Chapter 12
The Application Layer

There are very many applications that run over the IP. We can only give the briefest introduction to a few of these as many applications could warrant an entire course to themselves

Applications

- Telnet. Logging in to a remote machine for interactive use: 1969. Uses TCP
- File Transfer Protocol (FTP). A basic way of transferring files. TCP
- Simple Mail Transfer Protocol (SMTP). Email. TCP
- Network File System (NFS). Allows remote disks to appear as a local file system. UDP and TCP

Applications

- HyperText Transfer Protocol (HTTP). For requesting and transferring Web pages. Web pages are encoded using Hypertext Markup Language (HTML). TCP
- Wireless Application Protocol (WAP). A variant on HTTP more suited to the low bandwidths as supplied by mobile phone networks
- Internet Radio. Radio broadcasting
- Television over IP (IPTV). TV broadcasting

Applications

- Voice over IP (VoIP). Telephone over the Internet
- Many others. Finger, whois, LDAP, X Window System, ssh, pop, imap, USENET news, network time synchronisation, and so on

Applications

Telnet

- Telnet is a fairly simple application: it connects to a port and relays back and forth between the user's terminal and the program running on that port
- telnet mailhost.example.com 25

would connect to (probably) the mail server on that host

Applications

Telnet

- More usually telnet connects to a login program that runs on port 23
- This program would ask for a username and password and then log you in
- Telnet then relays characters between you and a shell running on the remote machine

Applications

Telnet

telnet example.com

Trying 10.0.0.1...

Connected to example.com

Escape character is '^]'

Welcome to Suse Linux 9.1 (i586) – Kernel 2.6.5-7-default (7).

example.com login: rjbradford

Password:

Last login Thu May 19 20:54:08 from fire

%

Applications

Telnet

- Telnet was one of the earliest applications developed for the Internet as its main purpose is to give remote access to machines and the original purpose of the Internet was to give remote access to machines

Applications

Telnet

- This connection is insecure in that data (including the password) is readable by machines that route the packets as they travel
- An eavesdropper could read confidential information as it goes past
- So telnet is no longer recommended and has been replaced by the more sophisticated *secure shell* (ssh)

Applications

FTP

- The *file transfer protocol* (FTP), with telnet and email, is another very early application
- Its purpose is to share files

Applications

```
% ftp example.com
Trying 172.17.2.1...
Connected to example.com
(172.17.2.1).
220 example.com FTP server ready.
Name (example.com:rjbradford):
rjbradford
331 Password required for rjbradford.
Password:
230 User rjbradford logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Applications

```
ftp> dir
227 Entering Passive Mode (172,17,2,1,247,214)
150 Opening ASCII mode data connection for directory
listing.
total 122
-rw-r--r--   1 1082     100        158535 Aug 14 18:46
6month.ps
drwxr-xr-x   6 1082     100          2048 Sep  4 19:20 Misc
-rw-r--r--   1 1082     100        129761 Dec 21  2000
mrbenn.ogg
-rw-r--r--   1 1082     100          7100 Jun 20  2000 tcp.eps
-rw-r--r--   1 1082     100           216 Apr  8  1999 test.c
226 Transfer complete.
ftp>
```

Applications

The primary use is to transfer files:

```
ftp> get test.c
local: test.c remote: test.c
227 Entering Passive Mode (172,17,2,1,71,184)
150 Opening BINARY mode data connection for test.c (216
bytes).
226 Transfer complete.
216 bytes received in 0.0199 secs (11 Kbytes/sec)
ftp>
```

Applications

FTP

- There is also a *put* command to send a file

