Programowanie współbieżne i rozproszone

WYKŁAD 5

Krzysztof Pancerz

Komunikowanie się procesów

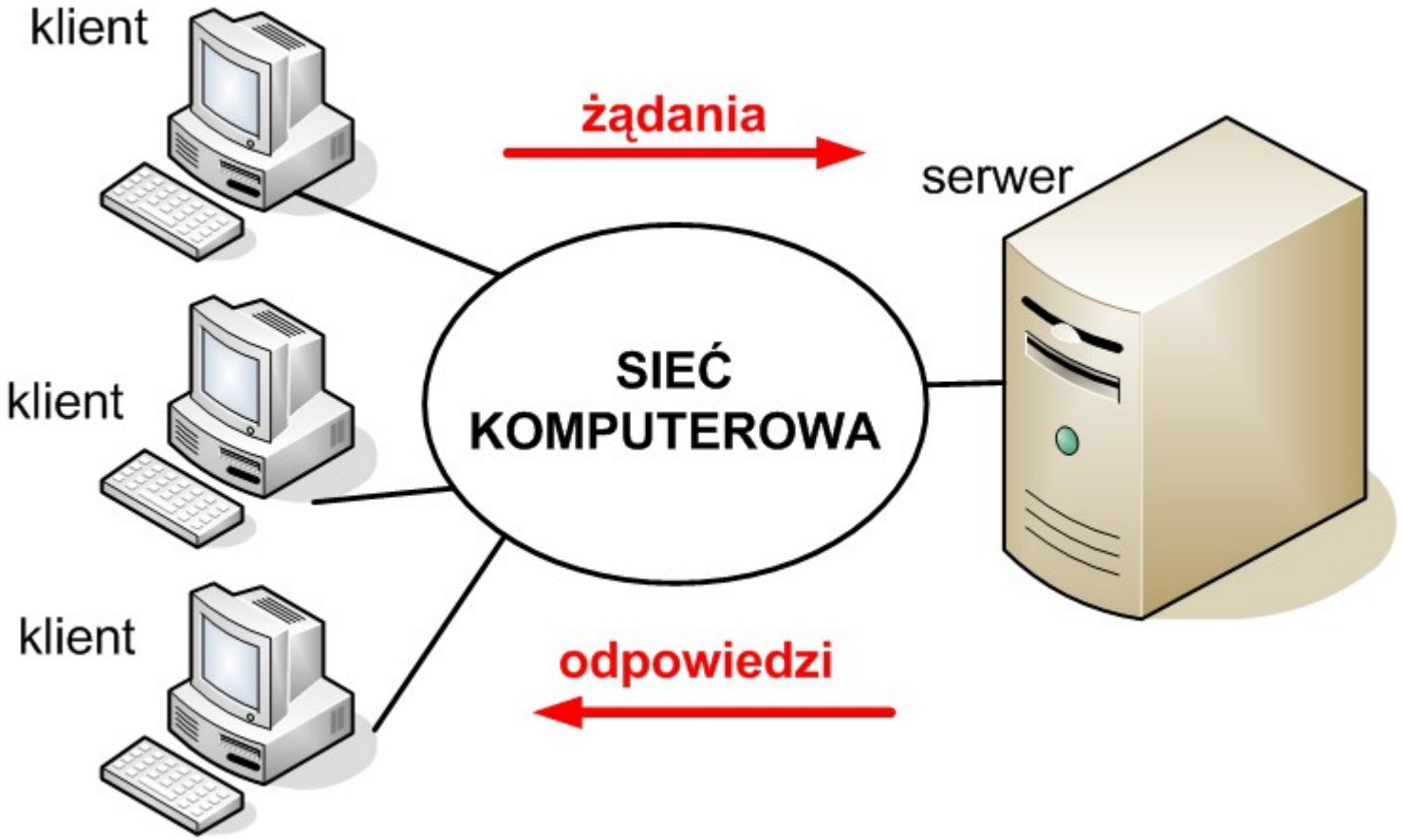
Z użyciem pamięci współdzielonej.

- wykorzystywane przede wszystkim w programowaniu wielowątkowym.

Za pomocą przesyłania wiadomości przez kanały komunikacyjne.

