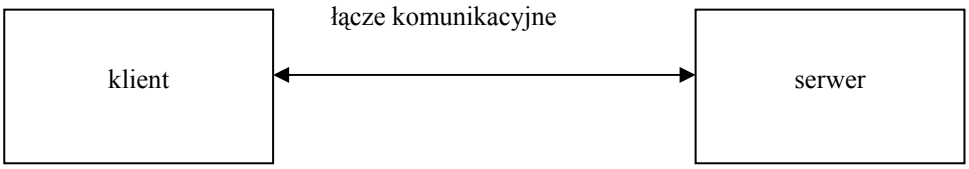
1. Model klient-serwer

1.1. Model komunikacji w sieci



Tradycyjny podziała zadań:
- ***Klient*** – strona żądająca dostępu do danej usługi lub zasobu
- ***Serwer*** – strona, która świadczy usługę lub udostępnia zasoby
- Komunikacja między klientem i serwerem realizowana jest w warstwie aplikacji.

Typowy schemat pracy klienta:
1. Nawiązuje kontakt z serwerem
2. Wysyła do serwera żądanie wykonania usługi i czeka na odpowiedź
3. Po otrzymaniu odpowiedzi od serwera kontynuuje działanie.

Typowy schemat pracy serwera:
1. Rozpoczyna pracę i zasypia czekając na klienta, który się z nim skontaktuje.
2. Gdy otrzyma zlecenie klienta budzi się i wykonuje usługę.
3. Po zakończeniu wykonywania usługi zasypia i czeka na nadejście następnego żądania.

Rodzaje usług
- standardowe – zdefiniowane w standardach RFC
- niestandardowe – wszystkie inne

1.2. Klient

Charakterystyka klienta
- wywoływany przez użytkownika, który chce skorzystać z usługi
- aktywnie inicjuje kontakt z serwerem
- działa lokalnie na komputerze użytkownika
- nie wymaga specjalnych uprawnień systemowych
- może kontaktować się z wieloma serwerami, ale w danej chwili aktywnie komunikuje się tylko z jednym

Wymagania dla klienta
- Przykłady usług standardowych:
  - zdalnie działający terminal (ang. *remote terminal client*) - protokół TELNET
  - poczta elektroniczna (ang. *electronic mail client*) – protokół SMTP
  - przesyłanie plików między komputerami (ang. *file transfer client*) protokół FTP
- Parametryzacja - dostęp do serwerów wielu usług
- Przykład:

```
telnet oceanic.wsisiz.edu.pl        (telnet)
telnet oceanic.wsisiz.edu.pl 25     (smtp)
```

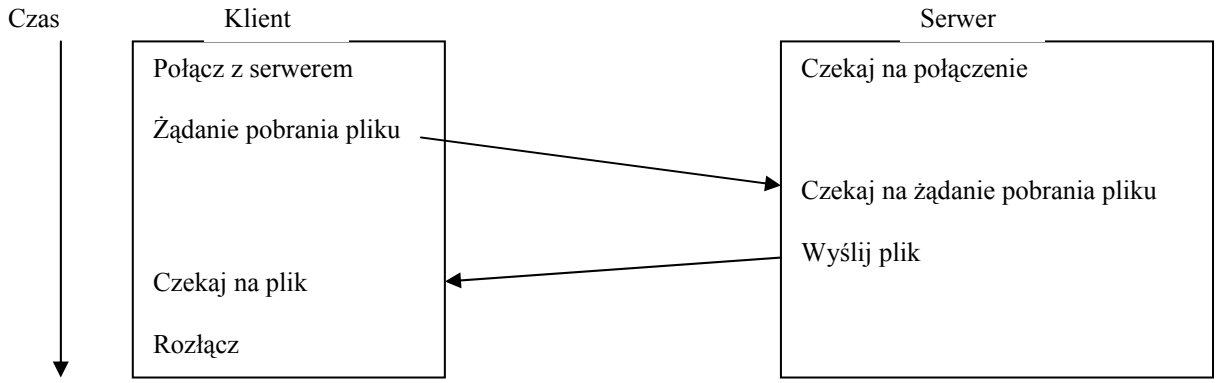
1.3. Serwer

Charakterystyka serwera
- uruchamiany automatycznie przy starcie systemu
- czeka pasywnie na zgłoszenia od dowolnych klientów
- działa na publicznie dostępnym komputerze
- jest specjalizowanym, uprzywilejowanym programem, którego zadaniem jest świadczenie konkretnej usługi
- zwykle świadczy jedną usługę, ale może obsługiwać wielu klientów