```
ftp> bye
221-You have transferred 216 bytes in 1 files.
221-Total traffic for this session was 923 bytes in 1
transfers.
221 Thank you for using the FTP service on
example.com.
```

Applications

FTP

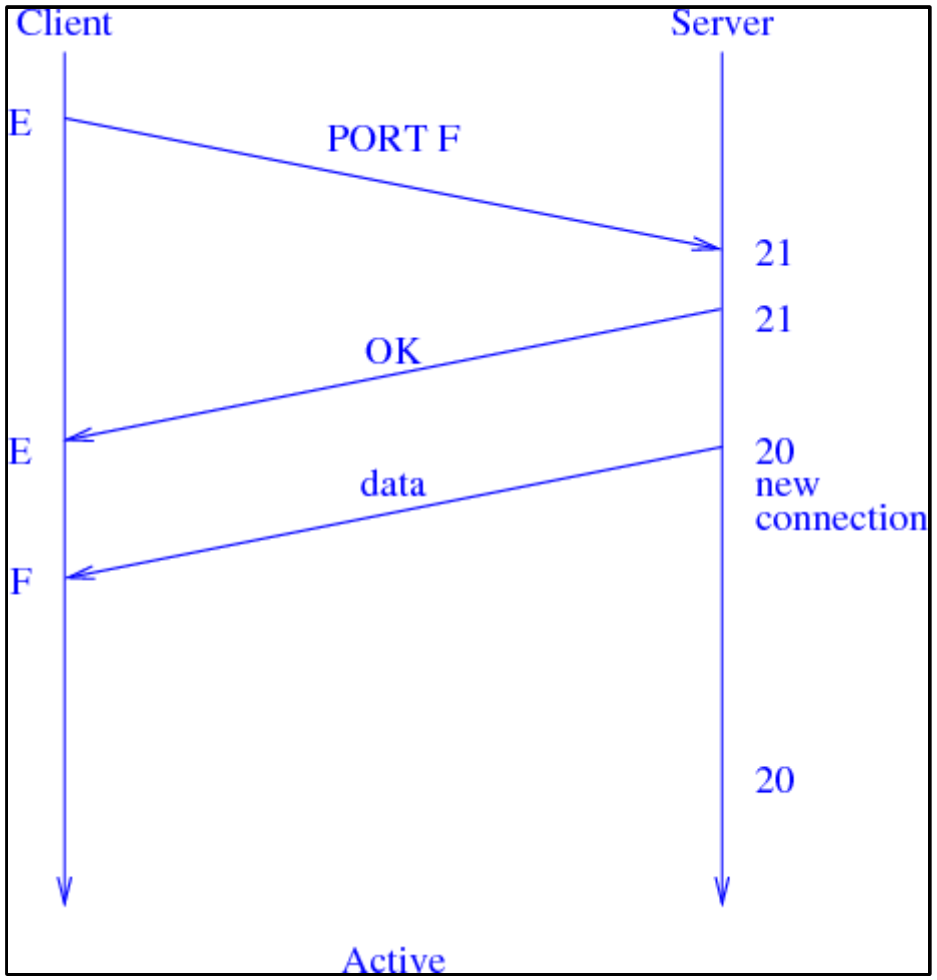
- The way files are copied seems eccentric to the modern eye: early systems used *active mode* transfer, while later systems use *passive mode*
- In active mode, a new reverse TCP connection is made from the server back to the client whenever data is to be transmitted and the data flows over that new connection

Applications

FTP

- This causes problems with firewalls (see later) and so *passive mode* was developed, where the client initiates the data connection