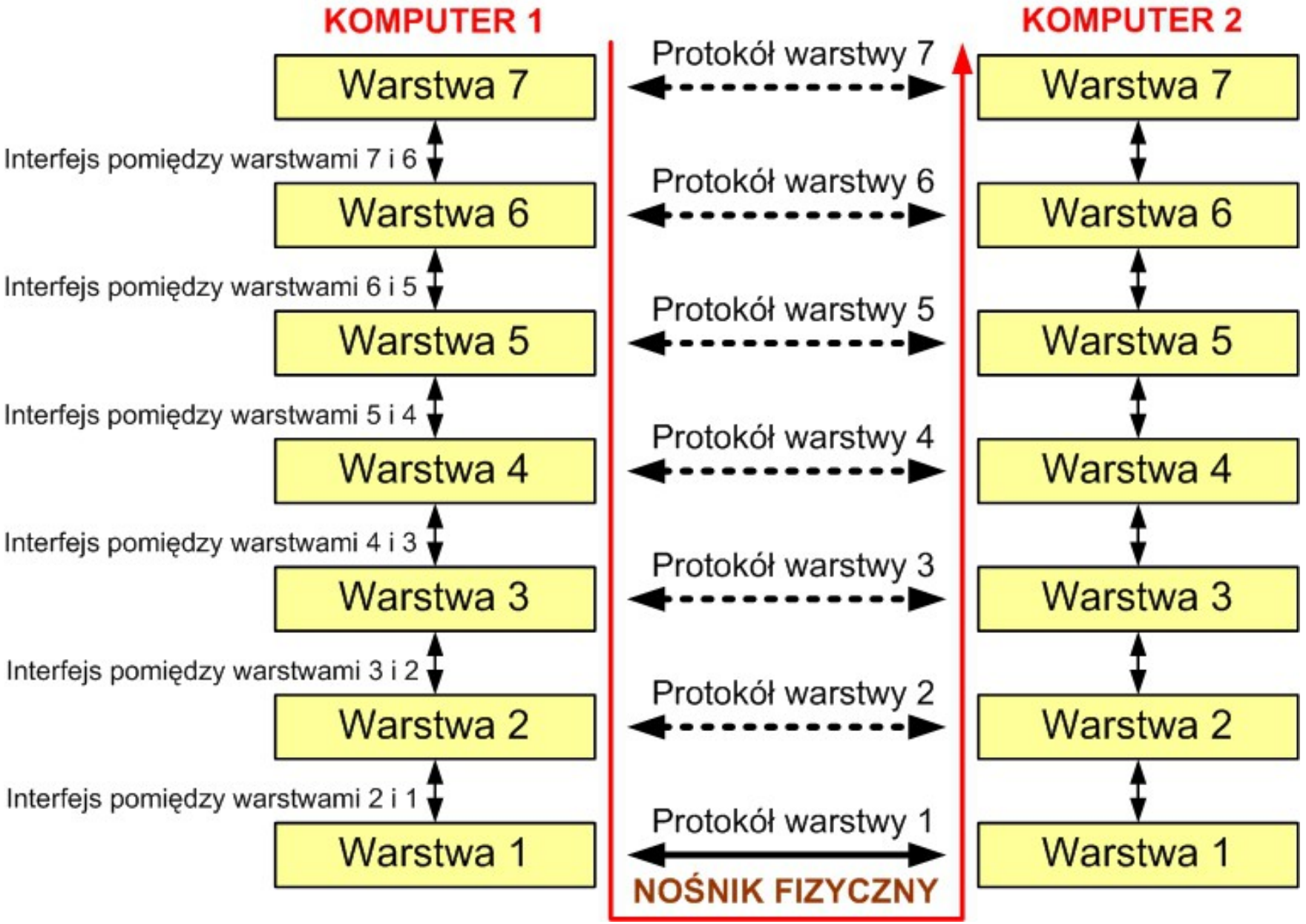
- wykorzystywane przede wszystkim w programowaniu rozproszonym.

Model klient - serwer



- Klient jest systemem, który zleca określone zadanie systemowi zwanemu serwerem.
- Serwer na żądanie klienta wykonuje określone usługi.

Ogólna architektura sieci



Model odniesienia ISO/OSI

Warstwa transportowa i warstwa sieciowa

- Protokół warstwy transportowej zapewnia logiczną komunikację między **procesami** uruchomionymi na różnych hostach.
- Protokół warstwy sieciowej zapewnia logiczną komunikację między **hostami**.

Gniazda

- **Gniazdo** (*socket*) identyfikuje punkt końcowy sieci.
- Paradygmat gniazd został zastosowany na początku lat 80-tych w systemie operacyjnym 4.2BSD Berkeley UNIX, stąd często mechanizm nazywany jest gniazdami Berkeley.
- Gniazda umożliwiają jednemu hostowi jednoczesną obsługę (w roli serwera) wielu klientów.

Gniazda

- Komunikacja przy użyciu gniazd wymaga pośrednictwa protokołów:
  - IP
    - warstwa sieciowa
  - TCP lub UDP
    - warstwa transportowa

Gniazda

- Gniazdo jest połączeniem numeru IP komputera i numeru portu, na którym odbywa się komunikacja.

