TIN
Techniki Internetowe
lato 2018

Grzegorz Blinowski

Instytut Informatyki

Politechniki Warszawskiej

Plan wykładów

2	Intersieć, ISO/OSI, protokoły sieciowe, IP
3	Protokół IP i prot. transportowe: UDP, TCP
4	Model klient-serwer, techniki progr. serwisów
5	Protokoły aplikacyjne: telnet, ftp, smtp, nntp, inne
6	HTTP
7, 8	HTML, XML
9, 10, 11	Aplikacje WWW, CGI, sesje, serwery aplikacji serwlety, integracja z backended SQL
12	Aspekty zaawansowane: wydajność, przenośność, skalowalność; klastering
13	Inne: P2P, SOAP, RDF, WSDL, ontologie
14	Wstęp do zagadnień bezpieczeństwa (IPSec, VPN, systemy firewall) oraz aspekty kryptograficzne (DES, AES, RSA, PGP, S/MIME), tokeny i akceleracja sprzętowa

Zdalna praca znakowa: telnet, rlogin (i rsh), ssh

Przykład - sesje zdalne

```
don:~> telnet charlie
Trying 192.168.1.5...
Connected to charlie.xyz.com.pl.
Escape character is '^]'.

SunOS 5.7

login: janek
Password:
Last login: Fri C
Sun Microsystems
charlie%
```

```
charlie% ctrl-]
telnet> ?
Commands may be abbreviated.  Commands

close           close current connectio
…
set             set operating parameter
unset           unset operating paramet
status          print status informatio
…
```

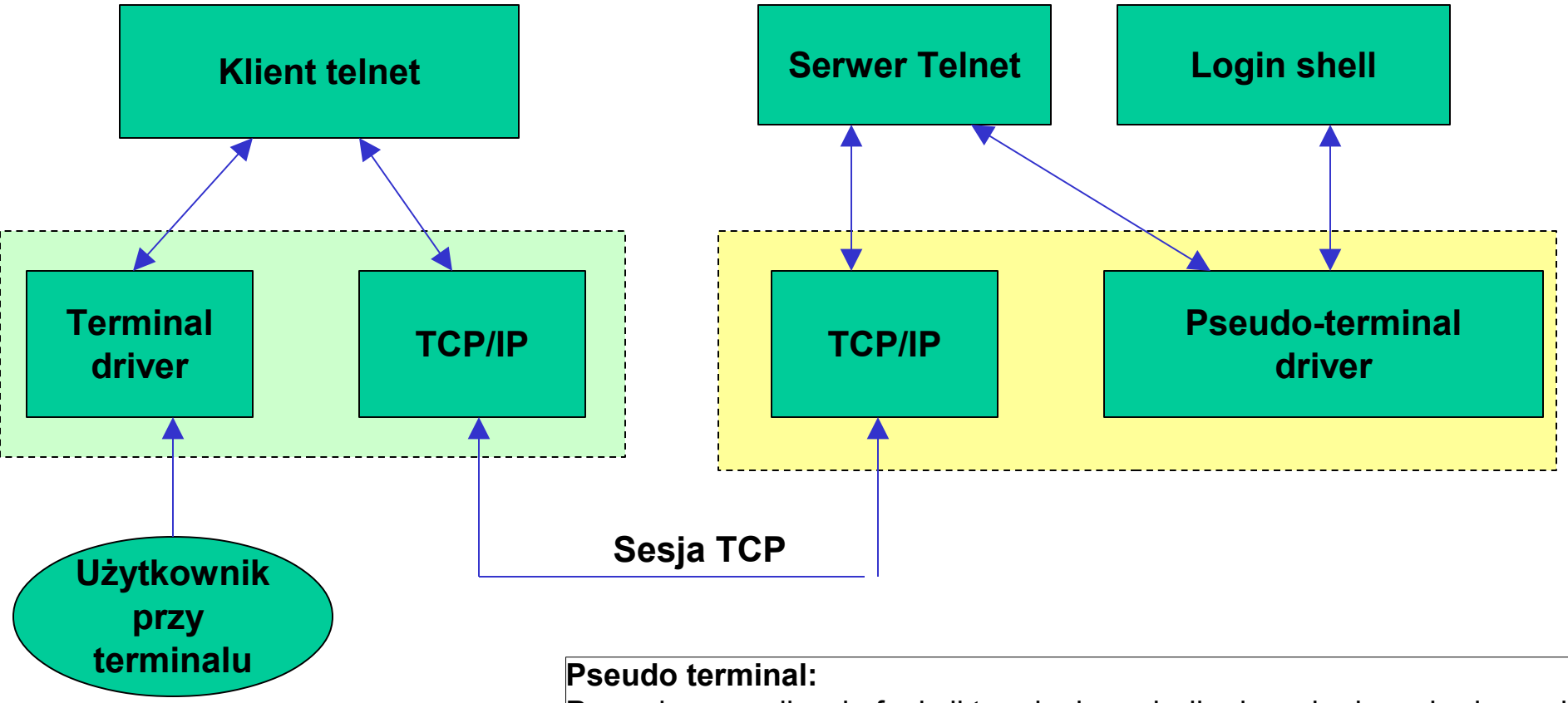
Telnet

RFC0158(!) - 1971,

RFC: 0854

- Podstawowy protokół pracy zdalnej w trybie tekstowym ("teletype over network")
- Telnet realizuje sesję zdalną - pozwala na zalogowanie się do odległej maszyny
- Obecny w każdym systemie "Unix-o podobnym" - często domyślnie włączony!
- Wykorzystuje TCP/IP, port TCP serwera: 23
- Znaczny wysiłek włożono w przenośność: liczne opcje negocjowane między klientem i serwerem
- Protokół o minimalnym (żadnym?) poziomie bezpieczeństwa
- Używany do testowania m.in.: SMTP, HTTP, FTP, POP3, IMAP
- Uwaga: telnet nie jest w pełni przeźroczysty dla danych 8-o bitowych (kod \0xff, sekwencja \0x0d \0x00)

Telnet



Schemat sesji klient-serwer dla prot. telnet

Pseudo terminal:
Pozwala na realizację funkcji terminalowych dla danych niepochodzących z "fizycznych" terminali - obsługa sygnałów, grupy terminalowe procesów - praca w tle (SIGINT SIGQUIT, SIGTSTP, SIGTTIN, SIGTTOU)

