Politechnika Łódzka
Wydział Elektrotechniki, Elektroniki, Informatyki i Automatyki

Programowanie sieciowe

Wykład 1
Techniki programowania
z wykorzystaniem gniazd

Instytut Informatyki Stosowanej
Politechniki Łódzkiej

Opracował: dr hab. inż. Radosław Wajman

Literatura

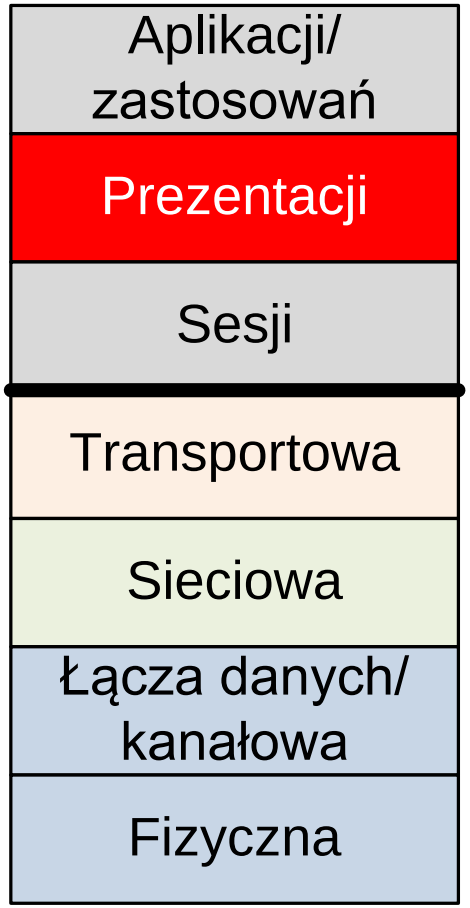
- R. Stevens: *UNIX Programowanie usług sieciowych* TOM 1, 2002
- R. Blum: *C# Network Programming*, Sybex, 2002
- Microsoft: *Network Programming in the .NET Framework*
  https://msdn.microsoft.com/en-us/library/4as0wz7t(v=vs.110).aspx
  https://msdn.microsoft.com/en-us/library/b6xa24z5(v=vs.110).aspx
- C# sockets code examples
  https://msdn.microsoft.com/en-us/library/w89fhyex(v=vs.110).aspx
- Oracle: *All about sockets*
  http://docs.oracle.com/javase/tutorial/networking/sockets/

Model OSI – *Open System Interconnection*

Warstwa aplikacji

- **Komunikacja z użytkownikiem**,
- **Wyświetlanie grafiki, tekstu** (przeglądarka WWW, gry multiplayer),
- Zapis/odczyt danych z dysku,

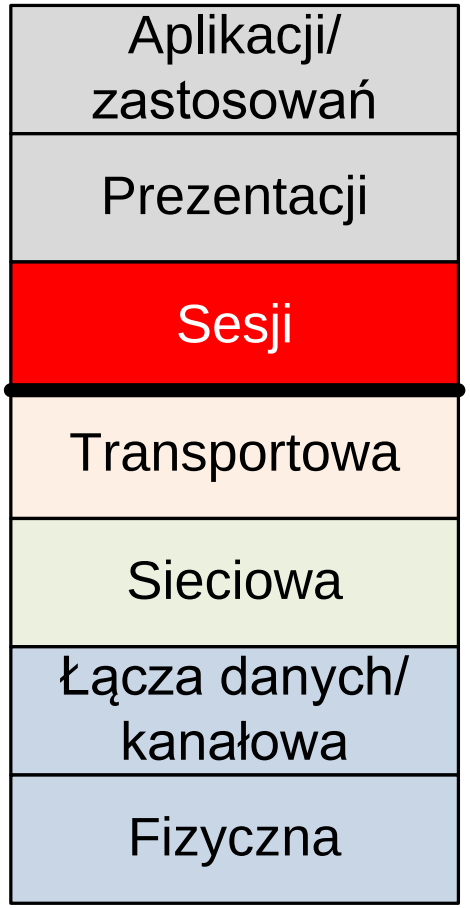
Model OSI – *Open System Interconnection*



Warstwa prezentacji

- Przygotowanie danych do wysłania
  - **Kompresja**,
  - **Szyfrowanie**,
  - **Serializacja** obiektów binarnych do np. formatu XML,
- Ustalenie kolejności bajtów do tzw. sieciowej kolejności bajtów (*network byte order*).
  - Pierwszeństwo bajtu bardziej znaczącego (*big-endian*).
  - 0xCAFFE001 -> CA, FF, E0, 01,
  - `htons(unsigned short)`, `htonl(unsigned long)`,
  - `ntohs(unsigned short)`,
  - `ntohl(unsigned long)`,

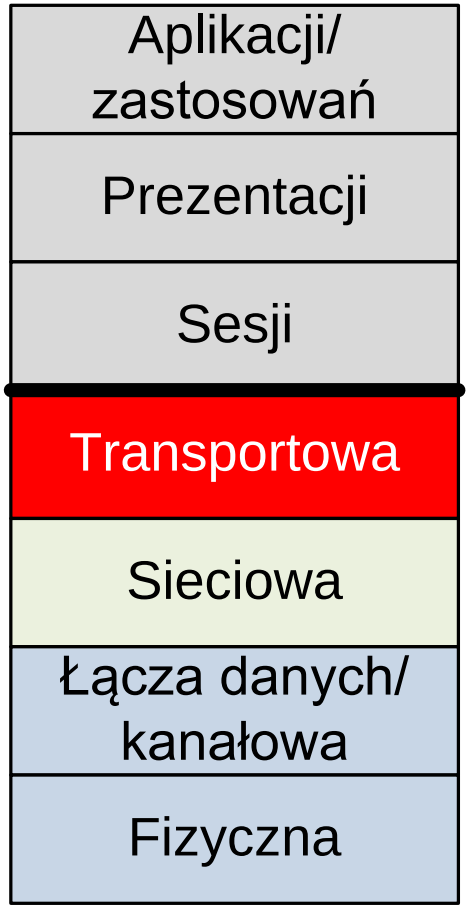
Model OSI – *Open System Interconnection*



