AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Gniazda TCP/UDP

Robert Straś

stras@agh.edu.pl

Wstęp

- UPEL (upel.agh.edu.pl/wiet)
  - Kurs: Systemy Rozproszone
  - Klucz: distributed

- Pokój wideokonferencyjny do odbioru zadań domowych, konsultacji i ewentualnie na wypadek konieczności przejścia na zajęcia zdalne:

  https://mche.webex.com/join/r.stras

Regulamin przedmiotu

- Pełny regulamin dostępny na UPEL

- Organizacja
  - 7 tematów zajęć
  - Każdy temat to laboratorium + zadanie
  - Harmonogram:

  L1 - L2 - Z1 - Z2

  L3 - L4 - Z3 - Z4

  L5 - L6 - L7 - Z5 - Z6 - Z7.

Regulamin przedmiotu

- Ćwiczenia
  - 0–5 pkt. za uczestnictwo (1 pkt. za obecność)
  - 0-10 pkt. zadania domowe (0-15 pkt. za ostatnie zadanie)
  - Dodatkowa aktywność (forum) do 10 pkt.
  - Ocena wg skali ocen AGH
  - 100% to 100 punktów
  - (max 110 pkt.)
- Zadania
  - muszą być zamieszczone w systemie UPEL do godz. 10:00 dnia, w którym pierwsza z grup prezentuje zadanie (t.j. poniedziałek godz. 12:50).

Regulamin przedmiotu

- Warunki zaliczenia przedmiotu:
  - Co najmniej 14 pkt. *łącznie* za uczestnictwo w zajęciach
  - Co najmniej 3 pkt. za *każde* zadanie domowe
  - Co najmniej 50 pkt. *łącznie*

- Egzamin

Regulamin przedmiotu

- Pozostałe informacje:
  - Wymagania odnośnie zadań
  - Ponowne oddawanie zadań
  - Ocena końcowa

Plan zajęć

- Przypomnienie wiadomości (TCP/UDP)
- Programowanie gniazd w języku Java
- Programowanie gniazd w języku Python
- Zadania na ćwiczeniach
- Zadanie domowe

Przypomnienie wiadomości

- TCP
- UDP
- Gniazda

TCP/UDP

- TCP
  - Połączeniowy
  - Punkt-punkt
  - Niezawodny
  - Kontrola przepływu danych
  - Strumieniowy
  - Relatywnie wolny (duży narzut)

TCP/UDP

- UDP
  - Bezpołączeniowy
  - Możliwy multicast
  - Zawodny
  - Brak kontroli przepływu
  - Datagramy
  - Szybki (mały narzut)

TCP/UDP

- Zastosowania TCP?
- Zastosowania UDP?
- Porty?

Gniazda

- Asocjacja
  - (protokół, adres lokalny, proces lokalny, adres obcy, proces obcy)

- Gniazdo
  - (protokół, adres lokalny, proces lokalny)

  lub
  - (protokół, adres obcy, proces obcy)

- TCP/UDP - identyfikacja procesów przez numer portu

Porty

- Dobrze znane porty – zarezerwowane, np.:
  - 21 FTP
  - 23 Telnet
  - 25 SMTP
  - 80 HTTP
  - …
- Porty efemeryczne – krótkotrwałe, przydzielane na czas połączenia

Polecenie netstat

- Windows oraz Unix
- Wyświetla używane gniazda
- Dodatkowe opcje (Windows):
  -p protokół (np. tcp/udp)
  -a wszystkie gniazda (domyślnie tylko połączone)
  -b nazwy procesów

Dane liczbowe - kolejność bajtów

- Kolejność bajtów w pamięci (*byte order / endianness)*
  - **Big endian** (Motorola, Sparc, **sieć**)
  - **Little endian** (Alpha, Intel)
  - Przykład:

  1 507 634 416 (dec) = 59DC ACF0 (hex)

Big endian

59	DC	AC	F0

Little endian

F0	AC	DC	59

Dane liczbowe – rozmiar typów

- Ile bajtów ma typ Integer?

PROGRAMOWANIE GNIAZD W JĘZYKU JAVA

Serwer TCP (Java)

```
ServerSocket serverSocket = null;
try {
    serverSocket = new ServerSocket(12345);
    while(true){

        Socket clientSocket = serverSocket.accept();

        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
        BufferedReader in = new BufferedReader(new
                InputStreamReader(clientSocket.getInputStream()));

        String msg = in.readLine();
        System.out.println("received msg: " + msg);
        out.println("Pong");
        clientSocket.close();
    }
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (serverSocket != null) serverSocket.close();
}
```

Klient TCP (Java)

```
Socket socket = null;
try {
    socket = new Socket(”localhost”, 12345);

    PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
    BufferedReader in = new BufferedReader(new
                    InputStreamReader(socket.getInputStream()));

    out.println("Ping");
    String response = in.readLine();
    System.out.println("received msg: " + response);
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (socket != null) socket.close();
}
```

Serwer UDP (Java)

```
DatagramSocket socket = null;
try {
    socket = new DatagramSocket(9876);
    byte[] receiveBuffer = new byte[1024];

    while(true) {
          DatagramPacket receivePacket =
              new DatagramPacket(receiveBuffer, receiveBuffer.length);
          socket.receive(receivePacket);

          String msg = new String(receivePacket.getData());
          System.out.println("received msg: " + msg);
    }
} catch (Exception e) {
    e.printStackTrace();
} finally {
    if (socket != null) socket.close();
}
```

