TCP/IP Standard Applications Telnet - SSH - FTP

Virtual Terminal, Secure Shell, File Transfer

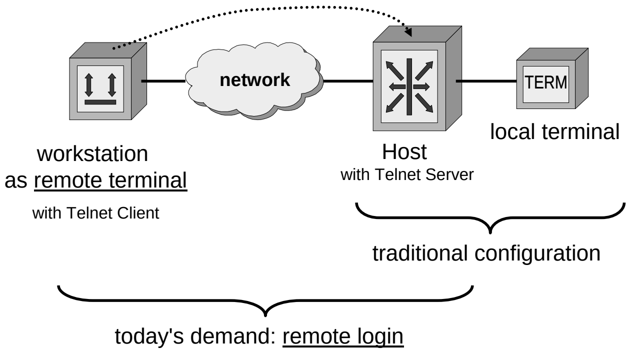
Agenda

- **Telnet**
- **SSH**
- **FTP**

What is Telnet?

- **Telnet is a standard method to communicate with another Internet host**
- **Telnet provides a standard interface for terminal devices and terminal-oriented processes through a network**
- **using the Telnet protocol user on a local host can remote-login and execute commands on another distant host**
- **Telnet employs a client-server model**
  - a Telnet client "looks and feels" like a Terminal on a distant server
  - even today Telnet provides a text-based user interface

Local and Remote Terminals



network

TERM

workstation as remote terminal

with Telnet Client

Host

with Telnet Server

local terminal

traditional configuration

today's demand: remote login

About Telnet

- **Telnet was one of the first Internet applications**
  - since the earliest demand was to connect terminals to hosts across networks
- **Telnet is one of the most popular Internet applications because**
  - of its flexibility (checking E-Mails, etc.)
  - it does not waste much network resources
  - because Telnet clients are integrated in every UNIX environment (and other operating systems)

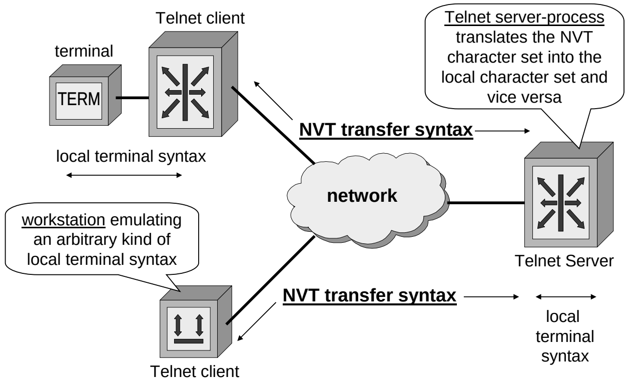
Telnet Basics

- **Telnet is connection oriented and uses the TCP protocol**
- **clients connect to the "well-known" destination port 23 on the server side**
- **protocol specification: RFC 854**
- **three main ideas:**
  - concept of *Network Virtual Terminals* (NVTs)
  - principle of *negotiated options*
  - a *symmetric view* of terminals and (server-) processes

Virtual Terminals

- **a Telnet Client can emulate the behaviour of a wide range of well-known real terminals**
- **internally, each end of a Telnet connection leads to a Network Virtual Terminal (NVT)**
- **an NVT provides a standard, network-wide, intermediate representation of a canonical terminal**
  - consisting of a display (printer) and a keyboard (line-buffered mode) in half-duplex mode
  - Telnet communications rely upon the "language" of NVT's
  - each local device characteristics are mapped to the NVT capabilities

Telnet Client - Server


Telnet client
terminal
TERM
local terminal syntax
Telnet server-process translates the NVT character set into the local character set and vice versa
NVT transfer syntax
network
Telnet Server
workstation emulating an arbitrary kind of local terminal syntax
NVT transfer syntax
local terminal syntax
Telnet client

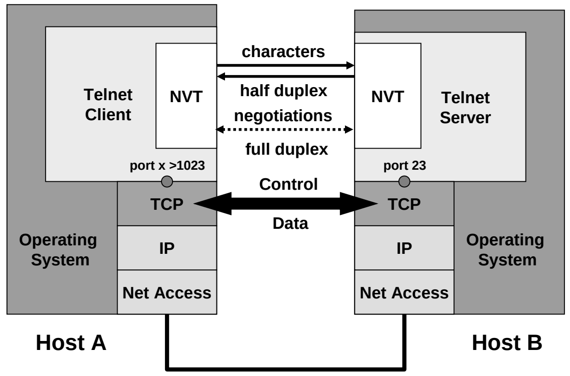
Half-Duplex Connection

- **a Telnet connection "itself" is running <u>full-duplex</u>**
  - e.g. both sides can send negotiation commands or signals at the same time
- **but at the users point of view, <u>NVT's</u> only communicate in a <u>half-duplex</u> way !**
  - to reduce network costs and the number of server interrupts, a Telnet-client accumulates NVT keyboard inputs in a buffer before sending it (e.g. line buffered)
  - on the other side the Telnet-server wants to send all data to the client's printer before the client continues
  - so a kind of token-principle has been specified: the GA-character (Go Ahead) **<u>can</u>** be send to notify the other side that the current sender has finished its transmission

Negotiating Options

- **in order to extend the rather poor capabilities of a NVT, Telnet provides a means for option-negotiating**
  - using commands like DO, DON'T, WILL, WON'T
  - e.g. for full screen mode, specify terminal type, etc...
- **symmetric view: both the server and the client may propose additional options to be used**