Wymagania dla serwera
- Działa w trybie uprzywilejowanym, jeśli musi mieć dostęp do zasobów chronionych
- Obsługa wielu klientów
  - sekwencyjna (serwer iteracyjny)
  - kilka zgłoszeń jednocześnie (serwer współbieżny)
- Zapewnienie bezpiecznej pracy:
  - **uwierzytelnianie** (ang. *authentication*) - sprawdzenie tożsamości klienta
  - **kontrola uprawnień** (ang. *autorization*) - sprawdzenie, czy dany klient ma prawo dostępu do usługi realizowanej przez serwer
  - **ochrona danych** (ang. *security*) - zabezpieczenie danych przez niezamierzonym naruszeniem lub ujawnieniem
  - **poufność** (ang. *privacy*) - zabezpieczenie przed dostępem nieupoważnionych użytkowników
  - **ochrona zasobów** (ang. *protection*) - zabezpieczenie zasobów systemowych przed niewłaściwym użyciem programów użytkowych działających w sieci

1.4. Protokół warstwy aplikacji

- Protokół jest to zbiór reguł, których muszą przestrzegać klient i serwer, aby mogły się ze sobą komunikować. Przykład:



- Przykład: protokół SMTP

Klient	**Serwer**
(otwiera połączenie)	
	220 poczta.onet.pl Wed, 10 Sep 2004 20:45:41 +0200
EHLO wp.pl	
	250-poczta.onet.pl Hello wp.pl [213.135.45.70], pleased to meet you 250-ENHANCEDSTATUSCODES 250-8BITMIME 250-SIZE 250-DSN 250-ONEX 250-ETRN 250-XUSR 250 HELP
MAIL FROM:adam@wp.pl	
	250 2.1.0 <adam@wp.pl>... Sender ok
RCPT To: dorota@poczta.onet.pl	250 2.1.5 <dorota@poczta.onet.pl>... Recipient ok
DATA	354 Enter mail, end with "." on a line by itself
From: adam@wp.pl To: dorota@poczta.onet.pl Subject: Test Czesc!. .	
	250 2.0.0 h8AIjfe01432 Message accepted for delivery
QUIT	221 2.0.0 poczta.onet.pl closing connection
(Połączenie zamknięte)	

Każdy przesłany wiersz kończy się znakami CRLF (carriage return, line feed)

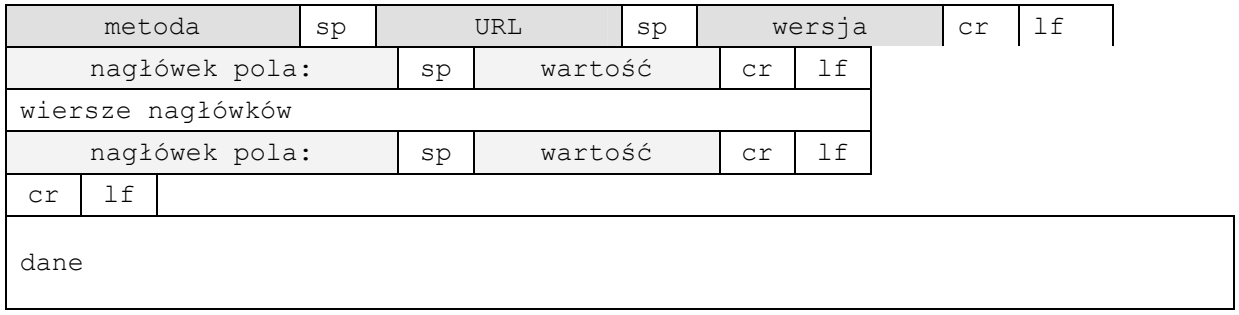
- Przykład: protokół HTTP – żądanie i odpowiedź

```
GET /~projekt/index.html HTTP/1.1.
Host: oceanic.wsisiz.edu.pl
Connection: close
```

