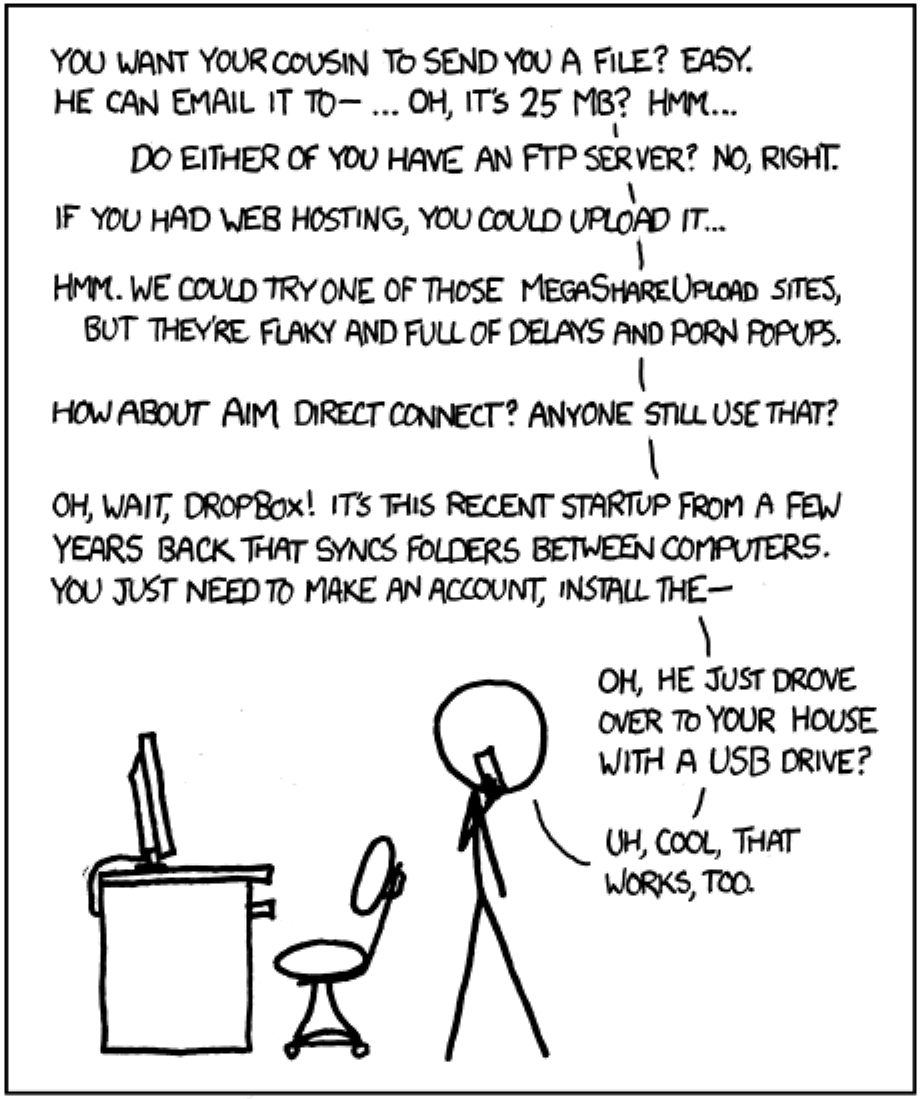
File Transfer Protocol (FTP) & SSH


YOU WANT YOUR COUSIN TO SEND YOU A FILE? EASY. HE CAN EMAIL IT TO— ... OH, IT'S 25 MB? HMM...
DO EITHER OF YOU HAVE AN FTP SERVER? NO, RIGHT.
IF YOU HAD WEB HOSTING, YOU COULD UPLOAD IT...
HMM. WE COULD TRY ONE OF THOSE MEGASHAREUPLOAD SITES, BUT THEY'RE FLAKY AND FULL OF DELAYS AND PORN POPUPS.
HOW ABOUT AIM DIRECT CONNECT? ANYONE STILL USE THAT?
OH, WAIT, DROPBOX! IT'S THIS RECENT STARTUP FROM A FEW YEARS BACK THAT SYNCS FOLDERS BETWEEN COMPUTERS. YOU JUST NEED TO MAKE AN ACCOUNT, INSTALL THE—
OH, HE JUST DROVE OVER TO YOUR HOUSE WITH A USB DRIVE?
UH, COOL, THAT WORKS, TOO.

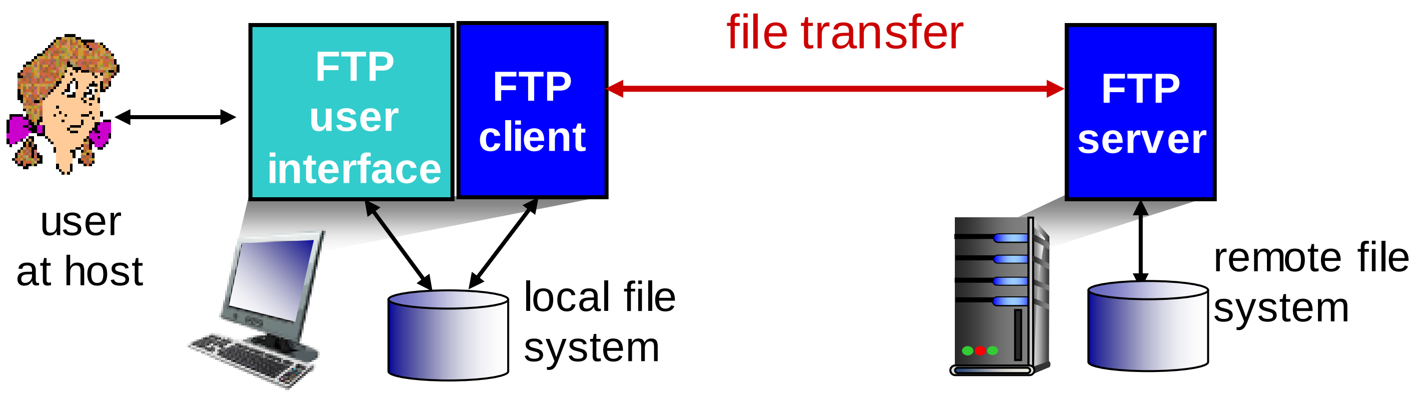
I LIKE HOW WE'VE HAD THE INTERNET FOR DECADES, YET "SENDING FILES" IS SOMETHING EARLY ADOPTERS ARE STILL FIGURING OUT HOW TO DO.

