Systemy rozproszone

Laboratorium 1

Gniazda TCP/UDP

Albert Gierlach

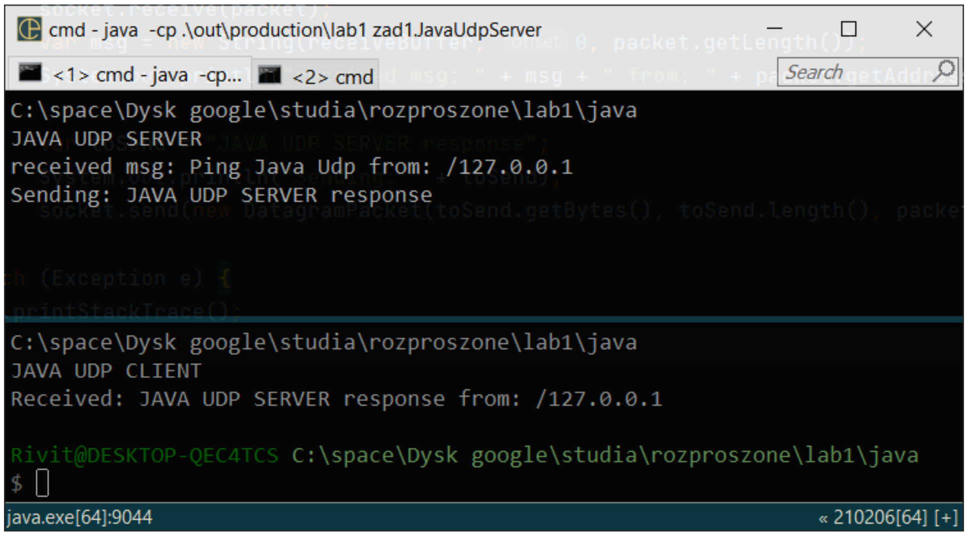
05.03.2021

1. Zadanie 1

```java
1  public class JavaUdpServer {
2      public static void main(String[] args) {
3          System.out.println("JAVA UDP SERVER");
4
5          int portNumber = 9008;
6          try (DatagramSocket socket = new DatagramSocket(portNumber)) {
7              var receiveBuffer = new byte[1024];
8
9              while (true) {
10                 Arrays.fill(receiveBuffer, (byte) 0);
11                 var packet = new DatagramPacket(receiveBuffer, receiveBuffer.length);
12                 socket.receive(packet);
13                 var msg = new String(receiveBuffer, 0, packet.getLength());
14                 System.out.println("received msg: " + msg + " from: " + packet.getAddress());
15
16                 var toSend = "JAVA UDP SERVER response";
17                 System.out.println("Sending: " + toSend);
18                 socket.send(new DatagramPacket(toSend.getBytes(), toSend.length(), packet.getAddress(), packet.getPort()));
19             }
20         } catch (Exception e) {
21             e.printStackTrace();
22         }
23     }
24 }
```

```java
1  public class JavaUdpClient {
2      public static void main(String[] args) {
3          System.out.println("JAVA UDP CLIENT");
4
5          int portNumber = 9008;
6          try (DatagramSocket socket = new DatagramSocket()) {
7              var address = InetAddress.getByName("localhost");
8              var sendBuffer = "Ping Java Udp".getBytes();
9
10             var sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length, address, portNumber);
11             socket.send(sendPacket);
12
13             var receiveBuffer = new byte[1024];
14             var receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
15             socket.receive(receivePacket);
16             var receivedString = new String(receiveBuffer, 0, receivePacket.getLength());
17             System.out.println("Received: " + receivedString + " from: " + receivePacket.getAddress());
18         } catch (Exception e) {
19             e.printStackTrace();
20         }
21     }
22 }
```

W klasie *JavaUdpServer* została dodana obsługa pobierania adresu nadawcy (metoda *.getAddress()*) oraz wysyłanie odpowiedzi poprzez metodę *.send()* na odpowiedni adres i port (linie 14-18). Klasa klienta została rozszerzona o odbieranie wiadomości od serwera oraz jej wypisanie (linie 13-17).



```
C:\space\Dysk google\studia\rozproszone\lab1\java
JAVA UDP SERVER
received msg: Ping Java Udp from: /127.0.0.1
Sending: JAVA UDP SERVER response
```

```
C:\space\Dysk google\studia\rozproszone\lab1\java
JAVA UDP CLIENT
Received: JAVA UDP SERVER response from: /127.0.0.1

Rivit@DESKTOP-QEC4TCS C:\space\Dysk google\studia\rozproszone\lab1\java
$
```

