Introduction to Network Security

Chapter 11

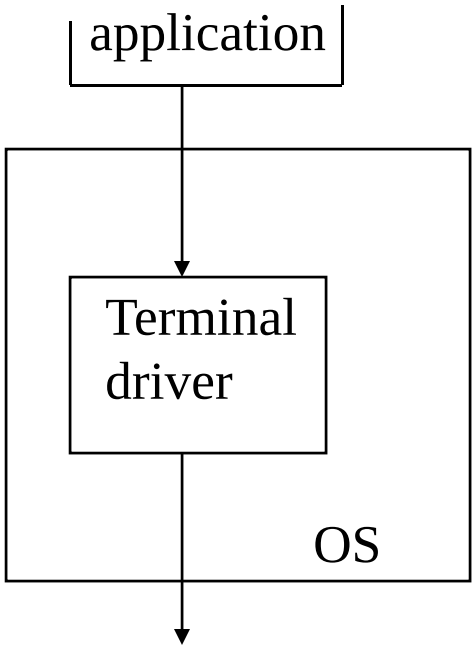
Remote Access Security

Topics

- Remote Access
  - Telnet
  - Rlogin
  - X-Windows
  - FTP
  - General Countermeasures
- Peer-to-Peer Protocols
- Anonymous services & Privacy
- General countermeasures

Telnet

TELNET: a Virtual Terminal Protocol that provides interactive access to remote computers

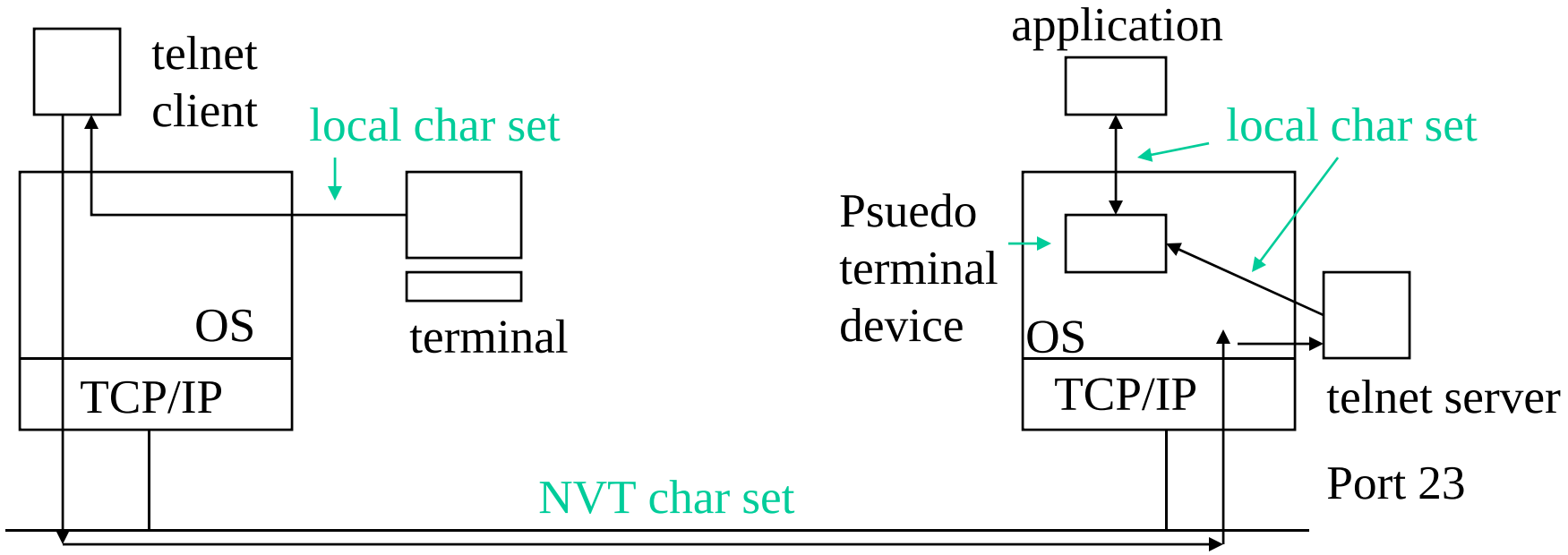


The protocol defines:

- Format of data
- How control signals are passed and how to distinguish them from data
- Data transfer mode
  (half/full duplex, sync/async)
- How out-of-band signals are passed
- How data delivery is controlled

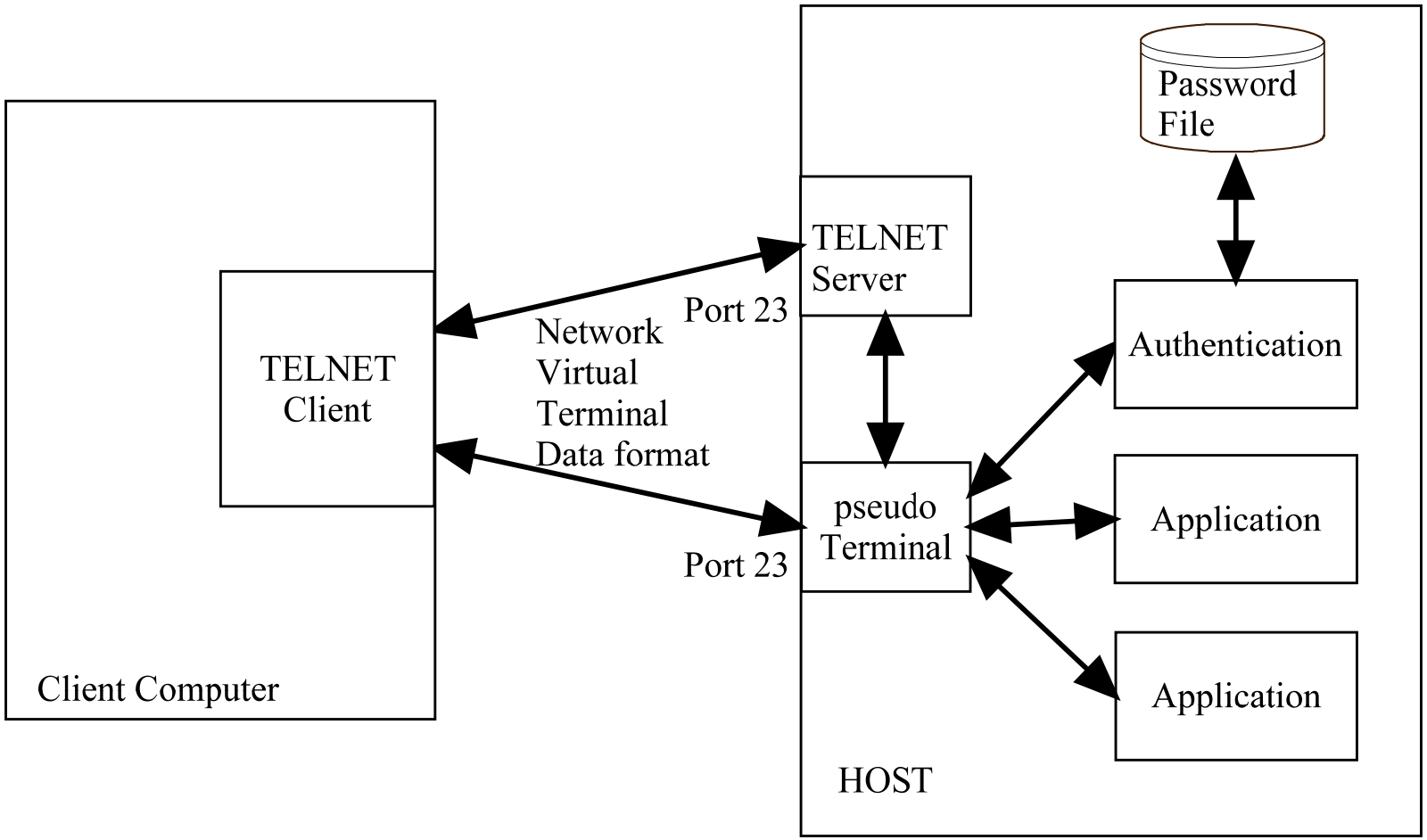
Telnet

NVT – Network Virtual Terminal



Local charsets of different OS's may not be compatible. When sending over the network, the local charset is translated to the common NVT charset by the telnet client. The telnet server then translates the NVT charset to the local charset

Telnet


Password
File
TELNET
Server
Port 23
Network
Virtual
Terminal
Data format
TELNET
Client
Authentication
pseudo
Terminal
Application
Port 23
Application
Client Computer
HOST

Telnet

The virtual terminal consists of a display and a printer

- Display
  - Characters are 7 bit ASCII
  - Operates in scroll mode with unlimited line length, unlimited lines per page
  - Must be able to generate control signals:

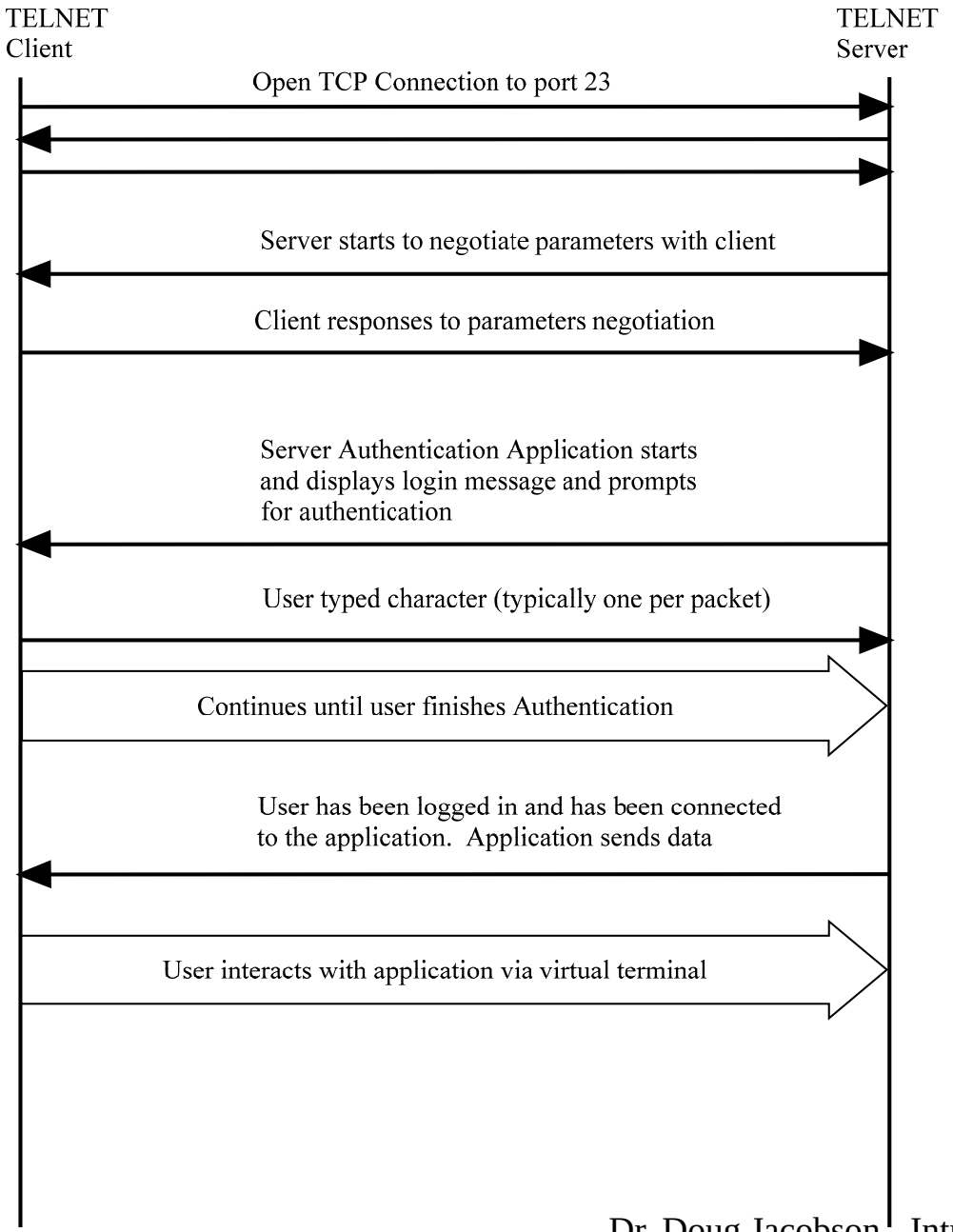
    Are You There — Interrupt Process

    Abort Output — Erase Character

    Erase line — Break
- Printer
  - Has unspecified line width and page length
  - Can print the 95 ASCII graphic characters
  - Can respond to the control codes:

    NUL — Line Feed — Carriage return

Telnet


TELNET
Client
TELNET
Server
Open TCP Connection to port 23
Server starts to negotiate parameters with client
Client responses to parameters negotiation
Server Authentication Application starts
and displays login message and prompts
for authentication
User typed character (typically one per packet)
Continues until user finishes Authentication
User has been logged in and has been connected
to the application. Application sends data
User interacts with application via virtual terminal

Telnet Commands

Definition	Abbr	code	
End of subnegotiation	SE	240	
No Operation	NOP	241	
Data Mark: A stream sync character	DM	242	
Break	BRK	243	
Interrupt Process		IOP	244
Abort Output	AO	245	
Are You There	AYT	246	
Erase Character		EC	247
Go Ahead: turn line around for half duplex		GA	249
Begin subnegotiation	SB	250	
WILL		251	
WONT		252	
DO		253	
DON'T		254	
Interpret as CMD		IAC	255

Telnet Commands

How to mix user data and commands:

user data:

command:

There is a special command to transfer 8 byte data

Telnet Options

- Options can be negotiated by telnet processes
- New options can be accommodated since they are not part of the standard
- Three categories
  1. Enhance, change, and refine NVT characteristics (e.g. line width)
  2. Change transfer protocol (e.g. suppress GO AHEAD)
  3. Information to be passed to the host (e.g. status, terminal type)

Telnet Options

This is just a subset of the options defined in many different RFC's:

ID	Name	RFC	Category
0	Binary transmission	856	2
1	echo	857	1
5	status	859	3
8	output line width		1
9	Output page size		1
10	Output <cr> disposition	652	1
24	terminal type	930	3
25	End of record	885	3

Telnet Negotiation

Option negotiation rules:

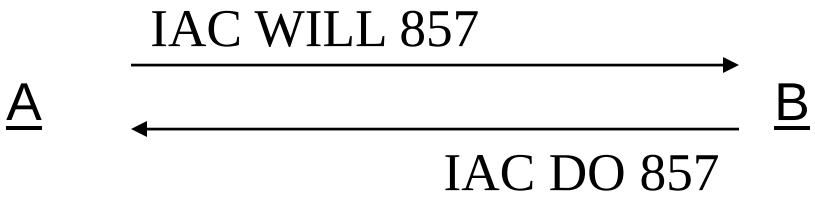
- May reject a request to enable an option
- Must accept a request to disable an option
- Options are not enabled until negotiation is complete
- Never negotiate an option that is already true

Telnet Negotiation

Option negotiation commands:

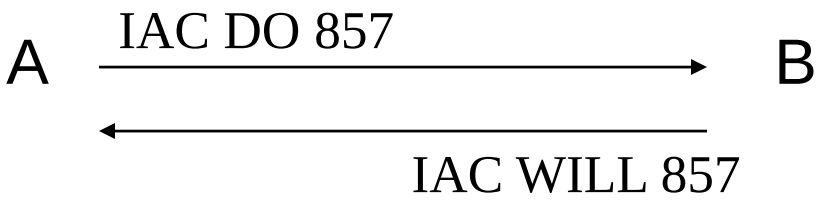
- WILL — Sender wants to enable the option
- WONT — Sender does not want to enable the option
- DO — Sender would like the other side to enable the option
- DON'T — Sender would not like the other side to enable the option

