16.11.2024 r.

PSI – Sprawozdanie – Zadanie 1.1

Autorzy
Krzysztof Gólcz 325159
Daniel Machniak 325190
Natalia Pieczko 325208

1. Treść zadania

Komunikacja UDP

Napisz zestaw dwóch programów – klienta i serwera wysyłające datagramy UDP. Wykonaj ćwiczenie w kolejnych inkrementalnych wariantach (rozszerzając kod z poprzedniej wersji). Klient jak i serwer powinien być napisany zarówno w C jak i Pythonie (4 programy). Sprawdzić i przetestować działanie „między-platformowe", tj. klient w C z serwerem Python i vice versa.

Z 1.1
Klient wysyła, a serwer odbiera datagramy o stałym rozmiarze (rzędu kilkuset bajtów). Datagramy powinny posiadać ustaloną formę danych. Przykładowo: pierwsze dwa bajty datagramu mogą zawierać informację o jego długości, a kolejne bajty kolejne litery A-Z powtarzające się wymaganą liczbę razy (ale można przyjąć inne rozwiązanie). Odbiorca powinien weryfikować odebrany datagram i odsyłać odpowiedź o ustalonym formacie. Klient powinien wysyłać kolejne datagramy o przyrastającej wielkości np. 1, 100, 200, 1000, 2000... bajtów. Sprawdzić, jaki był maksymalny rozmiar wysłanego (przyjętego) datagramu. Ustalić z dokładnością do jednego bajta jak duży datagram jest obsługiwany. Wyjaśnić.

2. Rozwiązanie

Przyjęta została następująca forma danych datagramu: pierwsze dwa bajty zawierają informację o długości datagramu, a kolejne to wiadomość składająca się z kolejnych liter od A do Z powtarzających się wymaganą liczbę razy.

2.1 Rozwiązanie w języku Python

Serwer

```
def main():
    args = parse_args()

    port = args.port
    host = args.host
    bufsize = args.bufsize

    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        s.bind((host, port))
        print(f"UDP server up and listening on {host}:{port}")
        print()
        print("-" * 50)
        while Work():
            data, address = s.recvfrom(bufsize)
            print(f"Recieved {len(data)} bytes from client {address}")
```

```
            response = confirm_datagram(data, bufsize)
            print(f"Sending response to client {address}: \"{response}\"")
            print("-" * 50)
            s.sendto(response.encode(), address)
```

Funkcja *parse_args()* odpowiedzialna jest za parsowanie podawanych argumentów: IP hosta, port I rozmiar bufforu. Gdy argumenty nie są zapewnione używane są wartości domyślne. W programie głównym *main*(), po przypisaniu odpowiednich wartości do zmiennych tworzone jest gniazdo UDP. Następnie za pomocą *bind*() przypisywany jest adres do gniazda. Serwer pętli odbiera dane o zadanym rozmiarze od klienta. Wywoływana jest funkcja *confirm_datagram()* zwracająca odpowiedź o poprawności otrzymanego datagramu.

```
def confirm_datagram(data, bufsize):
    if len(data) < 2:
        return "ERROR: Datagram not long enough"

    datagram_length_declared = int.from_bytes(data[:2], "big")
    msg = data[2:].decode()

    if datagram_length_declared > bufsize:
        return f"ERROR: Buffer size ({bufsize} bytes) exceeded. Declared
datagram length: {datagram_length_declared} bytes"

    if datagram_length_declared - 2 != len(msg):
        return f"ERROR: Incorret message length. Declared message length:
{datagram_length_declared - 2} bytes, but actual {len(msg)} bytes"

    letters = 'ABCDEFGHIJKLMNOPRSTUWXYZ'
    repeat_count = (datagram_length_declared - 2) // len(letters) + 1
    expected_seq = (letters * repeat_count)[:datagram_length_declared - 2]

    if msg != expected_seq:
        return "ERROR: Incorrent datagram format"

    return f"OK: Recieved message with length {len(msg)} bytes"
```