Telnet

- Trzy podstawowe serwisy:
  - Symetryczna traktowanie obydwu stron połączenia (w sensie transmisji danych)
  - **NVT** - Network Virtual Terminal - standardowy interfejs do zdalnych systemów
  - Mechanizm negocjacji opcji połączenia (poprzez NVT)
- Telnet pozwala na negocjowanie b. dużej liczby opcji między klientem i serwerem co pozwala na współpracę b. różnych systemów, m.in.: typ terminala, rozmiar okna, sterowanie przepływem, echo zdalne/lokalne, tryb liniowy. Opcje są negocjowane.
- Każdy znak podróżuje zazwyczaj w osobnym pakiecie ("tinygram") - echo jest zazwyczaj zdalne, co podwaja liczbę przesyłanych pakietów (zob. alg. Nagle'a limitujący liczbę niepotwierdzonych pakietów)

rlogin, rsh, rcp, ...

rlogin

- Mechanizm zdalnej pracy pochodzący z systemów BSD (i przeznaczony do współpracy wyłącznie systemów Unix – w przeciwieństwie do telnet):
  - rlogin - praca zdalna - odpowiednik telnet
  - rsh - zdalne wywoływanie poleceń
  - rcp, rdist - kopiowane plików na zdalne maszyny
- Wykorzystuje TCP, port TCP serwera: 513, klienta 1023(!)
- W stosunku do telnet:
  - dołączono mechanizmy (pół)automatycznej autoryzacji
  - możliwe jest standardowe przekierowanie we/wy (szczególnie przydatne przy rsh)
  - poziom bezpieczeństwa - podobny (praktycznie zerowy)

Rlogin, rsh inne

- Rsh jest wygodnym mechanizmem zdalnego wykonywania poleceń, np.:
  - W skrypcie na wielu maszynach
  - wykonywania zdalnych kopii, przykład
    dump <lokalne parametry> | rsh …
  - zdalnej instalacji: rdist (remote distribution)
- rsh/rlogin obsługuje także stderr poprzez dodatkowe gniazdo (co pozwala na rozdzielenie stdout i stderr i zwrotne przekazywnie sygnałów)
- Opcje rlogin są znacząco uproszczone w stosunku do telnet. Obłsugiwane są m.in.:
  - Sterowanie przepływem: Ctrl-Q/Ctrl-S
  - Zmiana rozmiaru okna
  - Sygnały (poprzez TCP OOB)

Rlogin, rsh

- **Przykłady:**

```
rlogin nazwa_maszyny
rlogin -l uzytkownik-zdalny nazwa_maszyny
```

- **Przekierowanie stdin/stdout w rsh:**

```
rsh nazwa_maszyny polecenie
rsh [-l uzytkownik-zdalny] nazwa_maszyny
  polecenie
rsh nazwa_maszyny 'polecenie arg1 arg2'
rsh nazwa_maszyny 'polecenie > /tmp/x'
rsh nazwa_maszyny 'polecenie arg' > /tmp/x
```