Example 1: Side A wants to enable ECHO (857), side B agrees

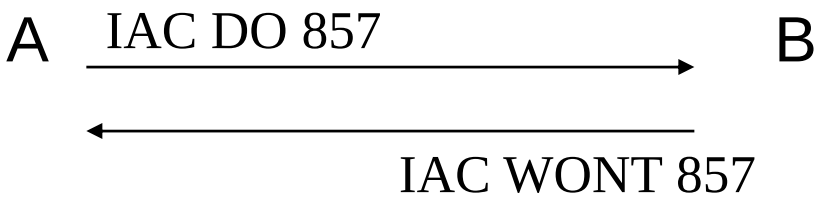


Telnet Negotiation

- Example 2: A would like B to enable ECHO, B agrees

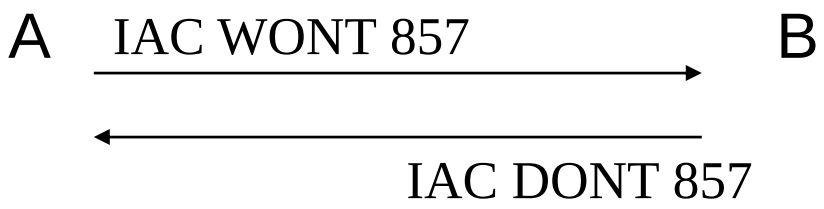


- Example 3: A would like B to enable ECHO, but B does not agree

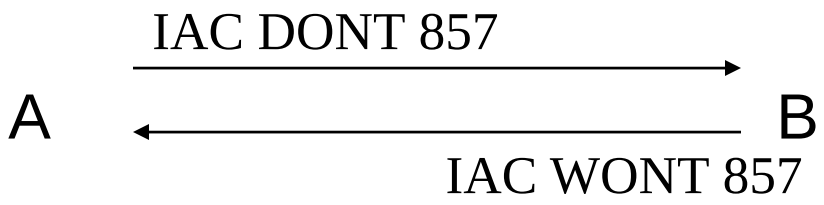


Telnet Negotiation

- Example 4: A would like to disable echo, B MUST agree



- Example 5: A would like B to disable echo, B must agree

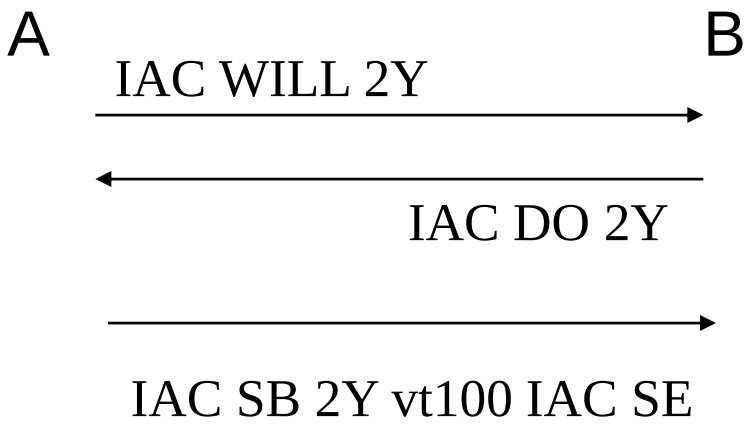


Telnet Negotiation

Suboptions

SE	240	suboption end
SB	241	suboption begin

Example: A wants to set the terminal type (2Y) to vt100

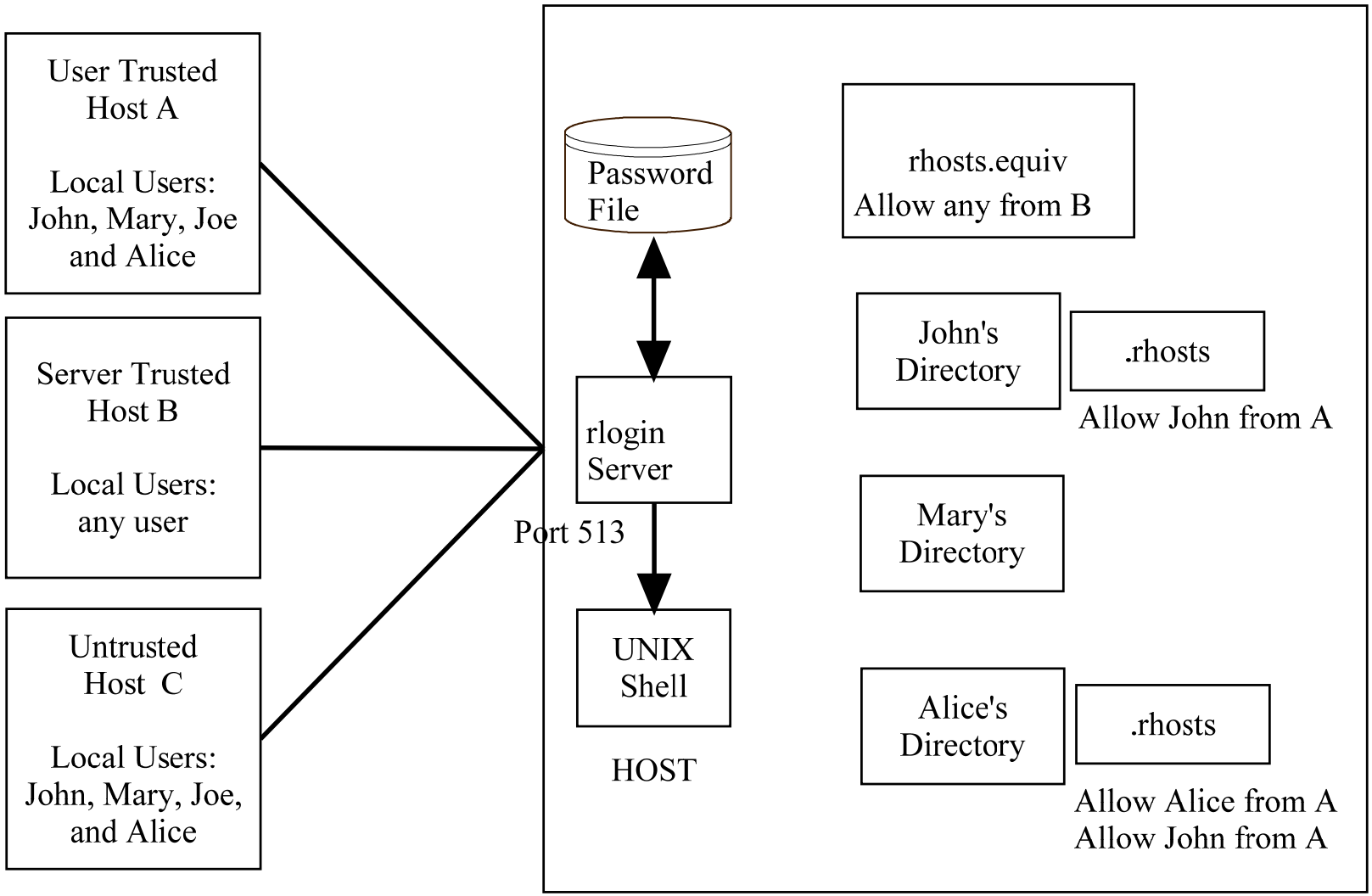


Direction	Data	Comments
C ← S	0xff 0xfd 0x01 0xff 0xfd 0x22 0xff 0xfb 0x05	IAC, Do Echo (request client echoes) IAC, Do linemode (request client sends a line at a time) IAC, Will Status (server wishes to send status info)
C → S	0xff 0xfb 0x01 0xff 0xfc 0x22 0xff 0xfe 0x05	IAC, Will Echo (client will echo characters) IAC, Won't linemode (Client will not do linemode) IAC, Don't Status (client does not want server to send status information)
C ← S	0xff 0xfe 0x01 0xff 0xfb 0x01	IAC, Don't Echo (tell client not to echo) IAC, Will Echo (tell client server will echo)
C → S	0xff 0xfc 0x01 0xff 0xfd 0x01	IAC, Won't Echo (tell server client will not echo) IAC, Do Echo (tell server it is OK to echo)
C ← S	\r\n Login:	Send authentication application prompt
C → S	j	First char of user name
C ← S	j	Echo of the character
		Repeat until enter key is pressed
C → S	\r\n	Send carriage return + linefeed
C ← S	\r\n	Echo carriage return + linefeed
C ← S	Password:	Send authentication application prompt
C → S	p	First char of password (server will not echo)
		Repeat until enter key is pressed
C → S	\r\n	Send carriage return + linefeed
C ← S	\r\n	Echo carriage return + linefeed
C ← S		User is now connected and server application will send message.

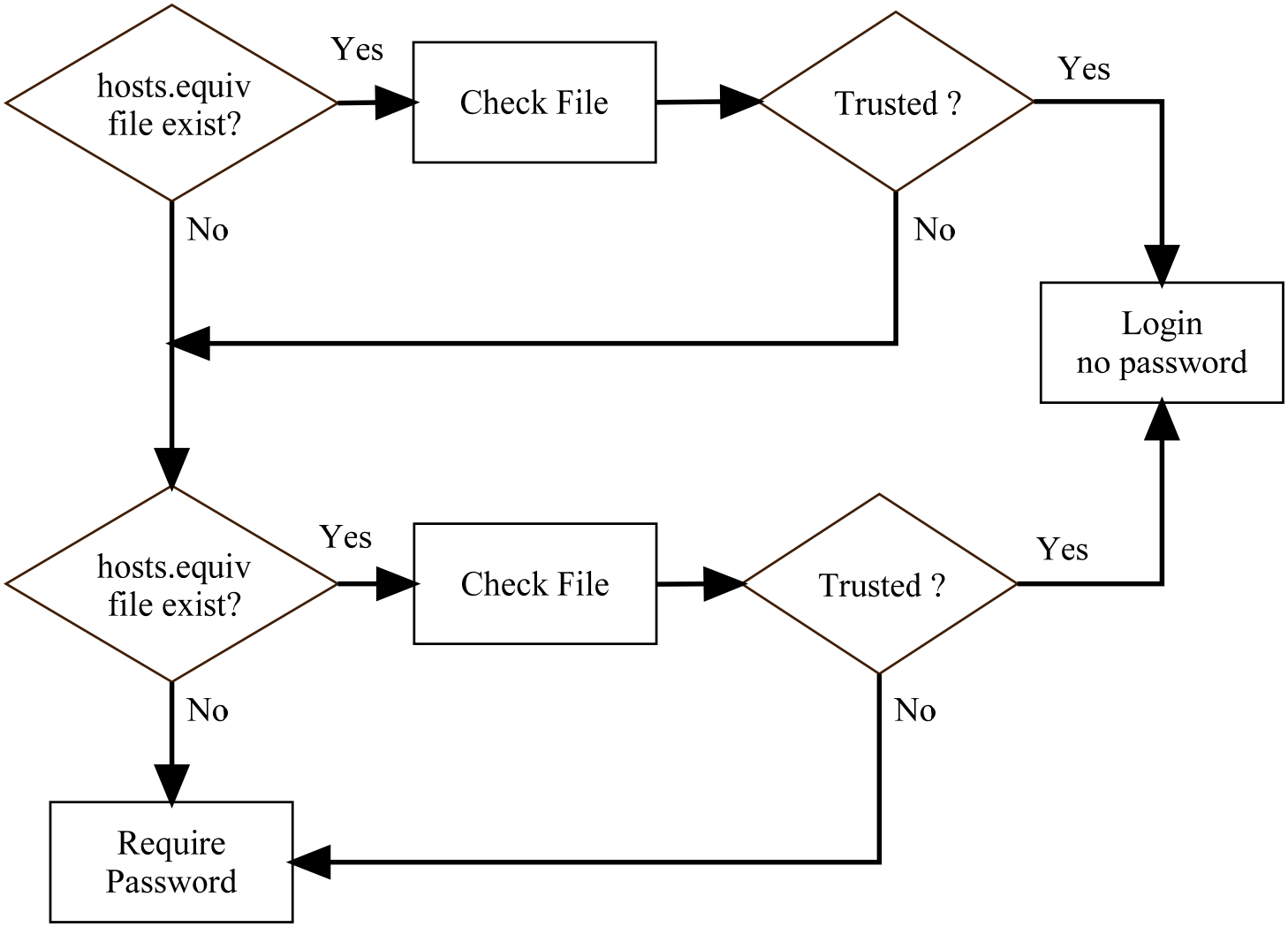
Rlogin

- Remote login (rlogin)
- Similar to telnet, but much simpler
- Designed for unix to unix communication
- Possible for hosts to login without a password
- Uses port 513
- Sequence:
  - Client sends: \0
local login name
\0
server login name
\0
terminal type
\0
  - Server sends: \0

Rlogin


User Trusted Host A
Local Users: John, Mary, Joe and Alice
Server Trusted Host B
Local Users: any user
Untrusted Host C
Local Users: John, Mary, Joe, and Alice
Password File
rlogin Server
Port 513
UNIX Shell
HOST
rhosts.equiv
Allow any from B
John's Directory
.rhosts
Allow John from A
Mary's Directory
Alice's Directory
.rhosts
Allow Alice from A
Allow John from A

rlogin server trust



rlogin trust

Client host	Client side user	Server side user	Result
A	John	John	Trusted
		Mary	Not Trusted
		Alice	Trusted
	Mary	John	Not Trusted
		Mary	Not Trusted
		Alice	Not Trusted
	Joe	John	Not Trusted
		Mary	Not Trusted
		Alice	Not Trusted
	Alice	John	Not Trusted
		Mary	Not Trusted
		Alice	Trusted
B	Any User	Any User	Trusted

rlogin trust

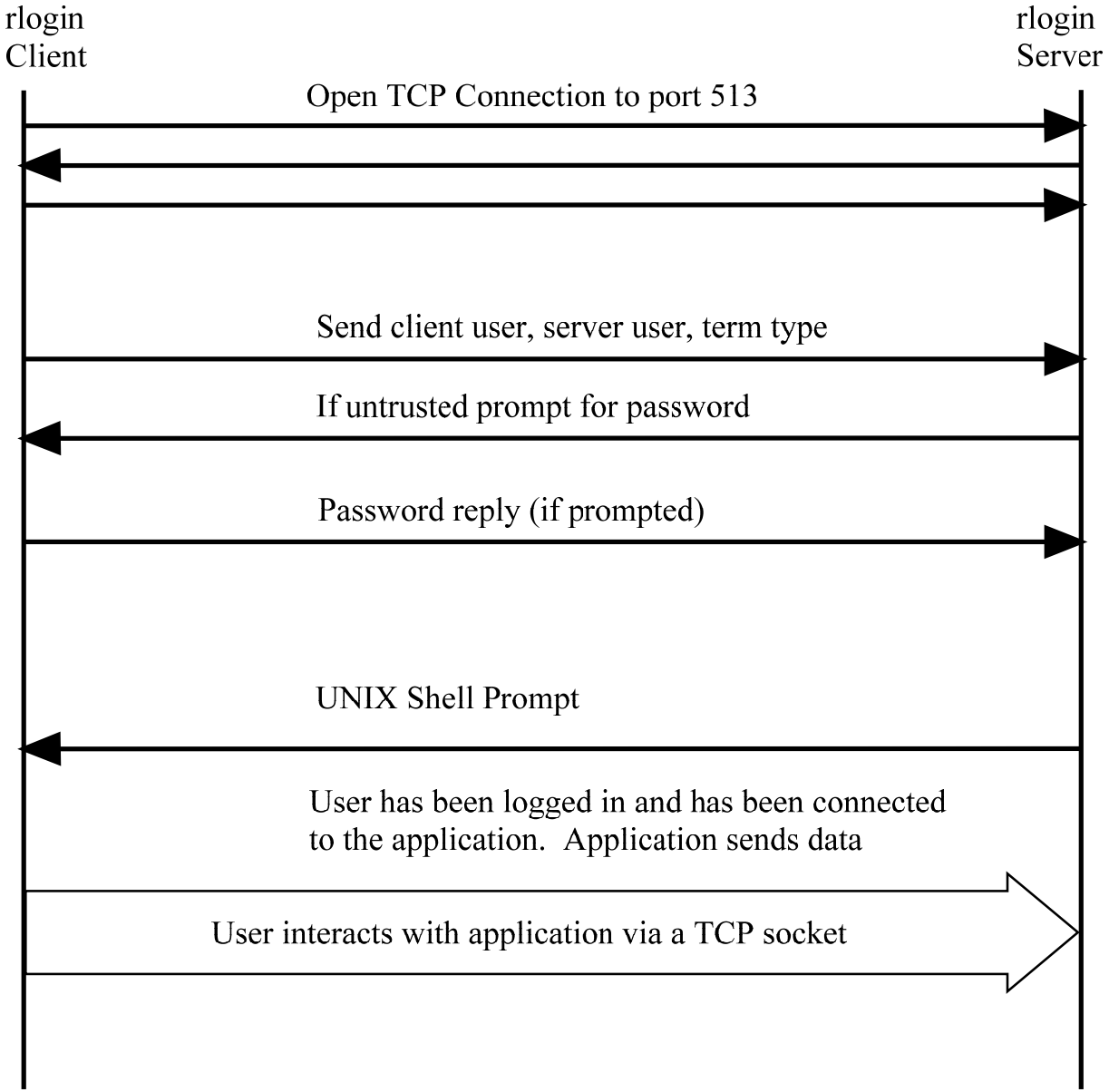
Client host	Client side user	Server side user	Result
C	John	John	Not Trusted
		Mary	Not Trusted
		Alice	Not Trusted
	Mary	John	Not Trusted
		Mary	Not Trusted
		Alice	Not Trusted
	Joe	John	Not Trusted
		Mary	Not Trusted
		Alice	Not Trusted
	Alice	John	Not Trusted
		Mary	Not Trusted
		Alice	Not Trusted