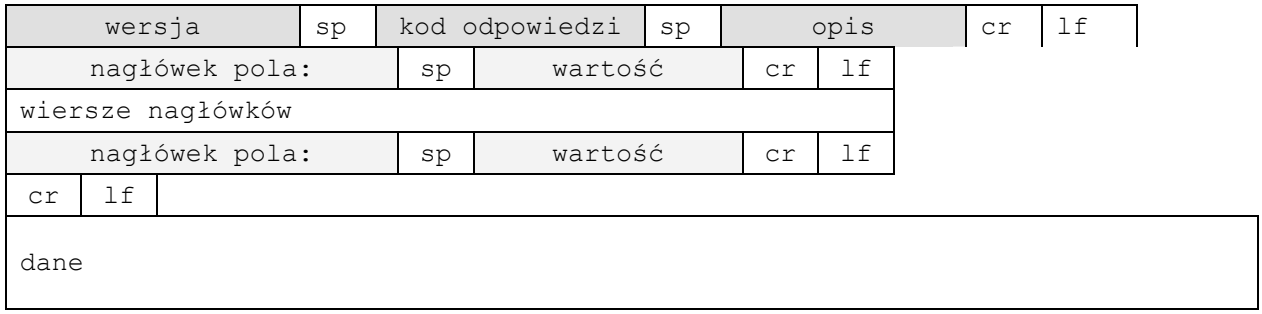
```
HTTP/1.1 200 OK
Date: Sat, 25 Feb 2006 09:09:01 GMT
Server: Apache
Last-Modified: Tue, 15 Nov 2005 14:54:42 GMT
ETag: "1c8627-c9e-4059bc4859080"
Accept-Ranges: bytes
Content-Length: 3230
Connection: close
Content-Type: text/html

dane dane dane ....
```

- Ogólna postać żądania:



- Ogólna postać odpowiedzi:



1.5. Komunikacja - protokół transportowy: TCP czy UDP

Typy komunikacji

- Wyróżnia się następujące typy komunikacji:
  - strumieniowa (*stream*) – pomiędzy dwoma punktami końcowymi przesyłany jest strumień bajtów (w obydwu kierunkach). Na strumień ten nie jest nakładana żadna określona struktura. Wysyłane dane mogą być agregowane lub dzielone. Przykład protokołu strumieniowego: TCP
  - datagramowa (*datagram*) – wysyłany jest pojedynczy komunikat od nadawcy do odbiorcy, bez nawiązywania połączenia. Przykład protokołu datagramowego: UDP
  - rozgłoszeniowa (*broadcast*) – komunikat jest wysyłany do wszystkich jednostek w sieci, nie trzeba ustalać odbiorcy, oparty jest na UDP.
  - rozgłoszeniowa ograniczona (*multicast*) – komunikat jest wysyłany do grupy odbiorców, trzeba taką grupę zdefiniować, oparty jest na UDP.

Protokoły warstwy transportowej

- Komunikacja jest realizowana w oparciu o protokoły warstwy transportowe, takie jak. TCP i UDP.
  - Protokół TCP – połączeniowy, niezawodny, dane są przesyłane w postaci strumienia bajtów
  - Protokół UDP – bezpołączeniowy, zawodny, dane przesyłane w postaci datagramów określonej długości, możemy je traktować tak jak rekordy
- Serwer:
  - ***połączeniowy*** (ang. *connection-oriented*) – oparty na TCP
  - ***bezpołączeniowy*** (ang. *connectionless*) – oparty na UDP
- O wyborze decyduje protokół aplikacyjny:
  - niezawodność
  - uporządkowanie pakietów
  - sterowanie przepływem
  - pełny duplex
  - narzuty czasowe
- Przykład: kiedy zazwyczaj stosuje się protokół UDP
  - wymaga tego protokół aplikacji – ale wtedy zawiera mechanizmy zapewniające niezawodność
  - czasochłonność obsługi lub opóźnienia uniemożliwiają poprawne działanie aplikacji
  - protokół aplikacji korzysta z trybu rozgłaszania

- Przykłady:

Aplikacja	Protokół warstwy aplikacji	Protokół transportowy
Poczta	SMTP (RFC 2821)	TCP
Zdalny dostęp do terminala	Telnet (RFC 854)	TCP
WWW	HTTP(RFC 2616)	TCP
Zdalne przesyłanie plików	FTP (RFC 959)	TCP
Zdalny serwer plików	NFS (McKusnik 1996)	UDP lub TCP
Strumieniowe usługi multimedialne	Często własność producenta	UDP lub TCP

1.6. Własny protokół warstwy aplikacji

- Projektowanie własnego protokołu wymaga podjęcia takich decyzji jak:
  - format wymienianych komunikatów,
  - czy serwer będzie bierny czy aktywny,
  - czy serwer ma przetwarzać polecenia pojedynczo,
  - czy protokół będzie wymagał nawiązania sesji,
  - czy protokół będzie tekstowy czy binarny ,
  - czy będzie uwzględniał uwierzytelnianie,
  - czy będzie zapewniał poufność.

Należy przeczytać:

Douglas E. Comer, David L. Stevens: *Sieci komputerowe TCP/IP, tom 3*: str. 35-48

W. Richard Stevens: *Unix, programowanie usług sieciowych, tom 1: API gniazda i XTI*: str. 82-137