Gdy długość otrzymanych danych jest mniejsza niż 2 bajty, pewne jest, że są one niepoprawne, ponieważ, format danych powinien zawierać 2 bajty przeznaczone na długość datagramu. Zwracana jest wiadomość o błędzie. Następnie dane są dekodowane i sprawdzane jest czy deklarowana długość danych nie przekracza wielkości buffora, faktyczna długość wiadomości odpowiada zadeklarowanej oraz czy znaki w wiadomości to kolejne litery z zakresu od A do Z. Zwracane są infromacje o odpowiednich błędach. Jeżeli dane są poprawne zwracana jest informacja o długości otrzymanych danych.

W pętli głównego programu, po sprawdzeniu poprawności danych na serwerze wyświetlana jest odpowiednia informacja. Odpowiedź serwera jest wysyłana również do klienta na odpowiedni adres.

Klient

Funkcja *parse_args()* odpowiedzialna jest za parsowanie podawanych argumentów: IP serwera oraz port, na który klient będzie przesyłał dane. W main() tworzone jest gniazdo UDP.

```
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        print("-" * 50)
        for datagram_size in range(1, BUFSIZE + 100, 100):
            try:
```

```
                data = generate_datagram(datagram_size)

                print(f"Sending datagram with length {datagram_size} bytes to server
{host}:{port}.")

                s.sendto(data, (host, port))
                response = s.recv(BUFSIZE)

                print(f"Received response from server {host}:{port}:
\"{response.decode()}\".")
                print("-" * 50)

            except ValueError as ve:
                print(ve)
                print("-" * 50)
            except OverflowError:
                print(f"ERROR: Cannot write {datagram_size} on two bytes")
                print("-" * 50)
                break
            except OSError:
                print(f"ERROR: Maximum OS datagram size exceeded. Declared datagram
length: {datagram_size} bytes")
                print("-" * 50)
                print()
                print("Looking for maximum OS datagram size...")
                print()
                print("-" * 50)
                break
```

W pętli generowane są kolejne rozmiary datagramów zaczynając od 1 do BUFSIZE + 100 z krokiem co 100. Za generowanie danych w odpowiednim formacie odpowiedzialna jest funkcja generate_*data*().

```
def generate_datagram(length):
    if length < 2:
        raise ValueError("ERROR: Datagram not long enough")

    letters = 'ABCDEFGHIJKLMNOPRSTUWXYZ'
    repeat_count = (length - 2) // len(letters) + 1
    msg = (letters * repeat_count)[:length - 2]
    datagram = length.to_bytes(2, "big") + msg.encode()

    return datagram
```

Gdy podana długość datagramu jest mniejsza niż minimalna (2) podnoszony jest wyjątek. Obsługiwany jest on w razie potrzeby w pętli w progrmie *main*().
Gdy dane uda się wygenerowac poprawnie wysyłane są za pomocą *sendto*() na serwer o odpowiednim adresie. Następnie korzystając z *recv*() odbierana jest odpowiedź serwera, która jest następnie wyświetlana.
Gdy pojawią się wyjątki są one odpowiednio obsługiwane. Wyjątek *OSError* świadczy o tym, że przekroczony został maksymalny rozmiar datagramu w systemie. Za odnalezienie tej wartości odpowiada kolejna pętla.

```
for size in range(datagram_size-1, 1, -1):
            try:
                data = generate_datagram(size)
                print(f"Trying to send {size} bytes.")
                s.sendto(data, (host, port))
```

```
                print(f"Datagram with length {size} bytes accepted")
                print("-" * 50)
                break
            except ValueError as e:
                print(e)
                print("-" * 50)
            except OverflowError:
                print(f"ERROR: Cannot write {size} on two bytes")
                print("-" * 50)
            except OSError:
                print(f"ERROR: Maximum OS datagram size exceeded. Declared datagram
length: {size} bytes")
                print("-" * 50)

        print()
        print(f"Found maximum OS datagram size: {size} bytes.")
```

W zakresie od ostatnio wygenerowanego datagramu zmniejszając o 1 bajt aż do 1 podejmowana jest próba wysłania datagramu tej długości (*size*) na serwer. Gdy to się uda pętla jest przerywana i wyświetlany jest maksymalny rozmiar datagramu, który jest obsługiwany. Wartość otrzymana podczas testów to **65507 bajtów**.