Symmetric Telnet Model



NVT's Character Set

- **NVT generally use the 8 bit data format**
- **however, NVT's basic character set is the US ASCII 7-bit code**
- **so an NVT can handle the printable characters with ASCII codes 32-126 plus a small set of control characters:**
  - NULL (NUL) - no operation
  - BELL (BEL) - produces an audible or visible signal
  - Back Space (BS) - moves the print head one character to the left margin
  - Horizontal Tab (HT) - moves the printer to the next horizontal tab stop
  - Line Feed (LF) - moves the printer to the next print line, keeping the same horizontal position
  - Vertical Tab (VT) - moves the printer to the next vertical tab stop
  - Form Feed (FF) - moves the printer to the top of the next page
  - Carriage Return (CR) - moves the printer to the left margin

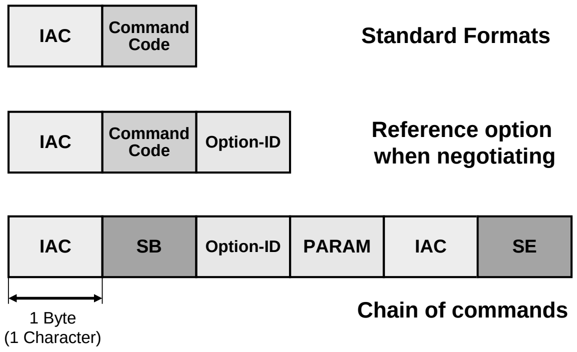
Internal Telnet Commands

- **for options negotiating and signalling purposes Telnet applies special command characters**
- **these commands have bit 8 set (code words 128-255)**
- **Telnet commands are prefixed with a special escape character: IAC - "Interpret As Command"**
  - code word 255
  - IAC is doubled if it appears in the normal data stream (only in the optional 8-bit mode - "IAC stuffing")

Internal Telnet Commands

- **all communication between client and server is handled with internal commands**
- **each command has 2 or 3 bytes length**
  - first byte: IAC
  - second byte: command code
  - possible third byte: referenced option when negotiating
- **the chain of commands can be even longer in case of sub-negotiating**
  - indicated with the command code SB (Subnegotiation Begin)
  - closed with the command code SE (Subnegotiation End)

Possible Internal Command Formats



Standard Formats

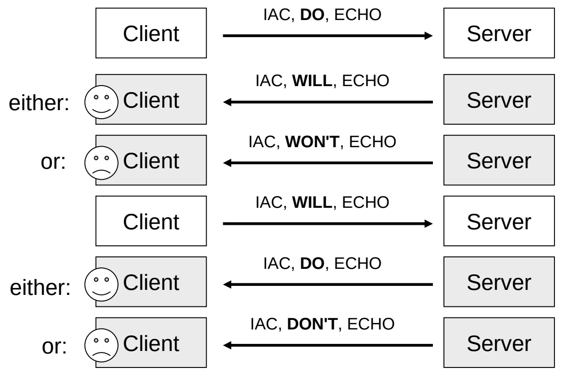
Reference option when negotiating

Chain of commands

Internal Telnet Commands - Overview

	SE	240	End of Subnegotiation
	NOP	241	No Operation
	DM	242	Data Mark (part of the Synch function)
	BRK	243	NVT character break
	GA	249	Go Ahead ("Token" for half duplex mode)
	SB	250	Begin of Subnegotiation
negotiation commands	**WILL**	251	Sender wants to enable an option
	WON'T	252	Sender do not want to enable an option
	DO	253	Sender asks Receiver to enable an option
	DON'T	254	Sender asks Receiver to not enable an option
	IAC	255	Interpret As Command

Command Examples for Negotiation



Client — IAC, **DO**, ECHO → Server

either: Client ← IAC, **WILL**, ECHO — Server

or: Client ← IAC, **WON'T**, ECHO — Server

Client — IAC, **WILL**, ECHO → Server

either: Client ← IAC, **DO**, ECHO — Server

or: Client ← IAC, **DON'T**, ECHO — Server

Important Telnet Options - Overview

0	**Transmit Binary**
1	**Echo**
3	**Suppress Go Ahead**
5	**Status**
6	**Timing Mark**
8	**Output Line Width**
9	**Output Page Size**
24	**Terminal Type**
35	**X Display Location**
39	**Telnet Environment Option**

Important Telnet Options (1)

- **Transmit Binary (Code 0)**
  - toggles from 7-bit ASCII code to 8-bit binary code with IAC stuffing
- **Echo (Code 1)**
  - received data characters will be echoed back to the sender
  - by default local echo (character on screen is echo of client keyboard) is enabled
- **Suppress Go Ahead (Code 3)**
  - toggles from the default half-duplex mode into full-duplex
- **Status (Code 5)**
  - verify the current status of remote Telnet options

Important Telnet Options (2)

- **Timing Mark (Code 6)**
  - causes the a time stamp to be inserted inside the data stream (for synchronisation purposes in full-duplex mode)
- **Terminal Type (Code 24)**
  - to signal some specific terminal type to be used
    - DEC VT-100, IBM 3270
- **Extended Options List (Code 255)**
  - if there is a demand for more than 256 Telnet options, this option can be used to negotiate the availability of an extended option list

Important Telnet Options (3)

- **Telnet Environment Option (Code 39)**
  - enables the server to use its client's environment variables
- **Output Line Width (Code 8)**
- **Output Page Size (Code 9)**
- **X Display Location (Code 35)**