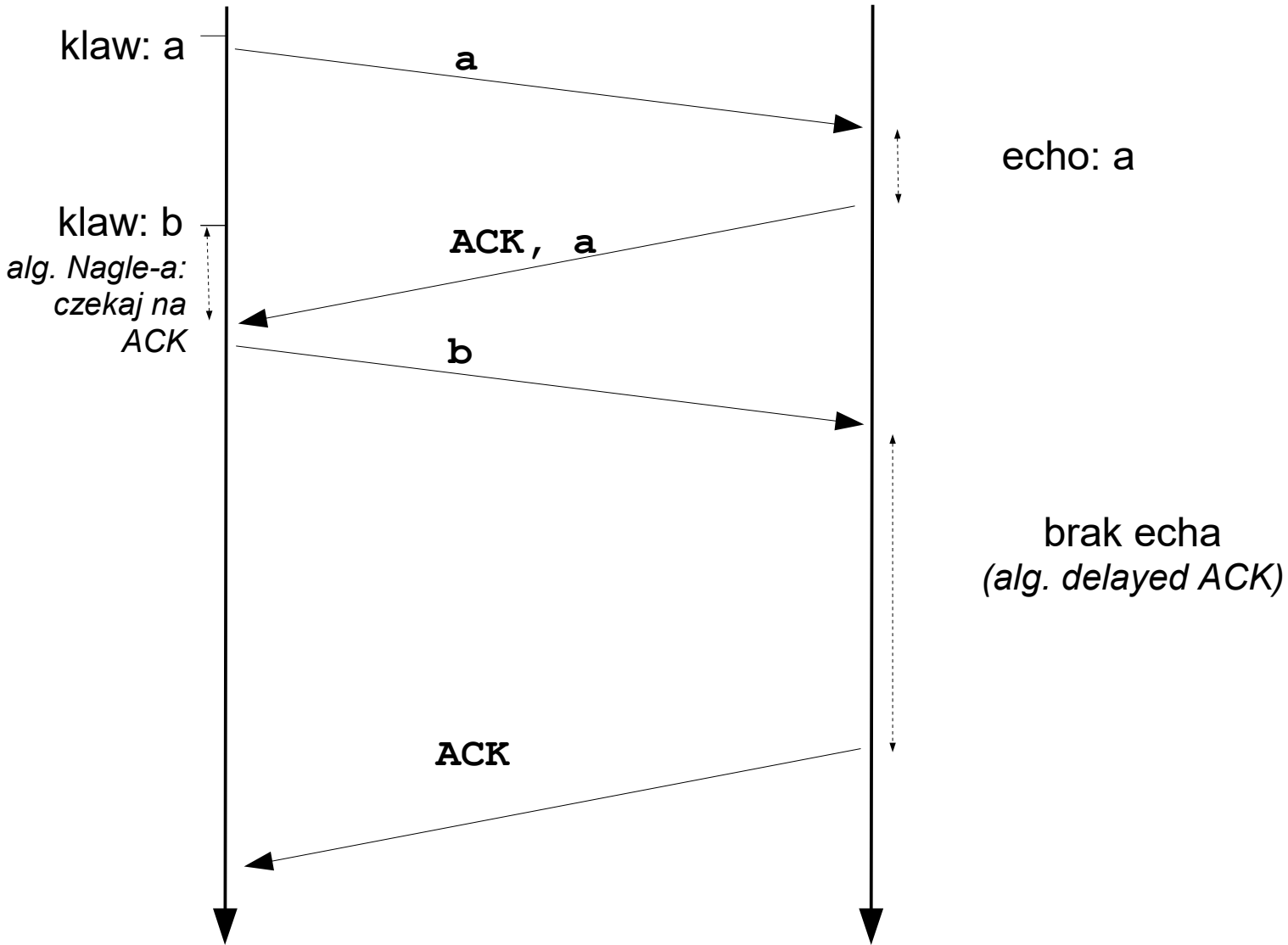
Rlogin - autoryzacja

- Autoryzacja opiera się na mechanizmie rcmd (funkcja):
  - **lokalnej** nazwie użytkownika, **zdalnym** koncie użytkownika
  - Po nawiązaniu połączenia nazwa użytkownika lokalna i zdalna są przesyłane do serwera otwartym tekstem
  - pliku **`.rhosts`** lokalnym dla zdalnego użytkownika
  - ogólnosystemowym pliku **`/etc/hosts.equiv`**
- Możliwe jest więc logowanie na arbitralne konto zdalne - zawsze będzie sprawdzany zdalny plik `~username/`**`.rhosts`**
- rsh - nie pyta o hasło, użytkownicy zaufani (trusted)
- Bezpieczeństwo opiera się na zaufaniu do zdalnego użytkownika root-a (prawo do otwarcia portu < 1024)
- rhosts: pozwala/zakazuje dostępu na podstawie nazw użytkowników, maszyn ("netgroups")

Praca terminalowa - alg. Nagle'-a i alg. opóźnionego potwierdzenia

- Założenie: w miarę możności unikać przesyłania pakietów małych (rozm. < MSS)
- Alg. Nagle'a - nie wysyłaj pakietu małego jeśli poprzednie nie zostały potwierdzone
  - przy trybie konwersacyjnym w wolnej sieci nastąpi sklejenie kilku mniejszych pakietów w jeden
- Alg. "delayed ACK" - nie wysyłaj potwierdzenia od razu po otrzymaniu danych
  - … może pojawią sie jakieś dane do wysłania zwrotnego (po co wysyłać samo potwierdzenie)
  - dobrze sie sprawdza ze zdalnym echem
- **Uwaga** - rozwiązania dobrze sprawują się przy pracy konwersacyjnej, często źle w przypadku innych prot.

Alg. Nagle'-a i alg. opóźnionego potwierdzenia (delayed ACK alg.)



Przekazywanie danych pilnych w prot. interaktywnych - Wstęp - opcje gniazd

```
int getsockopt(int sockfd, int level, int
  optname, char *optval, int *optlen);
int setsockopt(int sockfd, int level, int
  optname, char *optval, int optlen);
```

- level: IPPROTO_IP, IPPROTO_TCP, SOL_SOCKET
- optname - opcja
- optval - wartość ustawianego parametru (często: 0, 1)
- optlen - długość bufora opcji dla opcji niebinarnych (IP_OPTIONS)
- Opcja dla alg. Nagle-a: TCP_NODELAY

Przekazywanie danych pilnych w prot. interaktywnych

- Nadejście danych pilnych (urgent, OOB), powoduje wysłanie sygnału SIGURG
- Dane pilne (1 bajt) mogą:
  - być umieszczone w strumieniu (tj. buforze odbiorczym) danych "zwykłych" (in-line)
  - być umieszczone poza strumieniem danych zwykłych
  - W obydwu przypadkach w strumieniu danych "zwykłych" znajduje się znacznik pozwalający określić położenie danych pilnych
  - Na raz w strumieniu danych może być tylko jeden znacznik OOB (URG)

```
int off=0;
setsockopt(sockfd, SOL_SOCKET,
       SO_OOBINLINE, (char*)&off, sizeof(off));
```

Przekazywanie danych pilnych w prot. Interaktywnych c.d.

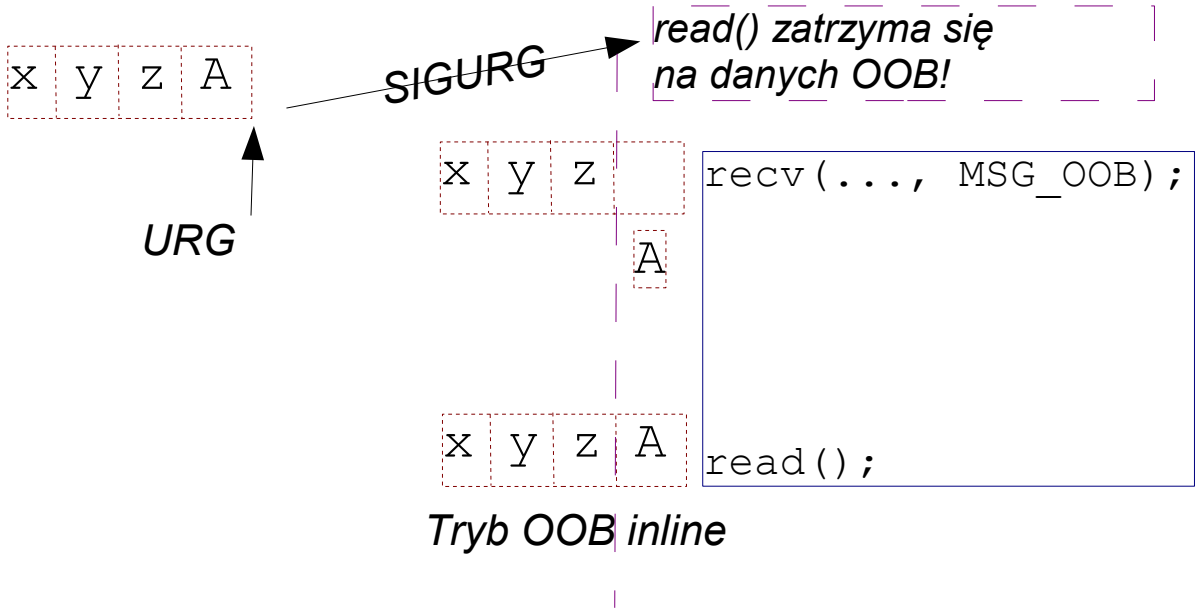
- Nadejście danych pilnych (urgent, OOB), powoduje wysłanie sygnału SIGURG
- Sygnał można przechwycić w f-kcji obsługi sygnału

```
int urg_flg=0;                  /* flaga globalna ! */
int urg_sig(int signum)         /* f-kcja obsługi */
    { urg_flg=1; }
...
    signal(SIGURG, urg_sig);
...
    while (...) {               /* główna pętla serwisowa */
      nsfd = accept( ... )
      ...
      if (urg_flg) do_urg(); /*
    }
```

Transmisja OOB

Nadawca *dane TCP* *Odbiorca*

```
write(s, "xyz", 3);
send(s, "A", 1, MSG_OOB);
```



Przekazywanie danych pilnych w prot. interaktywnych, c.d.