Rlogin commands

- Commands are distinguished by 0xFF
  - Remote flow control 0x10
  - Local flow control 0x20
  - Window size 0x80

  (asks client for current window size)
- Escape character: ~ ^d
- Everything is sent in clear text

rlogin


rlogin
Client
rlogin
Server
Open TCP Connection to port 513
Send client user, server user, term type
If untrusted prompt for password
Password reply (if prompted)
UNIX Shell Prompt
User has been logged in and has been connected
to the application. Application sends data
User interacts with application via a TCP socket

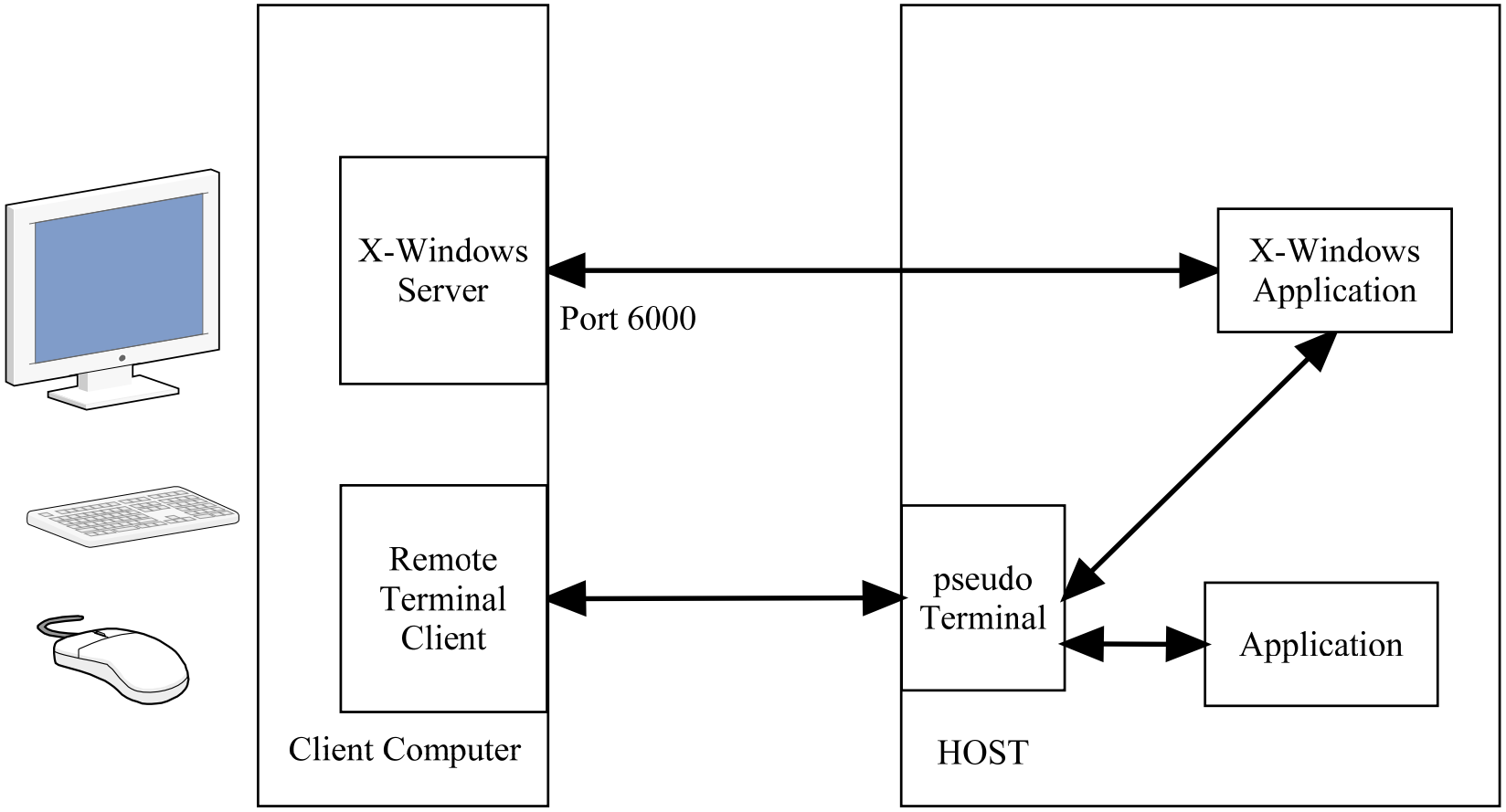
rlogin

Direction	Data	Comments
C → S	john 0x00 john 0x00 xterm\34800 0x00	Client side username Server side username Terminal type and speed
		If authentication is required (user is untrusted)
C ← S	Password:	Prompt for password
C → S	p	First char of password (server will not echo)
		Repeat until enter key is pressed
C → S	\r	Send carriage return
C ← S	\r\n	Echo carriage return + linefeed
		If authentication worked or user was trusted
C ← S	Data from server	User is now connected and server will display the UNIX shell prompt.

X windows

- The user sits on the server side of X windows
  - Usually telnet into client and start X window client
  - X windows then starts and the client authenticates to the X windows server
  - X windows sends information in clear text

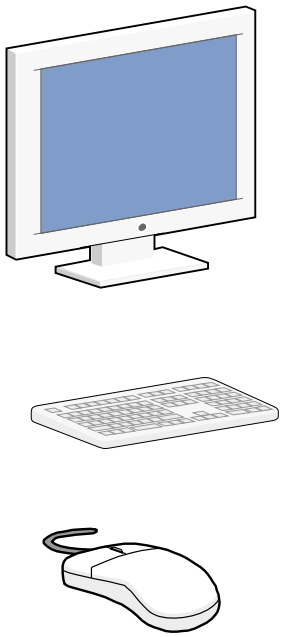
X-windows

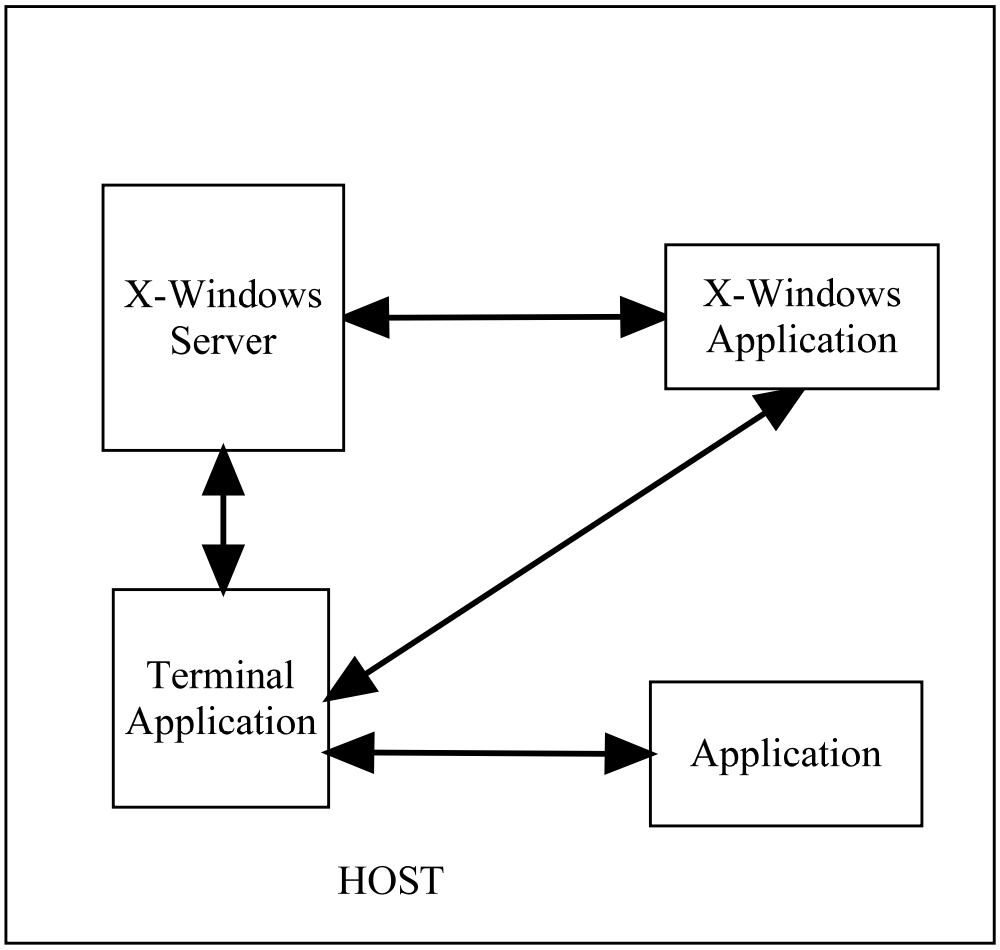

X-Windows Server
Port 6000
X-Windows Application
Remote Terminal Client
pseudo Terminal
Application
Client Computer
HOST

Communication

- In order for two programs to communicate in Unix, a pipe is created between the two processes
  - Pipe works like it sounds, put data in on one side comes out the other
  - Pipe created in the tmp directory
- Port 6000

Local X-Windows

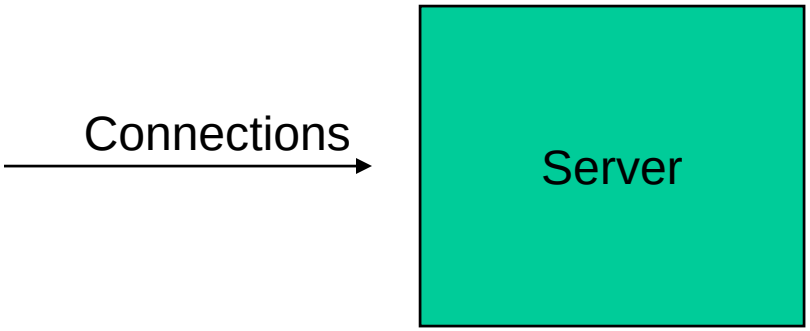




Server Side

- X windows offers up your computer to the outside world to manipulate
- Pc also has public domain X windows programs
- Xhost determines who can connect to your server
  - Xhost + would allow all to connect to one's X windows
- X windows is designed to allow applications control over the display
- Client side
  - How does client know which server to connect to
  - Variable called display
    - :0.0 display means local display
    - The second number is the monitor
    - If remote machine:0.0 which is set on the client
      - Tells X windows to point to server

Server Side cont…



- Authentication?
  - Xhost command, indicates who can connect to one's server, which is IP address based authentication
  - Xhost + allow all connections
  - Xhost - allows nobody

- Command set is designed to allow total control over input and display
  - Through X windows, hackers could
    - Capture screen
    - Capture keystrokes
    - Create, destroy windows
    - Enter key strokes into windows

Local Side

- Pipe
  - /tmp/.X11 …
  - Tmp directory is shared and is world read writable
  - Can do denial of service by deleting the pipe in the tmp directory
    - No new clients can connect
    - Current clients stay connected

Header Based

- For Telnet and rlogin there is not much of a header.
- X-Windows there is possible buffer overflow attacks.

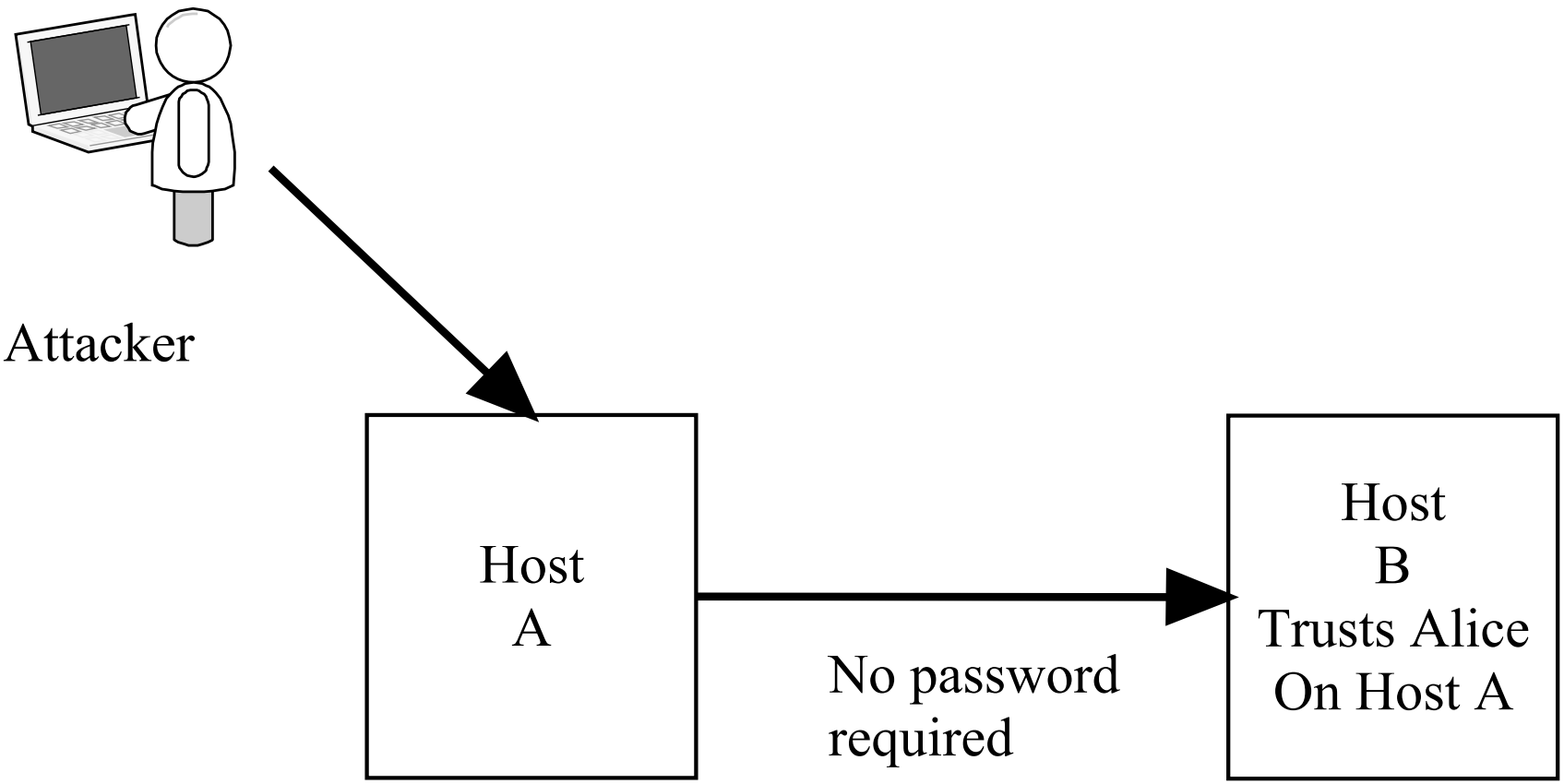
Protocol Based

- Telnet and rlogin have a simple protocol and there is not any attacks, other than telnet can be used to connect to any service (not really a flaw)
- X-Windows has some issues with the protocol since the protocol gives the application control over the remote computer.