Basic Set of Standard Functions

- **to ease the compatibility of different implementations**
  - a set of standard functions have been specified (= most important functions)
  - each of these commands initiates the processing of a well defined control function

IP	**244**	**Interrupt Process**
AO	**245**	**Abort Output**
AYT	**246**	**Are You There?**
EC	**247**	**Erase Character**
EL	**248**	**Erase Line**
SYNCH	**----**	**Synchronization**

Standard Functions - Explanation (1)

- **IP - Interrupt Process**
  - invokes a system function to suspend, interrupt, abort or terminate the operation of the (remote) process
- **AO - Abort Output**
  - forces the remote system to finish its output, even if there is any outstanding data
- **AYT - Are You There**
  - requires the remote system to send an optical (printable) or acoustic ("beep") signal to indicate that this system is still up and running
- **EC/EL - Erase Character/Line**
  - this function is typically used to edit keyboard input

Standard Functions - Explanation (2)

- **SYNCH - Synchronize**
  - processes in remote systems are sometimes hard to control because some control signals might be buffered anywhere between the sender and the receiver
    - e.g. caused by the networks flow control
  - the Telnet "Synch" mechanism consists of a TCP Urgent notification coupled with the Telnet DM (Data Mark) command
  - on receiving any data stream with the TCP-Urgent data bit set, a server discards all buffered data except commands
  - the Telnet DM-command signals that the desired commands have been already occurred and the server can return with normal processing the data stream

Synchronised Commands

- **the Telnet SYNCH function is applied on the most essential basic functions:**
  - AYT, AO, IP and BRK
- **that is, these characters are send in TCP segments with the Urgent data bit set, followed by a Telnet DM command**

Standard User Commands 1

- **generally Telnet supports the following basic commands:**
  - open <remote IP-address>
    - sets up connection to the remote host
  - close
    - closes connection to the remote host
  - quit, Ctrl-D
    - exits the current Telnet session
  - display
    - shows current Telnet variables
  - set <Telnet variable> <Value>
    - sets Telnet variables to some specific values
e.g. redefining escape sequence

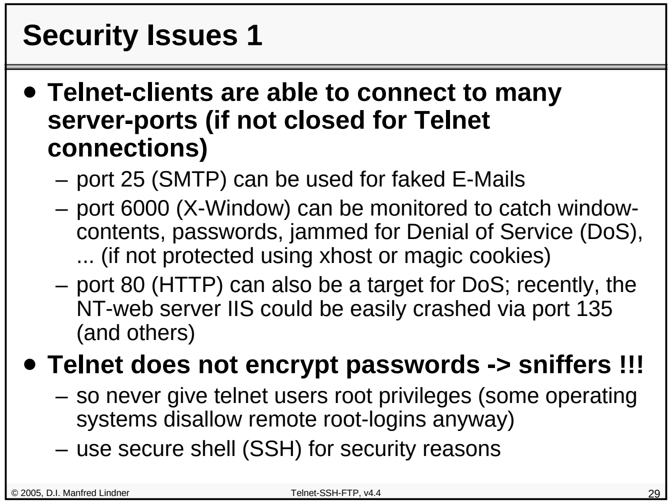
Standard User Commands 2

  - ?
    - help command
  - status
    - provides status information about the current session
  - type <terminal type>
    - enables further terminal functions e.g. VT220 or 3270 emulation
  - mode
    - toggle between ASCII and binary transmission mode

- **see actual User Manual !**

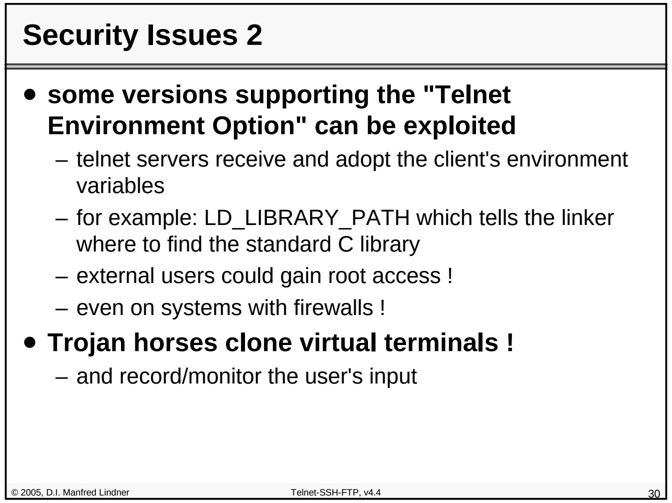
Telnet Applications

- **LYNX**
  - on requesting a web-page via Telnet the printer would display the unformatted HTML-source code
  - Lynx is a terminal-based Web-Browser upon Telnet which can interpret and format the HTML-tags
- **PINE**
  - sophisticated mail user agent
  - commonly started via a Telnet session



