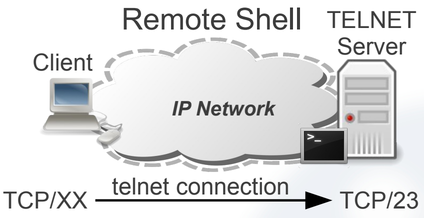
Basic Network Applications

Jose L. Muñoz, Oscar Esparza, Juanjo Alins, Jorge Mata
Telematics Engineering
Universitat Politècnica de Catalunya (UPC)

Outline

Outline

What is TELNET?



- TELNET or TELecommunication NETwork is a standard Internet protocol for emulating a terminal in a remote host using a TCP/IP network.
- The TELNET service follows the client/server model.
- In Linux (and most Unix-like systems) the client is called `telnet` and the server is a daemon called `telnetd`.
- TELNET is a well-known service over TCP and its default port is 23.

Practical Telnet

- Practical example:

```
$ telnet
telnet>open 192.168.0.1
```

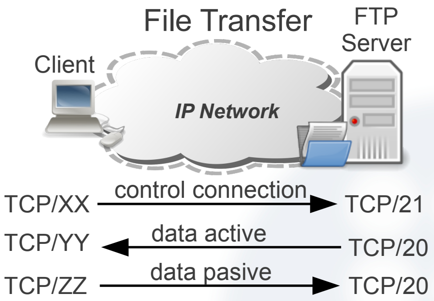
- or use parameters in the command-line:

```
$ telnet 192.168.0.1 23
```

- Once the connection is established, TELNET provides a bidirectional interactive text-oriented communication.
- The commands you type locally, are executed remotely in the host at the other end of the TCP/IP network.
- If at any moment you need to return to the `telnet` sub-shell, you can to type “CRL+ALTGR+]“.
- Because of security issues, the use of the telnet service has been reduced in favor of SSH (secure shell).

Outline

What is FTP? I



- File Transfer Protocol (FTP) is a standard Internet protocol for transmitting files between computers (hosts) on the Internet.
- FTP uses TCP/IP and it is based on a client-server model.
- FTP utilizes separate control and data connections between the client and server.

What is FTP? II

- The default server port for control data is 21 and the default server port for data exchanging is 20.
- The control connection remains open for the duration of the session and it is used for session administration (i.e., commands, identification, passwords) exchanged between the client and server.
- E.g. ”RETR filename“ would transfer the specified file from the server to the client.
- Due to this two-port structure, FTP is considered an out-of-band, as opposed to an in-band protocol such as TELNET.
- The server responds on the control connection with three digit status codes in ASCII with an optional text message.
- E.g. ”200 OK“ means that the last command was successful.

Active and passive modes

- FTP can be run in active or passive mode, which determine how the data connection is established:
    - **In active mode**, the client sends the server the IP address and port number on which the client will listen, and the server initiates the TCP connection. In situations where the client is behind a firewall and unable to accept incoming TCP connections, passive mode may be used.
    - **In passive mode**, the client sends a PASV command to the server and receives an IP address and port number in return. The client uses these to open the data connection to the server.

Practical FTP I

- The `ftp` client supports active and passive mode, ASCII and binary transmissions and only stream mode.
- To connect to a `ftpd` server, we can use one of the following options:

```
$ ftp name
$ ftp 192.168.0.1
$ ftp user@192.168.0.1
```

- To establish an FTP session you must know the ftp username and password.
- When you enter your own loginname and password, it returns the prompt ”ftp>“. This is a sub-shell in which you can type several subcommands.
- As summary of these subcommands is shown in following Table.

Practical FTP II

`open`	opens an FTP session.
`close`	closes and FTP session.
`quit`	exits `ftp`.
`rmdir`	removes a directory in the server.
`mkdir`	creates a directory in the server.
`delete`	removes files in the server.
`get`	downloads a file from the server to the client.
`mget`	downloads multiple files from the server.
`put`	uploads a file from the client to the server.
`mput`	uploads multiple files from the client to the server.
`type ascii`	selects ascii mode.
`type binary`	selects binary mode.
`!`	executes a command in the local shell.
`cd`	remotely change directory (in the server).
`lcd`	locally change directory (in the client).
`ls`	lists files in the remote directory.
`!ls`	lists files in the local directory.
`pwd`	print remote working directory.
`!pwd`	print local working directory.
`verbose`	show debug info.
`status`	show the configuration parameters.
`help`	help.