Authentication Based

- Telnet offers access to the remote machine and to the login prompt
- Rlogin does not need password unless setup correctly. Uses IP address for authenticator
- X-Windows
  - server can allow any machine to control it based on the IP address
  - Client uses machine authentication to allow a user to run the application

Authentication Stepping stone



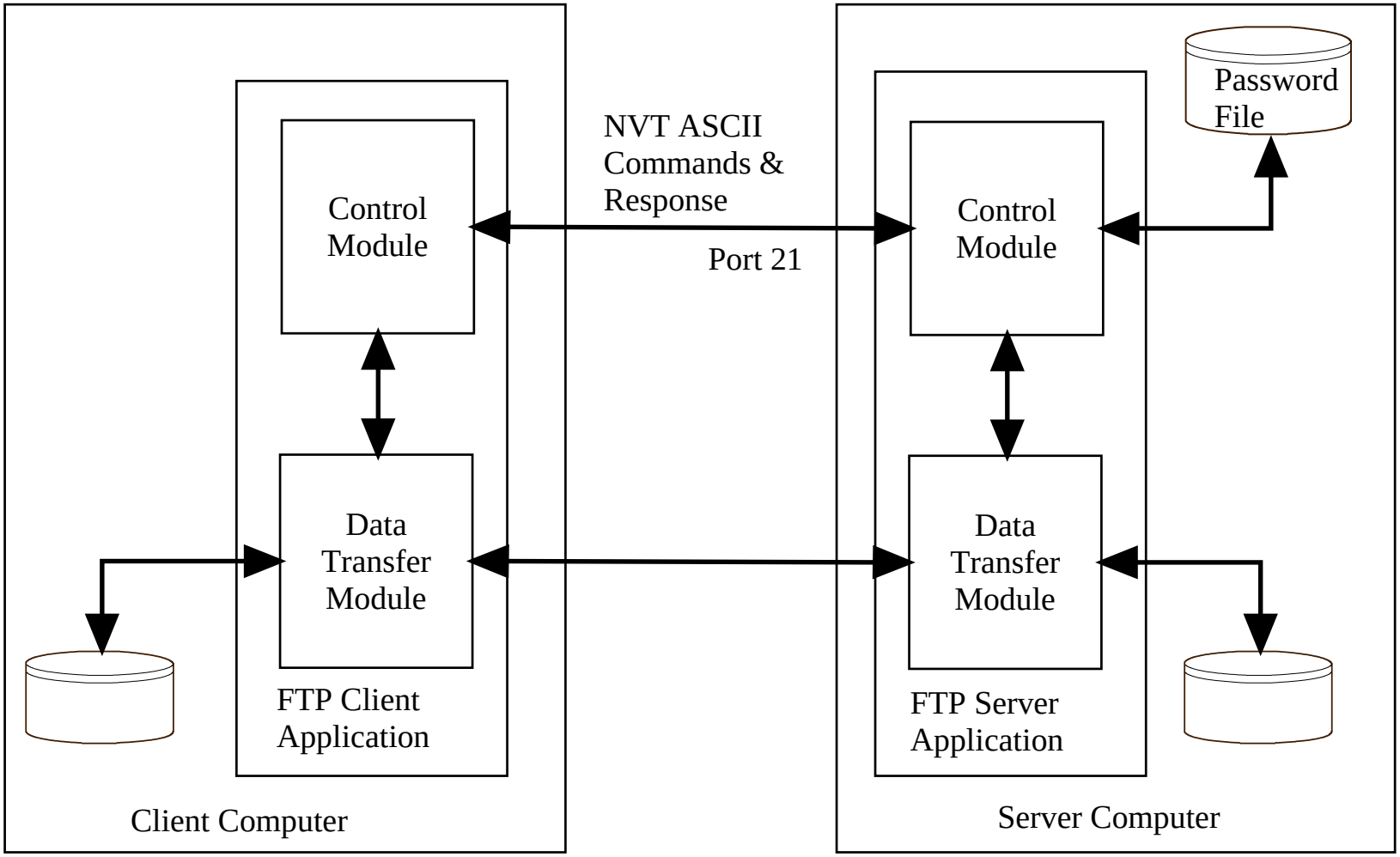
Traffic Based

- All three are clear text (sniffing)
  - Usernames & Password
  - Commands and text

FTP

- Commonly used files transfer protocol
- Uses a command channel and a data channel
- Command channel is used to control the FTP session and remains open for the entire FTP session.
- The data channel is used to transfer data between the client and the server
- A new data channel connection is opened for each data transfer.

FTP Command & Data Channels



FTP Commands

- The next slide lists the common FTP commands
- The commands are sent as ASCII text and the responses to the commands are also ASCII.

Command	Action
Authentication	
USER username	Send the username to the server
PASS password	Send the user password to the server
QUIT	Finish session
File Management	
CWD directory_name	Change directory on the server
CDUP	Change to the parent directory on the server
DELE filename	Delete the file from the server
LIST directory_name	List the files on the server
MKD directory_name	Make a new directory on the server
PWD	Print the current directory on the server
RMD directory_name	Delete a directory from the server
RNFR old_file_name	Name of file on the server to be renamed
RNTO new_file_name	Name of file on the server to rename the file to
Data Format	
TYPE (A, I)	Set data transfer type, A=ASCII, I=Image
Data port	
PORT 6 digit identifier	Client sends the port number for the server to connect to for the data transfer
PASV	Server send the port number for the client to connect to for the data transfer
File Transfer	
RETR filename(s)	Transfer the file(s) from the server to the client using the data connection
STOR filename(s)	Transfer the file(s) from the client to the server using the data connection
Miscellaneous	
HELP	Server will return information

Response codes

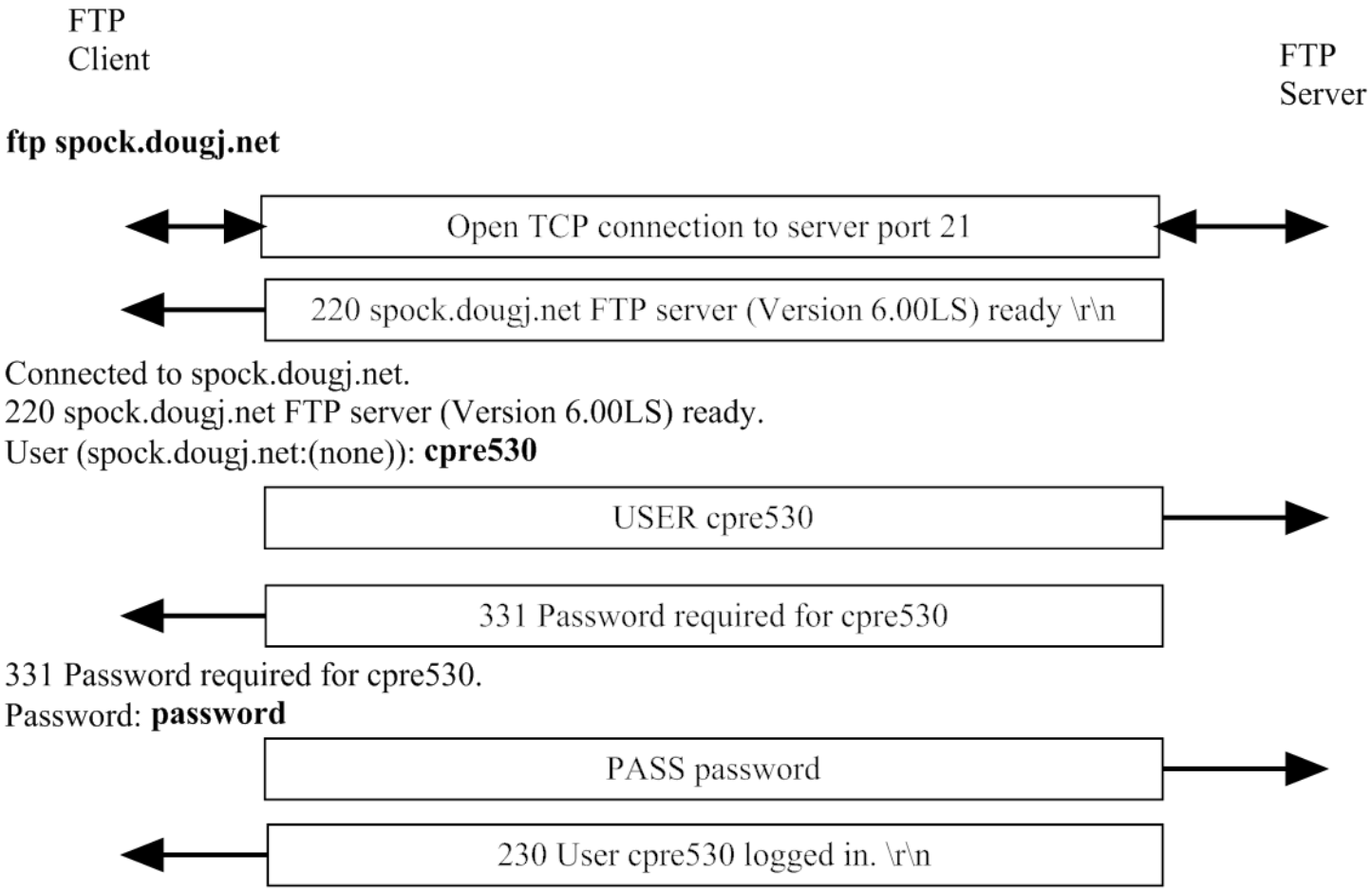
Code	Response Status
1XX	Positive Preliminary Reply – Indicates the server will respond with another response code before the client can continue.
2XX	Positive Completion Reply – Indicates the command was successful and a new command can be issued.
3XX	Positive Intermediate Reply – Indicates the command was successful, but the action is held up pending receipt of another command from the client.
4XX	Transient Negative Completion Reply – Indicates the command was not accepted, however the error is temporary.
5XX	Permanent Negative Completion Reply – Indicates the command was not accepted.

Code	Response type
X0X	Syntax Error or unimplemented commands
X1X	Information – reply to a request for information
X2X	Connections – Reply to a request for connection
X3X	Authentication – Reply to authentication commands
X4X	Unspecified
X5X	File System – Reply to file system based requests

Common Response Codes

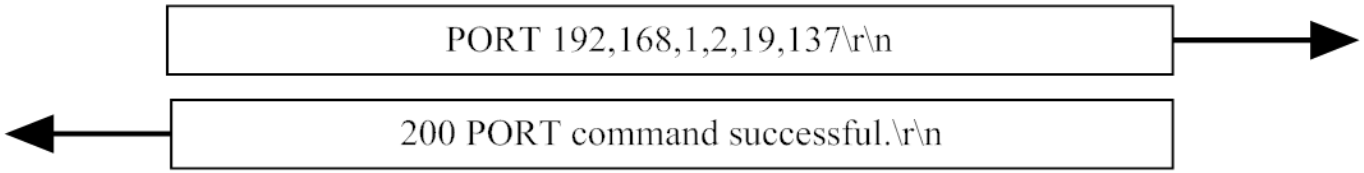
Code	Responses
150	Data connection will open
200	Command acknowledgement
220	Service ready
225	Data connection open
226	Closing data connection
230	User logged in
331	User needs password
425	Cannot open data connection
500	Syntax error
530	User login failure

FTP Protocol Exchange

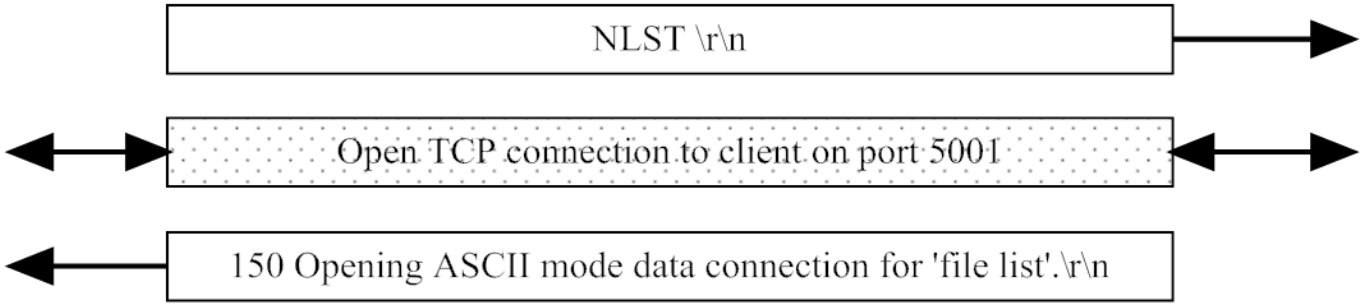


FTP Protocol Exchange

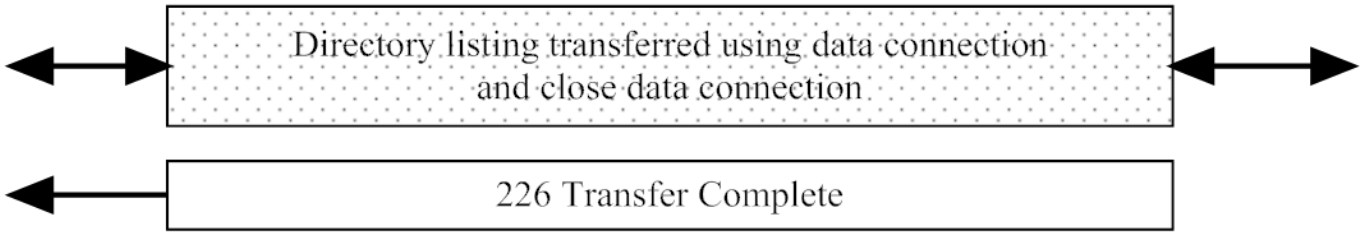
230 User cpre530 logged in.
ftp> **ls**



200 PORT command successful.

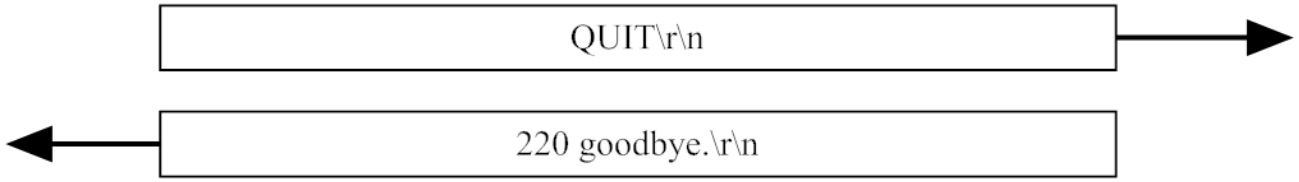


150 Opening ASCII mode data connection for 'file list'.



FTP Protocol Exchange

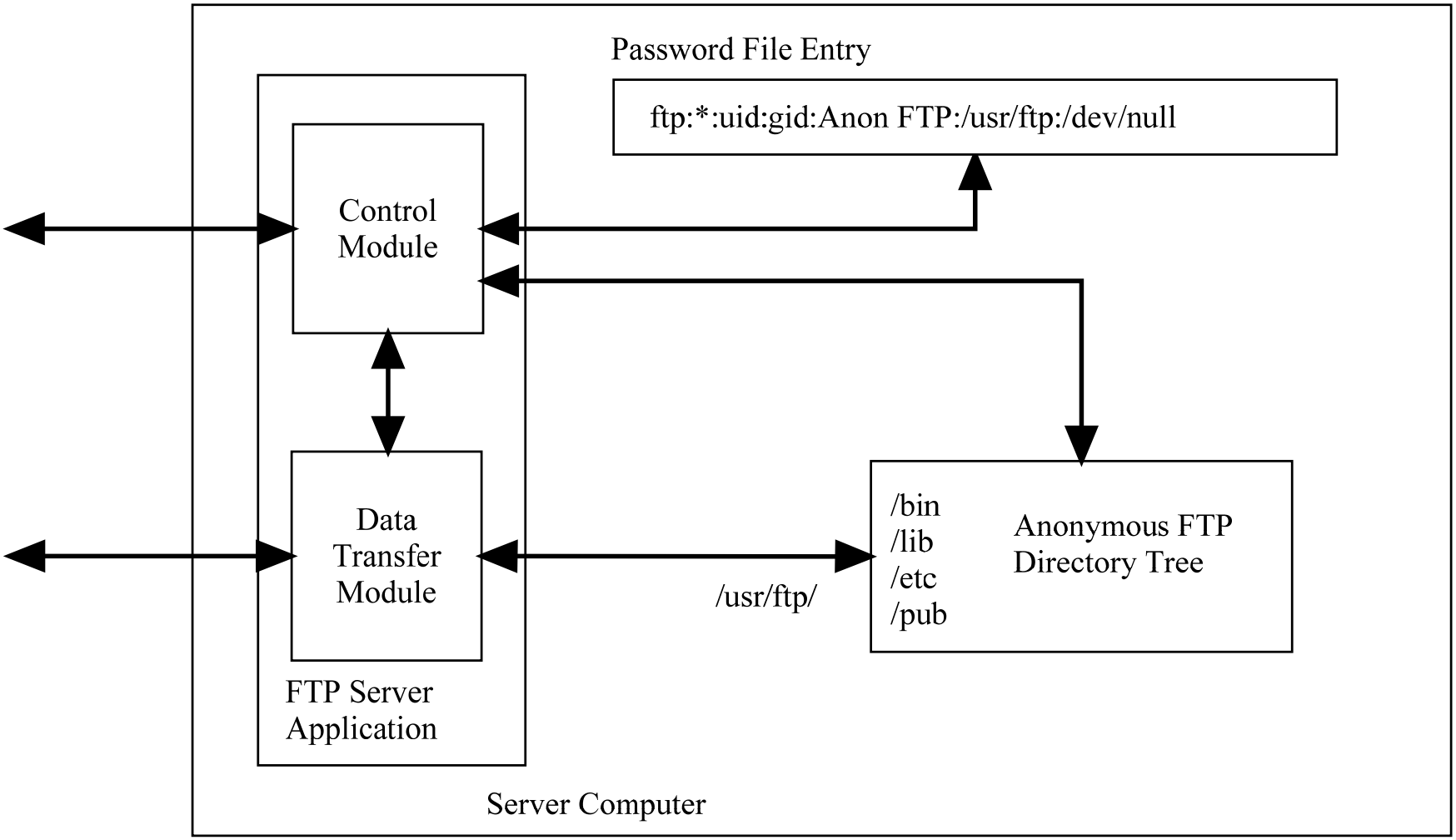
(list of files)
226 Transfer complete.
ftp: 463 bytes received in 0.00Seconds 463000.00Kbytes/sec.
ftp> **quit**



Anonymous FTP

- $ **ftp spock.dougj.net**
- Connected to spock.dougj.net.
- 220 spock.dougj.net FTP server ready.
- User (spock.dougj.net:(none)): **anonymous**
- 331 Guest login ok, type your name as password.
- Password:
- 230 Guest login ok, access restrictions apply.
- ftp>

Anonymous FTP Server



TFTP

Name (opcode)	Parameters	Function
RRQ (1)	Filename (var), 0x00 Mode (var), 0x00	Read request, mode is either netascii or octet
WRQ (2)	Filename (var), 0x00 Mode (var), 0x00	Write request, mode is either netascii or octet
DATA (3)	Block Number (2 bytes) Data (0-512 bytes)	Block number starts at 1, all blocks except the last block must be 512 bytes long. A block that is less than 512 bytes is used to indicate last block and the file transfer is done
ACK (4)	Block Number (2 bytes)	Used to acknowledge the data block
ERROR (5)	Error number (2 bytes) Error data (var), 0x00	Used to indicate an error, the error data is text data.

