Washington University in St. Louis

Engineering

4. File Transfer and Electronic Mail in the Internet

- FTP Protocol
- Secure Shell (SSH) and Secure FTP (SFTP)
- Components of an Email System
- Simple Mail Transfer Protocol
- Mail Access Protocols

Jon Turner – based partly on material from Kurose and Ross

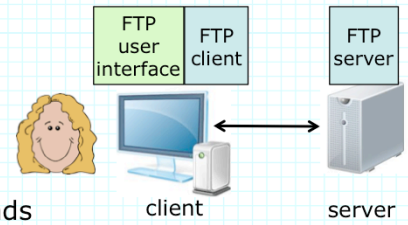
RFCs and the IETF

- Requests for Comments (RFC) are used to document the Internet protocols
  - » available at http://www.ietf.org/rfc.html
- RFCs are produced and maintained by Internet Engineering Task Force (ietf.org)
  - » open, volunteer standards organization responsible for developing and maintaining Internet protocols
    - anyone can join and participate
    - strong emphasis on technical competence
    - pragmatic philosophy of "rough consensus and running code"
    - companies work to influence process, but participants are treated as individuals
  - » most activities occur online through mailing lists
  - » three meetings per year – open to all

2

File Transfer Protocol (RFC 959)

- Used to transfer files between hosts
  - » client initiates "session" and issues commands
  - » server accepts connections from clients (port 21) and responds



- FTP uses separate "control" and "data" connections
  - » client connects on port 21 (the control connection) and can use it to browse remote file system
  - » when client issues file transfer command, server initiates a separate connection on port 20 (data connection) to xfer file
    - connection closed when transfer completes
    - server (usually) opens separate connection for each file transfer
  - » this style of control sometimes called "out-of-band" control
  - » server maintains session state (local directory being browsed)

3

FTP commands, responses

sample commands:

- sent as ASCII text over control channel
- `USER` *`username`*
- `PASS` *`password`*
- `LIST` return list of files in current directory
- `RETR filename` retrieves (gets) file
- `STOR filename` stores (puts) file onto remote host

sample return codes

- status code and phrase (as in HTTP)
- `331 Username OK, password required`
- `125 data connection already open; transfer starting`
- `425 Can't open data connection`
- `452 Error writing file`

More About FTP

- FTP can be used to transfer files between remote hosts
  - » client opens control connections to both servers to setup transfer
  - » one server initiates data connection to other and transfers file directly
- FTP supports a "passive mode"
  - » client establishes the data connection to the server
  - » allows connection to clients that are behind firewalls/NATs
- FTP is completely insecure
  - » sends user names, passwords and data in clear text
    - many systems no longer support it for this reason
    - anonymous FTP remains useful for public access to files
  - » common modern replacement involves transferring files using the Secure Shell (SSH) protocol – often referred to as SFTP

5

FTP Implementation Issues

- FTP client program
  - » command-line interpreter with "get", "put", "ls" commands
  - » or, GUI that shows local and remote files sytems and supports drag-and-drop interface
  - » uses the FTP protocol to interact with remote server
- FTP server (aka daemon)
  - » accepts connections from remote clients and interacts using the FTP protocol
  - » must accommodate differences among host operating systems and their file system interfaces
    - client and server hosts may use different data formats
- Key challenge is ensuring consistent operation between client and daemon
  - » multiple implementations of both; dealing with failures

6

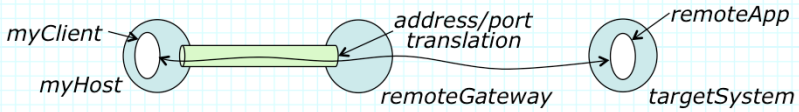
Exercises

1. FTP uses TCP. Why is this an appropriate choice?
2. In some respects, the FTP client program acts like a server. Explain.
3. When was the first RFC for FTP written? Who wrote it? When was RFC 959 written? Who wrote it?
4. List at least 5 commands that an FTP server must implement; list both the command name and briefly describe what it does.
5. Name the two character representations supported by FTP.
6. What are the three file structures supported by FTP?
7. What is the default port number used by a client when connecting to an FTP server?
8. The RFC for FTP refers to the port number used by the client program as U. This same port number is used for data connections. Why don't these two uses conflict?