Klient UDP (Java)

```
DatagramSocket socket = null;
try {
    socket = new DatagramSocket();

    InetAddress address = InetAddress.getByName("localhost");
    byte[] sendBuffer = "Ping".getBytes();

    DatagramPacket sendPacket =
          new DatagramPacket(sendBuffer, sendBuffer.length, address, 9876);
    socket.send(sendPacket);

} catch (Exception e) {
    e.printStackTrace();
} finally {
    if (socket != null) socket.close();
}
```

Java TCP, UDP

- Proszę uruchomić przykłady:
  - JavaTcpServer + JavaTcpClient
  - JavaUdpServer + JavaUdpClient
- Proszę skorzystać z polecenia netstat, aby zaobserwować otwarte porty

PROGRAMOWANIE GNIAZD W JĘZYKU PYTHON

Serwer UDP (Python)

```
import socket;

serverPort = 9009

serverSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
serverSocket.bind(('', serverPort))

buff = []

while True:
    buff, address = serverSocket.recvfrom(1024)
    print("received msg: " + str(buff, 'utf-8'))
```

Klient UDP (Python)

```
import socket;

serverIP = "127.0.0.1"
serverPort = 9009
msg = "Ping Python Udp!"

client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
client.sendto(bytes(msg, 'utf-8'), (serverIP, serverPort))
```

Python UDP

- Proszę uruchomić przykład:
  - PythonUdpServer + PythonUdpClient

Zadania - UDP

- Komunikacja dwukierunkowa
- Komunikacja pomiędzy różnymi językami
- Przesył napisów – kodowanie
- Przesył liczb – kolejność bajtów
- Projektowanie protokołu

Zadanie 1 (1 pkt)

- Zaimplementować dwukierunkową komunikację przez UDP Java-Java
  - Klient wysyła wiadomość i odczytuje odpowiedź
  - Serwer otrzymuje wiadomość i wysyła odpowiedź
  - Należy pobrać adres nadawcy z otrzymanego datagramu

Zadanie 2 (1 pkt)

- Zaimplementować komunikację przez UDP pomiędzy językami Java i Python
  - JavaUdpServer + PythonUdpClient
  - Należy przesłać wiadomość tekstową:

  ‘żółta gęś’

  (uwaga na kodowanie)

Zadanie 3 (1 pkt)

- Zaimplementować przesył wartości liczbowej w przypadku JavaUdpServer + PythonUdpClient
  - Symulujemy komunikację z platformą o innej kolejności bajtów: klient Python ma wysłać następujący ciąg bajtów:

```
msg_bytes = (300).to_bytes(4, byteorder='little')
```

  - Server Javy ma wypisać otrzymaną liczbę oraz odesłać liczbę zwiększoną o jeden

Zadanie 3 wskazówki

– Zamiana bajty –> int –> bajty w Javie:

```
int nb = ByteBuffer.wrap(buff).getInt();

buff = ByteBuffer.allocate(4).putInt(nb).array();
```

– Zamiana bajty –> int w Pythonie:

```
int.from_bytes(buff, byteorder='little')
```

Zadanie 4 (1 pkt)

- Zaimplementować serwer (Java lub Python) który rozpoznaje czy otrzymał wiadomość od klienta Java czy od klienta Python i wysyła im różne odpowiedzi (np. ‘Pong Java’, ‘Pong Python’)

ZADANIE DOMOWE

Zadanie domowe - Chat

- Napisać aplikację typu chat (5 pkt.)
  - Klienci łączą się serwerem przez protokół TCP
  - Serwer przyjmuje wiadomości od każdego klienta i rozsyła je do pozostałych (wraz z id/nickiem klienta)
  - Serwer jest wielowątkowy – każde połączenie od klienta powinno mieć swój wątek
  - Proszę zwrócić uwagę na poprawną obsługę wątków

Zadanie domowe c.d.

- Dodać dodatkowy kanał UDP (3 pkt.)
  - Serwer oraz każdy klient otwierają dodatkowy kanał UDP (ten sam numer portu jak przy TCP)
  - Po wpisaniu komendy ‘U’ u klienta przesyłana jest wiadomość przez UDP na serwer, który rozsyła ją do pozostałych klientów
  - Wiadomość symuluje dane multimedialne (można np. wysłać ASCII Art)

Zadanie domowe c.d.

- Zaimplementować powyższy punkt w wersji multicast (2 pkt.)
  - Nie zamiast, tylko jako alternatywna opcja do wyboru (komenda ‘M’)
  - Multicast przesyła bezpośrednio do wszystkich przez adres grupowy (serwer może, ale nie musi odbierać)

Zadanie domowe - uwagi

- Zadanie można oddać w dowolnym języku programowania
- Nie wolno korzystać z frameworków do komunikacji sieciowej – tylko gniazda! Nie wolno też korzystać z Akka

Zadanie domowe - uwagi

- Przy oddawaniu należy:
  - zademonstrować działanie aplikacji (serwer + min. 2 klientów)
  - omówić kod źródłowy
- Proszę zwrócić uwagę na:
  - Wydajność rozwiązania (np. pula wątków)
  - Poprawność rozwiązania (np. unikanie wysyłania wiadomości do nadawcy, obsługa wątków)

AGH

Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Dziękuję