RCP

- Based on rlogin
- If user is trusted copy will take place
- If user is not trusted copy will not take place.

Header & Protocol Based

- FTP has problems with buffer overflows
- Not many protocol attacks
  - One is an FTP redirect attack
  - Done by telneting to an FTP server that has exploit code.
  - Use ftp to transfer the code to another server

Redirect

- $ **telnet klingon.iseage.org 21**
- 220 klingon.iseage.org FTP server ready.
- **user anonymous**
- 331 Guest login ok, type your name as password.
- **pass doug**
- 230 Guest login ok, access restrictions apply.
- **port 192,168,1,40,0,25**
- 200 PORT command successful.
- **retr m1**
- 150 Opening ASCII mode data connection for 'm1' (84 bytes).
- 226 Transfer complete.
- **Quit**

File m1:
HELO cia.gov
MAIL FROM: badperson@cia.gov
RCPT TO: user
DATA
(any mail message)
.

Authentication-Based

- FTP Prompts for username and password
- Anonymous FTP with writable directories
- User based FTP server

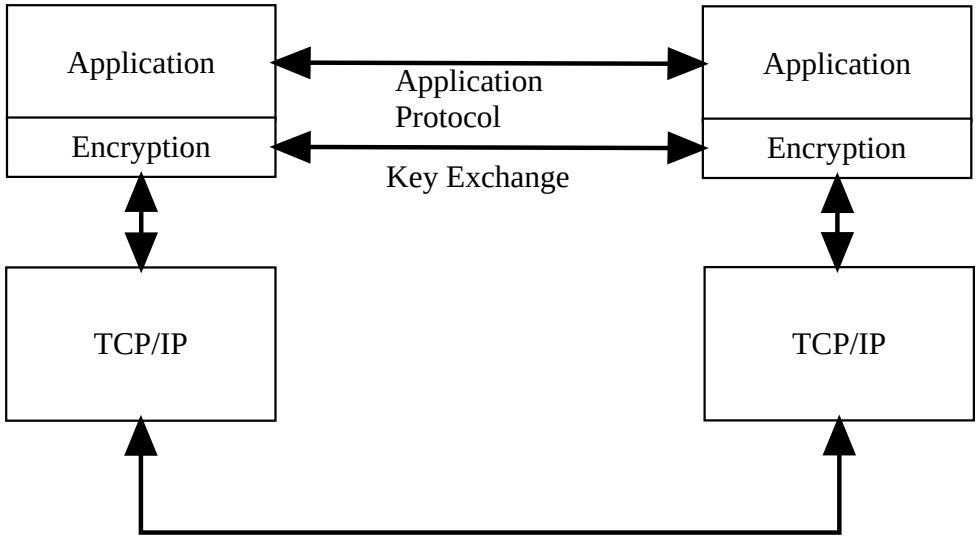
Traffic-Based

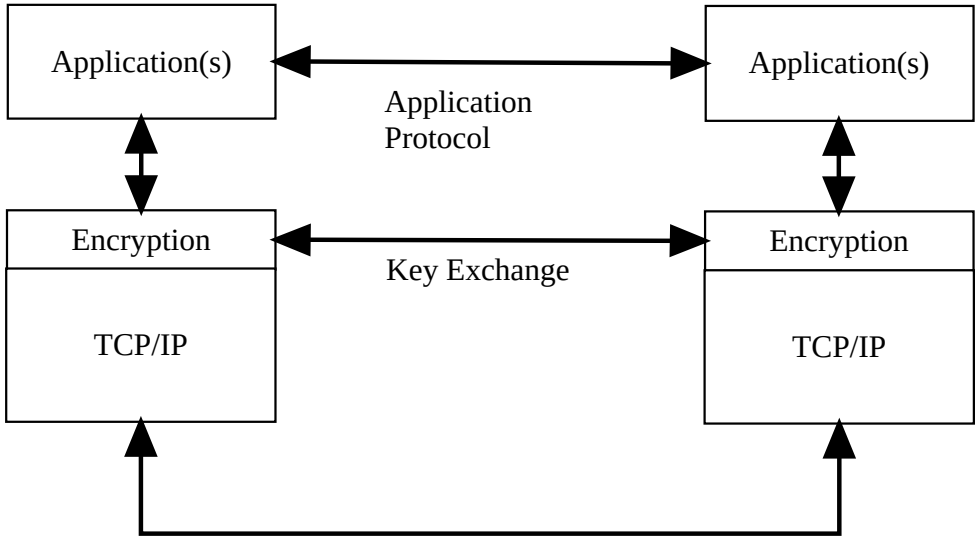
- Clear Text
- FTP can be flooded, massive uploads or downloads

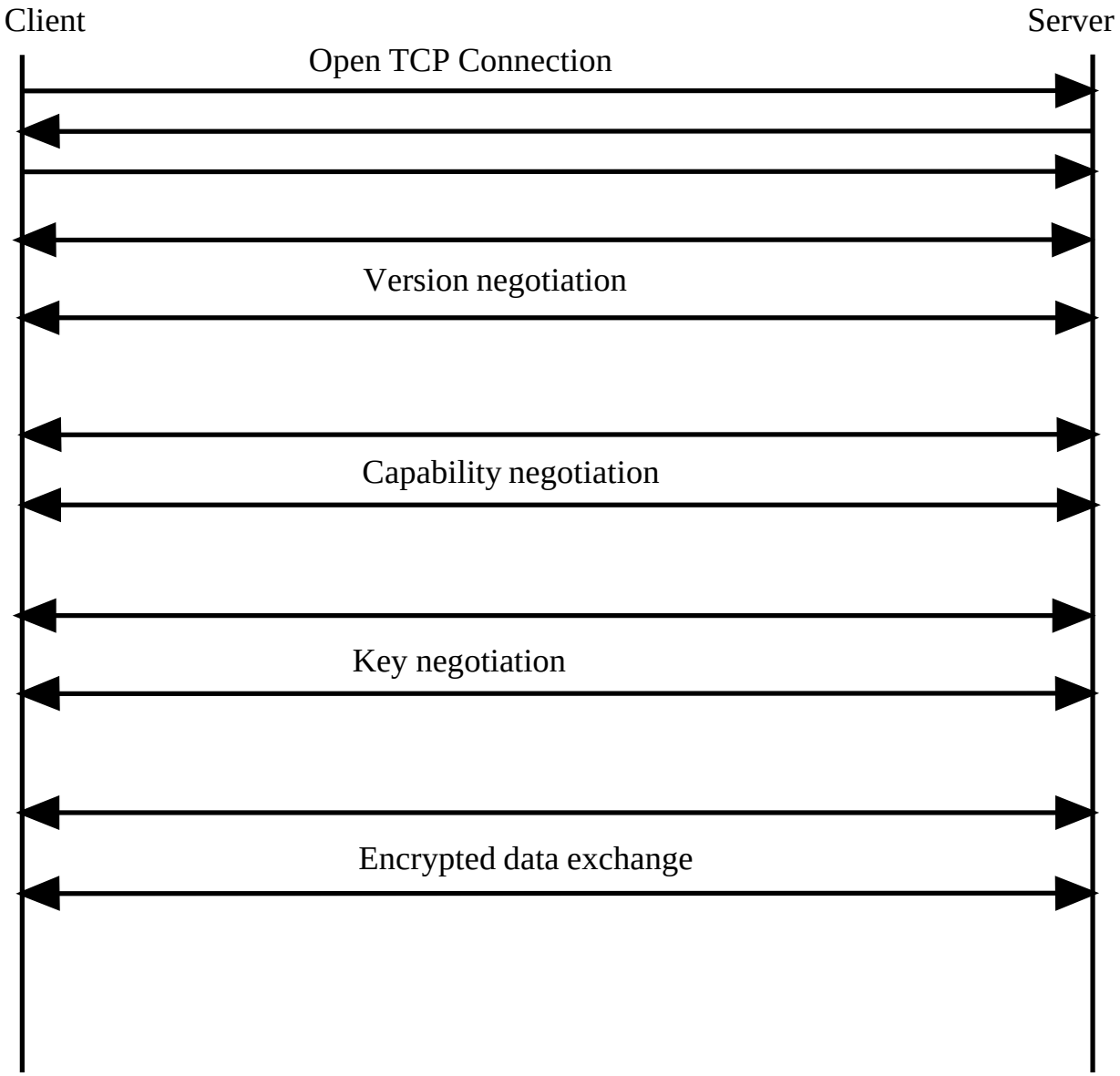
General FTP Countermeasures

- Encrypted Channels
- Encrypted copy & FTP

Encrypted Channels







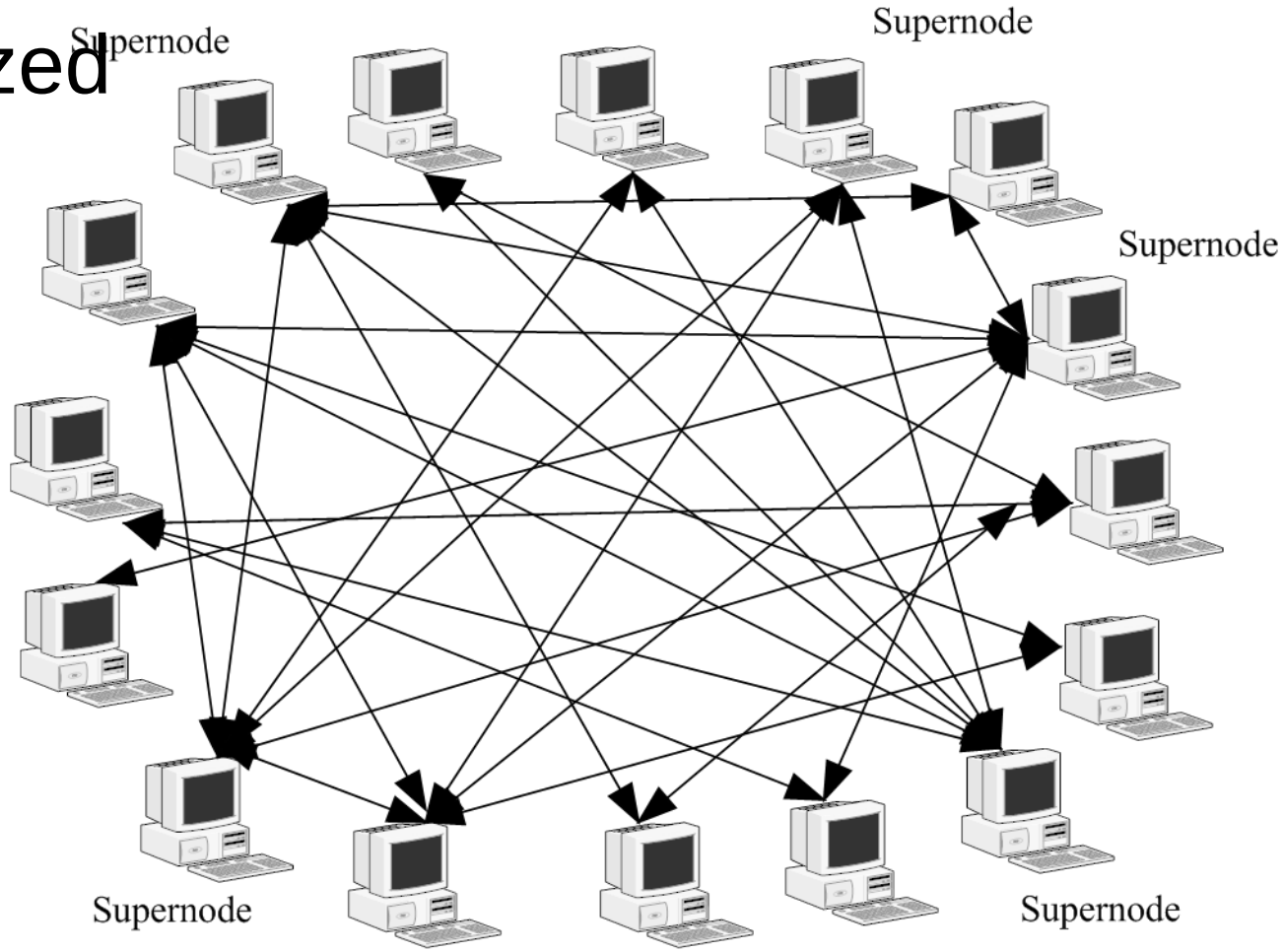
Encrypted protocols

Peer-to-Peer Topics

- We will look at examples of peer-to-peer protocols
  - Napster
  - KaZaA
  - Gnutella

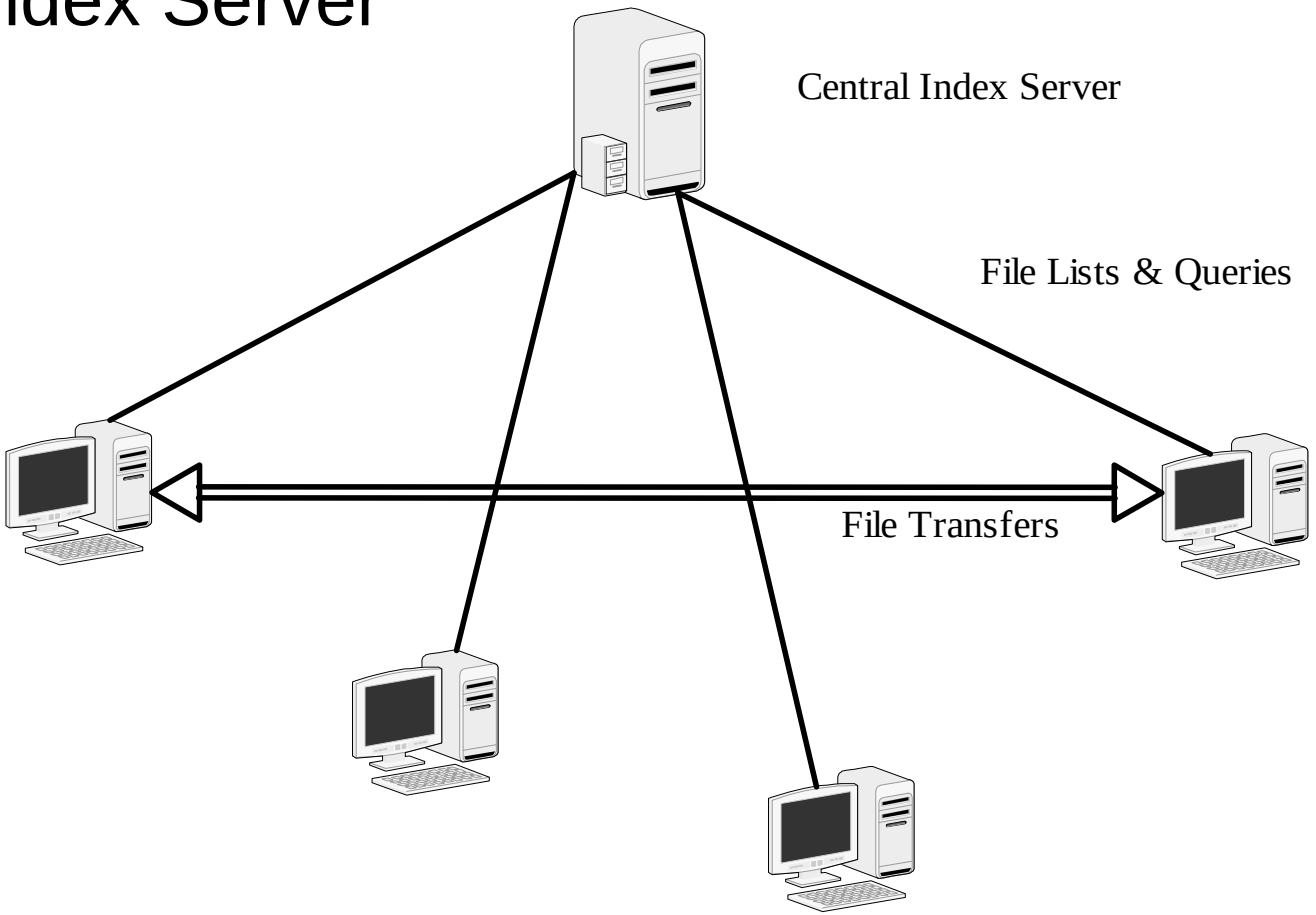
Peer to peer types

- Decentralized



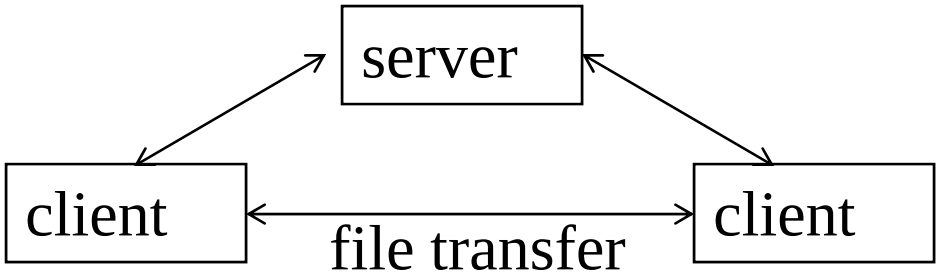
Peer to Peer types

- Central Index Server



Napster

- Napster is a controversial application that facilitates the sharing of music files
- User's can search for songs and download songs from another user's harddrive
- All clients connect to a central server



