Następnie klient odebiera potwierdzenie i przechodzi do następnego datagramu. Oba programy działają do momentu przerwania połączenia, które następuje w momencie próby wysłania zbyt dużej wiadomości.

```
def checkData(data: bytes) -> bool:
    if len(data) < 2:
        return False

    size = (data[0] << 8) + data[1]
    if size != len(data):
        return False

    for i in range(2, len(data)):
        if data[i] != ord('A') + (i - 2) % 26:
            return False

    return True
```

Wielkość największego obsługiwanego datagramu wynosi 65507 bajtów. Dla większych datagramów klient zwraca błąd: OSError: [Errno 90] Message too long. Teoretycznie maksymalna wielkość datagramu UDP wynosi 65535 bajtów, jednak faktyczny limit wynika z protokołu IPv4 i wynosi 65507 (65535 bajtów – 8-bajtów nagłówek UDP – 20-bajtów nagłowek IP).

Problemy

1. Błędna nazwa hosta

Podczas wywoływania polecenia `docker run` jako argument odpowiadający za nazwę hosta była podawana nazwa serwera. Jednak przez brak flagi --hostname nie dało się połączyć z serwerem. Początkowo zostało to napraione poprzez podawanie adresu IP serwera zamiast jego nazwy, jednak ostatecznie udało się zauważyć brak flagi --hostname.

2. Program działał poprawnie, ale tylko lokalnie

Gdy wywoływaliśmy program lokalnie uzyskiwaliśmy zadowalające wyniki, nie mieliśmy żadnych problemów, jednak po przeniesieniu go na system `bigubu` program przestał działać. Zostało to naprawione poprzez ustawienie "na sztywno" hostów oraz portów a następnie szukanie miejsc gdzie pojawiają się problemy.

3. Serwer nie wysyłał odpowiedzi do klienta

Napotkaliśmy problem przy próbie wysłania odpowiedzi do klienta zawierającej napis `"OK"`. Aby rozwiązać ten problem trzba było zamienić `"OK"` na `b"OK\x00"`, ponieważ serwer może wysyłać tylko dane w postaci bajtów.

```
if not checkData(data):
    print("Error in datagram")
    break

s.sendto(b"OK\x00", address)
```

Opis konfiguracji testowej

Adresy IP serwerów są tworzone automatycznie, klienci odwołują się do nich poprzez hostname. Serwery działają na portach 8000.

Przykład dla serwera pythonowego:

```
"ConfigOnly": false,
        "Containers": {

"468f5935169703698ec82485d9cf8d4862df43aa85cda304b86ede721b43e5d5": {
                "Name": "z34_p_server",
                "EndpointID":
"517be73eff0f5911659a69567bbe70cd0303627c0646f53e6b3a6d02713d39cf",
                "MacAddress": "02:42:ac:15:22:02",
                "IPv4Address": "172.21.34.2/24",
                "IPv6Address": "fd00:1032:ac21:34::2/64"
            }
        },
        "Options": {},
        "Labels": {}
```

Testy

W celu przetestowania programu sprawdzone zostały wszystkie konfiguracje między sobą.

Błąd wyrzucany przez klienta pythona:

```
Traceback (most recent call last):
  File "//./client.py", line 49, in <module>
    main(sys.argv[1:])
    ~~~~^^^^^^^^^^^^^^
  File "//./client.py", line 37, in main
    s.sendto(message, (host, port))
    ~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^
OSError: [Errno 90] Message too long
```

Błąd wyrzucany przez klienta C:

```
Sending 65507 bytes datagram...
Received 3 bytes from server
Sending 65508 bytes datagram...
send error!
: Message too long
```