Warstwa sesji

- Odpowiada za **nadzorowanie połączenia**, monitorowanie jego stanu,
- W przypadku **zerwania połączenia** program nadzorcy może ponawiać połączenie, np. *n* razy, po czym poinformować warstwę wyższą o błędzie
- **Ukrycie gniazd** przed warstwą prezentacji

Model OSI – *Open System Interconnection*



Warstwa transportowa

- Wykorzystywana najczęściej przez protokół **TCP** (*Transmission Control Protocol*) lub **UDP** (*User Datagram Protocol*).
- Można pominąć warstwę transportową i komunikować się bezpośrednio przy pomocy oprogramowania IPV4 oraz IPv6.
  - Służą do tego gniazda surowe (*raw sockets*)
  - Można tworzyć własne protokoły komunikacji
- Przyjmuje **strumień danych**, generuje **pakiety**

Model OSI – *Open System Interconnection*

Warstwa sieciowa

- Przesyłanie
- Obsługiwana przez oprogramowanie protokołów IPv4 oraz IPv6.
- Przyjmuje **pakiety**, generuje **datagramy IP**.

Model OSI – *Open System Interconnection*

Warstwa łącza danych

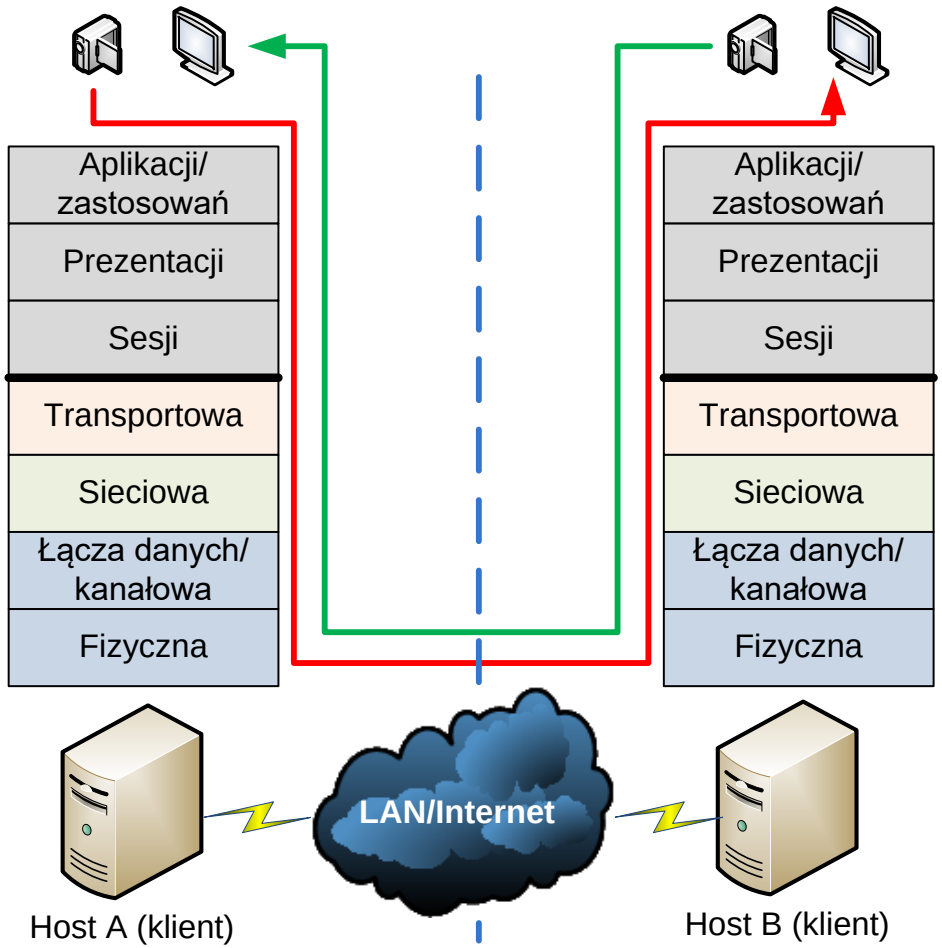
- Sterowniki dostarczane przez producenta sprzętu lub systemu operacyjnego,
- Ograniczenie wielkości segmentu do 1500 bajtów, MTU (*Maximum Transfer Units*)
- Przyjmuje **datagramy**, generuje **ramki.**