Napster

- Napster has a simple packet format:

Length	Type	Data

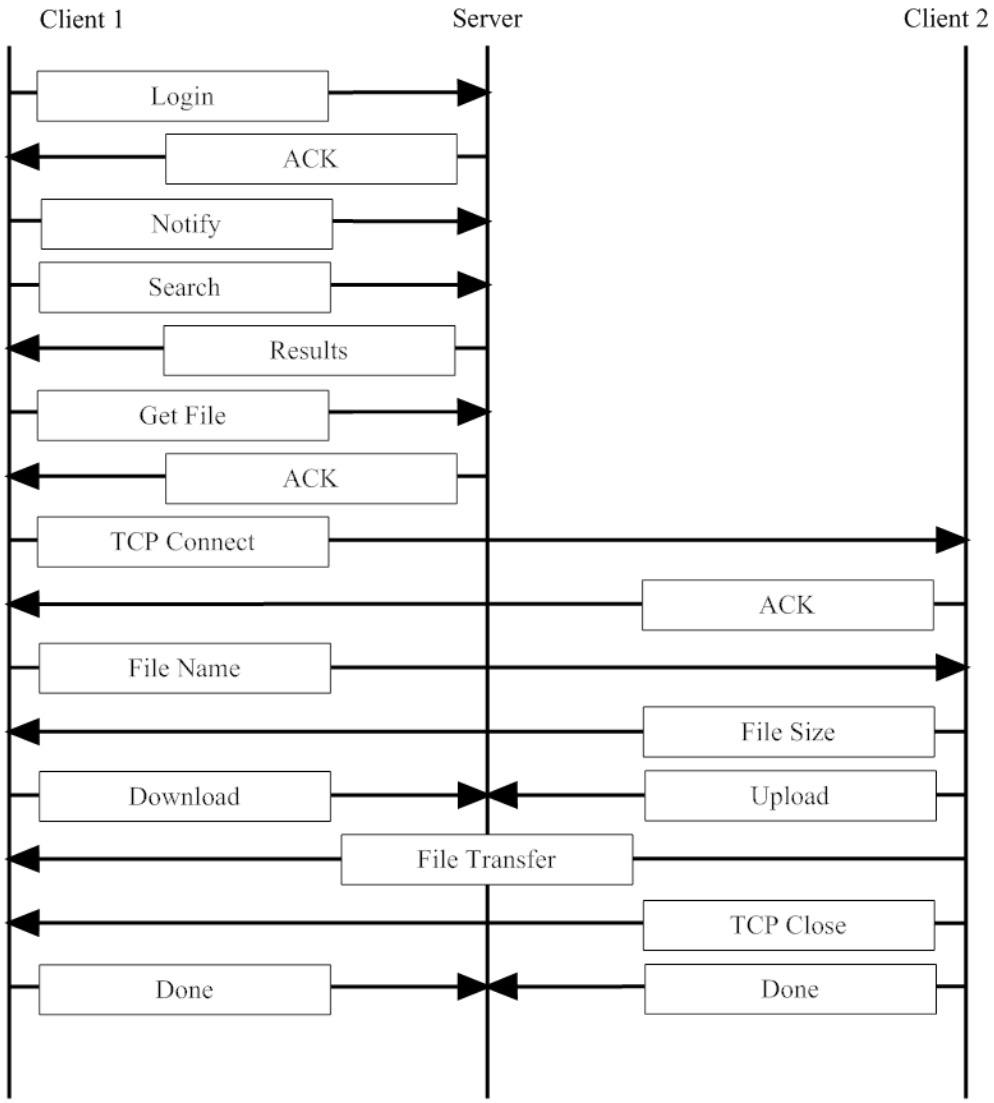
- The length and type fields are each 2 bytes
- Types:

2	Login	203	Get
3	Login Ack	204	Get Ack
100	Notify	218	Download
200	Search request	219	Download complete
201	Search reply	220	Upload
		221	Upload complete

Napster

- Sequence:
  - Log in to server
  - Notify the server of files you are sharing
  - Search for a file to download
  - Download the file
- The above sequence is illustrated on the next slide.
- For now, assume the user is not behind a firewall

Napster

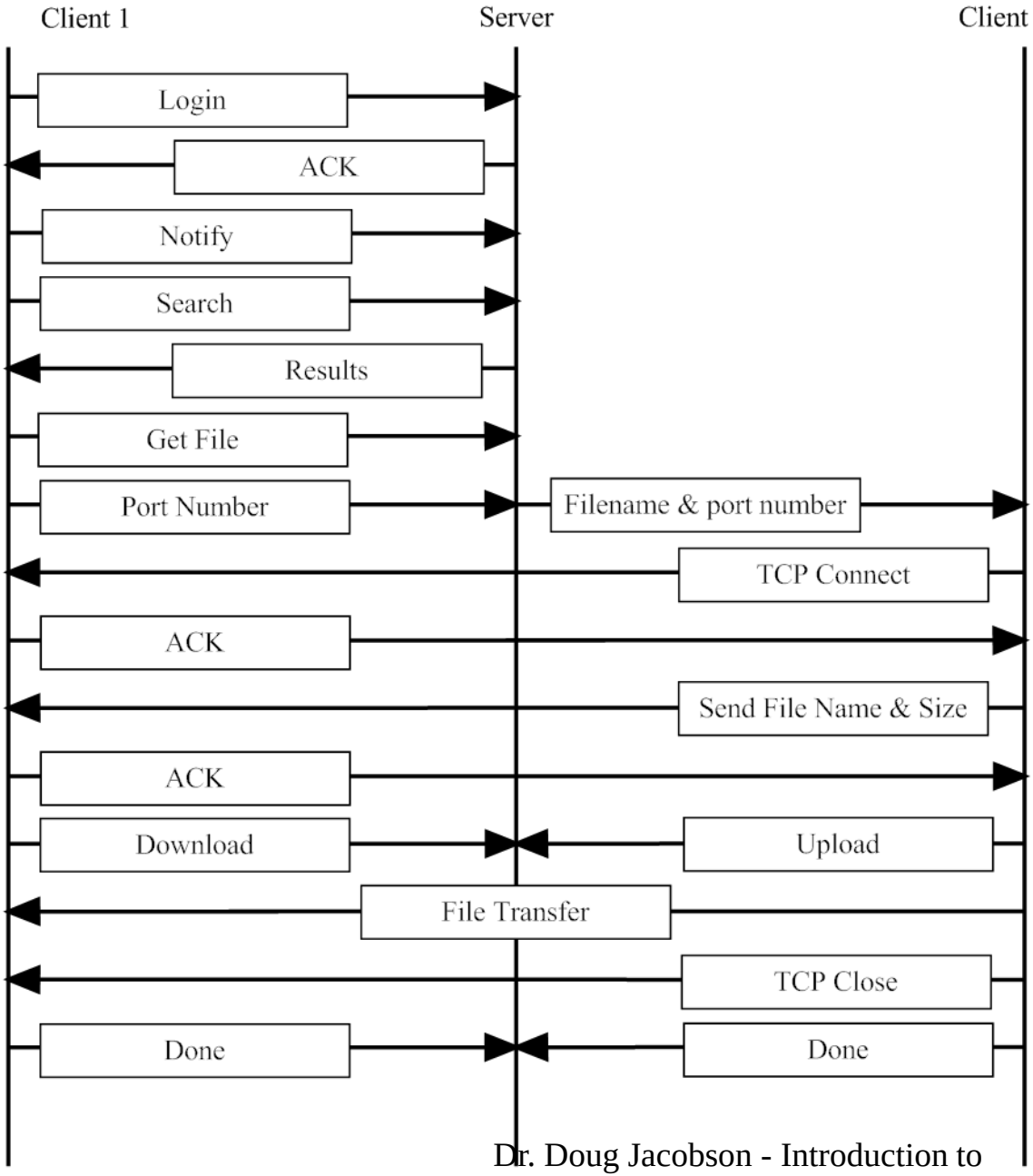


Length	Type	Data

Napster

- When client 1 is behind a firewall, the download is slightly different
- Client 1 tells the server the port to use
- The server then tells client 2 which port to use
- Client 2 sends the file to the specified port

Napster


Client 1
Server
Client 2
Login
ACK
Notify
Search
Results
Get File
Port Number
Filename & port number
TCP Connect
ACK
Send File Name & Size
ACK
Download
Upload
File Transfer
TCP Close
Done
Done

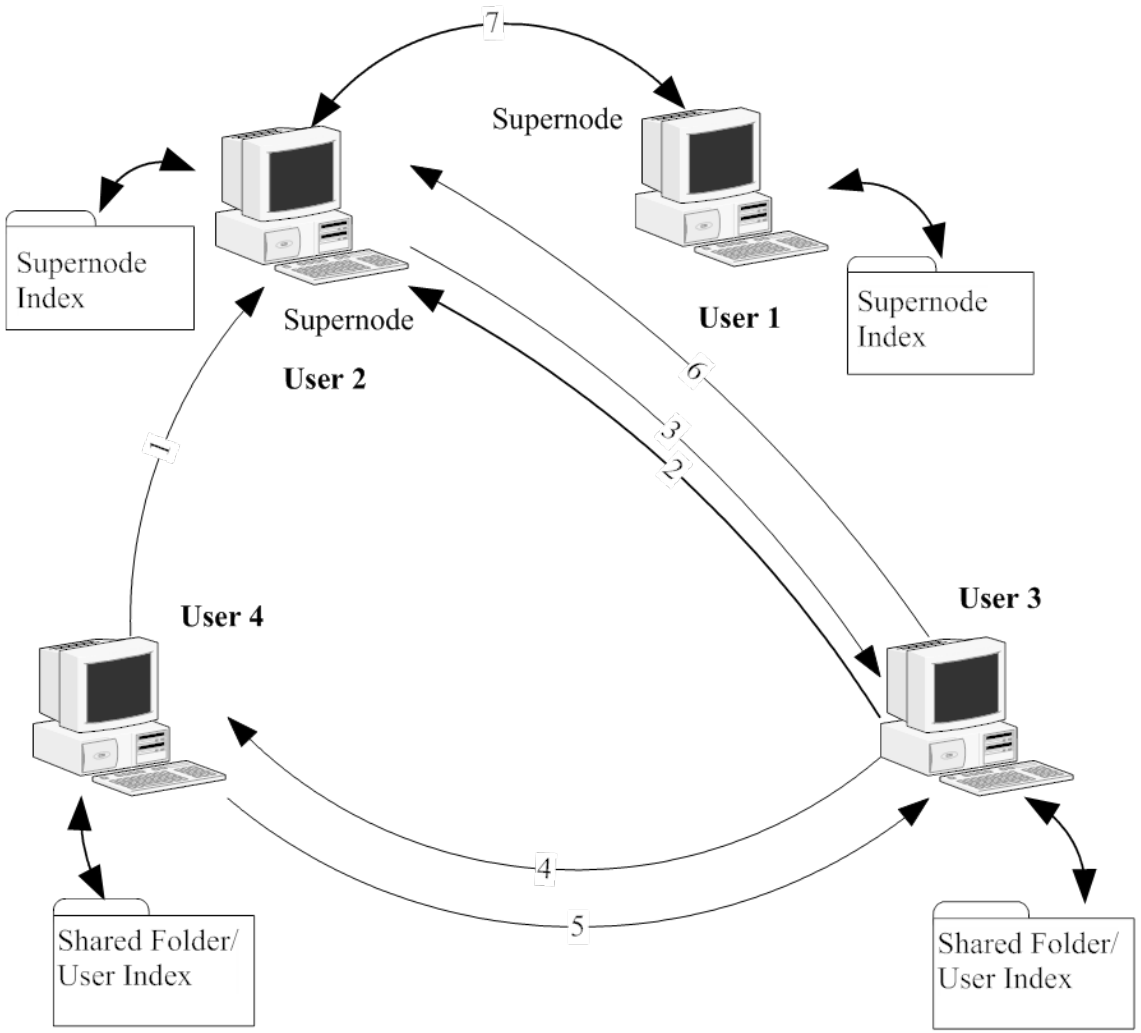
Napster Issues

- As shown in the preceding illustrations, the server is heavily involved in facilitating the transfer of files
- The server also keeps track of what is being transferred where
- This may have played a part in the case against Napster
- However, how can you verify that the filename accurately reflects the song transferred?