2.2 Rozwiązanie w języku C

Serwer

Argumenty wywołania przypisywane są do odpowiednich zmiennych: host, port i bufsize. Tworzone jest gniazdo UDP. W przypadku niepowodzenia obsługiwany jest wyjątek.

```
int sock;
    if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }
```

Następnie za pomocą *bind*() przypisywany jest adres do gniazda. Alokowane jest też miejsca w pamięci na buffor. Inicjowana jest struktura adresu klienta.

```
struct sockaddr_in server_addr;
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(port);
server_addr.sin_addr.s_addr = inet_addr(host);


if (bind(sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) == -1) {
    perror("Bind failed");
    close(sock);
    exit(EXIT_FAILURE);
}

char *buffer = malloc(bufsize);
if (!buffer) {
    perror("Memory allocation failed");
    close(sock);
    exit(EXIT_FAILURE);
}

struct sockaddr_in client_addr;
socklen_t client_len = sizeof(client_addr);
```

W pętli *while*(1) serwer odbiera datagram od klienta o danym adresie za pomocą *recvfrom*(). Informacje o odebranych danych są wyświetlane. Za sprawdzenie ich poprawności odpowiedzialna jest funkcja *confirm_datagram()*. Odpowiednia odpowiedź wysyłana jest do klienta za pomocą *sendto*(). Po zakończeniu pętli zwalniana jest pamięć i zamykane jest gniazdo.

```
while (1) {
        int received = recvfrom(sock, buffer, bufsize, 0, (struct sockaddr
*)&client_addr, &client_len);
        if (received == -1) {
            perror("Receive failed");
            printf("----------------------------------------\n");
            continue;
        }

        buffer[received] = '\0';
        printf("Received %d bytes from client %s:%d\n", received,
               inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));

        char response[DEFAULT_BUFSIZE];

        confirm_datagram(buffer, received, bufsize, response);

        printf("Sending response to client: %s\n", response);
        printf("----------------------------------------\n");

        sendto(sock, response, strlen(response), 0, (struct sockaddr
*)&client_addr, client_len);
}

free(buffer);
close(sock);
return 0;
```

Funkcja *confirm_data*() sprawdza, czy podany datagram ma minimalną długość 2 (miejsce za zapisanie długości datagramu). Następnie wiadomość jest dekodowana i sprawdzane jest czy zadeklarowana długość jest mniejsza niż rozmiar bufora, czy faktyczna długość wiadomości odpowiada tej zadeklarowanej i czy format danych jest poprawny. Jeżeli wszystko jest w porządku również zwracana jest odpowiednia informacja.

```
void confirm_datagram(const char *data, int length, int bufsize, char *response) {
    if (length < 2) {
        strcpy(response, "ERROR: Datagram not long enough");
        return;
    }

    int declared_length = ((unsigned char)data[0] << 8) | (unsigned char)data[1];
    const char *msg = data + 2;
    int msg_length = length - 2;

    if (declared_length > bufsize) {
        snprintf(response, bufsize, "ERROR: Buffer size (%d bytes) exceeded.
Declared datagram length: %d bytes.",
                 bufsize, declared_length);
        return;
    }

    if (declared_length - 2 != msg_length) {
```

```
        snprintf(response, bufsize, "ERROR: Incorrect message length. Declared: %d
bytes, Actual: %d bytes.",
                 declared_length - 2, msg_length);
        return;
    }

    if (compare_format(msg, msg_length, declared_length) != 0) {
        strcpy(response, "ERROR: Incorrect datagram format");
        return;
    }

    snprintf(response, bufsize, "OK: Received message with length %d bytes.",
msg_length);
}
```

