Gniazda sieciowe.

Ogólne informacje.

- Gniazda sieciowe:
  - programowy interfejs (API) protokołów TCP oraz UDP,
- Miejsce w modelu ISO/OSI:
  - 4. warstwa,
  - konsekwencje,
- Interfejs gniazd BSD:
  - Podobna koncepcja w różnych językach programowania: C/C++, Java, C# …
- Inne rozwiązania omawiane na ćwiczeniach bazują na gniazdach BSD,

Podstawowe pojęcia

- Asocjacja:
  - Piątka:
    - Protokół, adres nadawcy, port nadawcy, adres odbiorcy, port odbiorcy,
- Gniazdo:
  - Trójka:
    - Protokół, adres nadawcy, port nadawcy,
- Rodzaje portów:
  - Porty znane
    - FTP (20), SMTP (25), HTTP (80),
  - Porty efemeryczne.

Protokoły i ich cechy.

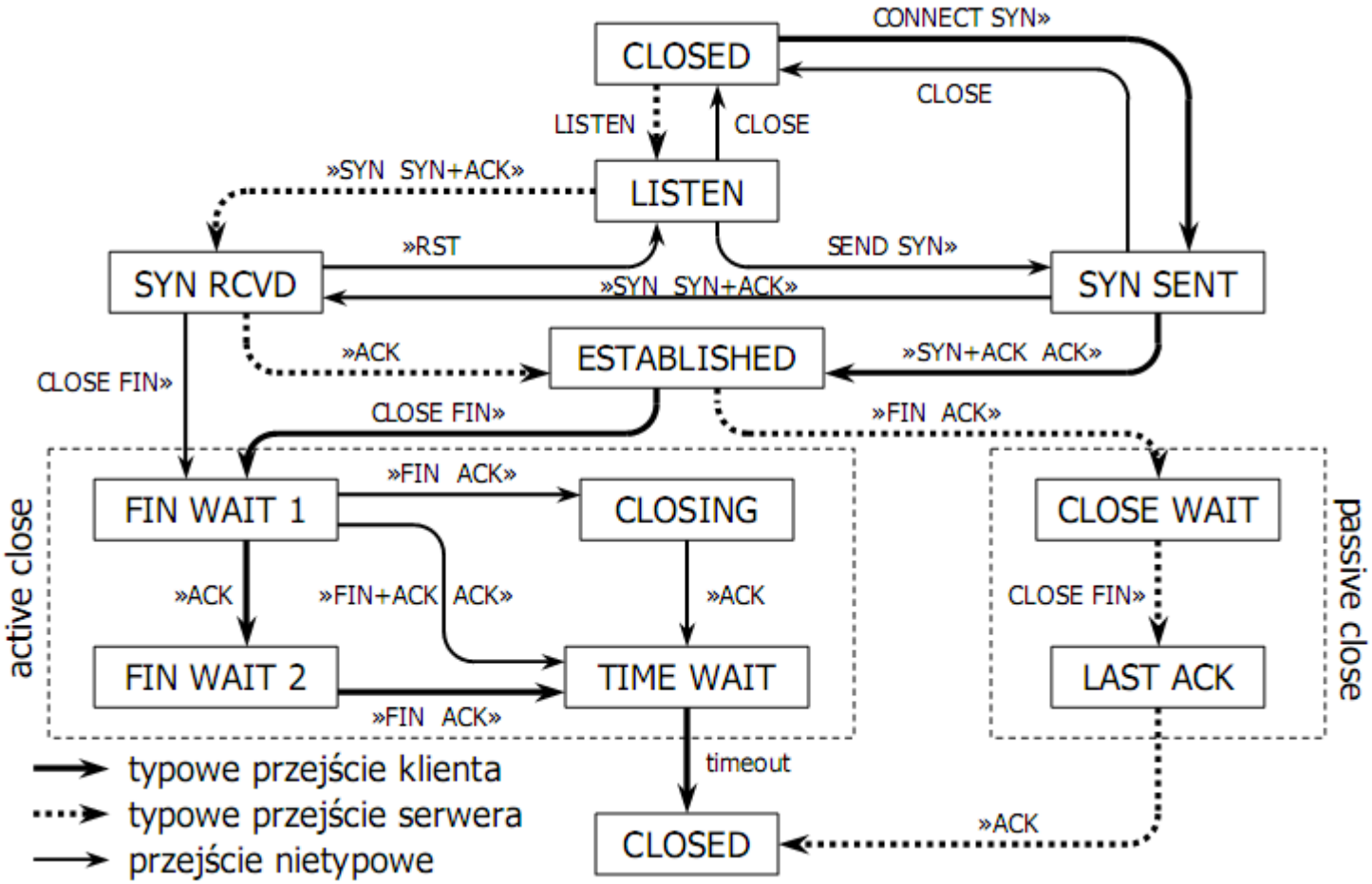
- Protokół sieciowy
  - Zbiór zasad nawiązywania łączności i wymiany danych między komputerami pracującymi w sieci.
- Podstawowe cechy
  - Połączeniowość
    - Modele komunikacji.
  - Rodzaj przesyłanych danych
    - Pakiety,
    - Komunikacja strumieniowa.
  - Niezawodność
    - Pewność dostarczenia,
    - Kolejność,
    - Duplikacja.