KaZaA

- Central Index server based (called super nodes)
- Uses Fasttrack protocol between server and client
  - Proprietary protocol
- All files have hash values
- Protocol between clients is HTTP 1.1

KaZaA



(1) Upload User's Index
(2) Search Supernode Index
(3) Search Results
(4) Request for File
(5) Send File
(6) File added to Supernode Index
(7) Queries

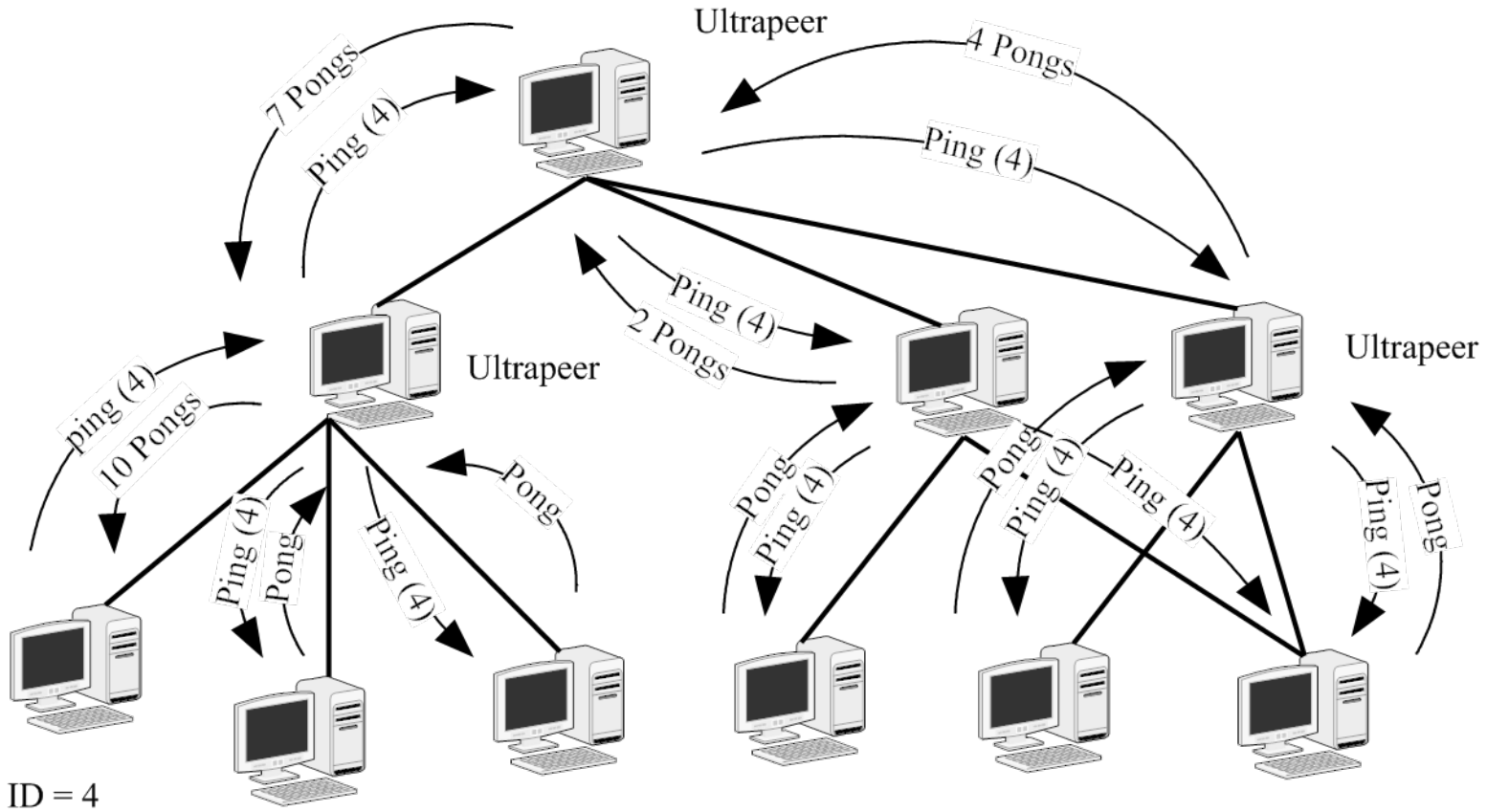
Decentralized Peer-to-Peer

- Limewire, Bearshare, Gnutella
- Peer-to-peer arrangement
- No central server
- Each client connects to 4 other clients, called servents
- Other clients connect to you
- Allows you to share and download any file type, not just music

Gnutella Protocol

- When you search for a file, you ask the servents nearest you, who ask the servents nearest them, and the search propagates in a daisy chain effect
- Logging in to the gnutella network generates a lot of traffic, as other people's searches are constantly propagating through you
- You can see what other people are searching for through you
- Gnutella clients are available for every platform. Some examples: BearShare, LimeWire

Gnutella Routing



Gnutella Ping and Pong

- The data section of the “pong” packet contains:
  - Port number of responding machine
  - IP address
  - Number of files shared (4 bytes)
  - Total kilobytes shared (4 bytes)
- “Ping” packets contain no data
- Each client periodically pings all connections nearest them

Gnutella Queries

- The “query” packet contains:
  - Minimum speed in kb/s (2 bytes)
  - Search string (length varies)
- The “query-hit” packet contains:
  - Number of hits (1 byte)
  - Port (2 bytes)
  - IP address (4 bytes)
  - Speed (2 bytes)
  - Result set (length varies)
    - Index (4 bytes), Filesize (4 bytes), Name (length varies)
  - Servent name, used for push (generally the IP address)

Gnutella Packet Format

ID	Payload	TTL	Hop	Length	Data

Gnutella Packet

Payload	
00	ping
01	Pong
80	Query
81	Query Hit

Port	IP	Number of files shared	Number of bytes shared

Pong Packet

Min Speed	String

Query packet

Hits	Port	IP	Speed	Results	IP

Query-Hit packet

Gnutella Push

- A “push” is used when the user is behind a firewall
- The “push” packet contains:
  - Servent ID
  - File index
  - IP address
  - Port

Header / Protocol Based

- Applications and protocol could be subject to these attacks.

Authentication Based

- Cannot trust source of files
- Anything can be shared
- Users that share can be traced

Traffic Based

- Can generate large amounts of traffic
- Super nodes can draw more traffic
- Sniffing is possible, but does not matter

Peer-to-Peer Countermeasures

- Port Blocking
- Content Blocking

Anonymous Services & Privacy Topics

- Anonymous services
  - Routing
  - Surfing
- Privacy on the Internet
- Proxy servers

Email Tracking

- www.readnotify.com
- Uses web bug tracking
- Keeps a log and emails you when the recipient opens the email.
- Looks like the email came from the sender, you send the email to:
  - user@domain.readnotify.com

Anonymous Email Services

- Login to a web site and send email from the site.
- Gmail, etc.
- Special sites for anonymous email
  - www.anonymousspeech.com

Privacy surfing the Internet

- Web servers can collect demographics about you
- www.privacy.net will show you all the things a webserver knows about you
- Examples:
  - Your browser type and Operating System
  - CPU type
  - whether JavaScript is enabled
  - Date/Time on your computer
  - Your IP address
  - Which plugins you have installed

Privacy on the Internet

- Once you login and give your email address, you are no longer anonymous
- Some web sites share your email address with other sites
- This can lead to you receiving spam from sites to which you've never disclosed your email
- Some sites store cookies on your harddrive. Amazon.com does this to recommend books based on your previous purchases.
- One way to surf privately: connect through a proxy

Proxy Servers

- A proxy is basically someone who makes requests on your behalf
- They were originally designed to cache information to prevent redundancy
- Suppose you (M) want to view a web page from server W. Here's how it would look without a proxy:

SIP = M S Port = ephemeral

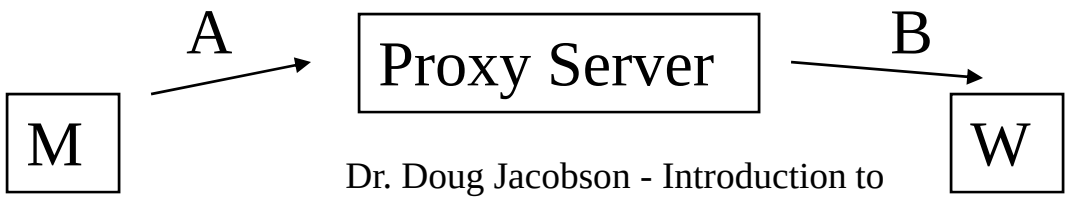
DIP = W D Port = 80

URL=http://w.com/path

Proxy Servers

- Here's how it would look if you used a proxy server. Two different packets are needed: packet A is generated by yourself, and packet B is generated by the proxy server

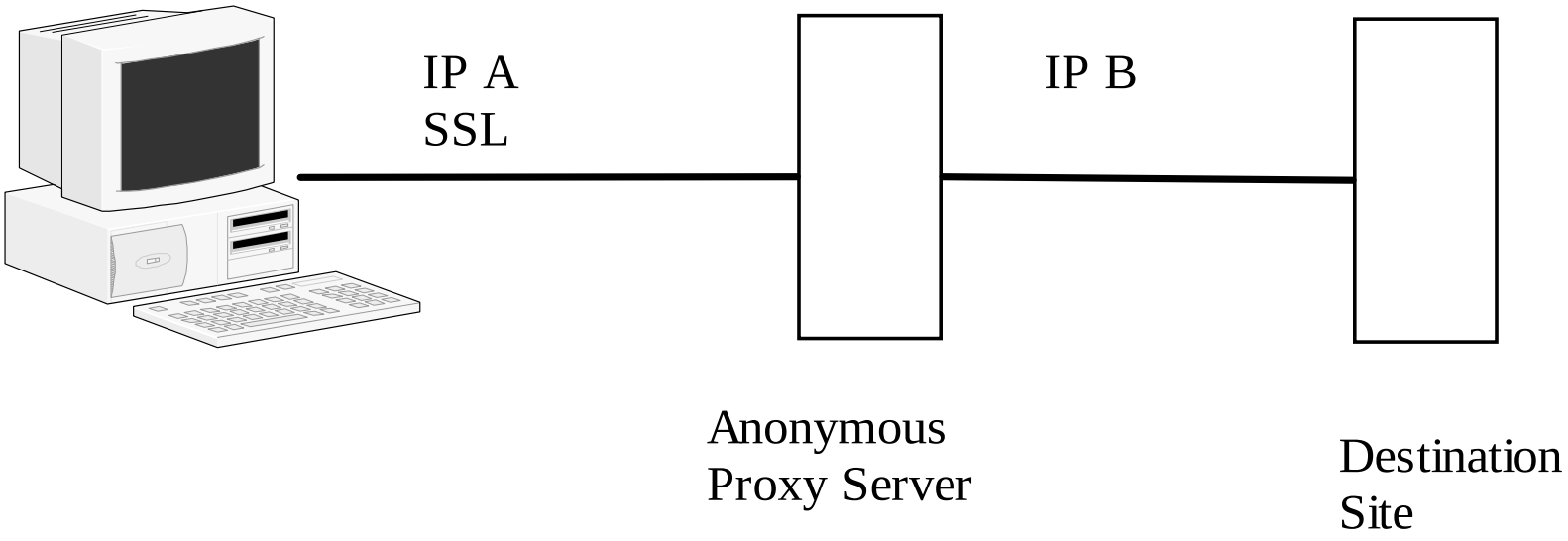
Packet A:		Packet B:	
SIP=M	SPort=?	SIP=P	SPort=?
DIP=P	DPort=	DIP=W	Dport=80
URL=http://w.com/path		URL=http://w.com/path	



Proxy Servers

- There are two reasons to be anonymous
  - Don’t want webservers to know who we are
  - Don’t want big brother (ie: your boss) to know what sites we are visiting
- A proxy can provide some amount of anonymity
- Examples of existing proxy servers used to provide anonymity:
  - anonymizer.com, safeweb.com, kaxy.com, the-cloak.com
- However, if your company does not wish you to be using these proxies, they can block access to them through their firewall.

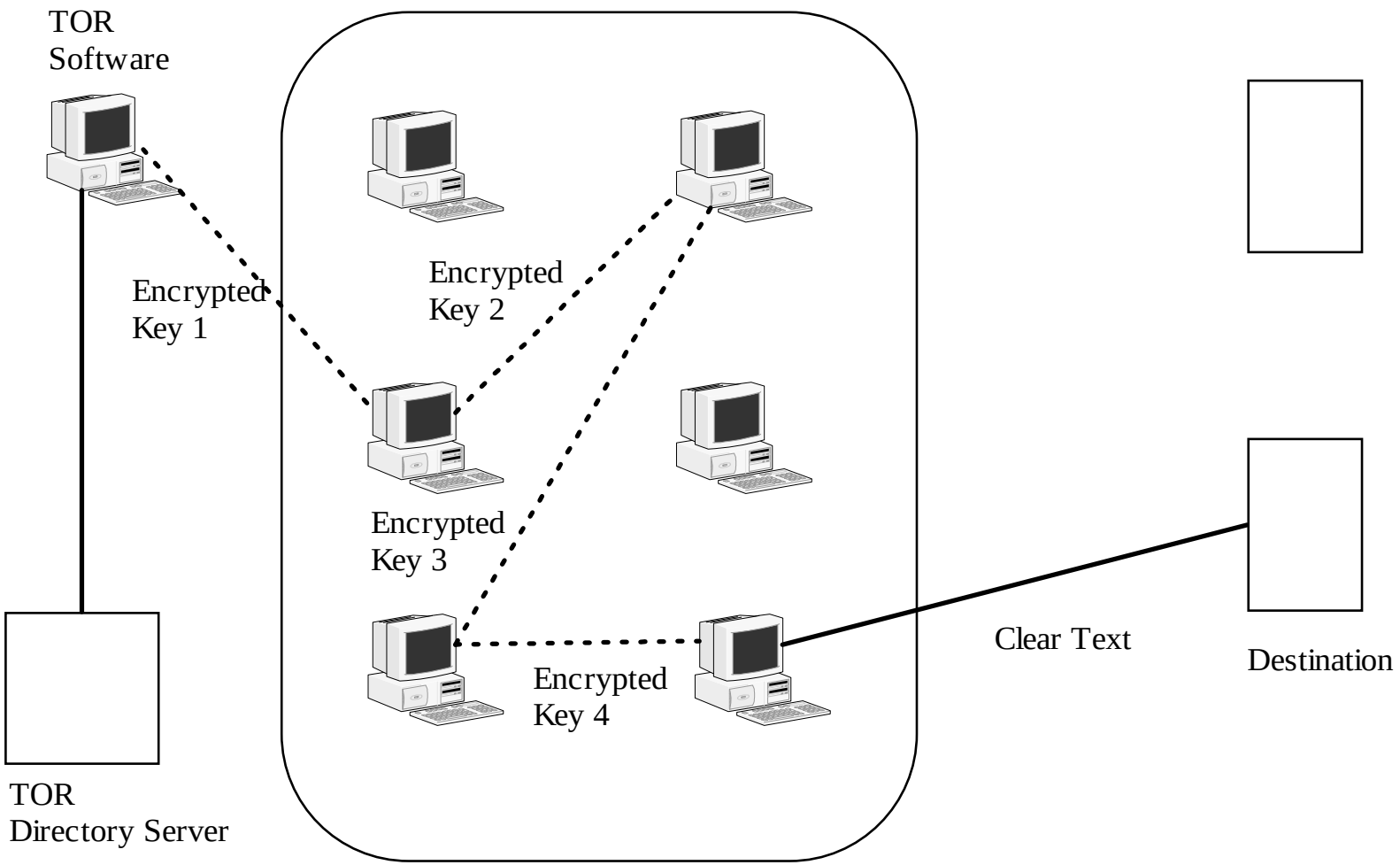
Secure Proxy Server



Proxy Servers

- However, TOR has a fix that prevents a company from blocking access to their site.
- It involves a a system called onion routing
- See diagram next slide

TOR


TOR
Software
Encrypted
Key 1
Encrypted
Key 2
Encrypted
Key 3
Encrypted
Key 4
Clear Text
Destination
TOR
Directory Server

TOR

- Starting host builds the connection one node at a time.
- The encryption keys are between each node and the starting point, so each node is unable to read the data
- Once the end node is reached the starting node has a key with each node.
- Destination host only sees the last node