Klient

Podobnie jak w przypadku serwera potrzebne wartości agrumentów wywołania programu są przypisywane do zmiennych (adres IP i port serwera). Tworzone jest gniazdo UDP, struktura adresu serwera i alokowana jest pamięć na datagram i odpowiedź od serwera.

```
int sock;
if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
    perror("Socket creation failed");
    exit(EXIT_FAILURE);
}

struct sockaddr_in server_addr;
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(port);
server_addr.sin_addr.s_addr = inet_addr(host);

char *datagram = malloc(BUFSIZE);
char *response = malloc(BUFSIZE);
if (!datagram || !response) {
    perror("Memory allocation failed");
    exit(EXIT_FAILURE);
}
```

W pętli, rozmiary datagramu (od 1 do BUFSIZE+100, z krokiem 100) są argumentem wywołania funkcji generującej datagram *generate_datagram*(). Obsługiwane są odpowiednie wyjątki w przypadku niepowodzenia tworzenia datagramu. Następnie korzystając z *sendto*() datagram wysyłany jest do serwera. Jeżeli wysłanie nie powiedzie się, oznacza to, że przekroczony został maksymalny rozmiar datagramu możliwy do wysłania w systemie operacyjnym. Następnie *recvfrom*() odpowiedź od serwera jest odbierana i wyświetlana.

```
int datagram_size;
for (datagram_size = 1; datagram_size <= BUFSIZE + 100; datagram_size += 100) {
    int ok = generate_datagram(datagram, datagram_size);

    if (ok == DATAGRAM_LEN_ERROR) {
        fprintf(stderr, "ERROR: Datagram not long enough.\n");
        printf("-------------------------------------------------\n");
        continue;
    };

    if (ok == OVERFLOW_ERROR) {
        fprintf(stderr, "ERROR: Cannot write %d on two bytes.\n", datagram_size);
        printf("-------------------------------------------------\n");
        break;;
    }
```

```
    printf("Sending datagram with length %d bytes to server %s:%d\n",
datagram_size, host, port);

    if (sendto(sock, datagram, datagram_size, 0, (struct sockaddr *)&server_addr,
sizeof(server_addr)) < 0) {
        fprintf(stderr, "Maximum OS datagram size exceeded. Declared datagram
length: %d bytes.", datagram_size);
        printf("-------------------------------------------------\n");
        printf("Looking for maximum OS datagram size...\n");
        printf("-------------------------------------------------\n");
        break;
    }

    socklen_t server_len = sizeof(server_addr);
    int received = recvfrom(sock, response, BUFSIZE, 0, (struct sockaddr
*)&server_addr, &server_len);
    if (received > 0) {
        response[received] = '\0';
        printf("Received response from server: \"%s\"\n", response);
    } else {
        perror("No response from server");
    }

    printf("-------------------------------------------------\n");
}
```

Gdy poprzednia pętla zostanie przerwana, ze względu na to, że zbyt duży datagram próbował być przesłany podejmowana jest próba znalezienia dokładnej wartości rozmiaru, który jest maksymalnym możliwym do przesłania. Dzieje się to na takiej samej zasadzie jak dla programu w Pythonie: w pętli zaczynając od wartości `datagram_size-1` zmniejszany jest rozmiar size i podejmowana jest próba wysłania datagramu o tej wielkości na serwer. Gdy datagram zostanie zaakkceptowany wyświetlany jest komunikat i pętla jest przerywana. Zwalniana jest wcześniej zaalokowana pamięć i zamykane jest gniazdo.

```
int size;
for (size = datagram_size - 1; size > 1; size--) {

    if (generate_datagram(datagram, size) == OVERFLOW_ERROR) {
        fprintf(stderr, "ERROR: Cannot write %d on two bytes.\n", datagram_size);
        printf("-------------------------------------------------\n");
        continue;;
    }

    printf("Trying to send %d bytes.\n", size);

    if (sendto(sock, datagram, size, 0, (struct sockaddr *)&server_addr,
sizeof(server_addr)) >= 0) {
        printf("Datagram with length %d bytes accepted\n", size);
        printf("-------------------------------------------------\n");
        break;
    } else {
        fprintf(stderr, "Maximum OS datagram size exceeded. Declared datagram
length: %d bytes.\n", size);
        printf("-------------------------------------------------\n");
    }
}

printf("\nFound maximum OS datagram size: %d bytes\n",  size);
```

```
free(datagram);
free(response);
close(sock);
return 0;
```