Practical FTP III

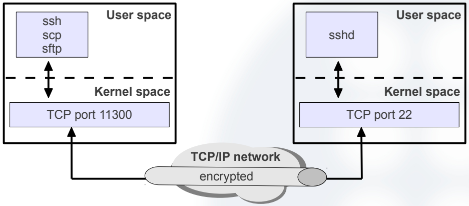
- To use the `ftp` client in passive mode (default is active mode) you can type either `ftp -p` or `pftp`.
- Finally, it is worth to mention that you can also use FTP through a Browser (and also with a several available graphical applications).
- Browsers such as Firefox allow typing the following in the URL bar:

```
ftp://ftp.upc.edu
ftp://ftpusername@ftp.upc.edu
ftp://ftpusername:password@ftp.upc.edu
```

Outline

Features of SSH

- The SSH protocol is used to obtain an encrypted end-to-end TCP connection between a client (`ssh`) and a server (`sshd`) over a TCP/IP network.



- The `sshd` daemon listens to port 22 TCP by default.
- This encrypted connection can be used to build serveral services.

Remote Terminal

- The `ssh` client can be used to connect to a `sshd` server to obtain a remote terminal.
- Same idea as TELNET, but secure.

```
user1$ ssh 192.168.0.1
```

- If you want to use a different user for login on the remote host, simply use remoteusername@hostname:

```
user1$ ssh user2@192.168.0.1
```

Secure Copy (SCP) I

- `scp` is a client program that uses the SSH protocol to send and receive files over an encrypted SSH connection.
- You can transfer files from the client to the server and vice versa.
- Client to server:

```
$ scp file.txt username@remotehost:
```

```
$ scp file.txt username@remotehost:anothername.txt
```

- If you want to copy a file to a **directory relative to the home directory** you can type:

```
$ scp file.txt username@remotehost:mydirectory/anothername.txt
```

Secure Copy (SCP) II

- You can also use **absolute paths** typing ”/“ after the colon.

```
$ scp file.txt username@remotehost:/tmp
```

- To **recursively** copy a directory to the server use the ”-r“ option.

```
$ scp -r Documents username@remotehost:
```

- To copy **from the server to the client** just reverse the from and to:

```
$ scp username@remotehost:file.txt .
```

Secure File Transfer Protocol (SFTP) I

- SFTP is a secure implementation of the traditional FTP protocol using a SSH session.
- Let us take a look at how to use the `sftp` command:

```
$ sftp user@hostname
```

For example:

```
user@192.168.0.1:~$ sftp 192.168.0.2
Connecting to 192.168.0.2...
user@192.168.0.2's password:
sftp> cd django
sftp> ls -l
drwxr-xr-x 2 user user 4096 Apr 30 17:33 website
sftp> cd website
sftp> ls
__init__.py __init__.pyc manage.py settings.py settings.pyc
urls.py urls.pyc view.py view.pyc
sftp> get manage.py
Fetching /home/user/django/website/manage.py to manage.py
/home/user/django/website/manage.py 100% 542 0.5KB/s 00:01
sftp>
```

Secure File Transfer Protocol (SFTP) II

- Some of the commands available under `sftp` are:
  - `cd`. Changes the current directory on the remote machine.
  - `lcd`. Changes the current directory on localhost.
  - `ls`. Lists the remote directory contents.
  - `lls`. Lists the local directory contents.
  - `put`. Send/upload files to the remote machine from the current working directory of the localhost. Supports wildcards for choosing files based on patterns (like *).
  - `get`. Receive/download files from the remote machine to the current working directory of the localhost. Supports wildcards for choosing files based on patterns (like *).

Start/Stop sshd

- The configuration of the sshd server is /etc/ssh/sshd_config.
- If you change the configuration of the daemon you have to stop and start it to apply the changes.
- Under Debian Linux use the script under the directory /etc/init.d:

```
# /etc/init.d/ssh stop
```

- To start the daemon type:

```
# /etc/init.d/ssh start
```

Outline

WEB Servers

- HTTP servers (or WEB servers) listen by default to TCP port 80.
- The “Apache WEB server” is widely spread.
- Current version is `apache2`.
- Debian-based distros store the Apache 2.0 configuration files in the directory /etc/apache2.
- The file ports.conf contains the Listen directives telling `apache2` what IP address and port to listen to.
- In the configuration files of the WEB server, we can also find a parameter called “Document Root”, which tells us where we can find the WEB resources.
- Typically, the Document root for `apache2` is /var/www.
- In this directory we can find a file called index.html that is an HTML file that contains the initial WEB page of the server.