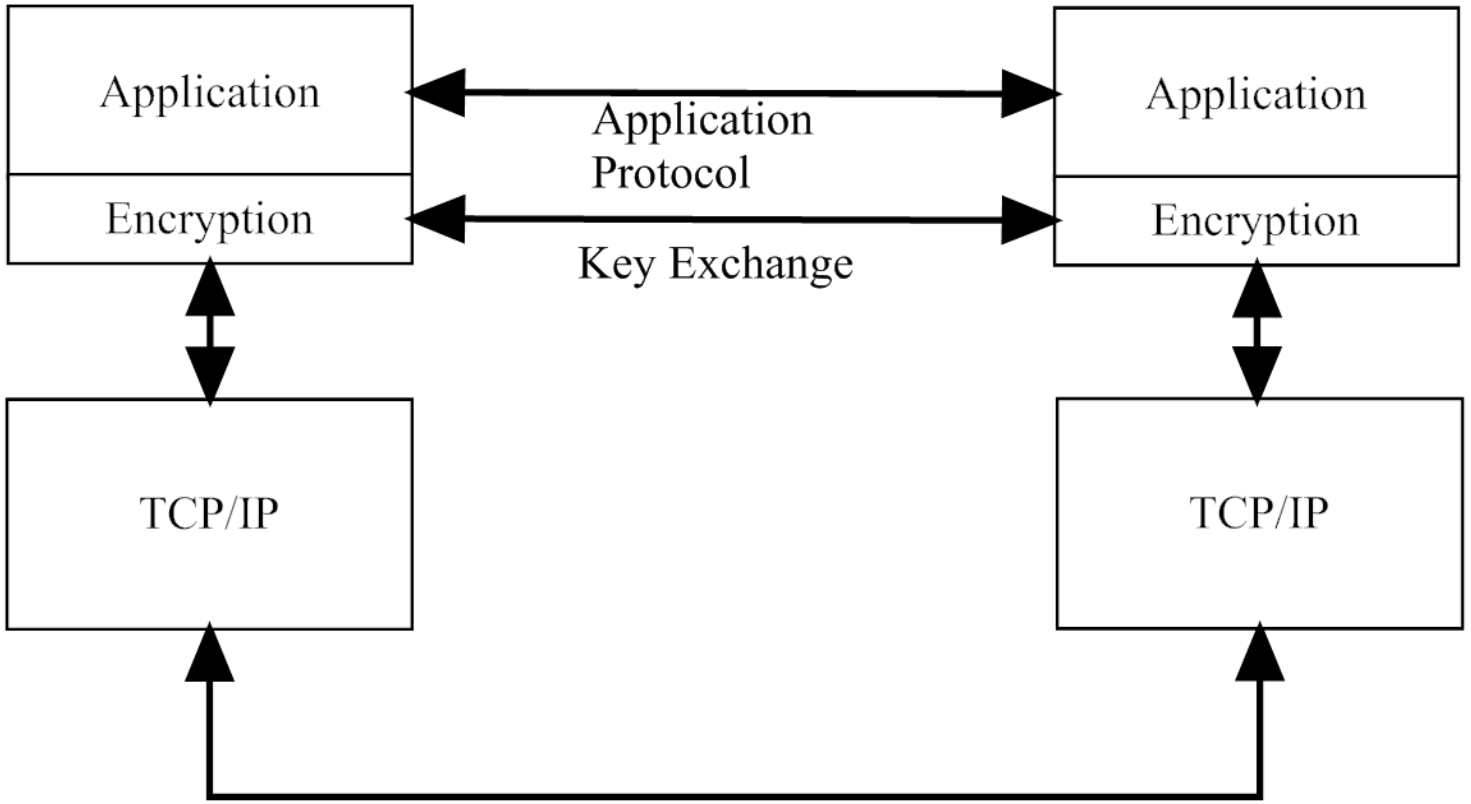
Security Issues

- Bypass company security policies
- Hard to stop

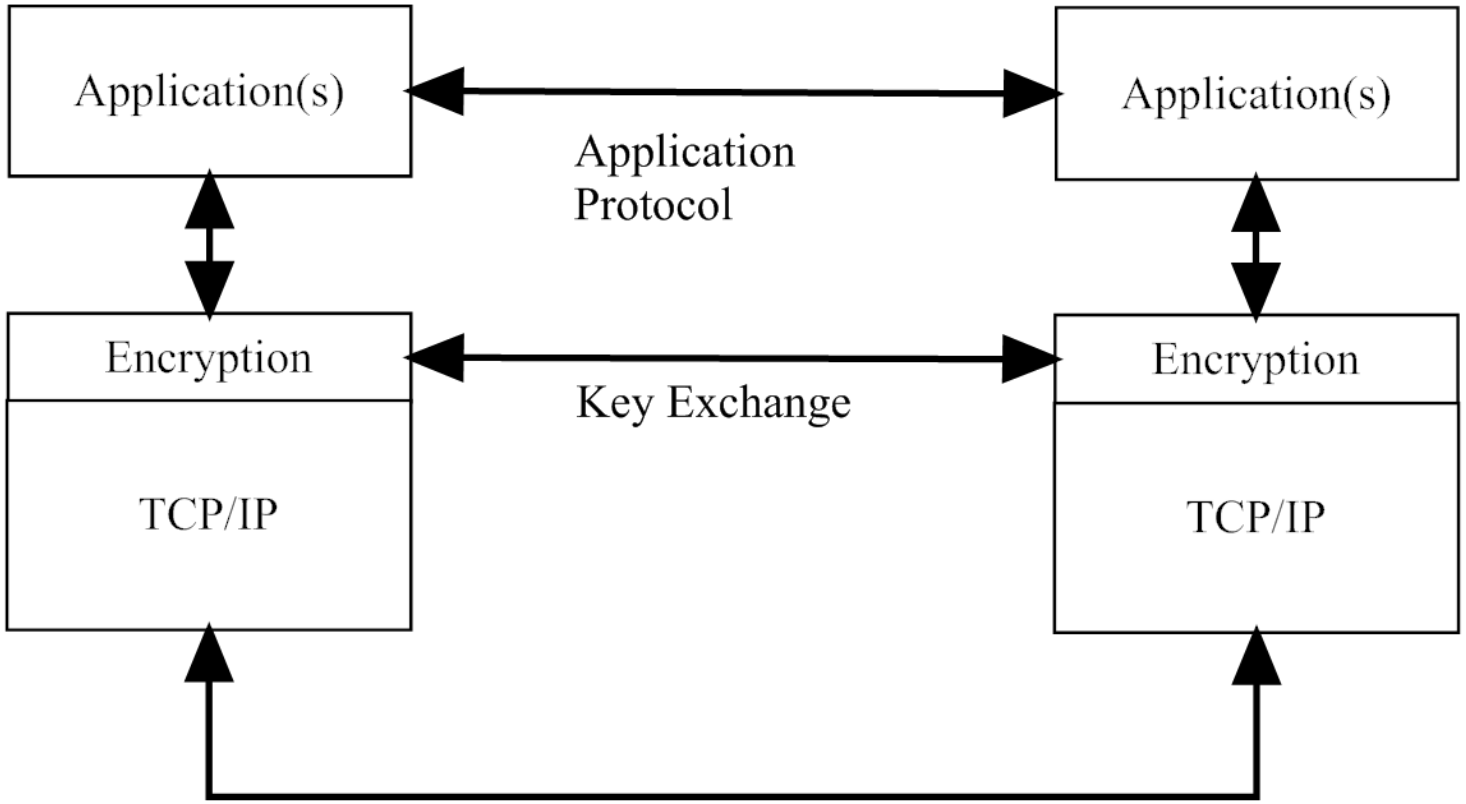
General Remote Access Countermeasures

- Encrypted remote access
  - Application-based
  - Tunnel-based
  - SSH
  - Remote desktop
  - Secure File transfer

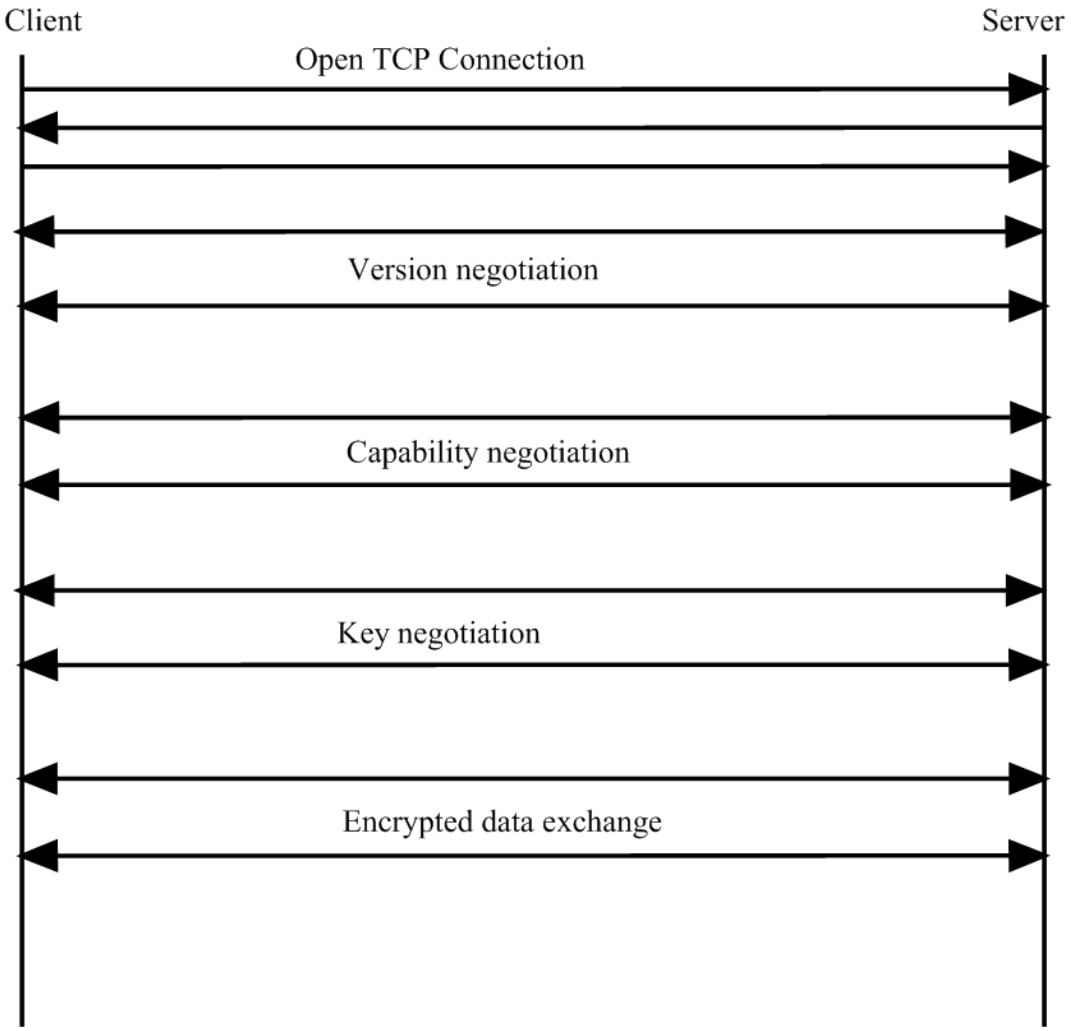
Application-Based Encryption



Tunnel-Based Encryption



Encrypted Remote access protocols



SSH

- SSH
  - Secure shell
  - Designed to replace rlogin, rsh, rcp
  - Provides
    - Authentication at the machine level, doesn't care about user authentication
    - Secure communication through encryption

SSH Details

- Strong Authentication
- Public domain software
- Some versions support compression of data
- Privacy
  - Key negotiation with symmetric key
  - Key exchange based on no trust of network
  - Multiple keys to deal with replay attacks
- Can provide secure X11 sessions
- Encrypt any traffic with SSH
- Same parameters as rlogin
- If other side doesn’t support SSH drops to rlogin

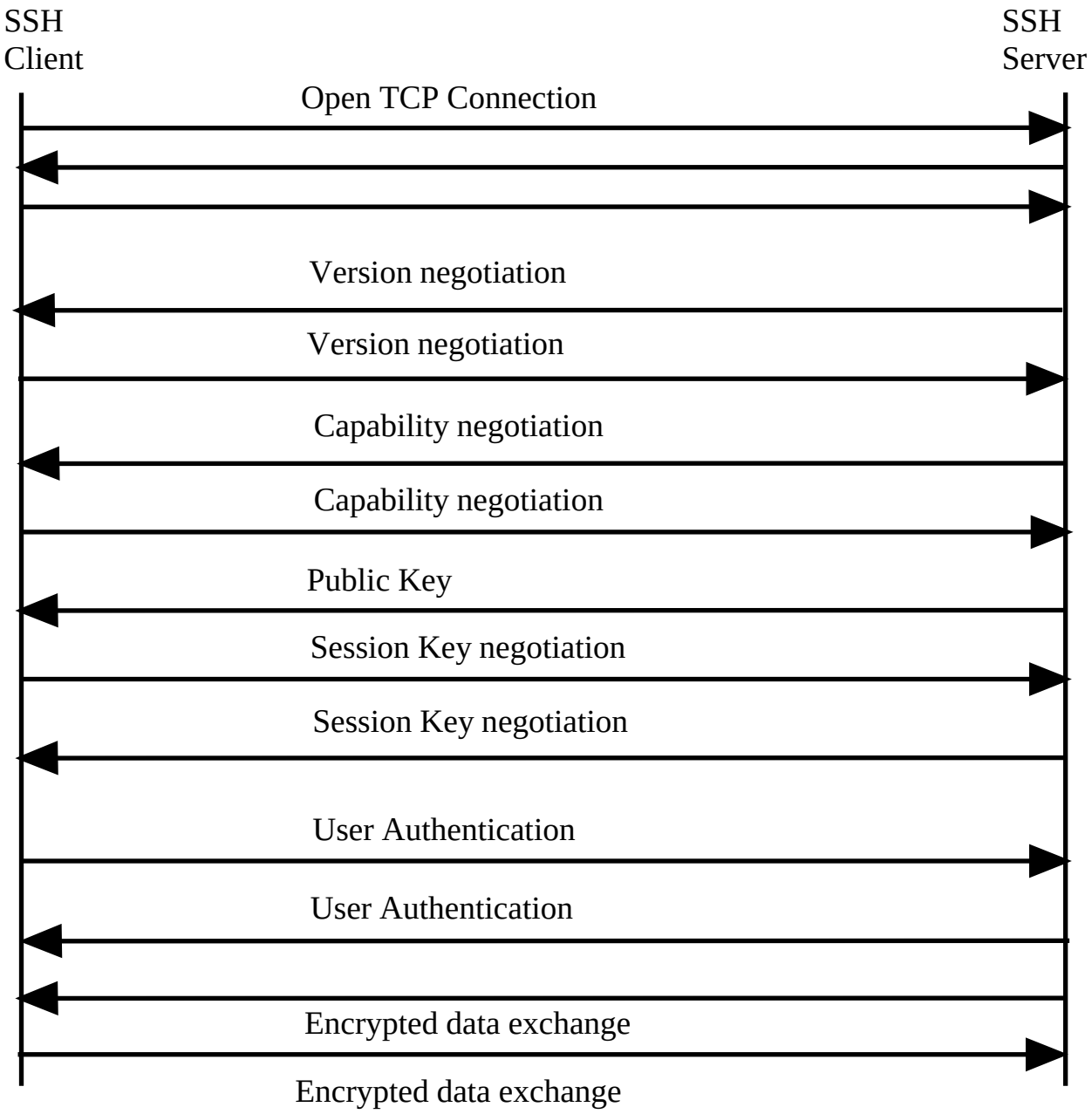
Details cont…

- Need server and client software
- Sshd server demon software
- Ssh is the client software
- Ssh keygen
  - Generates host key
- Ssh agent
  - Uses public and private key technique to get process started

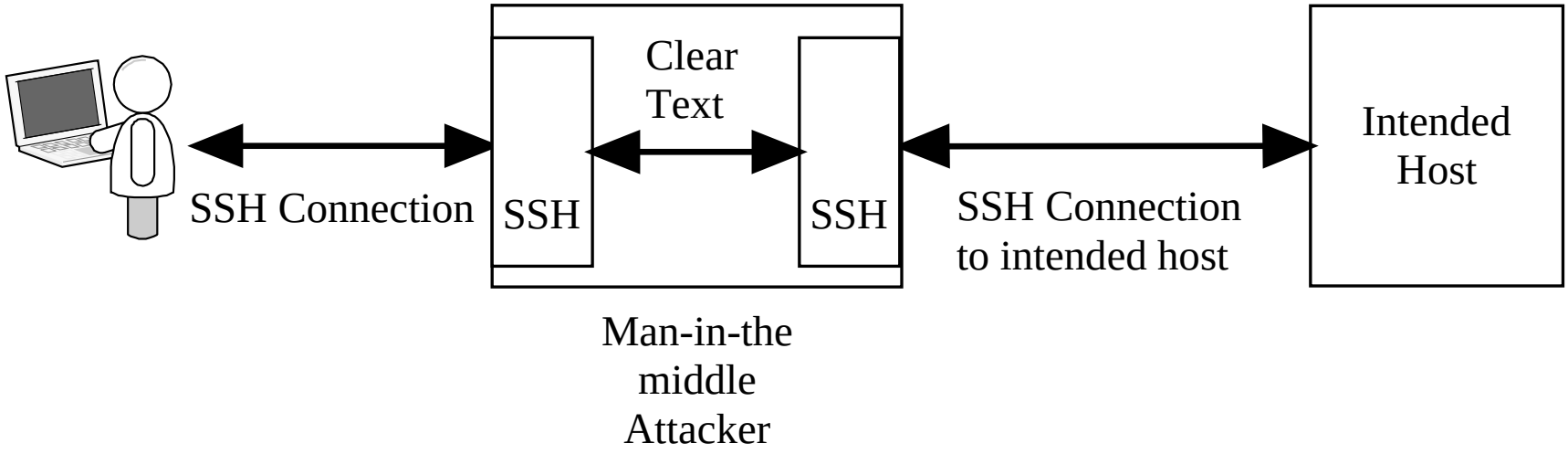
SSH Protocol

- Client sends query
- Server sends two public keys which is a 1024 bit client key and a server key which is a 768 bit key
- Server key recomputed every hour
- Client generates 256 bit random number which is the symmetric key, which is encrypted using the server and the host keys
- Server responds with ok which is encrypted with session key
- All traffic is now encrypted with session key
- Problems
  - Man in the middle attack
  - Putty is a man in the middle attack program

SSH


SSH Client
SSH Server
Open TCP Connection
Version negotiation
Version negotiation
Capability negotiation
Capability negotiation
Public Key
Session Key negotiation
Session Key negotiation
User Authentication
User Authentication
Encrypted data exchange
Encrypted data exchange

SSH Man in the Middle Attack



Remote Desktop

- Uses tunnel-based encryption
  - Via RDP or TLS (newer versions)
- Key exchange is similar to SSH
- Three levels
  - High (128 bit)
  - Medium (56 or 40 bit)
  - Low (56 or 40) only client to server data
- Subject to password guessing and man in the middle attacks

Secure File Transfer

- SFTP – uses SSH
- FTPS – uses SSL/TLS
- HTTPS – uses SSL/TLS