Funkcja genetate_datagram() generuje datagram o podawanej długości. Dwa pierwsze bity to długość datagramu a pozostała część to kolejne znaki od A do Z.

```
int generate_datagram(char *datagram, int length) {
    if (length < 2) return DATAGRAM_LEN_ERROR;

    if (length > BUFSIZE) return OVERFLOW_ERROR;


    datagram[0] = (length >> 8) & 0xFF;
    datagram[1] = length & 0xFF;

    const char letters[] = "ABCDEFGHIJKLMNOPRSTUWXYZ";
    uint letters_len = sizeof(letters) - 1;
    int i;
    for (i = 2; i < length; i++) {
        datagram[i] = letters[(i - 2) % letters_len];
    }
    return 0;
}
```

3. Konfiguracja testowa

3.1 Python - Serwer

Adres IP: 172.21.33.21
Port: 8000

3.2 C - Serwer

Adres IP: 172.21.33.11
Port: 8000

4. Testowanie

4.1 Serwer Python ←→ Klient Python

Serwer:

```
UDP server up and listening on 172.21.33.21:8080

--------------------------------------------------
Recieved 101 bytes from client ('172.21.33.2', 46344)
Sending response to client ('172.21.33.2', 46344): "OK: Recieved message with length 99 bytes"
--------------------------------------------------
Recieved 201 bytes from client ('172.21.33.2', 46344)
Sending response to client ('172.21.33.2', 46344): "OK: Recieved message with length 199 bytes"
--------------------------------------------------
Recieved 301 bytes from client ('172.21.33.2', 46344)
Sending response to client ('172.21.33.2', 46344): "OK: Recieved message with length 299 bytes"
--------------------------------------------------
Recieved 401 bytes from client ('172.21.33.2', 46344)
Sending response to client ('172.21.33.2', 46344): "OK: Recieved message with length 399 bytes"
--------------------------------------------------
Recieved 501 bytes from client ('172.21.33.2', 46344)
Sending response to client ('172.21.33.2', 46344): "OK: Recieved message with length 499 bytes"
--------------------------------------------------
Recieved 512 bytes from client ('172.21.33.2', 46344)
Sending response to client ('172.21.33.2', 46344): "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 601 bytes"
--------------------------------------------------
```

```
------------------------------------------------
Recieved 512 bytes from client ('172.21.33.2', 46344)
Sending response to client ('172.21.33.2', 46344): "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65401 bytes"
------------------------------------------------
Recieved 512 bytes from client ('172.21.33.2', 46344)
Sending response to client ('172.21.33.2', 46344): "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65501 bytes"
------------------------------------------------
Recieved 512 bytes from client ('172.21.33.2', 46344)
Sending response to client ('172.21.33.2', 46344): "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65507 bytes"
------------------------------------------------
```