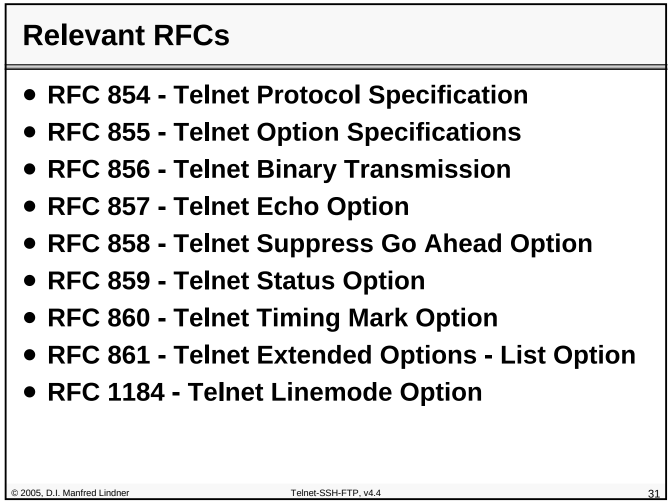
Security Issues 1

- **Telnet-clients are able to connect to many server-ports (if not closed for Telnet connections)**
  - port 25 (SMTP) can be used for faked E-Mails
  - port 6000 (X-Window) can be monitored to catch window-contents, passwords, jammed for Denial of Service (DoS), ... (if not protected using xhost or magic cookies)
  - port 80 (HTTP) can also be a target for DoS; recently, the NT-web server IIS could be easily crashed via port 135 (and others)
- **Telnet does not encrypt passwords -> sniffers !!!**
  - so never give telnet users root privileges (some operating systems disallow remote root-logins anyway)
  - use secure shell (SSH) for security reasons



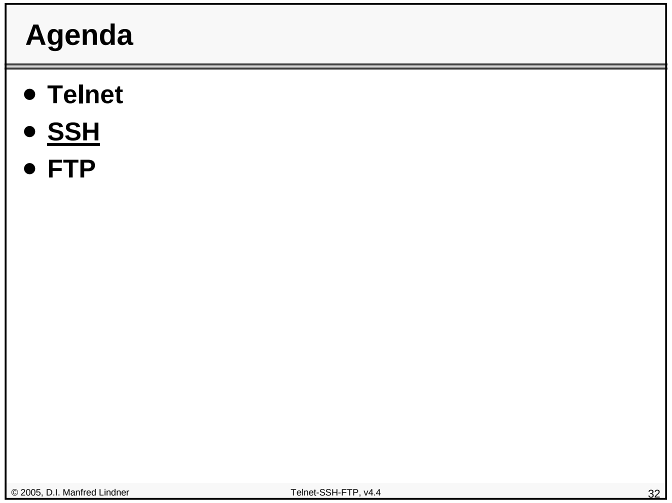
Security Issues 2

- **some versions supporting the "Telnet Environment Option" can be exploited**
  - telnet servers receive and adopt the client's environment variables
  - for example: LD_LIBRARY_PATH which tells the linker where to find the standard C library
  - external users could gain root access !
  - even on systems with firewalls !
- **Trojan horses clone virtual terminals !**
  - and record/monitor the user's input



Relevant RFCs

- **RFC 854 - Telnet Protocol Specification**
- **RFC 855 - Telnet Option Specifications**
- **RFC 856 - Telnet Binary Transmission**
- **RFC 857 - Telnet Echo Option**
- **RFC 858 - Telnet Suppress Go Ahead Option**
- **RFC 859 - Telnet Status Option**
- **RFC 860 - Telnet Timing Mark Option**
- **RFC 861 - Telnet Extended Options - List Option**
- **RFC 1184 - Telnet Linemode Option**



Agenda

- **Telnet**
- **SSH**
- **FTP**

SSH Basics

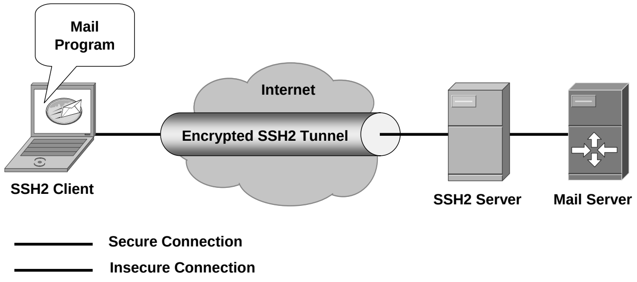
- **Secures connections over the Internet**
- **Encrypting all transmitted confidential data**
  - Passwords
  - Binary files
  - Administrative commands
- **Two versions of Secure Shell (not compatible)**
  - Secure Shell Version 1(SSH1 or SSH)
  - Secure Shell Version 2 (SSH2 or SecSH)
- **De-facto standard**
- **Client-server protocol**

SSH Basics

- **Solve two most acute problems in the Internet**
  - Secure remote terminal logins
    - ssh -l user-name machine-name
  - Secure remote command execution
    - ssh machine-name/path to exe-file
  - Secure file transfers
    - scp file user-name@machine-name
  - Port forwarding
    - ssh -L 3002:hostB:119 hostB
- **Tunnels TCP sessions over encrypted Secure Shell connection**
  - Secure the communication of other applications and protocols <u>without modifying</u> the applications

Principle

E-Mail Security through SSH2 Tunneling



Encryption

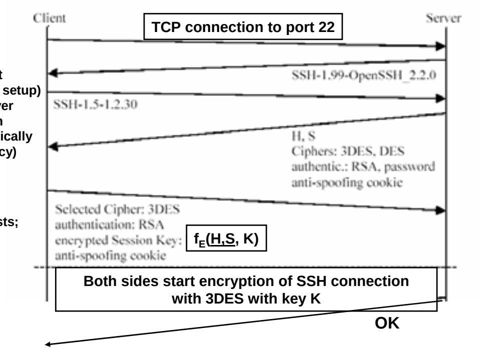
- **Support of the strongest available encryption algorithms**
  - 3DES
  - CAST-128
  - Twofish
  - AES
    - Advanced-Encryption-Standard (US)
    - 128-bit key!