Model OSI – *Open System Interconnection*

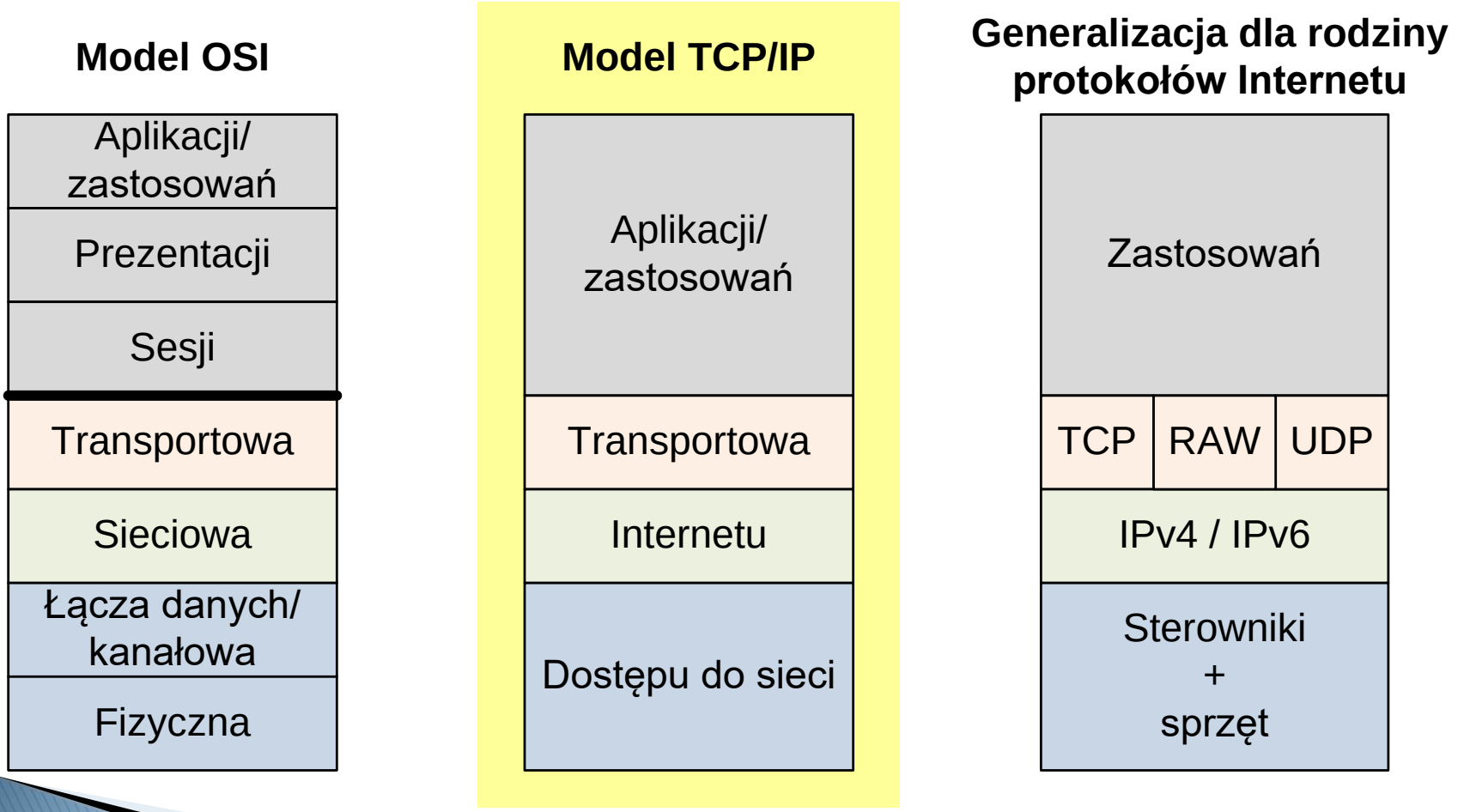
Warstwa fizyczna

- sprzęt
  - Karta sieciowa
  - Konwertery medium
  - Kable
- Informacje przesyłane jako **strumień bitów** (11010101010111011100 1 …)

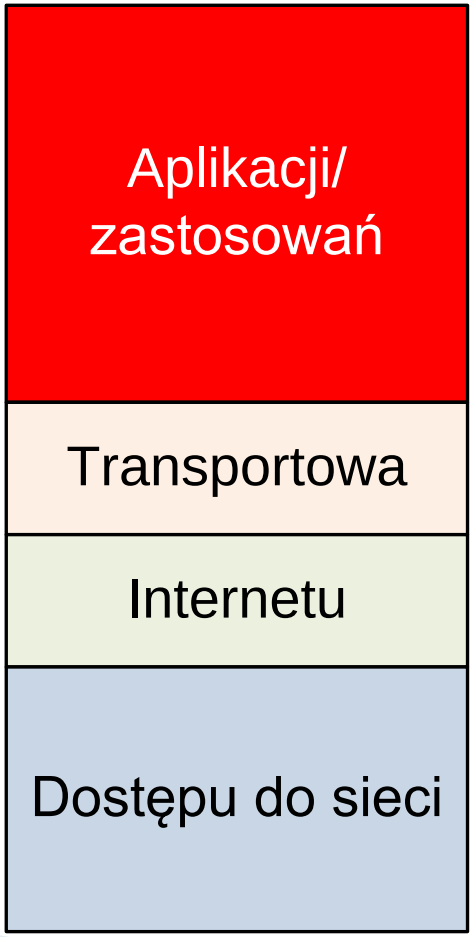
Model OSI – przepływ danych



Model OSI vs Model TCP/IP



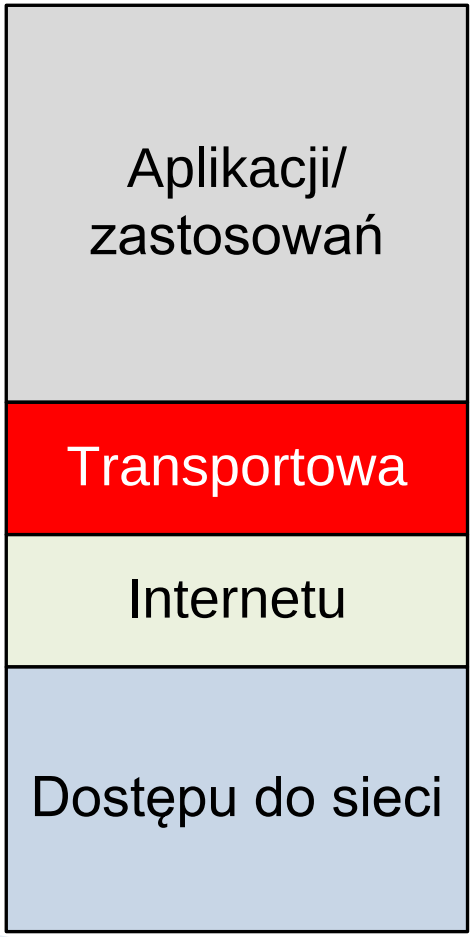
Model TCP



Warstwa aplikacji

- Kontakt z użytkownikiem lub procesem (GUI, Video)
- Transformacja danych do jednolitego formatu
- Dialog między aplikacjami zdalnymi, pracującymi wg założonego protokołu (np. FTP, HTTP)