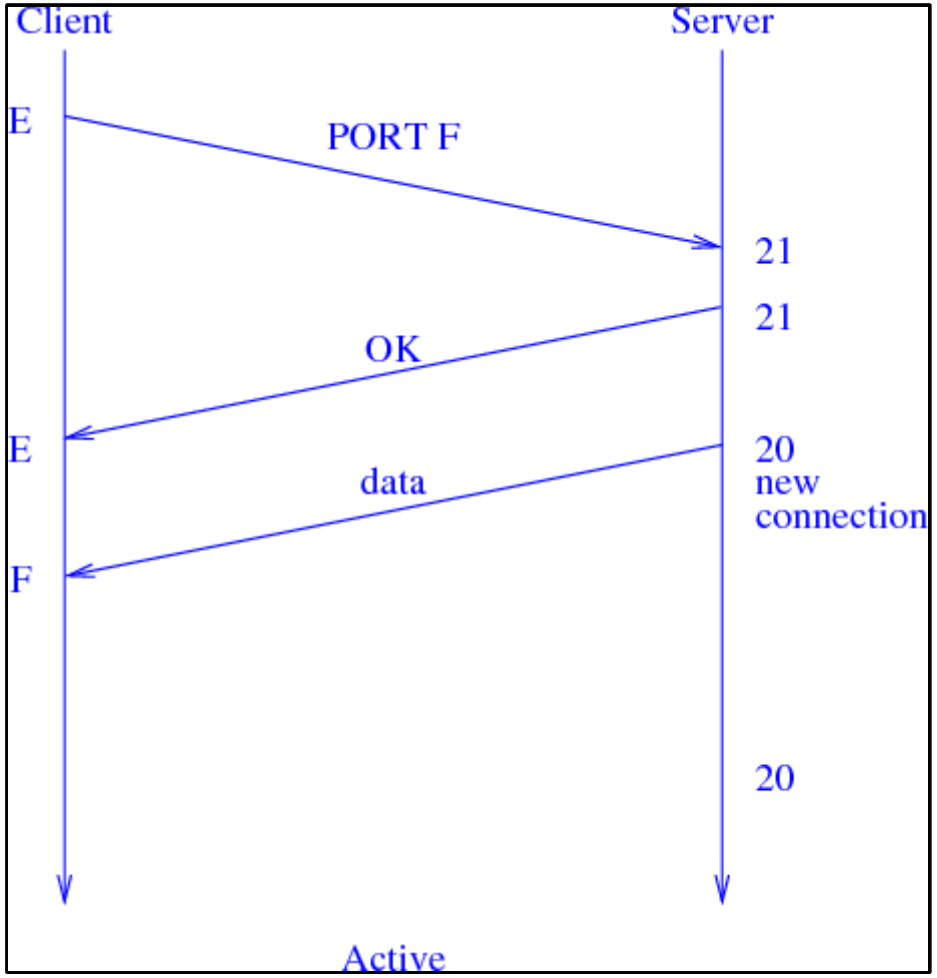
Applications



- In active FTP:

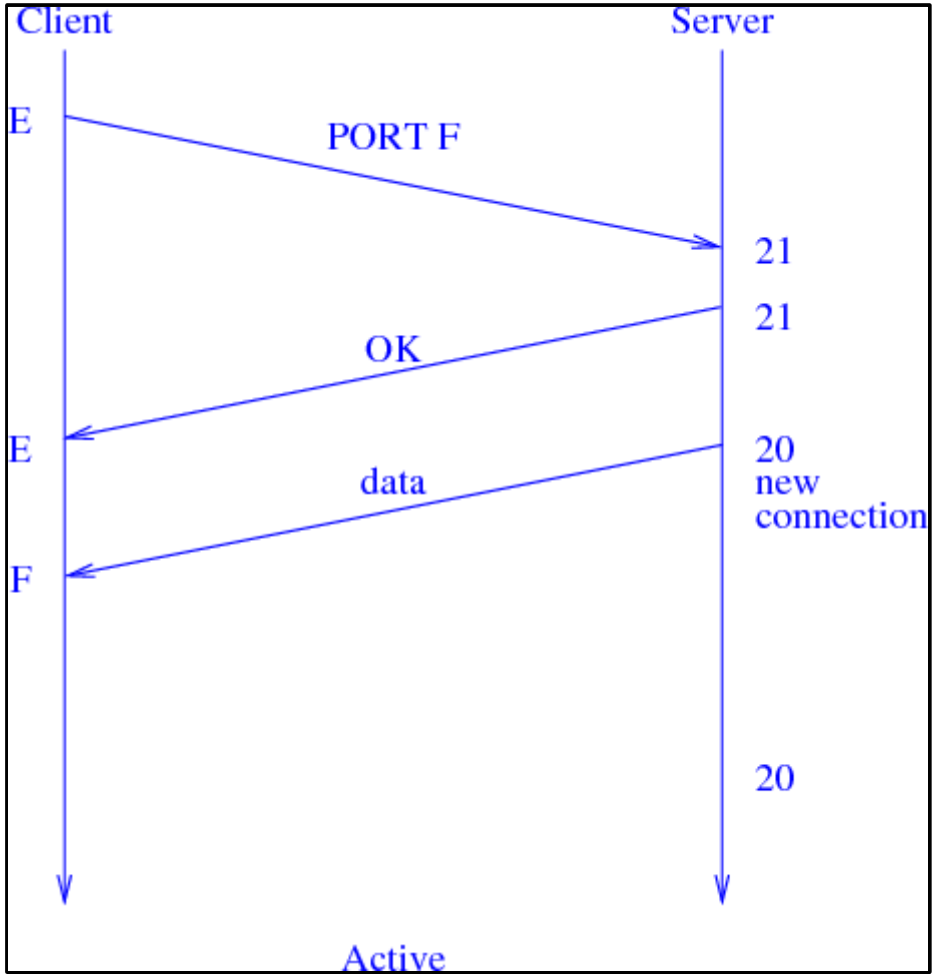
0. the client connects from some ephemeral port *E* to port 21 on the server. This is the control connection

Applications



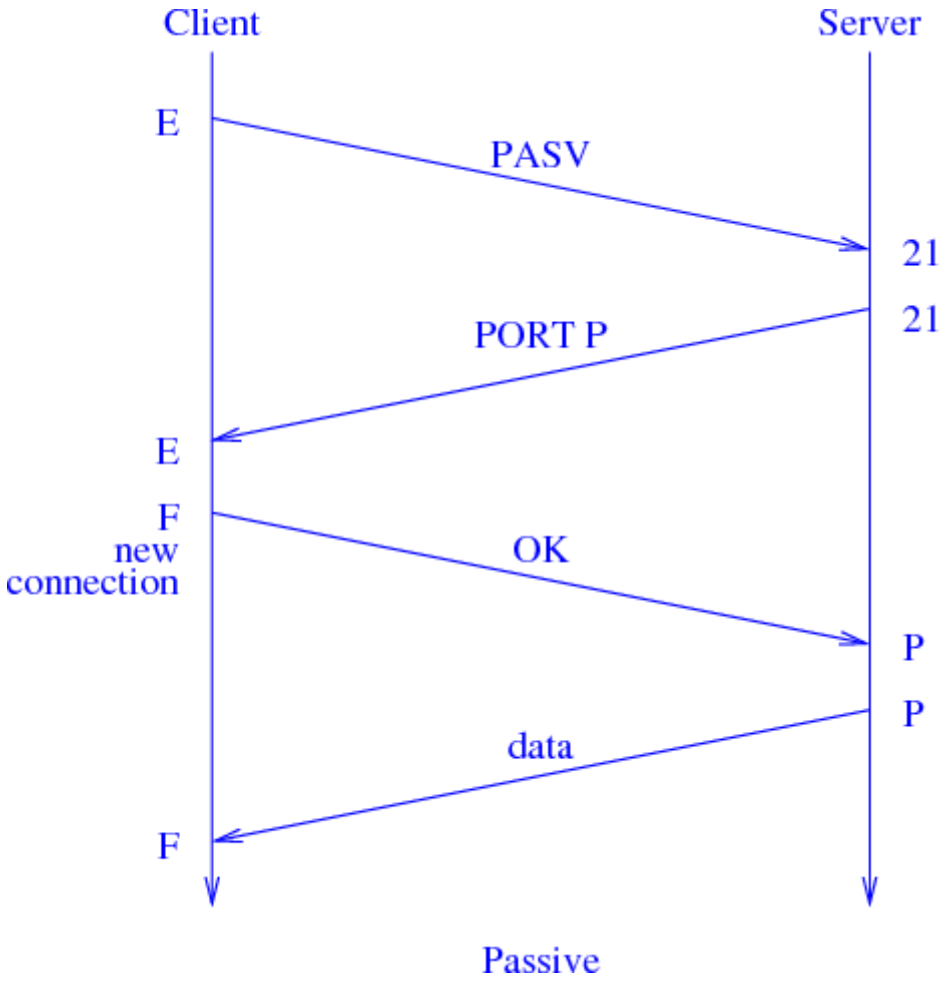
2. The client issues a PORT *F* commands, where *F* is an ephemeral port on the client. The client is listening on *F*

Applications



3. The server creates a new TCP connection from port 20 (FTP data port) to *F*. This connection is used to send data

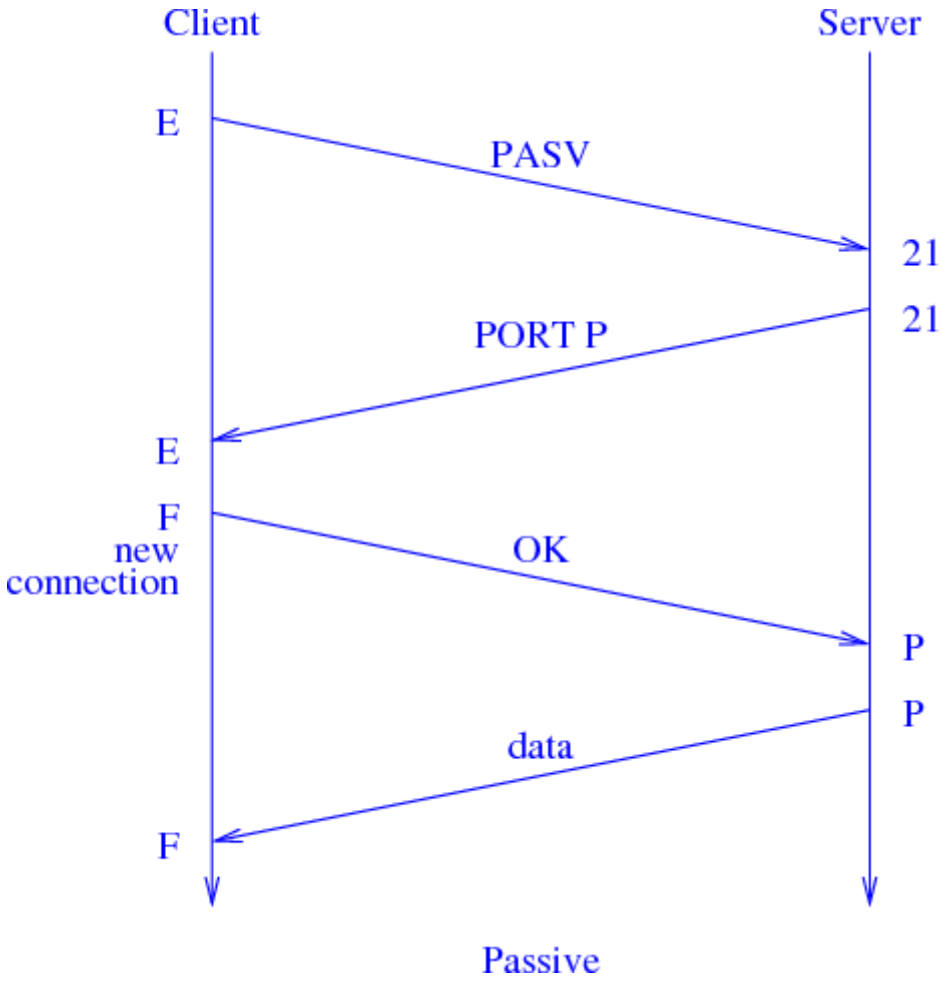
Applications



- In passive FTP:

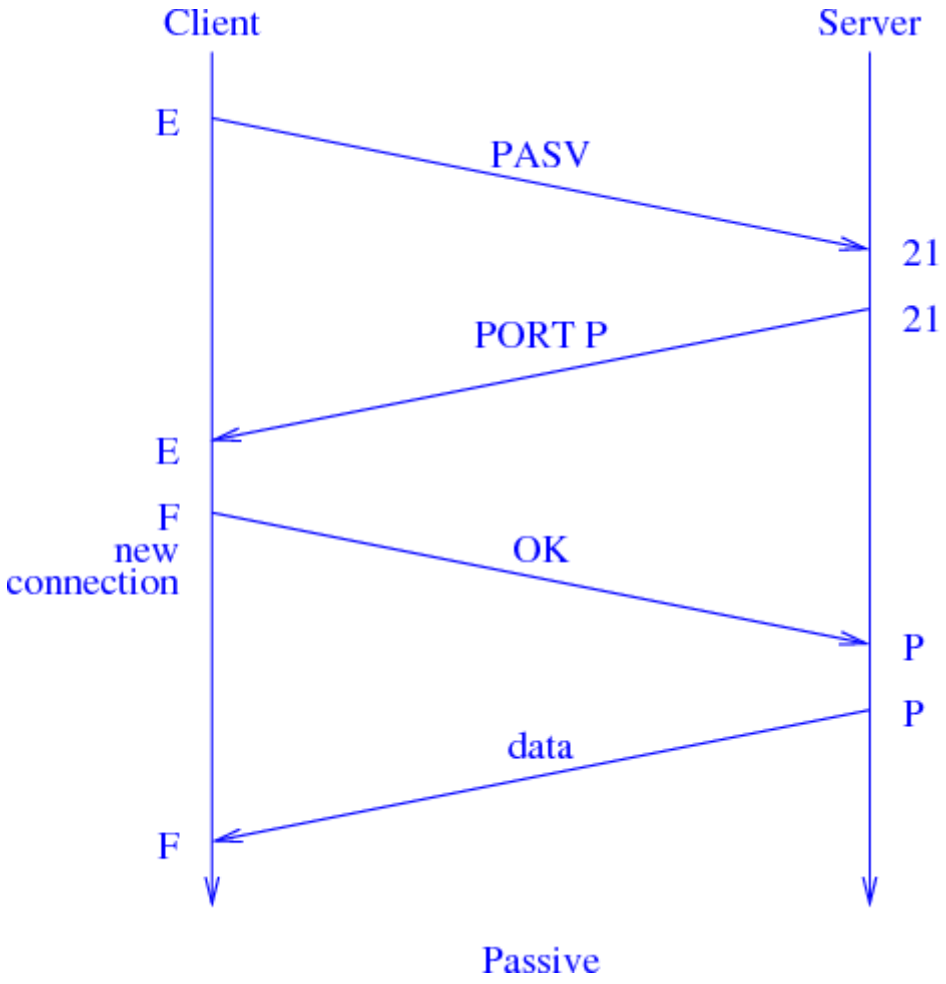
0. The client connects from ephemeral port *E* to the server port 21, as before