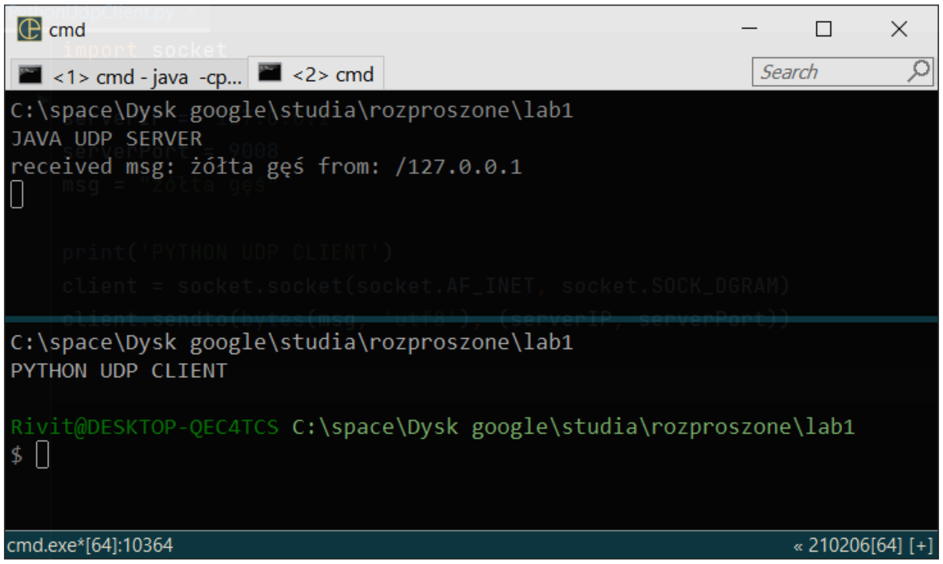
Rysunek 1: Wynik działania programów

2. Zadanie 2

```java
public class JavaUdpServer {
    public static void main(String[] args) {
        System.out.println("JAVA UDP SERVER");

        int portNumber = 9008;
        try (DatagramSocket socket = new DatagramSocket(portNumber)) {
            byte[] receiveBuffer = new byte[1024];

            while(true){
                Arrays.fill(receiveBuffer, (byte) 0);
                var receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);

                socket.receive(receivePacket);
                var msg = new String(receiveBuffer, 0, receivePacket.getLength());
                System.out.println("received msg: " + msg + " from: " + receivePacket.getAddress());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```python
1  serverIP = "127.0.0.1"
2  serverPort = 9008
3  msg = "żółta gęś"
4
5  print('PYTHON UDP CLIENT')
6  client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
7  client.sendto(bytes(msg, 'utf8'), (serverIP, serverPort))
```

Jedyna zmiana jaka została wprowadzona dotyczy programu napisanego w Pythonie. Kodowanie znaków zostało zmienione na UTF-8, aby wiadomość została poprawnie zdekodowana po stronie serwera (linia 7).



```
C:\space\Dysk google\studia\rozproszone\lab1
JAVA UDP SERVER
received msg: żółta gęś from: /127.0.0.1
```

```
C:\space\Dysk google\studia\rozproszone\lab1
PYTHON UDP CLIENT

Rivit@DESKTOP-QEC4TCS C:\space\Dysk google\studia\rozproszone\lab1
$
```

```
cmd.exe*[64]:10364                                         « 210206[64] [+]
```