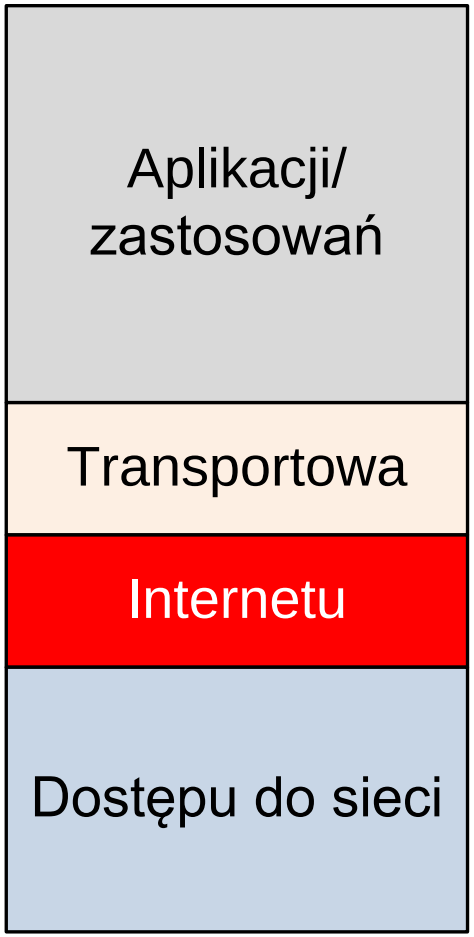
Model TCP



Warstwa transportowa

- Przesyłanie danych między aplikacjami (określanych na podstawie unikalnych par `numer_ip:port`)
- Obsługa wielu aplikacji jednocześnie; para `numer_ip:port` może być przyporządkowana **tylko do jednego** procesu
- *W modelu OSI to tutaj znajduje się oprogramowanie TCP*

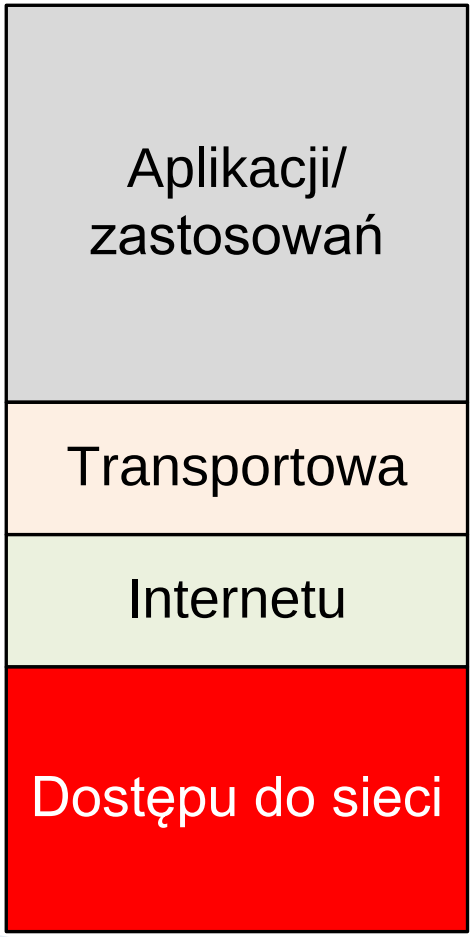
Model TCP



Warstwa Internetu

- Protokół IPv4 lub IPv6,
- Bazuje na adresie IP

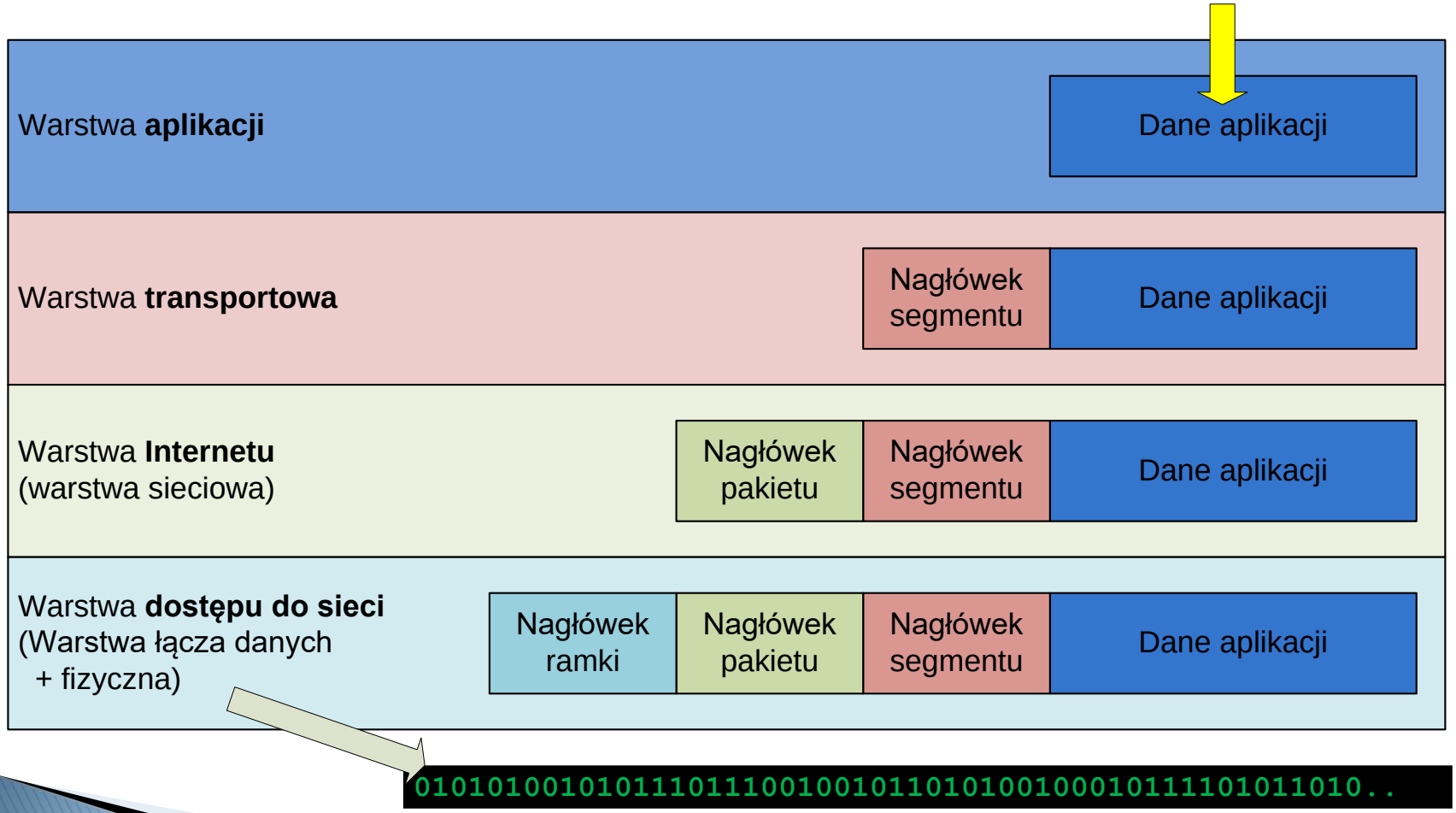
Model TCP



Warstwa dostępu

- Przekazywanie informacji przez fizyczne połączenie (nadawanie/odbiór)