Przykład:

Porty

- **Dobrze znane porty** (*well-known ports*) – porty o numerach od 0 do 1024 zarezerwowane na potrzeby standardowych usług warstwy aplikacji, np.:
  - FTP: port 21,
  - HTTP: port 80.
- Lista dobrze znanych portów zdefiniowana jest przez organizację IANA (*Internet Assigned Numbers Authority*):

http://www.iana.org/assignments/port-numbers

Protokół UDP

- **Protokół UDP** (*User Datagram Protocol*) realizuje usługę bezpołączeniowego dostarczania datagramów, tzn.:
  - nie ustanawia w żaden sposób połączenia,
  - nie sprawdza gotowości odległego komputera do odebrania przesyłanych danych ani poprawności ich dostarczenia.

Protokół UDP

- Połączenie UDP jest połączeniem punkt-punkt.
- Jednostką przesyłania danych dla protokołu UDP jest pakiet.
- Zaleta:
  - mała ilość przesyłanych informacji kontrolnych.

Protokół TCP

- **Protokół TCP** (*Transmision Control Protocol*) jest protokołem niezawodnym i połączeniowym.
- Połączenie TCP jest zawsze połączeniem punkt-punkt.
- Protokół TCP sprawdza czy dane zostały dostarczone poprawnie i w określonej kolejności.
- Dane dostarczane przez protokół TCP mogą być traktowane jak strumień.

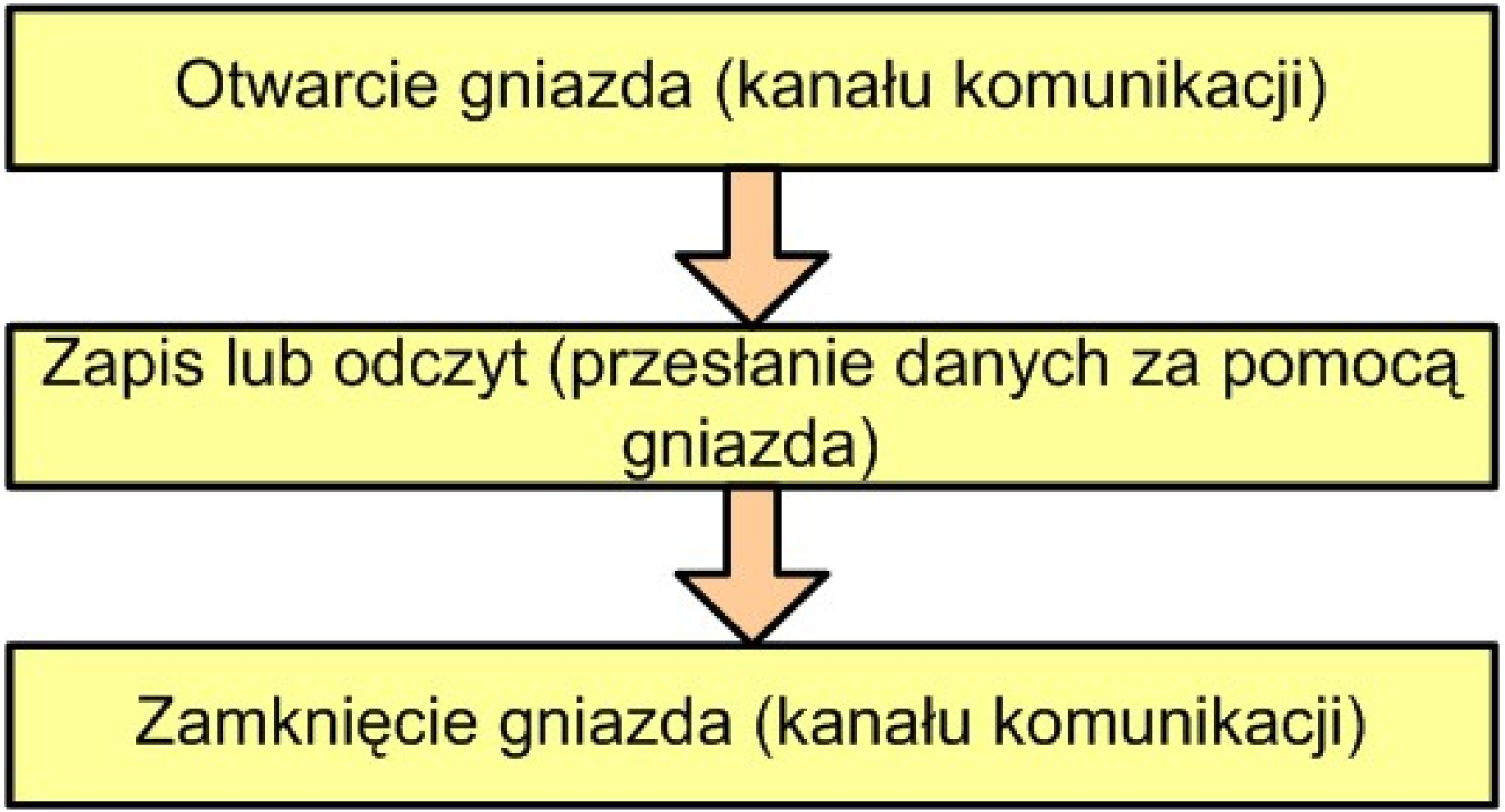
Protokół TCP (cd.)