Secure Shell (SSH)

- SSH is an application designed to replace remote login apps such as telnet and rlogin
  - » encrypts all data including user names and passwords
  - » basic usage: ssh userName@someServer.wustl.edu
    - gui-based clients also available, but command-line usage more versatile
- Remote command execution
  - » allows execution of single command on remote computer
  - » usage: ssh userName@someServer.wustl.edu command args...
- Public-key authentication
  - » by configuring key-pair on client and server, can bypass password dialog – generate key pair using ssh-keygen
    - place public key on server in ~/.ssh/authorized_keys, private key on client at ~/.ssh/id_rsa
- SFTP is a file transfer application based on SSH

SSH Tunneling



- SSH tunneling allows a host to login to a server in a remote network allowing secure access to services
- Command "ssh –L 4567:*targetSystem*:5678 *remoteGateway*" opens SSH connection to *remoteGateway* and creates a *tunnel*
  - » when local app opens TCP connection to *localhost*:4567, the connection is forwarded through the SSH tunnel to *remoteGateway* and from there to *targetSystem*:5678
  - » *remoteGateway* changes source address/port numbers so that *targetSystem* "thinks" connection comes from *remoteGateway*
- For more on ssh, see man pages (MacOs, Linux)
  - » *man ssh* and *man –s8 sshd*

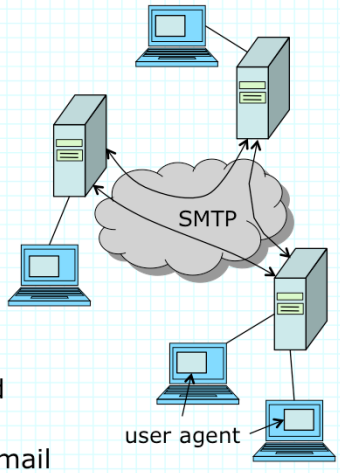
9

Exercises

1. Suppose you don't want to type your password whenever you login to shell.cec.wustl.edu. What can you do, so that you don't have to? Try it.
2. Consider the following situation. The TCP echo server from the previous lecture is running on a server called *secureServer*, which is behind a firewall that allows only ssh connections to pass through it. Explain how you can setup a tunnel using ssh to connect to *secureServer* and then access the echo server using the tunnel and the TCP echo client. Show all the commands used to do this.

Electronic Mail

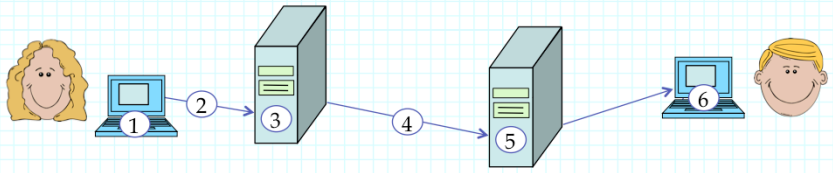
- Mail servers
  - » host users' mailboxes
  - » forward mail to other servers
    - messages held in outgoing *message queue* until transferred
- User agents – aka mail readers
  - » provide user interface for reading mail
  - » examples: Outlook, Thunderbird
  - » not required for web-based access
- Simple Mail Transfer Protocol (SMTP)
  - » used by originating mail server to send email to destination server
  - » also used by user agents to hand off email to "home server"



Electronic Mail: SMTP [RFC 2821]

- Uses TCP to reliably transfer email message from client to server, port 25
- Direct transfer: sending server to receiving server
- Three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- Command/response interaction
  - commands: ASCII text
  - response: status code and phrase
- Messages must be in 7-bit ASCII
  - requires non-ASCII data to be encoded as ASCII