Enkapsulacja danych w Modelu TCP/IP



Standardowe protokoły transportowe

- TCP (Transmission Control Protocol)
- UDP (User Datagram Protocol)

TCP – Transmission Control Protocol

- Protokół zorientowany na **połączenia**,
- Potwierdzenia odbioru danych przesyłane do nadawcy
  - W przypadku błędu przesyłu następuje **retransmisja** lub zerwanie połączenia
- Kontrola poprawności przesyłania danych,
- Odebrane dane przed przekazaniem do warstwy wyższej układane są w odpowiedniej kolejności
- **Większy narzut transmisji,**
- **Wysoki stopień niezawodności,**
- Komunikacja na dużych odległościach

UDP – User Datagram Protocol

- Brak mechanizmu połączeń,
  - Serwer może przyjąć datagramy od kilku różnych hostów na tym samym porcie,
- Brak kontroli zgubienia pakietu,
  - Poprawność transmisji określana jest na podstawie jedynie **sumy kontrolnej** datagramu,
  - Brak potwierdzenia otrzymania danych,
- Datagramy mogą zostać odebrane w różnej kolejności,
- Mniejszy narzut na transmisję,
- Pomimo swoich wad dobrze sprawdzają się w sieci lokalnej (gdzie niezawodność połączenia fizycznego jest wysoka)

Wybrane zagadnienia z programowania sieciowego (wszechobecne)

- Adres IP – identyfikacja hosta,
- Numer portu, – identyfikacja aplikacji,
- Para gniazdowa, – identyfikacja połączenia,
- Konwersja danych,
- Co to jest uchwyt?

Adres IP (dla IPv4)

- Liczba całkowita, 32-bitowa bez znaku; w języku C `unsigned long`, zakres: 0x00000000 – 0xFFFFFFFF (0 – 4,294,967,295),
- Umożliwia identyfikację komputera w sieci Internet
  - Przypadkiem szczególnym jest **współdzielenie łącza** (NAT/Maskarada)
- Dla użytkownika przewidziana jest postać „**A.B.C.D**”, gdzie litery to liczby 8-bitowe bez znaku.

Przykład:
0x7F000001 = 7F.00.00.01 = **127.0.0.1**
0xC0A81401 = C0.A8.14.01 = **192.168.20.1**

A jeśli programy **A** i **B** na hoście **M** będą chciały skomunikować się z odpowiednio programami **A** i **B** na hoście **N**?

Numer portu

- Liczba całkowita, 16-bitowa bez znaku; w języku C `unsigned short`, zakres: 0x0000 – 0xFFFF (0 – 65535),
- Ten sam numer (wartość) portu może być wykorzystana dla protokołu TCP i UDP,
- Umożliwia identyfikację połączenia dla danego numeru IP,
- Dany port może być przypisany tylko do jednej aplikacji,
- Para uporządkowana (`numer_ip:port`) **jednoznacznie** określa hosta oraz aplikację, z którą to połączenie jest nawiązane,
- Przykłady portów:
  - **53** – DNS,
  - **80** – Serwer HTTP, 8080 – najczęściej serwer proxy,
  - **21, 21** – Serwer FTP,
  - **25** – SMTP (poczta wychodząca)
  - **110** – POP3 (poczta przychodząca),
  - **22** – SSH

Konwersja danych

- Liczba 3735928559 (0xDEADBEEF) zapisana dane w formacie:

Little-endian (najmniej znaczący bajt jako pierwszy):

`EF BE AD DE`

- Procesory z rodziny x86 (Intel)

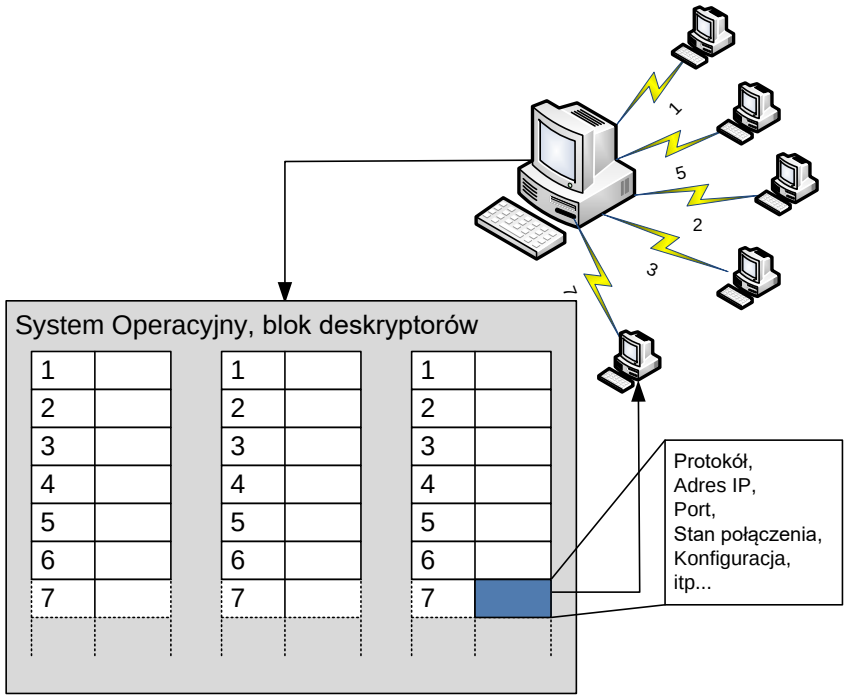
Big-endian (najbardziej znaczący bajt jako pierwszy):