http://xkcd.com/949/

Overview

- 2.1 Principles of network applications
  - App architectures
  - App requirements
- 2.2 Web and HTTP
- **2.3 FTP**
- 2.4 Electronic mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with UDP and TCP

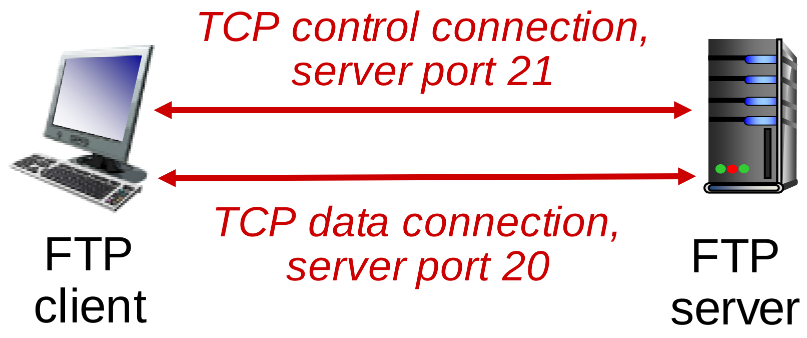
FTP: the file transfer protocol



- Transfer file to/from remote host
- Client/server model
  - *Client:* side that initiates transfer (either to/from remote)
  - *Server:* remote host
- FTP: RFC 959
- FTP server: port 21

Separate control/data connections

- FTP client contacts FTP server at port 21, using TCP
- Client authorized over control connection
- Client browses remote directory, sends commands over control connection
- When server receives file transfer command, *server* opens 2^{nd} TCP data connection (for file) *to* client
- After transferring one file, server closes data connection


TCP control connection, server port 21
TCP data connection, server port 20
FTP client
FTP server

- Server opens another TCP data connection to transfer another file
- Control connection: *"out of band"*
- FTP server maintains "state": current directory, earlier authentication

FTP commands & responses

Sample commands:

- Sent as ASCII text over control channel

`USER` ***`username`***

`PASS` ***`password`***

`LIST` return list of file 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`

Remote login

- TErminaL NETwork (TELNET)
  - 1969
  - TCP port 23
  - Insecure, usernames and passwords in the clear
- Secure shell (SSH)
  - 1995
  - TCP port 22

```
telnet> }quit
?Invalid command
telnet> exit
?Invalid command
telnet> q
Connection closed.
[tswalker@ryo-ohki ~]$
```

```
You have nyaned for 29 seconds!
```

http://www.flickr.com/photos/timjoyfamily/7409585334/

SSH

- SSH-TRANS
  - Provides transport layer security
  - Encrypted channel over TCP
  - Server authentication
- SSH-AUTH
  - Authentication of the client
- SSH-CONN
  - Allow multiplexing of secure channel
  - Port forwarding, allow applications to communicate over secure tunnel

SSH

- Client contacts server
  - Server sends its public encryption key
  - Client asked to accept the key, then stores for future connections to same server
  - First-time risk of imposter server
- Negotiates encryption protocol
  - Session key established for symmetric encryption, e.g. AES
- Client logs in via:
  - Password, public-key encryption, or host-based authentication (from a trusted server)

SSH tunneling

- Advantages:
  - SSH handles security so your app doesn't have to
  - Get through firewall (as long as SSH port 22 open)

```
vertanen@katie:~$ python TCPServer.py 12000
The server is ready to receive
```

```
c:\progs\python TCPClient.py katie.mtech.edu 12000
Traceback (most recent call last):
  File "TCPClient.py", line 21, in <module>
clientSocket.connect((serverName,serverPort))
  File "<string>", line 1, in connect
socket.error: [Errno 10060] A connection attempt failed
because the connected party did not properly respond after a
period of time, or established connection failed because
connected host has failed to respond
```

SSH tunneling

```
vertanen@katie:~$ python TCPServer.py 12000
The server is ready to receive
```

```
ssh -f vertanen@katie.mtech.edu -L 2345:katie.mtech.edu:12000 -N

Montana Tech of the University of Montana
Department of Computer Science

If you can't remember your password, email tipowell@mtech.edu
from your tech email.
vertanen@katie.mtech.edu's password:

c:\Dropbox\mtech\networks\progs\UpperCase>python TCPClient.py
localhost 2345
Input lowercase sentence:this is a test
From Server: THIS IS A TEST
```

Summary

- FTP
  - Unlike HTTP: out-of-band control channel
  - Unlike HTTP: maintains state
- SSH
  - Replacement for old and insecure TELNET
  - Provide secure login, file copying, tunneling, etc.