- Jednostką przesyłania danych dla protokołu TCP jest segment.
- Proces nawiązujący połączenie nazywany jest procesem klienta. Drugi proces nazywany jest procesem serwera.

Aplikacje sieciowe

- Przy tworzeniu aplikacji sieciowej należy zdecydować z jakiego protokołu (UDP czy TCP) warstwy transportowej aplikacja będzie korzystać.
- Wybór protokołu warstwy transportowej dokonywany jest przy definiowaniu gniazd.

Programowanie aplikacji sieciowych



Programowanie aplikacji sieciowych (cd.)

- Serwer tworzy gniazdo związane z określonym portem i na tym kanale komunikacyjnym oczekuje na prośbę połączenia od klienta.
- Inicjatywa połączenia wychodzi od klienta. Klient musi znać host serwera oraz numer portu otwartego do przyjmowania połączeń. Informacja ta podawana jest gdy klient tworzy swoje gniazdo związane z tak określonym adresem.

Programowanie aplikacji sieciowych (cd.)

- Serwer akceptuje połączenie od klienta i tworzy inne gniazdo do komunikacji z danym klientem tak, aby pozostać dostępnym dla innych klientów na kanale, gdzie oczekuje na połączenia.
- Z punktu widzenia klienta jest to zazwyczaj to samo gniazdo, na którym zainicjowane zostało połączenie.

Programowanie aplikacji sieciowych (cd.)

Rodzaje gniazd:

- Gniazdo serwera (reprezentowane w języku Java przez klasę ***ServerSocket***).
- Gniazdo klienta (reprezentowane w języku Java przez klasę ***Socket***) – używane przez klientów oraz przez serwery do komunikacji z klientami po zaakceptowaniu połączeń od nich.

Pakiet java.net

- Pakiet **`java.net`** zawiera zestaw narzędzi dla programowania sieciowego.

Wybrane klasy pakietu java.net

- **Inet4Address** - reprezentuje adres IP w wersji 4.
- **Inet6Address** - reprezentuje adres IP w wersji 6.
- **Socket** - reprezentuje gniazdo klienta.
- **ServerSocket** - reprezentuje gniazdo serwera.
- **DatagramSocket** - reprezentuje gniazdo implementujące mechanizm wysyłania i odbierania datagramów w protokole UDP.

Nawiązywanie połączenia z serwerem

```
Socket gniazdo=new Socket(serwer, port);
```

- **serwer -** adres (domenowy lub IP) serwera
- **port** - numer portu

Wczytywanie danych tekstowych z gniazda

```
InputStream strWeGniazda=gniazdo.getInputStream();

BufferedReader gniazdoOdczyt=new BufferedReader(new
  InputStreamReader(strWeGniazda));

String linia=gniazdoOdczyt.readLine();
```

Zapisywanie danych tekstowych do gniazda

```
OutputStream strWyGniazda=gniazdo.getOutputStream();

DataOutputStream strWy=new
  DataOutputStream(strWyGniazda);

strWy.writeBytes(linia);

strWy.flush();
```

Wczytywanie danych binarnych z gniazda

```
InputStream strWeGniazda=gniazdo.getInputStream();


DataInputStream gniazdoOdczyt=new
  DataInputStream(strWeGniazda);


int a=gniazdoOdczyt.readInt();


//float a=gniazdoOdczyt.readFloat();
//itp.
```

Zapisywanie danych binarnych do gniazda

```
OutputStream strWyGniazda=gniazdo.getOutputStream();

DataOutputStream strWy=new
  DataOutputStream(strWyGniazda);

strWy.writeInt(liczba);

//strWy.writeFloat(liczba);
//itp.
```

Wczytywanie danych serializowanych z gniazda

```
InputStream strWeGniazda=gniazdo.getInputStream();

ObjectInputStream gniazdoOdczyt=new
  ObjectInputStream(strWeGniazda);

Object obiekt=gniazdoOdczyt.readObject();
```

Zapisywanie danych serializowanych do gniazda

```
OutputStream strWyGniazda=gniazdo.getOutputStream();

ObjectOutputStream strWy=new
  ObjectOutputStream(strWyGniazda);

strWy.writeObject(obiekt);
```

Uruchamianie serwera i przyjmowanie żądań

```
ServerSocket gniazdoSerwera=new ServerSocket(port);

Socket gniazdo=gniazdoSerwera.accept();
```

Zamykanie gniazda

```
gniazdo.close();
```