Klient:

```
------------------------------------------------
ERROR: Datagram not long enough
------------------------------------------------
Sending datagram with length 101 bytes to server 172.21.33.21:8080.
Received response from server 172.21.33.21:8080: "OK: Recieved message with length 99 bytes".
------------------------------------------------
Sending datagram with length 201 bytes to server 172.21.33.21:8080.
Received response from server 172.21.33.21:8080: "OK: Recieved message with length 199 bytes".
------------------------------------------------
Sending datagram with length 301 bytes to server 172.21.33.21:8080.
Received response from server 172.21.33.21:8080: "OK: Recieved message with length 299 bytes".
------------------------------------------------
Sending datagram with length 401 bytes to server 172.21.33.21:8080.
Received response from server 172.21.33.21:8080: "OK: Recieved message with length 399 bytes".
------------------------------------------------
Sending datagram with length 501 bytes to server 172.21.33.21:8080.
Received response from server 172.21.33.21:8080: "OK: Recieved message with length 499 bytes".
------------------------------------------------
Sending datagram with length 601 bytes to server 172.21.33.21:8080.
Received response from server 172.21.33.21:8080: "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 601 bytes".
------------------------------------------------
```

```
------------------------------------------------
Sending datagram with length 65501 bytes to server 172.21.33.21:8080.
Received response from server 172.21.33.21:8080: "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65501 bytes".
------------------------------------------------
ERROR: Cannot write 65601 on two bytes
------------------------------------------------
```

```
------------------------------------------------
ERROR: Cannot write 65536 on two bytes
------------------------------------------------
Trying to send 65535 bytes.
ERROR: Maximum OS datagram size exceeded. Declared datagram length: 65535 bytes
------------------------------------------------
```

```
------------------------------------------------
Trying to send 65508 bytes.
ERROR: Maximum OS datagram size exceeded. Declared datagram length: 65508 bytes
------------------------------------------------
Trying to send 65507 bytes.
Datagram with length 65507 bytes accepted
------------------------------------------------

Found maximum OS datagram size: 65507 bytes.
```

4.2 Serwer Python ←→ Klient C

Serwer:

```
UDP server up and listening on 172.21.33.21:8080

------------------------------------------------
Recieved 101 bytes from client ('172.21.33.2', 32912)
Sending response to client ('172.21.33.2', 32912): "OK: Recieved message with length 99 bytes"
------------------------------------------------
Recieved 201 bytes from client ('172.21.33.2', 32912)
Sending response to client ('172.21.33.2', 32912): "OK: Recieved message with length 199 bytes"
------------------------------------------------
Recieved 301 bytes from client ('172.21.33.2', 32912)
Sending response to client ('172.21.33.2', 32912): "OK: Recieved message with length 299 bytes"
------------------------------------------------
Recieved 401 bytes from client ('172.21.33.2', 32912)
Sending response to client ('172.21.33.2', 32912): "OK: Recieved message with length 399 bytes"
------------------------------------------------
Recieved 501 bytes from client ('172.21.33.2', 32912)
Sending response to client ('172.21.33.2', 32912): "OK: Recieved message with length 499 bytes"
------------------------------------------------
Recieved 512 bytes from client ('172.21.33.2', 32912)
Sending response to client ('172.21.33.2', 32912): "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 601 bytes"
------------------------------------------------
```

.
.
.

```
------------------------------------------------
Recieved 512 bytes from client ('172.21.33.2', 32912)
Sending response to client ('172.21.33.2', 32912): "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65401 bytes"
------------------------------------------------
Recieved 512 bytes from client ('172.21.33.2', 32912)
Sending response to client ('172.21.33.2', 32912): "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65501 bytes"
------------------------------------------------
Recieved 512 bytes from client ('172.21.33.2', 32912)
Sending response to client ('172.21.33.2', 32912): "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65507 bytes"
------------------------------------------------
```

Klient:

```
------------------------------------------------
ERROR: Datagram not long enough.
------------------------------------------------
Sending datagram with length 101 bytes to server 172.21.33.21:8080
Received response from server: "OK: Recieved message with length 99 bytes"
------------------------------------------------
Sending datagram with length 201 bytes to server 172.21.33.21:8080
Received response from server: "OK: Recieved message with length 199 bytes"
------------------------------------------------
Sending datagram with length 301 bytes to server 172.21.33.21:8080
Received response from server: "OK: Recieved message with length 299 bytes"
------------------------------------------------
Sending datagram with length 401 bytes to server 172.21.33.21:8080
Received response from server: "OK: Recieved message with length 399 bytes"
------------------------------------------------
Sending datagram with length 501 bytes to server 172.21.33.21:8080
Received response from server: "OK: Recieved message with length 499 bytes"
------------------------------------------------
Sending datagram with length 601 bytes to server 172.21.33.21:8080
Received response from server: "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 601 bytes"
------------------------------------------------
```