Applications



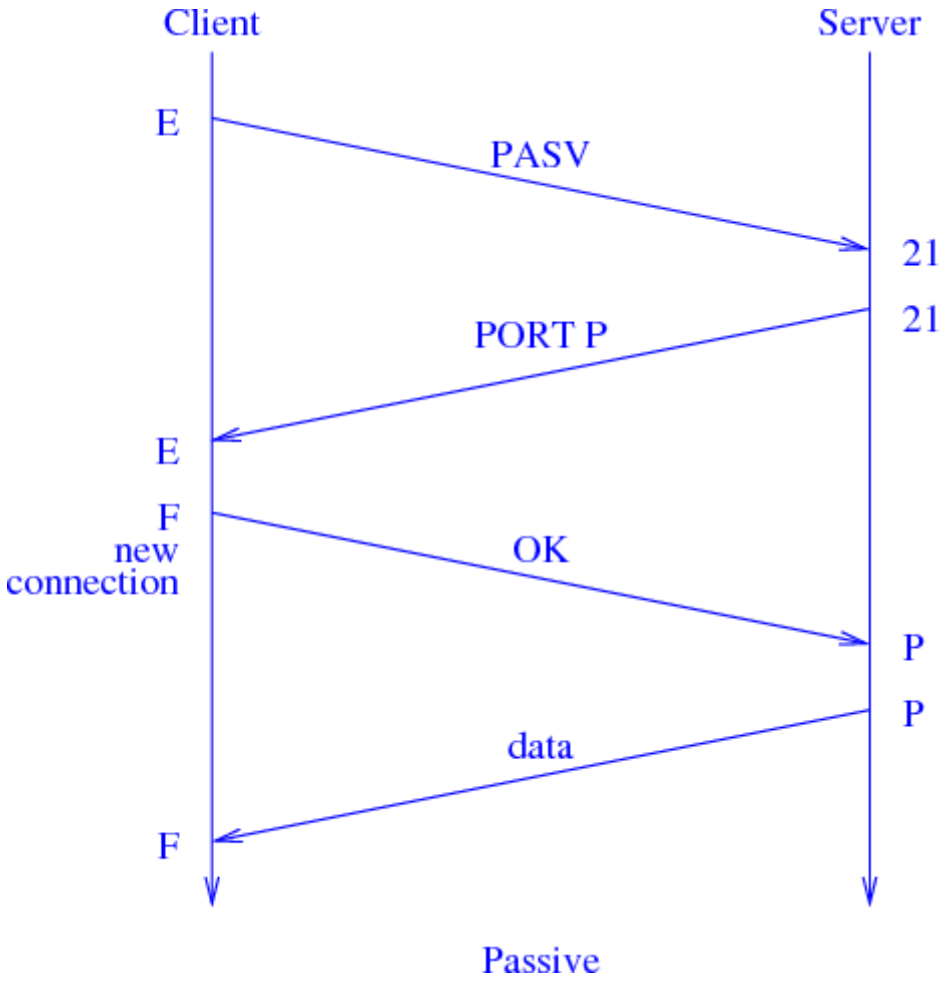
2. The client issues a PASV command to indicate it wants a passive connection

Applications



3. The server replies with a PORT *P* command and listens on port *P*

Applications



4. The client creates a new connection from an ephemeral port to port *P*. Data is then sent on this connection

Applications

FTP

- The difference between active and passive is that the client creates the data connection (from within the firewall) rather than the server (outside the firewall, trying to get in)
- Most FTP programs support the old active mode, just in case the remote server does not support passive mode

Applications

FTP

- The use of FTP has declined massively and these days most data travels over HTTP (Web)
- On the other hand, FTP is still widely implemented (even in Web browsers) as is still often used as it is very efficient at large data transfers

Applications

FTP

- A version of FTP is used in the *GRID*, which needs to move around the huge datasets that big science produces

Applications

SMTP

- The *Simple Mail Transfer Protocol* (SMTP) is the protocol behind email
- Email is one of the Internet's great successes
- Also one of its biggest problems: spam

Applications

SMTP

- SMTP was developed early in the life of the Internet and uses human-readable messages, such as HELO, MAIL FROM, and so on. This helped in debugging

Applications