WEB Browsers (clients)

- To access to the resources of a WEB server you have to use a WEB browser.
- The WEB browser is the client application of this service.
- We will use `firefox` (graphical browser) and `lynx` (textual browser).
- With `firefox` you have to type in the URL bar the address of the server.
- For example: http://10.1.1.1.
- Using `lynx` you can type in a terminal the following command to access to a WEB server:

```
$ lynx http://10.1.1.1
```

Start/Stop `apache2`

- If you change the configuration of the daemon you have to stop and start it to apply the changes.
- As most of the network daemons, `apache2` can be started and stopped under Debian Linux using a script under the directory /etc/init.d.
- In particular, to stop `apache2` type:

```
# /etc/init.d/apache2 stop
```

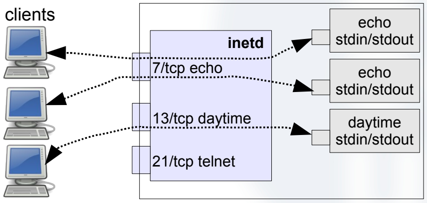
- To start the daemon type:

```
# /etc/init.d/apache2 start
```

Outline

What is `inetd`

- The `inetd` server is sometimes referred to as the Internet ”super-server“ or ”super-daemon“ because `inetd` can manage connections for several services.
- When a connection is received by `inetd`, it determines which program the connection is destined for.
- Spawns the particular process and delegates the socket to it.



Files: /etc/inetd.conf /etc/services /var/log/daemon.log

An inetd Service I

- The main configuration file is /etc/inetd.conf.
- Let's configure the service ”my-echo“ on port 12345.
- Let's add the following line in /etc/inetd.conf:

```
my—echo stream tcp nowait root /root/my—echo.sh
```

- You can restart the `inetd` super-daemon with:

```
# /etc/init.d/openbsd-inetd reload
```

- Restart `inetd` and observe the log file where `inetd` writes its messages (in a Debian system is /var/log/daemon.log).

An `inetd` Service II

- Taking a look at the log file you will observe it is complaining about the port name:

```
# tail -f /var/log/daemon.log
inetd[1173]: my-echo/tcp: unknown service
```

- Let’s add the port number for our service ”my-echo“ in the file /etc/services.
- The file /etc/services is the system file to map a port number/protocol to a service name.
- In this example, we will add the following line to /etc/services:

```
my-echo             12345/tcp         # By  telematics
```

- Restart `inetd` and try to connect to the service.

An inetd Service III

- Let's observe the log file:

```
# tail -f /var/log/daemon.log
execv /root/my-echo.sh: No such file or directory
```

- Create the file my-echo.sh with the following content:

```
cat
```

- The command `cat` without parameters reads writes to STDOUT what it reads from STDIN.
- Try to connect to the service and observe the log file:

```
# tail -f /var/log/daemon.log
inetd[1204]: execv /root/my-echo.sh: Permission denied
```

An inetd Service IV

- Give permissions to the script:

```
# chmod u+x /root/my-echo.sh
```

- Try to connect to the service and observe the log file:

```
# tail -f /var/log/daemon.log
inetd[1207]: execv /root/my-echo.sh: Exec format error
```

- We need a script ”formally correct“, to do so, you must include the initial line with the corresponding interpreter (/bin/bash in our example):

```
#!/bin/bash
cat
```

Stand-alone vs. `inetd` Services

- **Stand-alone:**
  - When a daemon is started stand-alone it must open its service port for listening and manage incoming service requests.
- **`inetd`:**
  - When a service is managed with `inetd`, the server program can just use STDIN and STDOUT, which are inherited from `inetd`.
  - In this case, as mentioned, the server program does not need any specific code to manage the network.
- Check our echo service: open two clients with `netcat` and check the open files of `inetd` and of the processes executed by `inetd`.