.
.
.

```
------------------------------------------------
Sending datagram with length 65501 bytes to server 172.21.33.21:8080
Received response from server: "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65501 bytes"
------------------------------------------------
ERROR: Cannot write 65601 on two bytes.
------------------------------------------------
```

.
.
.

```
--------------------------------------------------
ERROR: Cannot write 65601 on two bytes.
--------------------------------------------------
Trying to send 65535 bytes.
Maximum OS datagram size exceeded. Declared datagram length: 65535 bytes.
--------------------------------------------------
```

.
.
.

```
--------------------------------------------------
Trying to send 65508 bytes.
Maximum OS datagram size exceeded. Declared datagram length: 65508 bytes.
--------------------------------------------------
Trying to send 65507 bytes.
Datagram with length 65507 bytes accepted
--------------------------------------------------

Found maximum OS datagram size: 65507 bytes
```

4.3 Serwer C ←→ Klient C

Serwer:

```
UDP server listening on 172.21.33.11:8080
--------------------------------------
Received 101 bytes from client 172.21.33.2:53003
Sending response to client: OK: Received message with length 99 bytes.
--------------------------------------
Received 201 bytes from client 172.21.33.2:53003
Sending response to client: OK: Received message with length 199 bytes.
--------------------------------------
Received 301 bytes from client 172.21.33.2:53003
Sending response to client: OK: Received message with length 299 bytes.
--------------------------------------
Received 401 bytes from client 172.21.33.2:53003
Sending response to client: OK: Received message with length 399 bytes.
--------------------------------------
Received 501 bytes from client 172.21.33.2:53003
Sending response to client: OK: Received message with length 499 bytes.
--------------------------------------
Received 512 bytes from client 172.21.33.2:53003
Sending response to client: ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 601 bytes.
--------------------------------------
```

.
.
.

```
--------------------------------------
Received 512 bytes from client 172.21.33.2:53003
Sending response to client: ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65401 bytes.
--------------------------------------
Received 512 bytes from client 172.21.33.2:53003
Sending response to client: ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65501 bytes.
--------------------------------------
Received 512 bytes from client 172.21.33.2:53003
Sending response to client: ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65507 bytes.
--------------------------------------
```

Klient:

```
------------------------------------------------
ERROR: Datagram not long enough.
------------------------------------------------
Sending datagram with length 101 bytes to server 172.21.33.11:8080
Received response from server: "OK: Received message with length 99 bytes."
------------------------------------------------
Sending datagram with length 201 bytes to server 172.21.33.11:8080
Received response from server: "OK: Received message with length 199 bytes."
------------------------------------------------
Sending datagram with length 301 bytes to server 172.21.33.11:8080
Received response from server: "OK: Received message with length 299 bytes."
------------------------------------------------
Sending datagram with length 401 bytes to server 172.21.33.11:8080
Received response from server: "OK: Received message with length 399 bytes."
------------------------------------------------
Sending datagram with length 501 bytes to server 172.21.33.11:8080
Received response from server: "OK: Received message with length 499 bytes."
------------------------------------------------
Sending datagram with length 601 bytes to server 172.21.33.11:8080
Received response from server: "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 601 bytes."
------------------------------------------------
```

.
.
.

```
------------------------------------------------
Sending datagram with length 65501 bytes to server 172.21.33.11:8080
Received response from server: "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65501 bytes."
------------------------------------------------
ERROR: Cannot write 65601 on two bytes.
------------------------------------------------
```

.
.
.

```
------------------------------------------------
ERROR: Cannot write 65601 on two bytes.
------------------------------------------------
Trying to send 65535 bytes.
Maximum OS datagram size exceeded. Declared datagram length: 65535 bytes.
------------------------------------------------
```

.
.
.

```
------------------------------------------------
Trying to send 65508 bytes.
Maximum OS datagram size exceeded. Declared datagram length: 65508 bytes.
------------------------------------------------
Trying to send 65507 bytes.
Datagram with length 65507 bytes accepted
------------------------------------------------

Found maximum OS datagram size: 65507 bytes
```