```
220 yourhost.bath.ac.uk ESMTP Postfix
HELO myhost
250 yourhost.bath.ac.uk
MAIL FROM: me@myhost
250 Ok
RCPT TO: you@yourhost
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Date: Fri, 20 May 2005 09:37:30 +0100
From: me@myhost
To: you@yourhost
Subject: sending email

This is an example email.
.
250 Ok: queued as 1BC31201B82
```

An SMTP exchange

- Lines with numbers are from the server, the others are from the client

Applications

```
220 yourhost.bath.ac.uk ESMTP Postfix
HELO myhost
250 yourhost.bath.ac.uk
MAIL FROM: me@myhost
250 Ok
RCPT TO: you@yourhost
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Date: Fri, 20 May 2005 09:37:30 +0100
From: me@myhost
To: you@yourhost
Subject: sending email

This is an example email.
.
250 Ok: queued as 1BC31201B82
```

- The numbers are return codes, indicating success, failure or other states
- 220: service ready
- 250: success
- 354: start mail input

Applications

220 yourhost.bath.ac.uk ESMTP Postfix
HELO myhost
250 yourhost.bath.ac.uk
MAIL FROM: me@myhost
250 Ok
RCPT TO: you@yourhost
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Date: Fri, 20 May 2005 09:37:30 +0100
From: me@myhost
To: you@yourhost
Subject: sending email

This is an example email.
.
250 Ok: queued as 1BC31201B82

- Client starts with a HELO. More modern versions of SMTP use EHLO: this indicates the client supports later extensions to SMTP

Applications