Wczytaj bajt danych OOB inline odrzucając wszystkie poprzedzające go dane:

```
int  ioctl(int fd, int cmd, char *arg);
```

```
for (;;){
  if (ioctl(s, SIOCATMARK, &mark)<0){
    perror("ioctl");
    break; }
  if (mark)              // czy doszliśmy do OOB?
    break;
  read(s, trash, sizeof trash); // nie, odrzuć dane
}
if (recv(s, &oob, 1, MSG_OOB) < 0) { // weź bajt OOB
perror("recv"); …
}
```

SSH – Secure Shell

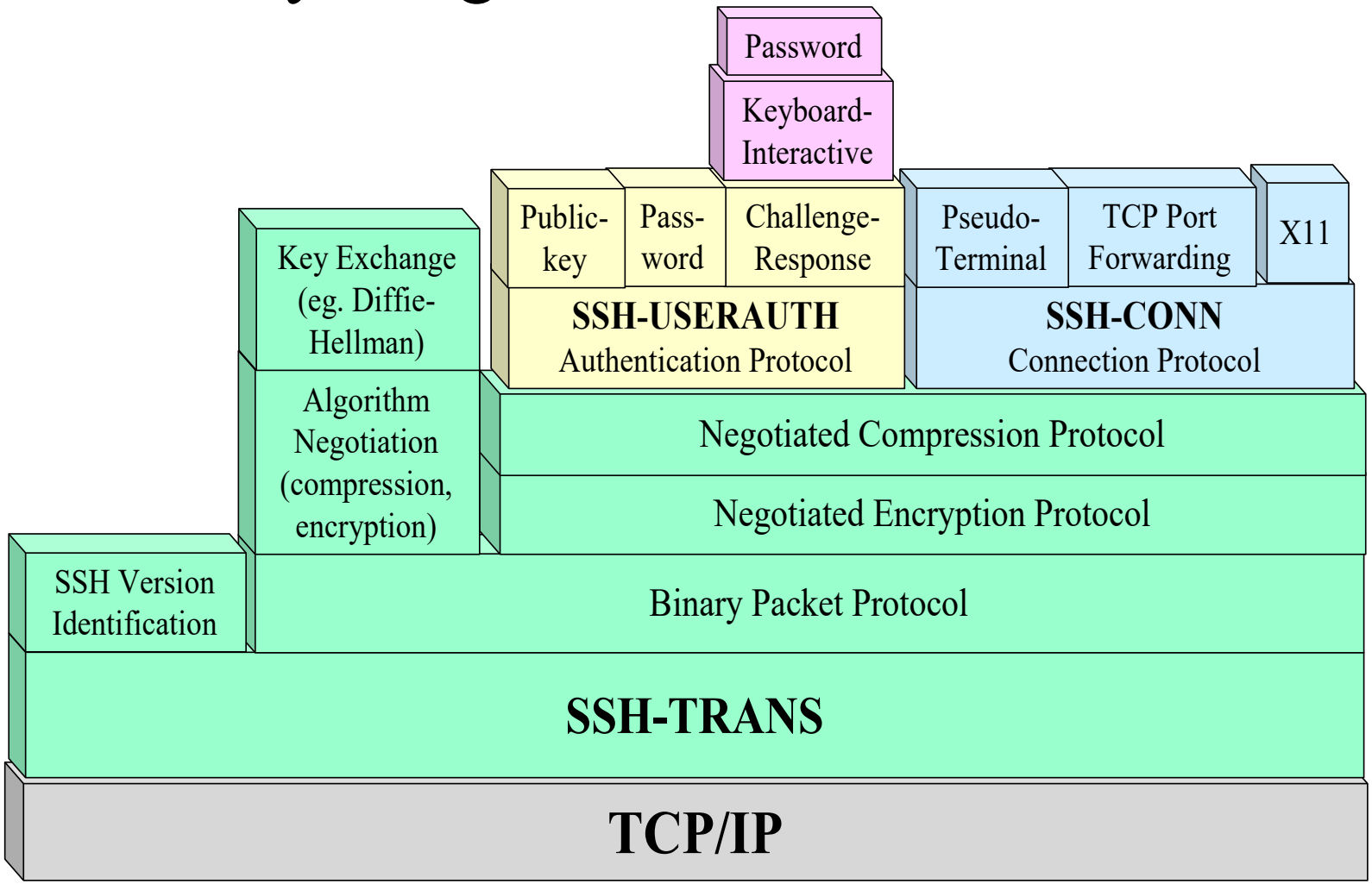
- SSH -standard de facto dla terminalowej pracy zdalnej i transferu plików (sftp)
- Autor: Tatu Ylonen (1995), Helsinki University of Technology, Finlandia
- Przewidziany jako następca telnet i usług “r” BSD
- Może pracować w trybie “port forward” obsługując protokół TCP/IP w tym połączenia X11 (X-Window)
- Używa portu 22
- RFC:
  - *SSH Assigned Numbers (RFC 4250)*
  - *SSH Protocol Architecture (RFC 4251)*
  - *SSH Authentication Protocol (RFC 4252)*
  - *SSH Transport Layer Protocol (RFC 4253)*
  - *SSH Connection Protocol (RFC 4254)*