`DE AD BE EF`

- Procesory 68000 Motorola

Format sieciowy (*Network Byte Order*):
big endian

Uchwyt (deskryptor)



- Uchwyt jest **formą** wskaźnika do struktury systemowej opisującej dany zasób,
- W przeciwieństwie do wskaźnika, uchwyt umożliwia **weryfikacje** istnienia informacji, na którą wskazuje

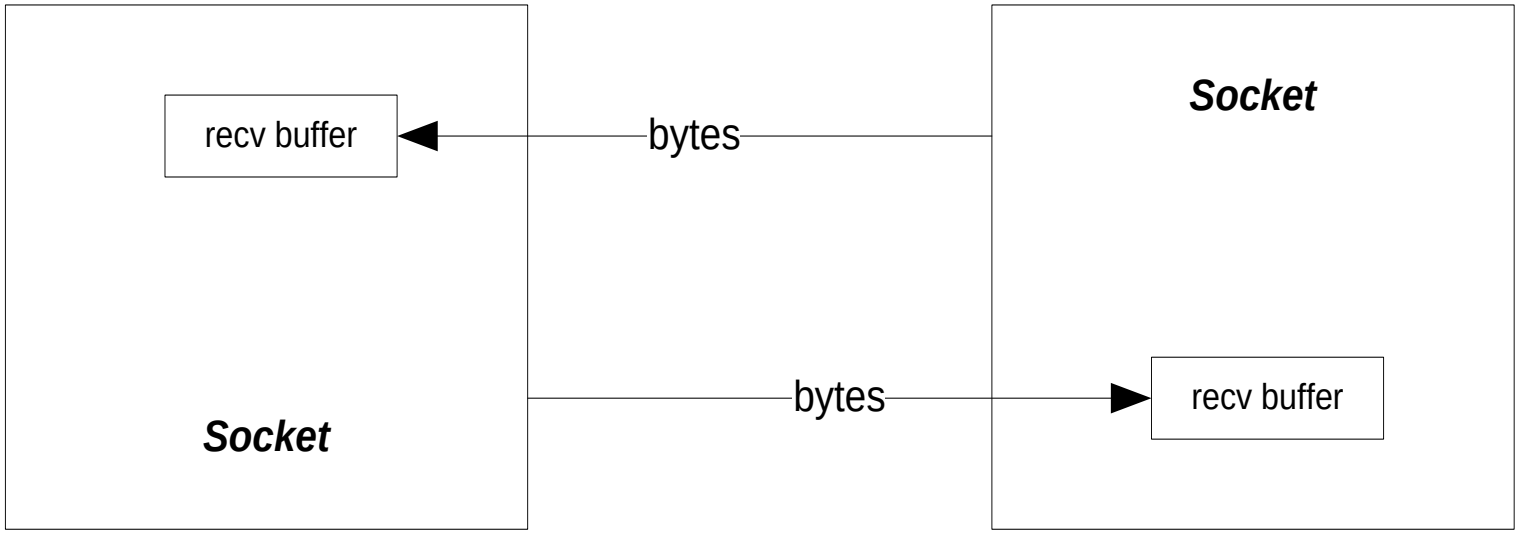
Co to są gniazda – sockets?

- Gniazda dostarczają interfejsu wielu protokołom komunikacji sieciowej
- są uchwytem (deskryptorem)

- Służą do ustalenia połączenia pomiędzy komputerami dla potrzeb wysyłania i odbierania danych.

- Gniazda gwarantują jednoczesne istnienie wielu połączeń klienckich z jednym serwerem.

Logiczna struktura gniazda



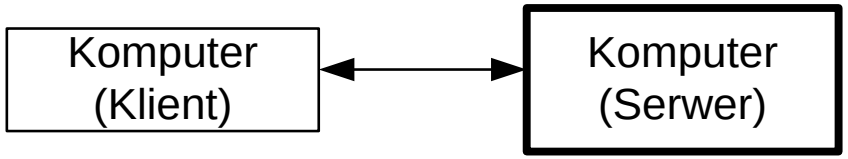
Para gniazdowa

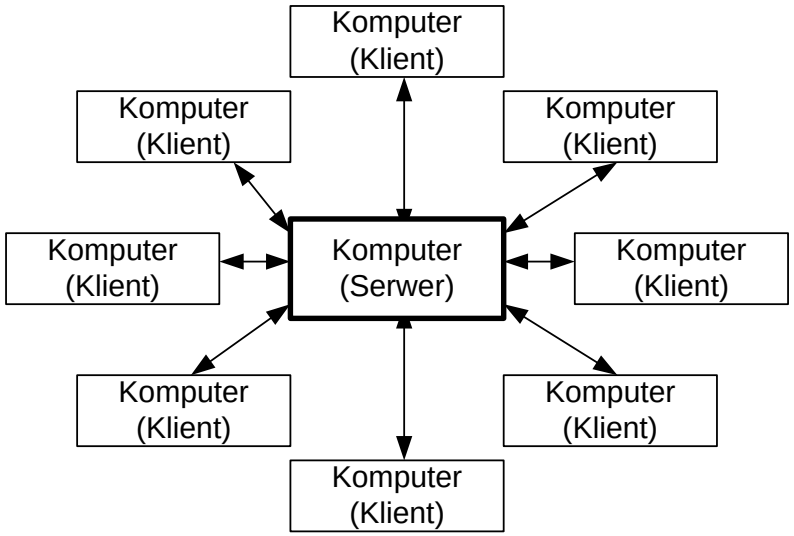
- **Cztery elementy** definiujące dwa punkty końcowe połączenia:
  - Adres IP lokalny,
  - Port lokalny,
  - Adres IP zdalny,
  - Port zdalny

- Ta Czwórka umożliwia **jednoznaczną** identyfikację danego połączenia TCP w sieci.

- W przypadku protokołu UDP czwórka jednoznacznie określa źródło i cel datagramu znajdującego się w sieci.