Method	SSH1	SSH2
DES	X	-
3DES	X	X
IDEA	X	-
Blowfish	X	X
Twofish	-	X
Arcfour	-	X
AES	-	X
Cast128-cbc	-	X

SSH (v1) Procedures 1

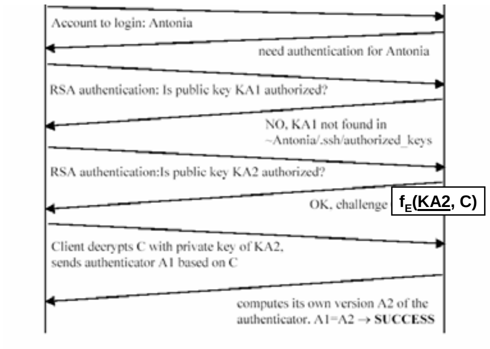
H … public-key RSA Host (generated by ssh server setup)
S … public-key RSA Server (generated by connection start and changed periodically for perfect forward secrecy)

client checks if H, S already know in file ~User/.ssh/known_hosts;
if yes -> ok
if no -> ask user


Client
Server
TCP connection to port 22
SSH-1.99-OpenSSH_2.2.0
SSH-1.5-1.2.30
H, S
Ciphers: 3DES, DES
authentic.: RSA, password
anti-spoofing cookie
Selected Cipher: 3DES
authentication: RSA
encrypted Session Key: $f_E(\underline{H,S}, K)$
anti-spoofing cookie
Both sides start encryption of SSH connection with 3DES with key K
OK

Client performs server authentication by decryption of received message OK

SSH (v1) Procedures 2


Account to login: Antonia
need authentication for Antonia
RSA authentication: Is public key KA1 authorized?
NO, KA1 not found in ~Antonia/.ssh/authorized_keys
RSA authentication:Is public key KA2 authorized?
OK, challenge $f_E(\underline{KA2}, C)$
Client decrypts C with private key of KA2, sends authenticator A1 based on C
computes its own version A2 of the authenticator. A1=A2 → SUCCESS

SSH1 vs. SSH2

- **Two entirely different protocols**
- **SSH1 uses server and host keys to authenticate**
- **SSH2 only uses host keys**
- **SSH2 encrypt different parts of the packet**
- **SSH2 is a complete rewrite of the protocol**
- **SSH2 is more secure**
- **Where to get:**
  - OpenSSH -> http://www.openssh.com/
    - ssh, scp, sftp, sshd, stfp-server
  - PuTTY -> http://www.chiark.greenend.org.uk/~sgtatham/putty/
    - Telnet and SSH client
  - SSH Tectia -> http://www.ssh.com/

Agenda

- **Telnet**
- **SSH**
- **FTP**

File Transfer Protocol FTP (RFC 959)

- **the way information is stored depends on the architecture of the underlying system**
  - hardware- and software-architecture (HW - processor; SW - operating system)
  - data types and coding styles
  - file organization and access methods
- **two approaches possible for exchanging files between different systems**
  - definition of <u>virtual files</u> and translation to real files
  - <u>reduction:</u> extract some few fundamental properties from many individual properties

Virtual File Approach

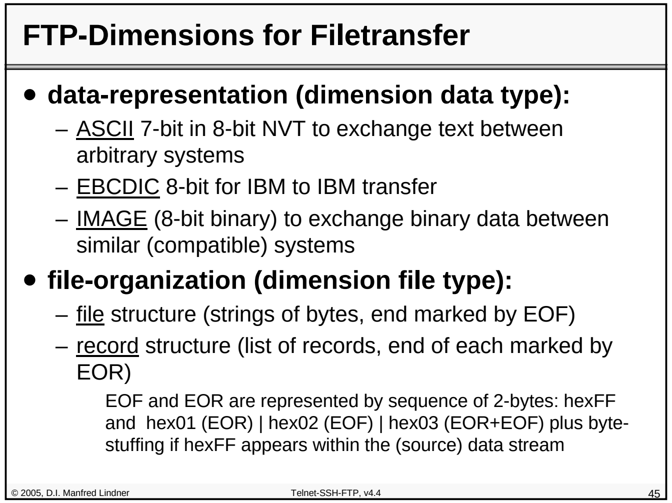
- **all possible representations must be considered**
- **translators from real to virtual file-systems and vice versa must be implemented**
  - complex and difficult to realize
  - advantages: operating systems working with virtual file-systems can easily support a variety of real file-systems
- **examples**
  - ISO FTAM protocol (layer 7)
    - FTAM (File Transfer, Access and Management) also allows to manage a remote file-system
  - Linux Kernel
    - using an internal virtual file-system it was easy to implement support for HPFS, NTFS, FAT, OS/2, System V, UFS, and other file-systems

Reduction Approach

- **based upon common fundamental properties of each file-system**
  - data types, file organization, file ownership and access authority, symbolical names for file identification, I/O-operations, etc.
  - only fundamental views and manipulation operations
    - easy to implement and powerful
  - no translation necessary between different systems
    - application itself is responsible for the appropriate data format
- **<u>example: FTP</u>**