SSH – Secure Shell

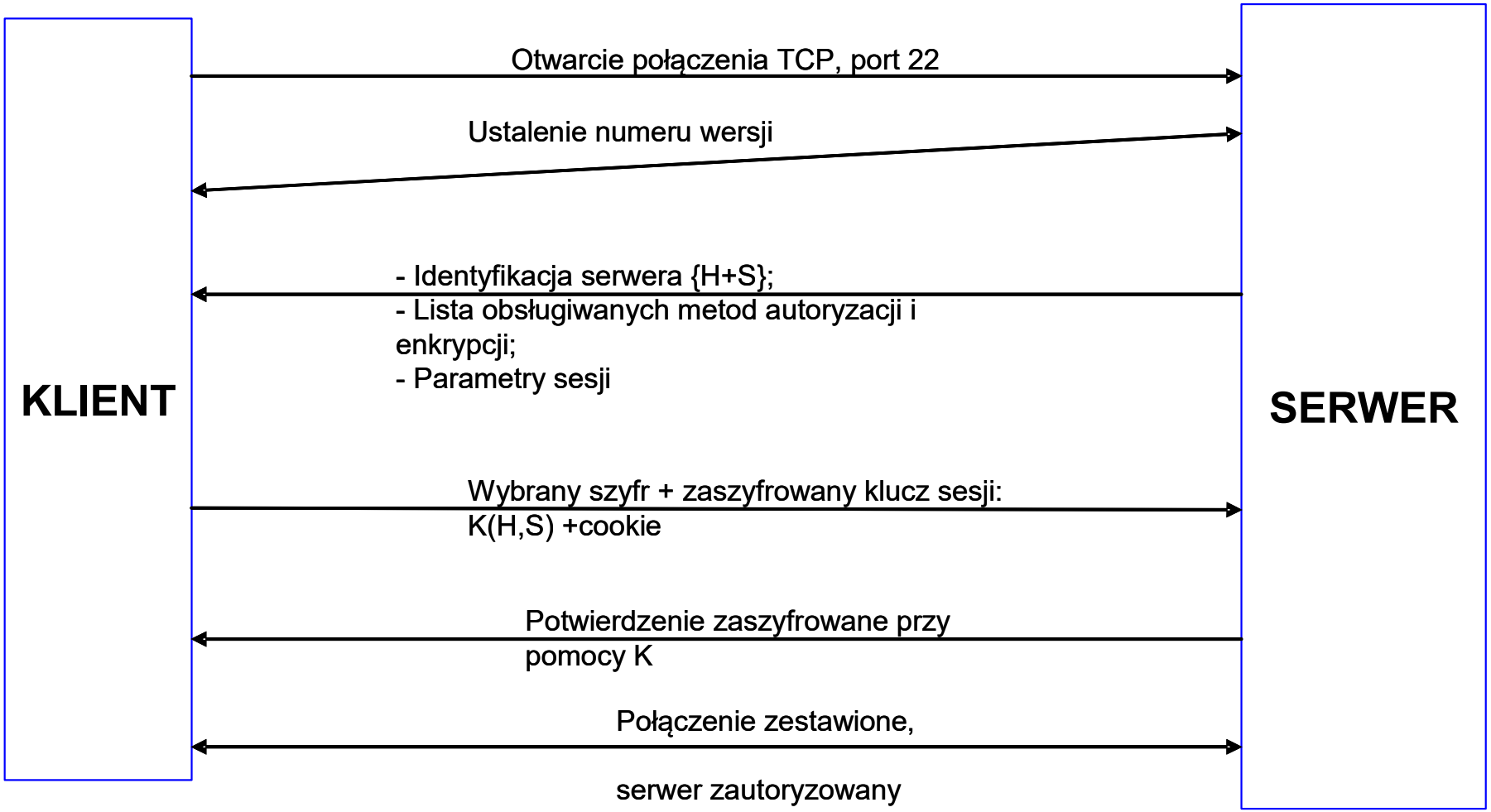
- SSH zapewnia:
  - Poufność (Szyfrowanie)
  - Integralność (kontrola poprawności)
  - Autoryzację (klucze serwerowe i aut. użytkownika)
- Jest odporny na: podsłuchiwanie, IP/DNS spoofing, MITM, przejmowanie połączeń (hijacking)
- Nie jest odporny na: łamanie hasła, ataki na poziomie IP i TCP, statystyczną analizę ruchu
- Wersje: SSH-1 i SSH-2 (problemy licencyjne)
- Platformy: głównie Unix, także Windows (np. putty)

SSH-2

Layering of SSH-2 Protocols



SSH -protokół (1)



H – host key
S- Server Key
cookie- sekwencja 8 losowych bajtów
K- klucz sesji

```
bash-3.00$ ssh -v mulder
OpenSSH_4.3p1, OpenSSL 0.9.7g 11 Apr 2005
debug1: Connecting to mulder [172.22.110.5] port 22.
debug1: Connection established.
...
debug1: Remote protocol version 1.99, remote software version OpenSSH_4.5
debug1: match: OpenSSH_4.5 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_4.3
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client aes128-cbc hmac-md5 none
debug1: kex: client->server aes128-cbc hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
...
debug1: Host 'mulder' is known and matches the RSA host key.
debug1: Found key in /home/gjb/.ssh/known_hosts:19
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEWKEYS sent
...
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey,password,keyboard-interactive
debug1: Next authentication method: publickey
debug1: Trying private key: /home/gjb/.ssh/identity
...
debug1: Next authentication method: password
gjb@mulder's password:
debug1: Authentication succeeded (password).
..
debug1: Entering interactive session.
```

SSH – protokół (2)

- Mogą być używane różne algorytmy kryptograficzne, także w zależności od kierunku przepływu danych
- W transmisji danych używane szybkie alg. CBC (3DES, Blowfish)
- Integralność zapewniona przez Mac (HMAC-SHA1/MD5) (w SSH-2, w SSH-1 zwykłe CRC)
- Autoryzacja serwera: RSA (klucz serwera, trzeba raz zainstalować lub zaakceptować “ręcznie”), albo Diffie-Helman:

```
The authenticity of host 'gw (10.0.0.10)' can't be established.
RSA key fingerprint is
81:f5:da:26:77:31:fc:51:64:3f:97:ec:d7:e9:97:ab.
Are you sure you want to continue connecting (yes/no)?
```

SSH – protokół (3)