```
220 yourhost.bath.ac.uk ESMTP Postfix
HELO myhost
250 yourhost.bath.ac.uk
MAIL FROM: me@myhost
250 Ok
RCPT TO: you@yourhost
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Date: Fri, 20 May 2005 09:37:30 +0100
From: me@myhost
To: you@yourhost
Subject: sending email

This is an example email.
.
250 Ok: queued as 1BC31201B82
```

- HELO gives the name of the client, which can be checked against where the email is actually coming from

Applications

```
220 yourhost.bath.ac.uk ESMTP Postfix
HELO myhost
250 yourhost.bath.ac.uk
MAIL FROM: me@myhost
250 Ok
RCPT TO: you@yourhost
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Date: Fri, 20 May 2005 09:37:30 +0100
From: me@myhost
To: you@yourhost
Subject: sending email

This is an example email.
.
250 Ok: queued as 1BC31201B82
```

- MAIL FROM: the sender of the email
- RCPT TO: the intended recipient(s) of the email
- DATA: the start of the message itself
- The text up to this point is called the *envelope*

Applications

```
220 yourhost.bath.ac.uk ESMTP Postfix
HELO myhost
250 yourhost.bath.ac.uk
MAIL FROM: me@myhost
250 Ok
RCPT TO: you@yourhost
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Date: Fri, 20 May 2005 09:37:30 +0100
From: me@myhost
To: you@yourhost
Subject: sending email

This is an example email.
.
250 Ok: queued as 1BC31201B82
```

- The email contains its own conventional header (layering)
- The addresses here need not be the same as in the envelope: the email is delivered to the addresses in the envelope
- This can be an issue

Applications

```
220 yourhost.bath.ac.uk ESMTP Postfix
HELO myhost
250 yourhost.bath.ac.uk
MAIL FROM: me@myhost
250 Ok
RCPT TO: you@yourhost
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Date: Fri, 20 May 2005 09:37:30 +0100
From: me@myhost
To: you@yourhost
Subject: sending email