Protokół TCP

- Cechy
  - Połączeniowy
    - full-duplex,
    - dwa końce połączenia.
  - Strumieniowy
  - Niezawodny
  - Dodatkowo
    - Kontrola przepływu,
    - Kontrola przeciążenia,

Protokół TCP.

- Maszyna stanów:



Protokół UDP

- Cechy
  - Bardzo prosty,
  - Bezpołączeniowy,
  - Datagramowy,
  - Zawodny.

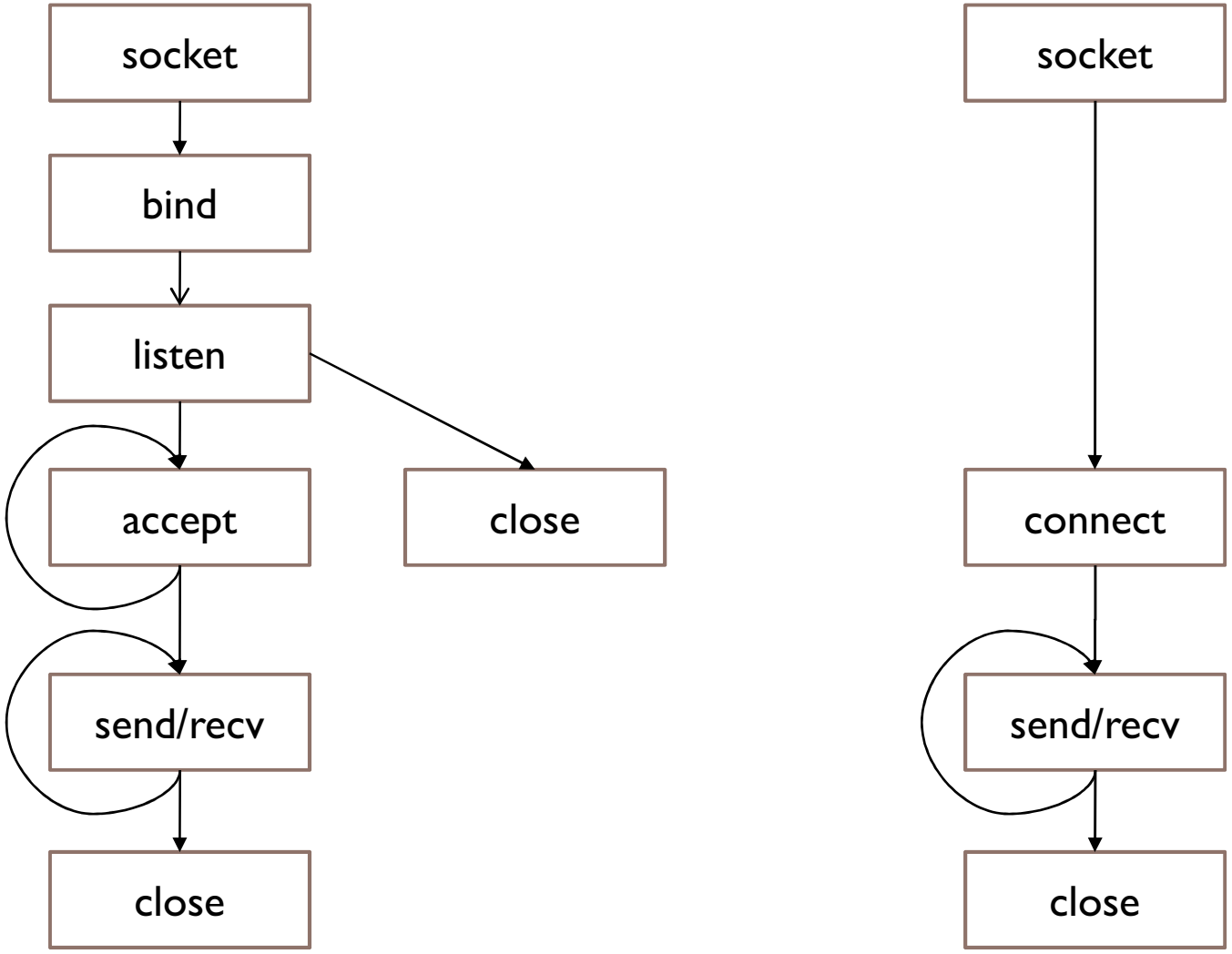
Programowanie TCP

Podstawowe elementy API. Standard BSD

- API gniazd BSD:

Element	Opis
socket	reprezentacja gniazda lub asocjacji,
bind	wiąże proces i adres z gniazdem,
listen	tworzy kolejkę zgłoszeń
accept	oczekuje na zgłoszenie
connect	ustanawia połączenie
send	wysłanie danych
recv	odebranie danych
close	zamyka połączenie

Schemat działania. Klient - serwer



Przykład C. Klient

```
 1:  struct sockaddr_in serv_addr;
 2:
 3:  int fd = socket(AF_INET, SOCK_STREAM, 0);
 4:
 5:  bzero((char*)&serv_addr,  sizeof(serv_addr);
 6:  serv_addr.sin_family = AF_INET;
 7:  serv_addr.sin_addr.s_addr = inet_addr(„10.1.2.3“);
 8:  serv_addr.sin_port = htons(12345);
 9:
10:  connect(fd, (struct sockaddr*) &serv_addr,  sizeof(serv_addr));
11:  send(fd, ...);
12:  ...
13:  close(fd)
14:
```

Przykład C. Serwer

```
 1:  struct sockaddr_in serv_addr, cli_addr;
 2:  int fd, cli_fd, cli_len;
 3:
 4:  fd = socket(AF_INET, SOCK_STREAM, 0);
 5:
 6:  bzero((char*)&serv_addr, sizeof(serv_addr));
 7:  serv_addr.sin_family = AF_INET;
 8:  serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
 9:  serv_addr.sin_port = htons(12345);
10:
11:  bind(fd, (struct sockaddr*)&serv_addr,  sizeof(serv_addr));
12:
13:  listen(fd, 5);
14:
15:  while (1)
16:     {
17:             cli_fd = accept(fd, (struct sockaddr*)&cli_addr, &cli_len);
18:             recv(cli_fd, ...);
19:             ...
20:             close(cli_fd);
21:     }
22:
23:  close(fd);
24:
```

Przykład Java. Klient.

```
 1:  Socket socket;
 2:
 3:  try {
 4:    socket = new Socket(host, port);
 5:
 6:    InputStreamReader inreader =
 7:              new InputStreamReader(socket.getInputStream());
 8:    BufferedReader in = new BufferedReader(inreader);
 9:
10:    PrintStream out = new PrintStream(socket.getOutputStream());
11:
12:    String line;
13:    line = in.readLine();
14:    out.println(line);
15:
16:  } catch (Exception e) {
17:     System.err.println("Error: " + e.getMessage());
18:  } finally {
19:      socket.close();
20:  }
21:
```

Przykład Java. Serwer.

```
1:  ServerSocket sock;
2:  String line;
3:  try {
4:    sock = new ServerSocket(port);
5:    Socket client;
6:
7:    while(true){
8:     client = sock.accept();
9:
10:    InputStreamReader inread = new InputStreamReader(client.getInputStream());
11:    BufferedReader in = new BufferedReader(inread);
12:    PrintStream out = new PrintStream(client.getOutputStream());
13:
14:    line = in.readLine();
15:    out.println(line);
16:
17:    client.close();
18:    }
19:   sock.close();
20: } catch (IOException e) {
21:    System.err.println("Network Error: " + e.getMessage());
22: }
```

Przykład C#. Klient

```
 1:  IPEndPoint serverAddress =
 2:     new IPEndPoint(IPAddress.Parse("127.0.0.1"),  1000);
 3:
 4:  Socket socket = new Socket(AddressFamily.InterNetwork,
 5:     SocketType.Stream, ProtocolType.Tcp);
 6:
 7:  socket.Connect(serverAddress);
 8:
 9:  StreamReader sr = new StreamReader(new NetworkStream(socket));
10:  StreamWriter sw = new StreamWriter(new NetworkStream(socket));
11:
12:  data = sr.ReadLine();
13:  sw.WriteLine(input);
14:  sw.Flush();
15:
16:  sr.Close();
17:  sw.Close();
18:  socket.Close();
```