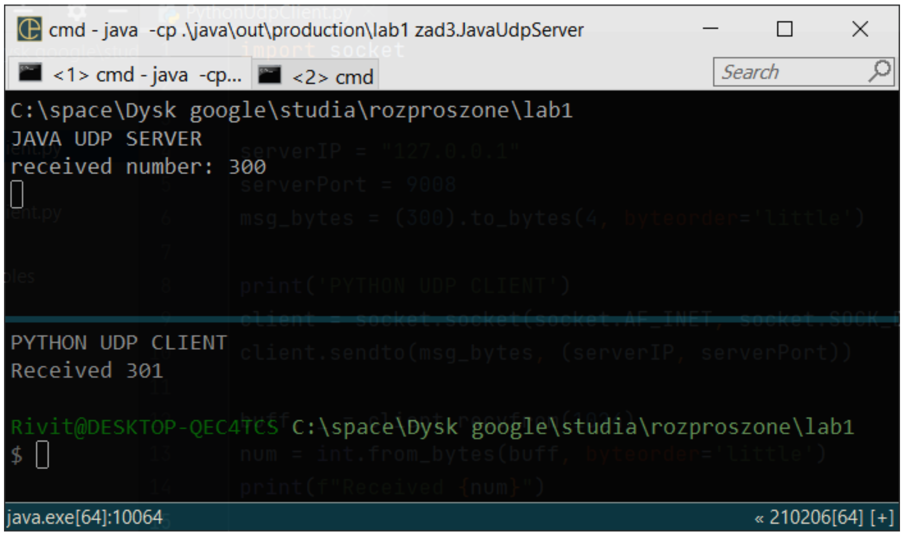
Rysunek 2: Wynik działania programów

3. Zadanie 3

```java
public class JavaUdpServer {
    public static void main(String[] args) {
        System.out.println("JAVA UDP SERVER");

        int portNumber = 9008;
        try (DatagramSocket socket = new DatagramSocket(portNumber)) {
            byte[] receiveBuffer = new byte[1024];

            while(true){
                Arrays.fill(receiveBuffer, (byte) 0);
                var receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
                socket.receive(receivePacket);

                var num = ByteBuffer.wrap(receiveBuffer).order(ByteOrder.LITTLE_ENDIAN).getInt();
                System.out.println("received number: " + num++);

                var buff = ByteBuffer.allocate(4).order(ByteOrder.LITTLE_ENDIAN).putInt(num).array();
                socket.send(new DatagramPacket(buff, buff.length, receivePacket.getAddress(), receivePacket.getPort()));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```python
serverIP = "127.0.0.1"
serverPort = 9008
msg_bytes = (300).to_bytes(4, byteorder='little')

print('PYTHON UDP CLIENT')
client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
client.sendto(msg_bytes, (serverIP, serverPort))

buff, _ = client.recvfrom(1024)
num = int.from_bytes(buff, byteorder='little')
print(f"Received {num}")
```

Klasa *JavaUdpServer* została rozszerzona o poprawną obsługę zamiany kolejności bajtów (metoda *.order()*), które są odczytywane z pakietu datagramowego. Następnie zwiększamy otrzymaną liczbę, zamieniamy ją na format little endian i odsyłamy do klienta (linie 14-18). Kod klienta sprowadza się do zakodowania liczby do postaci little endian, wysłania jej oraz oczekiwania na odpowiedź od serwera.



```
C:\space\Dysk google\studia\rozproszone\lab1
JAVA UDP SERVER
received number: 300

PYTHON UDP CLIENT
Received 301

Rivit@DESKTOP-QEC4TCS C:\space\Dysk google\studia\rozproszone\lab1
$
```