This is an example email.
.
250 Ok: queued as 1BC31201B82
```

- The end of the email is indicated by a lone full-stop
- Full-stops at the start of lines in the message are stuffed by an extra full-stop

Applications

```
220 yourhost.bath.ac.uk ESMTP Postfix
HELO myhost
250 yourhost.bath.ac.uk
MAIL FROM: me@myhost
250 Ok
RCPT TO: you@yourhost
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Date: Fri, 20 May 2005 09:37:30 +0100
From: me@myhost
To: you@yourhost
Subject: sending email

This is an example email.
.
250 Ok: queued as 1BC31201B82
```

- To be safe, only ASCII encoded messages should be sent
- More interesting data can be encapsulated by MIME

Applications

SMTP

- Email has been a huge success
- But (at the time of writing) perhaps 80% of the world's email is *bulk unsolicited commercial email*, also known as *spam*
- SMTP was developed in a safe academic environment and no thought to authentication was given

Applications

SMTP

- When you receive an email, you have no idea who sent it: the FROM field is useless as it can be anything the sender wants it to be
- Spammers use this openness to their advantage

Applications

SMTP

- As sending email is so cheap they only need a tiny fraction of a percent response to make it profitable: no matter that it abuses and annoys 99.9% of the public
- But spam is not the only problem: MIME encoding allows arbitrary data to be sent

Applications

SMTP

- In particular, you can send executable code in email
- This allows worms and viruses (collectively called *malware*) to propagate
- Many tools and techniques have arisen to combat the problem of spam and malware

Applications

SMTP

- Virus filters: emails are checked for malicious code before they are delivered. These filters need continual updating with the patterns they require to recognise the latest viruses

Applications

Spam

- Spam filters: emails are checked to see if they are likely to be spam. Spam messages share many distinguishing features as they all have the same purpose: get as many emails to as many people as fast as possible

Applications

Spam

- This makes them amenable to various analyses: the envelope, the sender, the layout of the content, the nature of the content, and so on
- Spam filters are quite good these days, but they need continual updating as spammers change their tactics

Applications

Spam

- Some countries have tried passing laws against spam, to spectacularly little effect
- Law have just made spammers go underground and start using even more dubious techniques

Applications

Spam

- There are secure versions of SMTP, namely *secure SMTP* (SMTPS or SSMTP)
- This is SMTP over an encrypted layer SSL/TLS (see later)
- This makes email secure against being read while in transit, but it does nothing for the spam problem

Applications

Spam

- Proposed defences against spam are many and varied
  - blacklisting: keep a list of addresses associated with spammers and refuse to accept email from them. This can be made efficient by using DNS and *Realtime Blackhole Lists* (RBLs) to lookup up IP addresses

Applications

Spam

- whitelisting: keep a list of IP addresses associated with people you trust and only accept email from them. This works reasonably well for individuals, but not for companies who want emails from new customers

Applications

Spam

- greylisting: keep a whitelist, but if an email arrives from a new address return a temporary delivery failure message and put the address on the whitelist (with an expiry date). The normal action of a sender is to wait a little and resend the email, which would then be accepted. Spammers rarely use RFC-compliant mailers and are in such a rush it is unlikely they will retry

Applications

Spam

- *Sender Policy Framework* (SPF, aka *Sender Permitted From*) can help against forging the “From” field. It again uses DNS to look up to see if the From address is permitted to send from the actual IP address the email came from
- A similar idea is *Domain Keys*, that uses DNS to store public encryption keys that can be used to check signatures on emails

Applications

Spam

- Analysis of the message content
  - “Click here”
  - Few words but a large picture
  - Use of HTML
  - Text colour the same as the background colour

and so on for a large number of features the would not be normal for an email written by a human

Applications

Spam

- We could try to fix SMTP itself. Suggestions include:
  - require computational challenges: to reduce the rate a spammer can send messages the receiver could require the sender to perform some time-consuming computational challenge
  - charge a small amount, say 1/10p, for each email as this would ruin the economics of mass emailing

Applications

Spam

- We could try to fix SMTP itself. Suggestions include:
  - replace the whole of SMTP by something else. There have been no serious contenders and the logistics of introducing a new system would be immense