Połączenia między komputerami





- **Serwer-klient** – najczęściej spotykana relacja
- Jeden serwer może obsługiwać wiele klientów jednocześnie (np. komunikatory, serwery WWW)
- Komputer-komputer (**P2P, Peer to peer**) – bezpośrednie połączenie między komputerami

Techniki transmisji danych

- Transmisja **jeden-do-jednego** (ang. *unicast*), w których pojedynczy pakiet danych przesyłany jest od nadawcy do jednego odbiorcy, dokładnie pod jeden adres

- TCP
- UDP

Techniki transmisji danych

- Transmisja **jeden-do-wielu**
  (ang. *multicast*)
  składa się z pojedynczego pakietu danych, który adresowany jest do grupy odbiorców, przy czym router docelowy może wysyłać pakiety nie tylko do użytkowników końcowych, ale także do innych routerów

- UDP

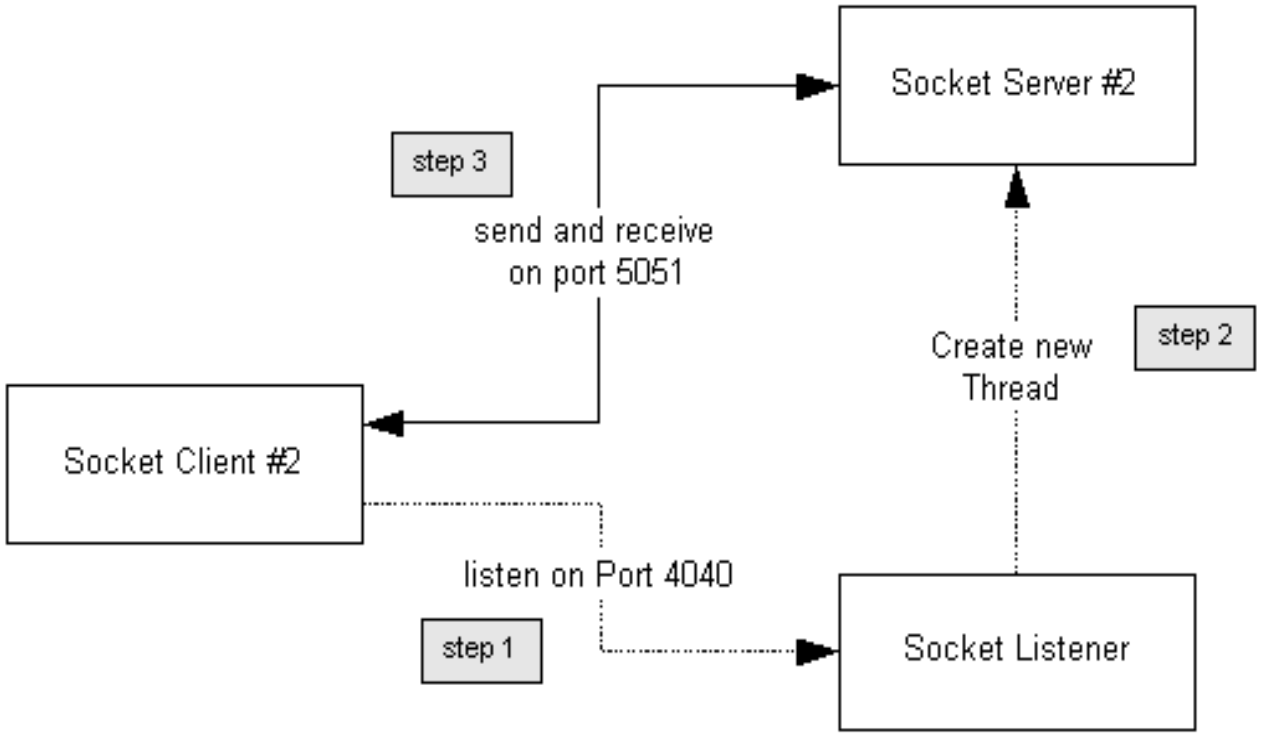
Techniki transmisji danych

- Transmisje **rozgłoszeniowe**
  (ang. *broadcast*)
  składają się z pojedynczego pakietu danych, kopiowanego i przesyłanego do wszystkich węzłów sieciowych.
- Pakiet jest adresowany przez węzeł źródłowy specjalnym adresem rozsyłającym, a następnie przesyłany do sieci, która tworzy kopie pakietu i wysyła je do każdego węzła sieci.
- UDP

Funkcje gniazd

- Istnieje kilka trybów pracy gniazd.
  - Najpowszechniejszym trybem jest **tryb słuchania** (ang. *Listener*), który realizuje oczekiwanie żądania połączenia na dowolnym porcie.
  - Gdy nadejdzie takie żądanie od aplikacji klienta, Słuchacz tworzy nowy wątek, a w nim nowe gniazdo w **trybie serwera** na innym porcie.
  - Od tego czasu klient i serwer mogą na tym nowym porcie komunikować się. Jeden wysyła dane, a drugi je odbiera.
  - W międzyczasie Słuchacz wraca do słuchania na zdefiniowanym dla siebie porcie.

Gniazdo Klienta, Serwera i Słuchacza



Transfer danych z użyciem gniazd

- Gniazdo „odbierające” (klienta lub serwera) posiada bufor, w którym przechowuje dane do czasu, kiedy wątek aplikacji odbiorcy je przeczyta.
  - Jeżeli bufor odbierania przepełni się, wątek wysyłania zostaje zblokowany do czasu, aż odbiorca przeczyta dane z bufora i go wyczyści.
    - To właśnie z tego powodu dobrą praktyką jest przydzielenie zadania opróżniania bufora i kolejkowania danych do oddzielnego wątku (odbiór danych może zablokować tylko jeden wątek a nie całą aplikacje).
  - Jeżeli z kolei bufor ulegnie przepełnieniu w trakcie wysyłania danych, do nadawcy wróci żądanie ponownego wysłania mniejszej porcji danych.
  - Jeżeli bufor odbierający jest pusty, odbieranie danych również blokuje aplikację.
  - Jeżeli bufor posiada jakieś dane, ale w ilości mniejszej niż oczekiwana do przeczytania, polecenie czytania zwróci liczbę faktycznie odebranych danych.

Długość wysyłanych danych

- Problem jest wtedy, gdy odbiorca nie będzie znał faktycznej długości wysłanych do niego danych.
  - Rozwiązuje się to poprzez wprowadzenie specjalnych ograniczników, komunikatów o stałej długości lub nagłówków (metadanych).