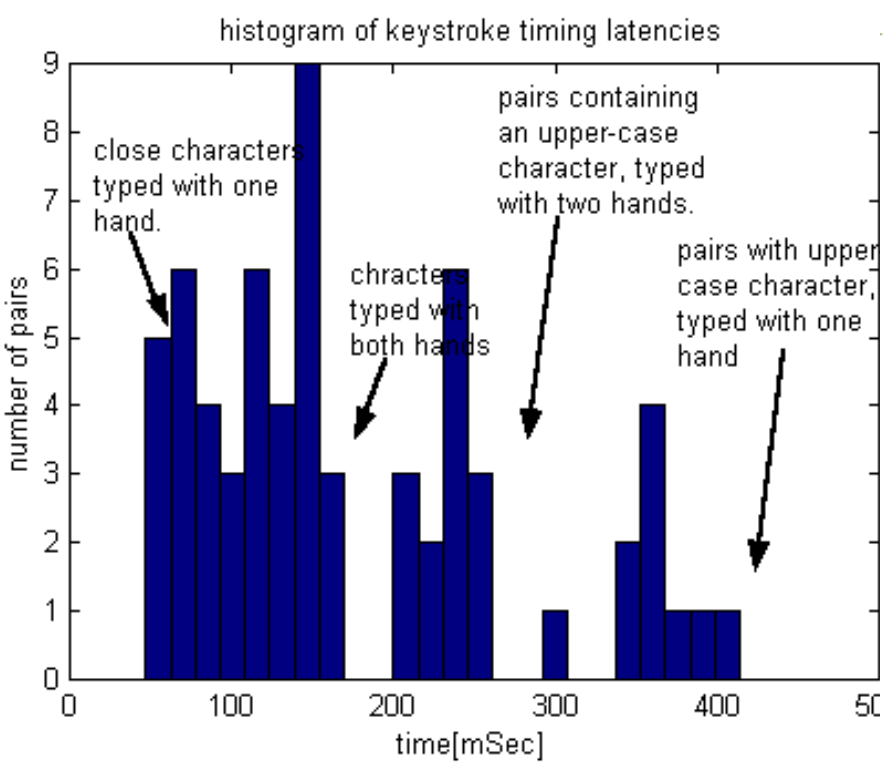
- W wyniku wymiany kluczy (RSA, DH) otrzymujemy ***K*** – wspólny sekret oraz hash ***H***, z których obliczamy:
  - Klucze szyfrujące K-do-S, S-do-K
  - Klucze podpisujące (integralność) K-do-S, S-do-K
- Klucze sesji zmieniane co pewien czas (np. Co 1 GB przesłanych danych)
- Autoryzacja klienta odbywa się dopiero po fazie autoryzacji serwera i zestawieniu sesji (nast. slajd)

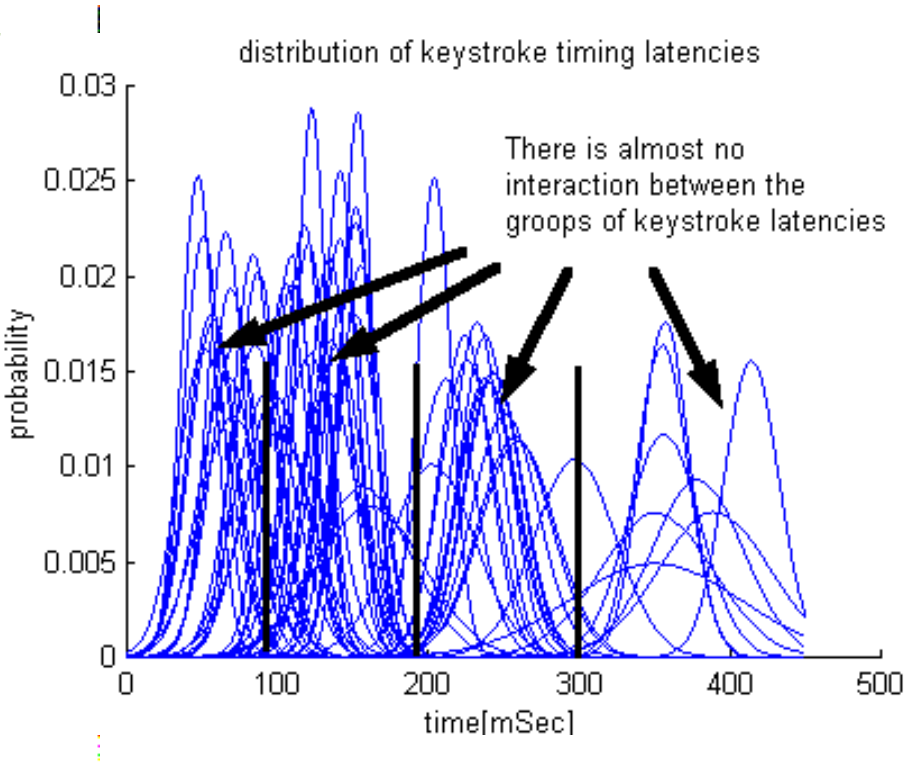
SSH -protokół (4)

- Po autoryzacji serwera autoryzacja klienta, metody:
  - Hasło (OTP)
  - Klucz publiczny
  - Rhosts, RhostsRSA
  - Kerberos

- Integralność danych: CRC (SSH-1), sumy krypto. (SSH-2)
- Stosowana jest też kompresja gzip

Dygresja: atak statystyczny na SSH





- "Key-Stroke Timing and Timing Attack on SSH" *http://comnet.technion.ac.il/~cn19s01*
- Informacja wysyłana po jednym znaku, zależności czasowe pozwalają na wydobycie danych dzięki statystycznej analizie ruchu
- Obrona – dodawania losowych, pustych komunikatów

FTP - File Transfer Protocol

FTP

- Podstawowy protokół wymiany plików - TCP/IP, **RFC 959**, porty 21 i 20
- klient-serwer - wymaga procesu serwera obsługującego FTP na porcie 21 - transfer plików do/z serwera
- Interaktywny
- Prot. stanowy - serwer przechowuje takie informacje jak: aktualny katalog, tryb transmisji, i wiele innych
- Zakłada autoryzację, zwyczajowo pozwala też na dostęp anonimowy
- Cechą charakterystyczną jest komunikacja przez dwa porty (21 - control, 20 - data)

Sesja FTP

3-cyfrowy kod

5xx - wymaga akcji

1xx=OK, I will
2xx=OK, done.
3xx=OK so far
4xx=NO, temp

```
klient%  ftp ftp.ii.pw.edu..pl
Connected to ftp ftp.ii.pw.edu..pl
220 Welcome to II PW FTP Server
530 Please login with USER and PASS.
Name (ftp.ii.pw.edu..pl): anonymous
331 Please specify the password.
Password: XXXXXXX
230 Login successful. Remember - the computer is your friend
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd pub
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxrwxrwx    2 0        0            2048 Aug 23 12:56 mirrors
226 Directory send OK.
ftp> quit
221 Goodbye.
klient%
```

Funkcje klienckie FTP
(klient Unix / linia poleceń)