Difference: FTP - File Server OS

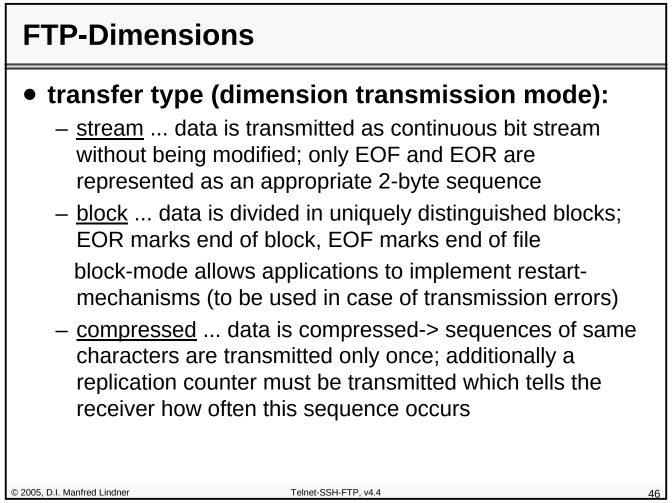
- **FTP: *Sharing by File Transfer***
  - files are copied and forwarded to the local system; the original file remains unchanged
- **File Server OS: *Online Sharing Systems***
  - allows multiple users to share a file over a network
  - files from a fileserver can be accessed and manipulated like local files
  - examples: Novell File Server, Sun NFS, IBM Lan Manager



FTP-Dimensions for Filetransfer

- **data-representation (dimension data type):**
  - ASCII 7-bit in 8-bit NVT to exchange text between arbitrary systems
  - EBCDIC 8-bit for IBM to IBM transfer
  - IMAGE (8-bit binary) to exchange binary data between similar (compatible) systems
- **file-organization (dimension file type):**
  - file structure (strings of bytes, end marked by EOF)
  - record structure (list of records, end of each marked by EOR)

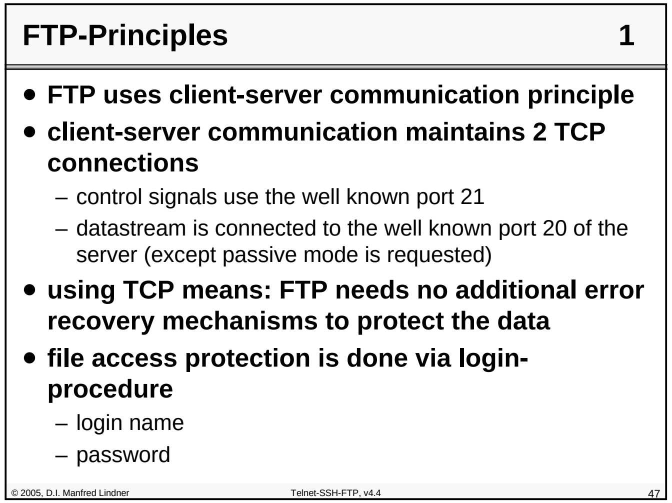
    EOF and EOR are represented by sequence of 2-bytes: hexFF and hex01 (EOR) | hex02 (EOF) | hex03 (EOR+EOF) plus byte-stuffing if hexFF appears within the (source) data stream



FTP-Dimensions

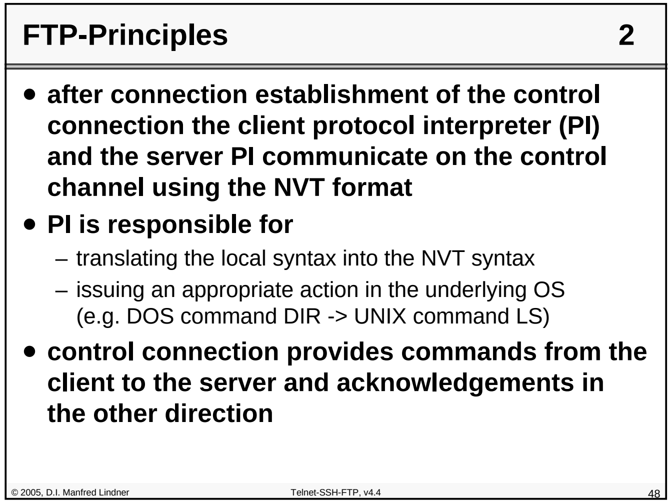
- **transfer type (dimension transmission mode):**
  - stream ... data is transmitted as continuous bit stream without being modified; only EOF and EOR are represented as an appropriate 2-byte sequence
  - block ... data is divided in uniquely distinguished blocks; EOR marks end of block, EOF marks end of file

    block-mode allows applications to implement restart-mechanisms (to be used in case of transmission errors)
  - compressed ... data is compressed-> sequences of same characters are transmitted only once; additionally a replication counter must be transmitted which tells the receiver how often this sequence occurs



FTP-Principles 1

- **FTP uses client-server communication principle**
- **client-server communication maintains 2 TCP connections**
  - control signals use the well known port 21
  - datastream is connected to the well known port 20 of the server (except passive mode is requested)
- **using TCP means: FTP needs no additional error recovery mechanisms to protect the data**
- **file access protection is done via login-procedure**
  - login name
  - password



FTP-Principles 2

- **after connection establishment of the control connection the client protocol interpreter (PI) and the server PI communicate on the control channel using the NVT format**
- **PI is responsible for**
  - translating the local syntax into the NVT syntax
  - issuing an appropriate action in the underlying OS (e.g. DOS command DIR -> UNIX command LS)
- **control connection provides commands from the client to the server and acknowledgements in the other direction**