Scenario: Alice sends email to Bob



1) Alice uses UA to compose message "to" `bob@someschool.edu`
2) Alice's UA sends message to her mail server; message placed in message queue
3) Client side of SMTP opens TCP connection with Bob's mail server
4) SMTP client sends Alice's message over the TCP connection
5) Bob's mail server places the message in Bob's mailbox
6) Bob invokes his user agent to read message

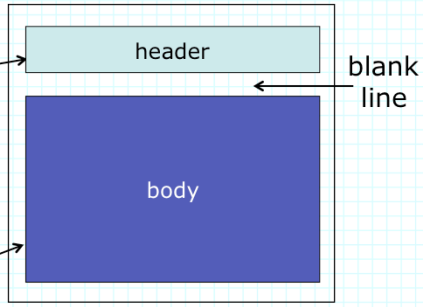
Sample SMTP Interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Mail Message Format [RFC 822]

- Note: mail message formats defined separately from SMTP
  - » SMTP defines only how messages are exhanged
- Header lines, e.g.,
  - » To:
  - » From:
  - » Subject:

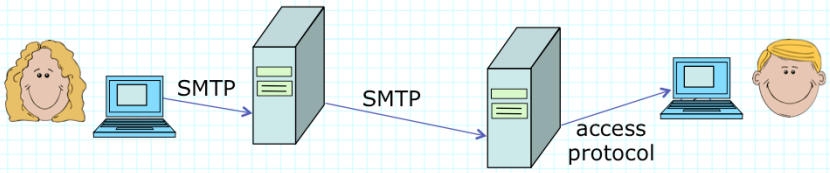
  different from SMTP commands!
- Body
  - » the "message", ASCII characters only



Exercises

1. The SMTP RFC talks about SMTP servers acting as "relays". Explain difference between a transfer that involves relays and one that does not. Describe a common situation that uses a relay.
2. Modern implementations of SMTP support an "extension mechanism". How does an extended SMTP recognize a peer that also supports extensions? What is the minimum set of commands that a modern SMTP implementation is required to support?
3. What does the RFC say about the problem of email "spoofing"? What approaches do they recommend for dealing with this? How do they justify the vulnerability of SMTP to spoofing?
4. Explain what a "trace record" is, in the context of SMTP. Identify the trace records in a recent email you have received.
5. How does an SMTP server respond to a "RCPT TO" command when it knows that the specified email address is not valid?
6. Explain what the VRFY command is. Give an example of its use, and the response from the server.

Mail Access Protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - » POP: Post Office Protocol [RFC 1939]
    - authorization (agent <-->server) and download
  - » IMAP: Internet Mail Access Protocol [RFC 1730]
    - more features (more complex)
    - manipulation of stored msgs on server
  - » HTTP: gmail, Hotmail, Yahoo! Mail, etc.

POP3 Protocol

Authorization phase

- client commands:
  - **`user`**: declare username
  - **`pass`**: password
- server responses
  - **`+OK`**
  - **`-ERR`**

Transaction phase (client)

- **`list`**: list message numbers
- **`retr`**: retrieve message by number
- **`dele`**: delete
- **`quit`**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully
   logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 2 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing
   off
```

POP3 (more) and IMAP

more about POP3

- previous example uses "download and delete" mode.
- Bob cannot re-read e-mail if he changes client
- "download-and-keep": copies of messages on different clients
- POP3 is stateless across sessions

IMAP

- keeps all messages in one place: at server
- allows user to organize messages in folders
- keeps user state across sessions:
  - » names of folders and mappings between message IDs and folder name

What Makes a Good Protocol?

- Key objective: make it easy to build software that uses the protocol correctly
  - » a successful protocol will be used in many different applications, running on different devices, written by different people
    - e.g. servers, laptops, tablets, cell phones
    - all combinations must work!!
  - » so, protocol definition should be precise and easy to understand
    - can be hard to achieve both
    - internet takes pragmatic approach combining detailed but informal protcol descriptions (RFCs) plus reference implementations
- Text-based protocols are easier to get right
  - » easy to observe protocol interactions and find problems
  - » no byte order bugs
  - » comes at some cost: longer messages, more complex parsing of received messages (and associated coding effort)

20