- open, user
- dir, ls
- cd, lcd, pwd
- binary, ascii, cr
- put, get
- mput, mget, mdelete, mls
- prompt
- nmap
- ntrans

- open [port]
- dir [rdir] [nazwa-pliku]
- cd/lcd - zdalny/ lokalany
- cr - transl. CR/LF
- put plik-lok [plik-odlegl]
- można stosować wzorce

- translacja nazw:
  `nmap $1.$2 $1_$2`
- translacja znaków

Elementy protokołu FTP

Przykładowe polecenia:

- wysyłane w trybie ASCII do portu 21
- **USER** ***username***
- **PASS** ***password*** ***(otwartym tekstem!)***
- **LIST** - zwraca listę plików
- **RETR** **`filename`** pobiera zawartość pliku
- **STOR** **`filename`** wgrywa zawartość pliku na serwer

Typowe odpowiedzi:

- Kod statusu i opis
- **331** **`Username OK, password required`**
- **125** **`data connection already open; transfer starting`**
- **425** **`Can't open data connection`**
- **452** **`Error writing file`**

FTP

- Protokół bazuje na NVT ASCII (z telnet)
- Typy danych: ASCII (NVT-ASCII), image (binarne, 8-o bitowe liczby), local, EBCDIC
- Przy transferze danych klient otwiera port, zaś serwer inicjuje do niego połączenie z danymi wykorzystując port 20 (*czy zachowana jest unikalność asocjacji?)*

  sockets API: `bind(...)`

- Tryb PASV (passive) jeśli serwer nie może zainicjować połączenia do klienta

Polecenia protokołu FTP

- Dostęp: USER, PASS, ACCT (account), REIN (reinitialize - jak logout, bez zamykania poł.), QUIT
- katalogi: CWD (zmiana), CDUP (cd ..)
- Transfer danych: PORT h1,h2,h3,h4,p1,p2; PASV, TYPE (A - ascii, I- image, …)
- Serwis: RETR, STOR, STOU (store unique), APPE (append), ALLO (allocate storage), RNFR/RNTO (rename from/to), ABOR, DELE, RMD/MKD (remove/make directory), PWD, LIST, STAT (status)
- Inne: NOOP, HELP
- Mało używane: struktura pliku (file, record, page); tryby transmisji: strumieniowy, blokowy oraz z kompresją

Kody statusu FTP

- jednolinijkowy:

```
123 opis kodu
```

- wielolinijkowy:

```
123-opis kodu
 nastepna linijka
      234 liczby dozwolone
123 ostatnia linia
```

- Statusy:
  - **1yz** - Positive Preliminary reply (będzie dalszy ciąg, nie wysyłać poleceń)
  - **2yz** - Positivie completion (można przesłać nast. polecenie)
  - **3yz** - Positive Intermediate (OK, wysłać dalszy ciąg polecenia)
  - **4yz** - Transient Negative
  - **5yz** - Permanent Negative

Kody statusu FTP

- **x0z Syntax** - błędy skladni, inne błędy poleceń, niezaimplementowane polecenia
- **x1z Information** - odpowiedzi o charakterze informacyjnym (status, pomoc)
- **x2z Connections** - Odpowiedzi na polecenia sterujące i połączeniowe
- **x3z Authentication and accounting** - login
- **x4z Unspecified as yet**.
- **x5z File system** - związane z systemem plików serwera
- Przykłady:
  - 211 System status, or system help reply
  - 250 Requested file action okay, completed