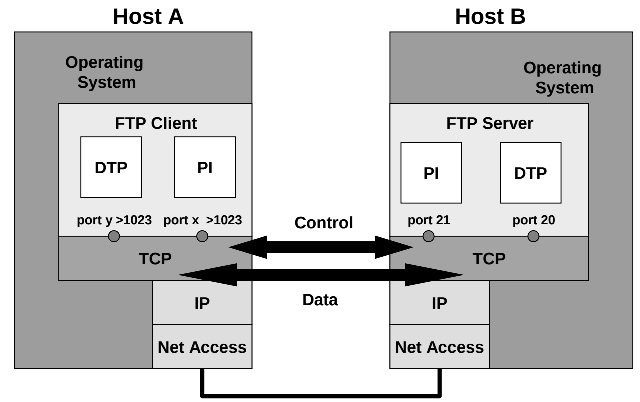
FTP-Principles 3

- **if a command issues a data transfer**
  - – a client DTP (Data Transfer Process) and a server DTP are started to maintain a separate TCP- connection
- **the separate TCP connection for date transfer can be established in two ways**
  - – the client specifies via control connection a portnummer to which the server setups a TCP connection from port 20 (<u>active mode, default mode</u>)
  - – the client requests via control connection passive mode and receives a new port number (> 1023) from the server to which the client establishes the separate TCP connection (<u>passive mode; firewall-friendly</u>)

FTP-Principles 4

- **all data transmission flows over this channel**
- **at the end this connection is closed and the DTP's terminate**
- **this procedure is repeated for each data transmission**
  - – half duplex !

FTP Internal Processes



Control Commands 1

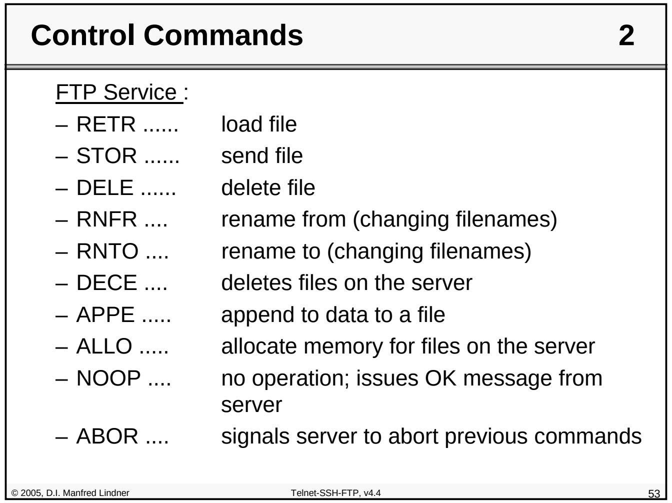
- **commands of the control connection from the client to the server (NVT-format):**

<u>Login Procedure</u>:

– USER	provides username for login
– PASS	provides password of the user; NOTE: transmitted in plain text !!!

<u>Directory Navigation/Creation:</u>

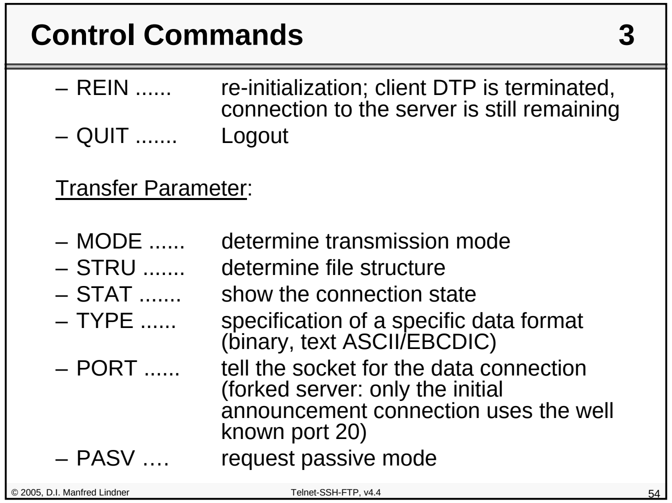
– LIST	list the directory content
– CWD	change the directory
– CDUP	change to the upper directory level
– MKD	create directory
– RMD	remove directory



Control Commands 2

FTP Service :

– RETR	load file
– STOR	send file
– DELE	delete file
– RNFR	rename from (changing filenames)
– RNTO	rename to (changing filenames)
– DECE	deletes files on the server
– APPE	append to data to a file
– ALLO	allocate memory for files on the server
– NOOP	no operation; issues OK message from server
– ABOR	signals server to abort previous commands

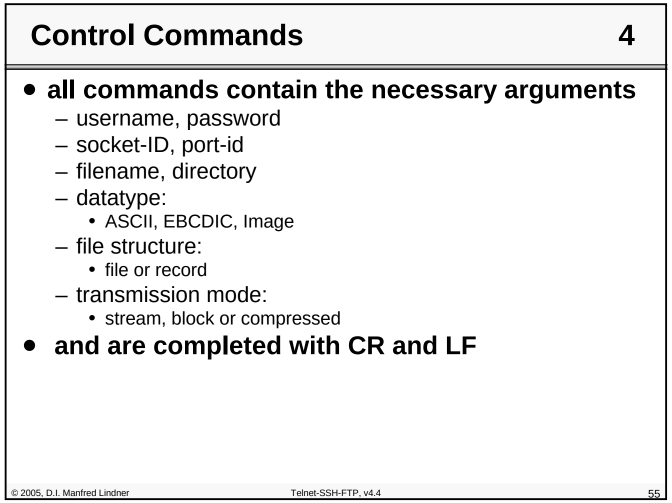


Control Commands 3

– REIN	re-initialization; client DTP is terminated, connection to the server is still remaining
– QUIT	Logout