Przykład C#. Serwer

```
 1:  IPEndPoint address = new IPEndPoint(IPAddress.Any,  1000);
 2:
 3:  Socket socket = new Socket(AddressFamily.InterNetwork,
 4:      SocketType.Stream, ProtocolType.Tcp);
 5:
 6:  socket.Bind(address);
 7:  socket.Listen(10);
 8:
 9:  while (true) {
10:     Socket client = socket.Accept();
11:
12:     data = new byte[1024];
13:     client.Receive(data);
14:
15:     client.Send(data, recv, SocketFlags.None);
16:
17:     client.Close();
18:  }
19:  socket.Close();
20:
```

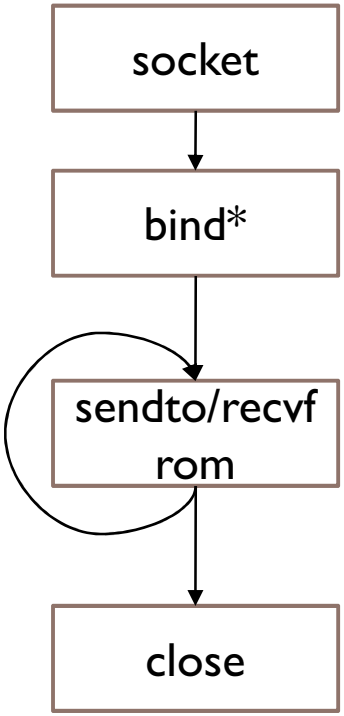
Programowanie UDP

Podstawowe elementy API.

- API gniazd BSD

Element	Opis
socket	reprezentacja gniazda lub asocjacji
bind	wiąże proces i adres z gniazdem
sendto	wysłanie informacji
recvfrom	odebranie danych
close	usuwa gniazdo

Schemat działania



Przykład C.

```
 1:  struct sockaddr_in serv_addr, cli_addr, sndr_addr;
 2:  int fd, sndr_len;
 3:  fd = socket(AF_INET, SOCK_DGRAM, 0);
 4:
 5:  bzero((char*)&serv_addr, sizeof(serv_addr));
 6:  serv_addr.sin_family = AF_INET;
 7:  serv_addr.sin_addr.s_addr = inet_addr("10.1.2.3");
 8:  serv_addr.sin_port = htons(12345);
 9:
10:  bzero((char*)&cli_addr, sizeof(cli_addr));
11:  cli_addr.sin_family = AF_INET;
12:  cli_addr.sin_addr.s_addr = inet_addr(INADDR_ANY);
13:  cli_addr.sin_port = htons(0);
14:
15:  bind(fd, (struct sockaddr*)&cli_addr, sizeof(cli_addr));
16:
17:  sendto(fd, ..., (struct sockaddr*)&serv_addr, sizeof(serv_addr));
18:  ...
19:  recvfrom(fd, ..., (struct sockaddr*)&sndr_addr, &sndr_len);
20:
21:  close(fd);
```

Przykład Java.

```
 1:  DatagramSocket serverSocket = new DatagramSocket(10001);
 2:
 3:  byte[] receiveData = new byte[1024];
 4:  byte[] sendData = new byte[1024];
 5:
 6:  DatagramPacket receivePacket =
 7:      new DatagramPacket(receiveData, receiveData.length);
 8:  serverSocket.receive(receivePacket);
 9:
10:  InetAddress IPAddress = receivePacket.getAddress();
11:  int port = receivePacket.getPort();
12:
13:  DatagramPacket sendPacket =
14:      new DatagramPacket(sendData, sendData.length, IPAddress, port);
15:
16:  serverSocket.send(sendPacket);
17:
```

Przykład C#.

```
1:  byte[] data = new byte[1024];
2:  string input, stringData;
3:
4:  IPEndPoint address = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 1000);
5:
6:  Socket server = new Socket(AddressFamily.InterNetwork,
7:                       SocketType.Dgram, ProtocolType.Udp);
8:
9:  server.SendTo(data, data.Length, SocketFlags.None, address);
10:
11: IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
12: EndPoint Remote = (EndPoint)sender;
13:
14: data = new byte[1024];
15: int recv = server.ReceiveFrom(data, ref Remote);
16:
17: server.Close();
```

Wnioski.

- Sposoby implementacji serwera
  - Interacyjny,
  - Wielowątkowy.
- Niski poziom
  - Ręczna serializacja danych,
  - Różna kolejność bajtów w pamięci:
    - big-endian vs little-endian,
    - język C: zestaw metod *htonl, htons, ntohl, ntohs,*
    - język Java: automatycznie przez jvm,
    - Język C#: automatycznie przez clr,
  - Problemy z reprezentacją danych
    - Reprezentacja łańcuchów znaków w C oraz w Java/C#.