Rysunek 3: Wynik działania programów

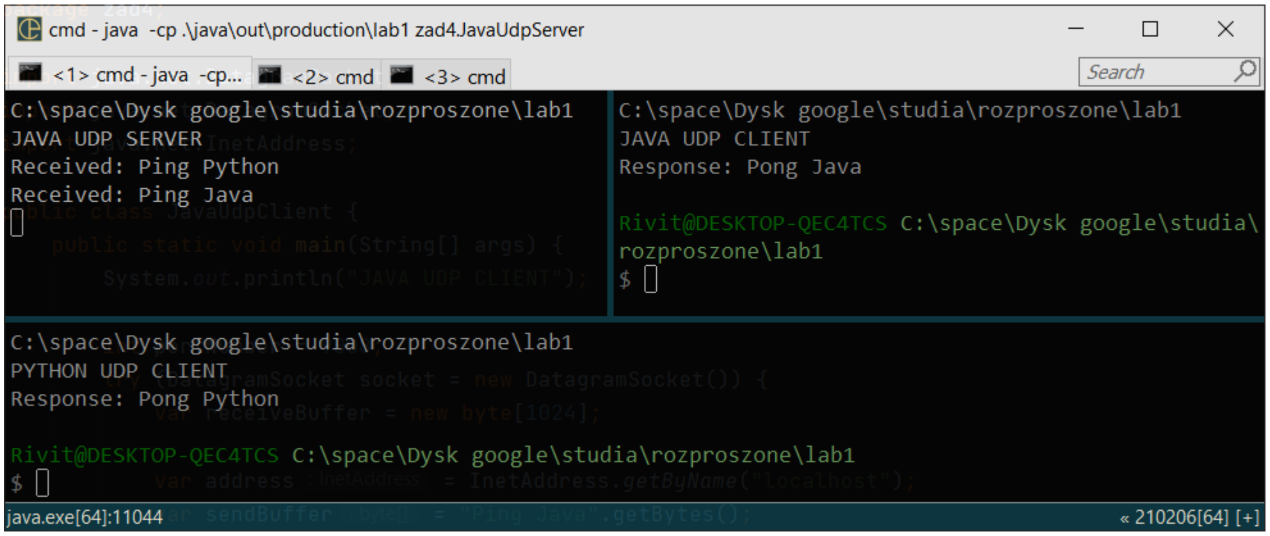
4. Zadanie 4

```java
public class JavaUdpServer {
    public static void main(String[] args) {
        System.out.println("JAVA UDP SERVER");

        int portNumber = 9008;
        try (DatagramSocket socket = new DatagramSocket(portNumber)) {
            byte[] receiveBuffer = new byte[1024];

            while(true){
                Arrays.fill(receiveBuffer, (byte) 0);
                var receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
                socket.receive(receivePacket);

                var msg = new String(receiveBuffer, 0, receivePacket.getLength());
                System.out.println("Received: " + msg);
                var response = "";
                if(msg.toLowerCase().contains("python")){
                    response = "Pong Python";
                }else if(msg.toLowerCase().contains("java")){
                    response = "Pong Java";
                }

                socket.send(new DatagramPacket(response.getBytes(), response.length(), receivePacket.getAddress(), receivePacket.
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```java
public class JavaUdpClient {
    public static void main(String[] args) {
        System.out.println("JAVA UDP CLIENT");

        int portNumber = 9008;
        try (DatagramSocket socket = new DatagramSocket()) {
            var receiveBuffer = new byte[1024];

            var address = InetAddress.getByName("localhost");
            var sendBuffer = "Ping Java".getBytes();
            var sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length, address, portNumber);
            socket.send(sendPacket);

            var receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
            socket.receive(receivePacket);
            var receivedString = new String(receiveBuffer, 0, receivePacket.getLength());
            System.out.println("Response: " + receivedString);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```python
serverIP = "127.0.0.1"
serverPort = 9008
msg = "Ping Python"

print('PYTHON UDP CLIENT')
client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
client.sendto(bytes(msg, 'utf8'), (serverIP, serverPort))

buff, _ = client.recvfrom(1024)
print(f"Response: {buff.decode()}")
```

Server rozpoznaje typ klienta od którego dostał wiadomość poprzez sprawdzenie treści wiadomości (linie 14-23). Jeśli treść wiadomości zawiera słowo "python" to oznacza, że wiadomość przyszła od klienta napisanego w języku Python - analogicznie z językiem Java. Klienci czekają na odpowiedź serwera, a następnie ją wypisują.



```
C:\space\Dysk google\studia\rozproszone\lab1
JAVA UDP SERVER
Received: Ping Python
Received: Ping Java
```

```
C:\space\Dysk google\studia\rozproszone\lab1
JAVA UDP CLIENT
Response: Pong Java

Rivit@DESKTOP-QEC4TCS C:\space\Dysk google\studia\rozproszone\lab1
$
```

```
C:\space\Dysk google\studia\rozproszone\lab1
PYTHON UDP CLIENT
Response: Pong Python

Rivit@DESKTOP-QEC4TCS C:\space\Dysk google\studia\rozproszone\lab1
$
```

Rysunek 4: Wynik działania programów