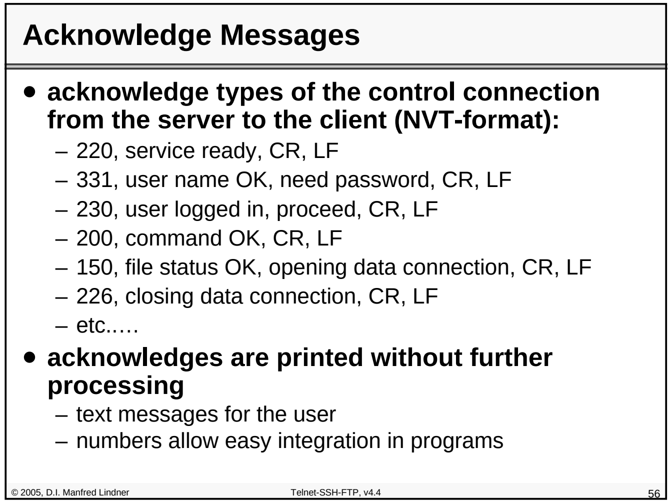
Transfer Parameter:

– MODE	determine transmission mode
– STRU	determine file structure
– STAT	show the connection state
– TYPE	specification of a specific data format (binary, text ASCII/EBCDIC)
– PORT	tell the socket for the data connection (forked server: only the initial announcement connection uses the well known port 20)
– PASV ….	request passive mode



Control Commands 4

- **all commands contain the necessary arguments**
  - username, password
  - socket-ID, port-id
  - filename, directory
  - datatype:
    - ASCII, EBCDIC, Image
  - file structure:
    - file or record
  - transmission mode:
    - stream, block or compressed
- **and are completed with CR and LF**



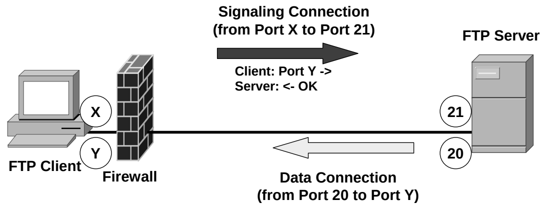
Acknowledge Messages

- **acknowledge types of the control connection from the server to the client (NVT-format):**
  - 220, service ready, CR, LF
  - 331, user name OK, need password, CR, LF
  - 230, user logged in, proceed, CR, LF
  - 200, command OK, CR, LF
  - 150, file status OK, opening data connection, CR, LF
  - 226, closing data connection, CR, LF
  - etc.….
- **acknowledges are printed without further processing**
  - text messages for the user
  - numbers allow easy integration in programs

Acknowledge Coding

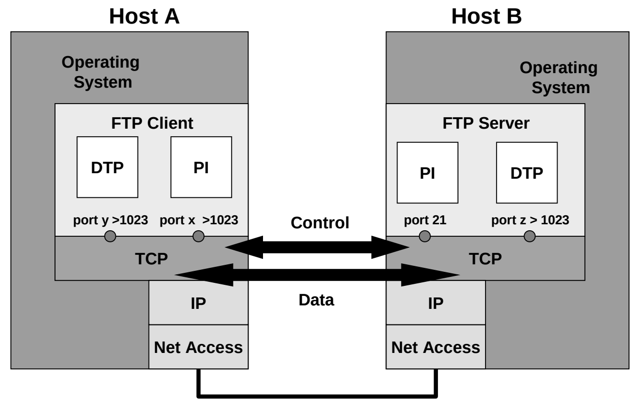
- **<1bc> ... premature positive-acknowledge**
- **<2bc> ... completion-positive-acknowledge**
- **<3bc> ... meantime positive-acknowledge**
- **<4bc> ... transient negative-acknowledge**
- **<5bc> ... permanent negative-acknowledge**
- **<a0c> ... concerns syntax**
- **<a1c> ... concerns commands questioning information**
- **<a2c> ... concerns state of connection**
- **<a3c> ... concerns commands for identification**
- **<a5c> ... concerns file system commands**
- **<ab_> ... detailed acknowledge information**

Operation Mode - Classic

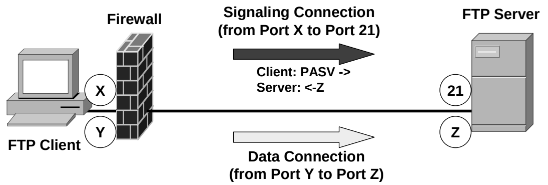


- **Firewall problems**
  - Blocks all incoming connections
- **Old Mode**

FTP Internal Processes (Passive Mode)



Operation Mode - Passive



- **Only outbound connections**
  - No Firewall problems
  - see RFC 1123, 1579, 1635
- **Best mode in secure environment**

User Interface

- **many FTP client software support the following commands through the user interface**

– open	open a FTP connection to a server
– user	announce a new user
– dir, ls	show the directory content
– pwd	show current directory
– cd	change current directory
– lcd	change local directory !
– binary	switch into the image mode
– text	switch into the text-mode (ASCII/EBCDIC) (default?)

Further User Commands

– delete	delete a file on the remote system
– get	receive a file from the server
– put	send a file to the server
– rename	rename a file
– mget	receive multiple files from the server
– mput	send multiple files to the server
– mkdir	create a directory
– rmdir	remove a directory
– exit/quit ...	close the connection to the server
– status	show the connection state
– ?	give help

NOTE: all commands relate to the remote filesystem (filesystem of the server); some commands have local meaning if preceded by a "l"