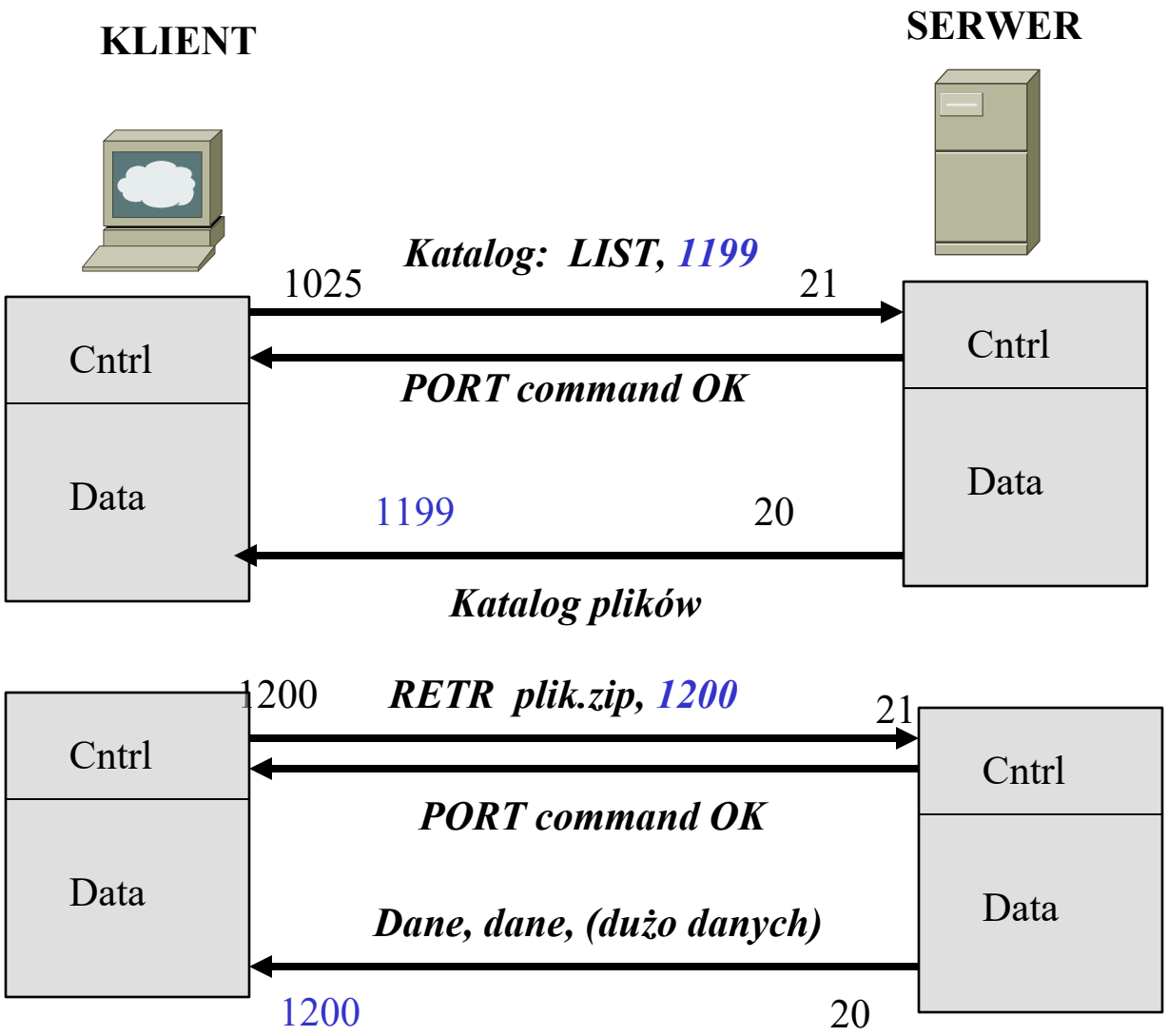
Kody statusu FTP

- 200 Command okay.
- 211 System status, or system help reply.
- 212 Directory status.
- 213 File status.
- 221 Service closing control connection.
- 225 Data connection open; no transfer in progress.
- 226 Closing data connection. Requested file action successful (for example, file transfer or file abort).
- 227 Entering Passive Mode (h1,h2,h3,h4,p1,p2).
- 230 User logged in, proceed.
- 250 Requested file action okay, completed.
- 257 "PATHNAME" created.
- 331 User name okay, need password.
- 350 Requested file action pending further information.

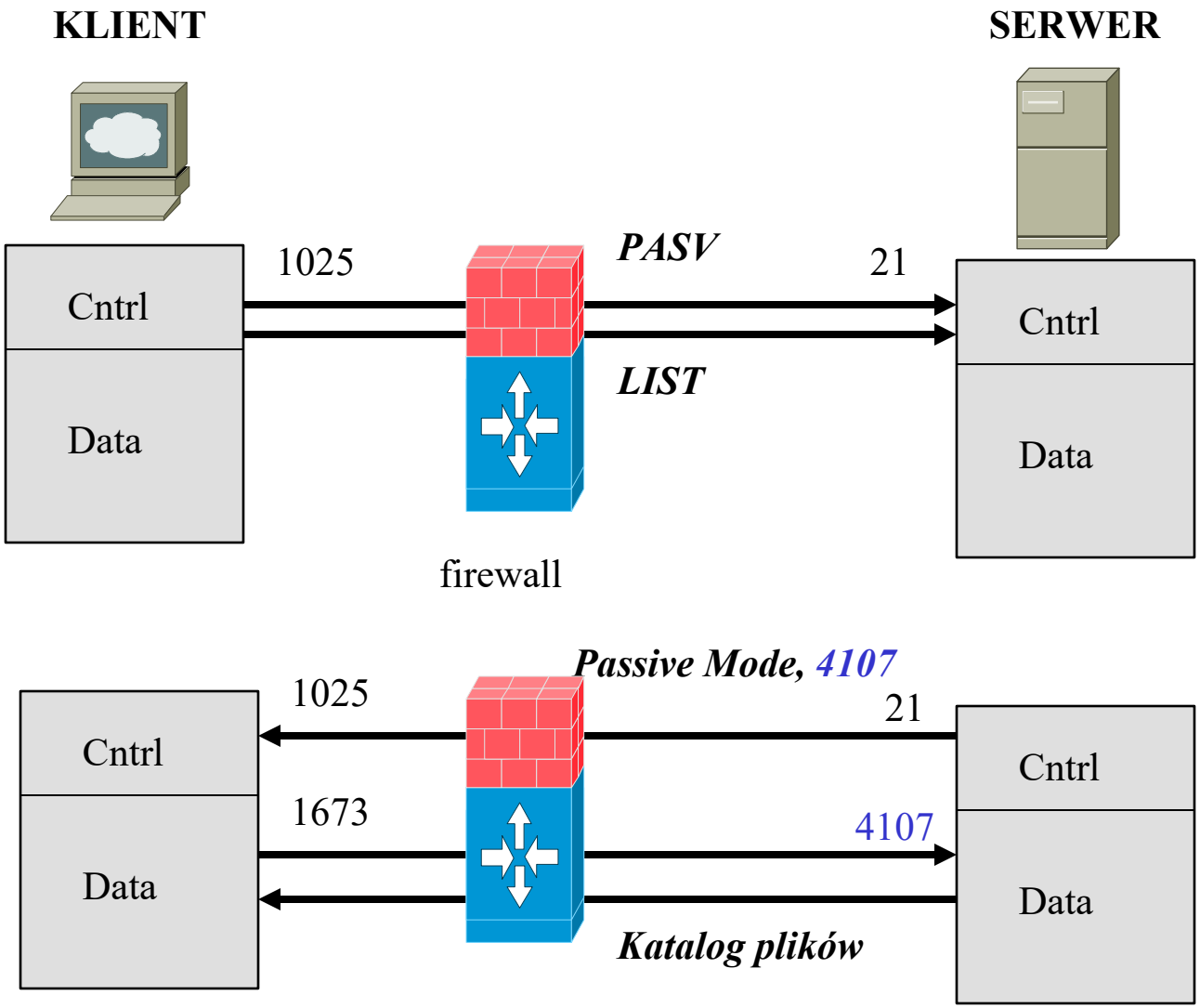
Kody statusu FTP

- 421 Service not available, closing control connection. This may be a reply to any command if the service knows it must shut down.
- 425 Can't open data connection.
- 426 Connection closed; transfer aborted.
- 450 Requested file action not taken. File unavailable (e.g., file busy).
- 451 Requested action aborted: local error in processing.
- 452 Requested action not taken. Insufficient storage space in system.
- 500 Syntax error, command unrecognized..
- 501 Syntax error in parameters or arguments.
- 502 Command not implemented.
- 503 Bad sequence of commands.
- 504 Command not implemented for that parameter.
- 530 User name okay, need password

Sesja FTP - tryb zwykły



Sesja FTP - tryb PASV



Zmiana nr portu w API gniazd

```
int on=1;
setsockopt(sockfd, SOL_SOCKET,
      SO_REUSEADDR, (char*)&on, sizeof(on));
bind(sockfd, (struct sockaddr*)&newsin,
   sizeof(newsin));
```

- Włączenie opcji SO_REUSEADDR pozwoli na ponowne dowiązanie do gniazda adresu, w szczególności pozwala to na zmianę numeru portu.