4.4 Serwer C ←→ Klient Python

Serwer:

```
UDP server listening on 172.21.33.11:8080
----------------------------------------
Received 101 bytes from client 172.21.33.2:37878
Sending response to client: OK: Received message with length 99 bytes.
----------------------------------------
Received 201 bytes from client 172.21.33.2:37878
Sending response to client: OK: Received message with length 199 bytes.
----------------------------------------
Received 301 bytes from client 172.21.33.2:37878
Sending response to client: OK: Received message with length 299 bytes.
----------------------------------------
Received 401 bytes from client 172.21.33.2:37878
Sending response to client: OK: Received message with length 399 bytes.
----------------------------------------
Received 501 bytes from client 172.21.33.2:37878
Sending response to client: OK: Received message with length 499 bytes.
----------------------------------------
Received 512 bytes from client 172.21.33.2:37878
Sending response to client: ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 601 bytes.
----------------------------------------
```

.
.
.

```
----------------------------------------
Received 512 bytes from client 172.21.33.2:37878
Sending response to client: ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65401 bytes.
----------------------------------------
Received 512 bytes from client 172.21.33.2:37878
Sending response to client: ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65501 bytes.
----------------------------------------
Received 512 bytes from client 172.21.33.2:37878
Sending response to client: ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65507 bytes.
----------------------------------------
```

Klient:

```
----------------------------------------------
ERROR: Datagram not long enough
----------------------------------------------
Sending datagram with length 101 bytes to server 172.21.33.11:8080.
Received response from server 172.21.33.11:8080: "OK: Received message with length 99 bytes.".
----------------------------------------------
Sending datagram with length 201 bytes to server 172.21.33.11:8080.
Received response from server 172.21.33.11:8080: "OK: Received message with length 199 bytes.".
----------------------------------------------
Sending datagram with length 301 bytes to server 172.21.33.11:8080.
Received response from server 172.21.33.11:8080: "OK: Received message with length 299 bytes.".
----------------------------------------------
Sending datagram with length 401 bytes to server 172.21.33.11:8080.
Received response from server 172.21.33.11:8080: "OK: Received message with length 399 bytes.".
----------------------------------------------
Sending datagram with length 501 bytes to server 172.21.33.11:8080.
Received response from server 172.21.33.11:8080: "OK: Received message with length 499 bytes.".
----------------------------------------------
Sending datagram with length 601 bytes to server 172.21.33.11:8080.
Received response from server 172.21.33.11:8080: "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 601 bytes.".
----------------------------------------------
```

.
.
.

```
Sending datagram with length 65501 bytes to server 172.21.33.11:8080.
Received response from server 172.21.33.11:8080: "ERROR: Buffer size (512 bytes) exceeded. Declared datagram length: 65501 bytes.".
----------------------------------------------
ERROR: Cannot write 65601 on two bytes
----------------------------------------------
```

.
.
.

```
--------------------------------------------------
ERROR: Cannot write 65536 on two bytes
--------------------------------------------------
Trying to send 65535 bytes.
ERROR: Maximum OS datagram size exceeded. Declared datagram length: 65535 bytes
--------------------------------------------------
```

.
.
.

```
------------------------------------------------
Trying to send 65508 bytes.
ERROR: Maximum OS datagram size exceeded. Declared datagram length: 65508 bytes
------------------------------------------------
Trying to send 65507 bytes.
Datagram with length 65507 bytes accepted
------------------------------------------------

Found maximum OS datagram size: 65507 bytes.
```

5. Wnioski

Wartość maksymalnego rozmiaru przyjętego datagramu wynosi 65507 bajtów. W nagłówku IP pole Total Length ma wartość 16, czyli rozmiar 16 bitów, co oznacza, że mogą być tam przechowywane wartości od 0 do 65535 (2^{16}-1). Wielkość nagłówka IP wynosi 20B, a nagłówka UDP 8B.
Nasz wynik to oczekiwana wielkość, ponieważ 65535 – (20 + 8) = **65507**.