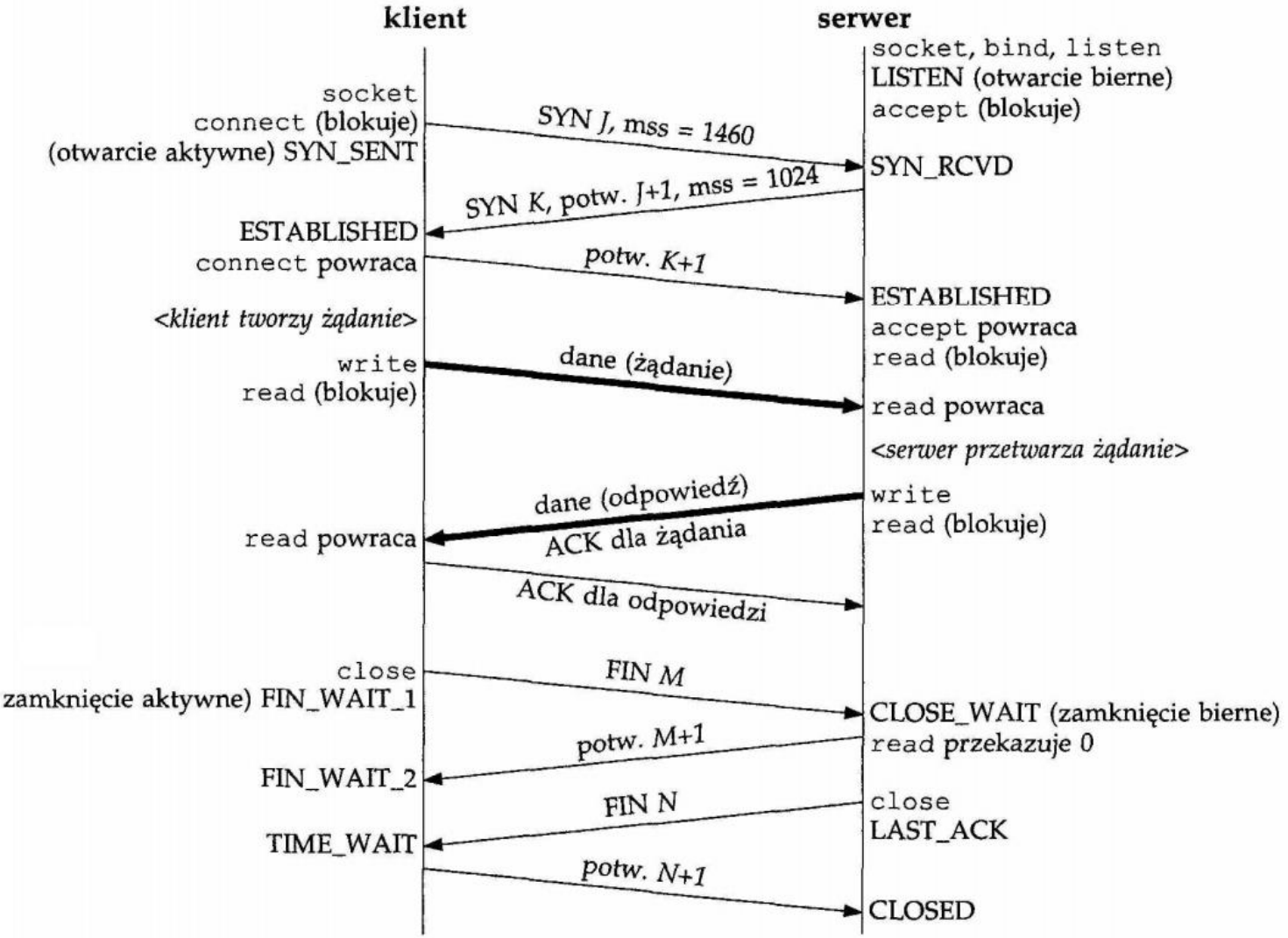
Zestawienie funkcji interfejsu gniazd Berkeley

Typ funkcji	Nazwa funkcji	Opis funkcji
Przydzielanie zasobów	socket	Tworzy gniazdo komunikacyjne i zwraca jego deskryptor
Zwalnianie zasobów i zamykanie połączenia	closesocket	Zamyka połączenie i likwiduje gniazdo komunikacyjne
	shutdown	Zamyka połączenie w jednym lub obydwu kierunkach
Nawiązywanie połączenia	connect	Inicjuje połączenie
	bind	Związuje gniazdo z określonym portem i numerem IP
	listen	Wprowadza lokalny port w bierny tryb pracy
	accept	przyjmuje zgłoszone połączenie
Wysyłanie danych	send	wysyła datagram
	sendto	wysyła datagram pod wskazany adres docelowy
	sendmsg	wysyła datagram
	write	wysyła dane przez połączenie
Odbieranie danych	read	odczytuje dane z połączenia
	recv	pobiera kolejny datagram
	recvfrom	odbiera datagram i zapamiętuje adres adawcy
	recvmsg	pobiera kolejny datagram
Funkcje informacyjne i konfiguracyjne	getpeername	podaje adres komputera zdalnego z którym nawiązano połączenie
	getsockopt	Pobranie i ustawienie opcji gniazda
	setsockopt	

Funkcje gniazd a rozmowa telefoniczna

Nazwa funkcji	Porównanie
socket	Otrzymanie numeru telefonu od operatora
bind	Poinformowania innych o naszym numerze telefonu, pod który mogą oni zadzwonić
listen	Włączenie telefonu, aby mógł oczekiwać na połączenie
connect	Wybranie numeru telefonu i zainicjowanie połączenia
accept	Odebranie rozmowy. Numer dzwoniącego tym razem znany dopiero po odebraniu
send/recv	Prowadzenie rozmowy mówienie/słuchanie
DNS	Szukanie abonenta w książce telefonicznej
close	Rozłączenie połączenia

Wymiana pakietów przez połączenie TCP
uzgodnienie trójfazowe



Źródło: [1]

Kolejność wykonywania funkcji gniazdowych klienta TCP

Źródło: *Programowanie usług sieciowych.* R. Stevens