Imperial College London

Computer Networks and Distributed Systems

Part 6 – Application Layer

Course 527 – Spring Term 2015-2016

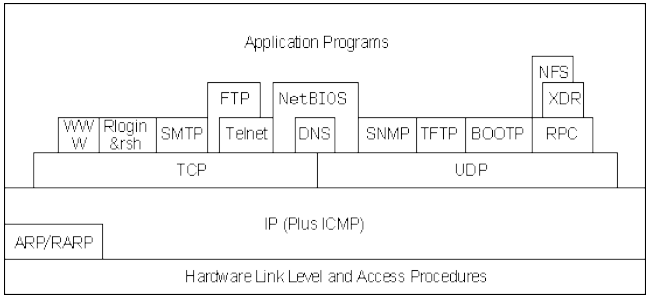
Emil C Lupu & Daniele Sgandurra

e.c.lupu@imperial.ac.uk, d.sgandurra@imperial.ac.uk

Part 6 – Contents

Application Layer

- Application level protocols
  - Domain name resolution (DNS)
  - Email (SMTP, POP, IMAP)
  - World Wide Web (HTTP)



Client

Initiates connection

Often user-invoked app running on local machine
- e.g. web browser, email clients

Uses service/resource from server

Resource use usually temporary

Server

Waits for connections
- Handles multiple clients

Special purpose app providing service/resource
- e.g. web server, chat server

Provides controlled access to resources/services

Often started at boot time
- e.g. "daemon" in UNIX; "service" under WIN

Peer-to-Peer

Not all communication client/server based

Peer-to-peer model
- Hosts operating together rather than one sided
- Offering of resources/service to one another
  - e.g. chat, file trading, Internet telephony

Often implemented as client/server on same host

Centralised server may coordinate

Application Layer Protocols

Clients, servers, peers need communication protocols

Web:	**HTTP**
Email protocols:	**SMTP, MIME, POP, IMAP**
File transfer:	**FTP**
Name resolution:	**DNS**

- Not user application but service

Names vs IP Addresses

IP addrs good for identification of network interfaces, but:

- Addr **`146.169.14.6`** not very memorable
- Machines, services may move between networks

Use meaningful, user-friendly names in addition to IP addrs

- e.g. **`columbia.doc.ic.ac.uk`** instead of **`146.169.7.41`**

Names assigned independently of IP addresses

- Names fixed when IP addrs change for technical reasons

Alias names can identify services

- Independent of machine providing service
- e.g. **`www.doc.ic.ac.uk`** for **`linnet.doc.ic.ac.uk`**

Mapping Names to IP Addrs

Need mapping between names and numbers

Local file
- Large and hard to maintain
- Was used in early ARPANET

Centralised server
- Clients query database to perform name resolution
- Bottleneck, single point of failure, admin hard

Distributed look-up system
- Domain Name System (DNS)

Domains Name System (DNS)

Internet is inter-network of autonomous networks

- Must avoid conflicts between names
- Must support independent administration of names

DNS names form hierarchies

- e.g. **`columbia.doc.ic.ac.uk`**

Requires uniqueness of complete name only

- Like postal address
- e.g. **`fred.foo.com`** different to **`fred.bar.com`**

Names and Domains: Externally Managed

Top-level structure conveys meaning

- `.uk` = United Kingdom
  - Country ID from global naming standard
- `.ac` = Academic network
  - Standard sub-domain within UK
- `.ic` = Imperial College
  - Name assigned by UK academic net administration
  - Owned by Imperial College

Within each domain naming managed independently

- e.g. `co.uk` is independent of `co.fr`

Top Level Domains (TLDs)

Domain	Specification	Example
`com`	Commercial	`ibm.com`
`net`	Network providers	`internic.net`, `demon.net`
`edu`	Educational institution	`mit.edu`
`gov`	Government organisation	`whitehouse.gov`
`mil`	US Military organisation	`navy.mil`
`org`	Other organisation	`linux.org`, `redcross.org`
<country code>	Domain administered by country (ISO 3166)	`fr`, `uk`, `zw`
2nd level country domains	Sub-domains to TLD administered by organisation for that country	`ac.uk`, `co.uk`

Name Assignment

ICANN authorises registrars for `.uk`, `.com`, `.org`

- Non-profit corp. with various Internet responsibilities

In UK, **Nominet** assigns names

- 2nd level names (co, ltd etc.) used by Nominet only
- TLD & 2nd level names not allowed as 3rd level names
  - `nhs.co.uk` not allowed as `nhs.uk` is 2nd level domain
  - `net.org.uk` not allowed as `.net` is TLD
- One character 3rd level names reserved
- `ac.uk`, `gov.uk`, `nhs.uk`, `police.uk`, `mod.uk` not controlled by Nominet

UK 2nd Level Domains

`.co.uk`	Commercial enterprises
`.me.uk`	Personal domains
`.org.uk`	Non-commercial organisations
`.plc.uk` `.ltd.uk`	Registered company names only
`.net.uk`	Internet Service Providers
`.sch.uk`	Schools
`.ac.uk`	Academic Establishments
`.gov.uk`	Government Bodies
`.nhs.uk`	NHS Organisations
`.police.uk`	UK Police Forces
`.mod.uk`	Ministry of Defence

Names and Domains: Internally Managed

Local domains follow organisational structure

- **`.doc`** = Dept. of Computing
  - Assigned by College admin
- **`.columbia`** = Machine in DoC
  - Assigned by Dept admin

Type of name may reflect type of machine, naming conventions.

- e.g. in DoC birds are servers, colours are printers

Names may reflect services machine provides

- e.g. **`www`**, **`mail`**

DNS Name Resolution

Host knows local DNS server to ask for names

- Local database of names maintained manually

Authoritative results

- Returned from name server managing that name

Local DNS servers do not move often

- May be hard-coded or given by DHCP
- DNS server known by its IP address
- Domain will typically have 2-3 DNS servers e.g., ns0, ns1 in doc.ic.ac.uk or ic.ac.uk.

Recursive DNS Lookups

If name cannot be resolved locally:

- Make new request to server up the hierarchy
- DNS server now becomes DNS client

When the top level is reached:

- Go down to required domain

Repeated until someone knows name,

- or decided that name not resolvable

DNS Servers are not required to support recursive queries.

Example: Recursive DNS Lookup

vm-shell1.doc.ic.uk → **www.csail.mit.edu**

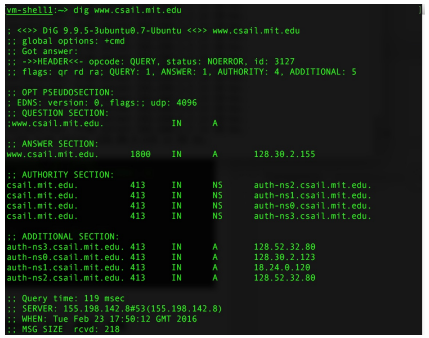
Vm-shell1.doc.ic.ac.uk -> **dns0.doc.ic.ac.uk**

dns0.doc.ic.ac.uk queries **ns.ic.ac.uk**

ns.ic.ac.uk queries **edu-server.net**

edu-server.net queries **mit.edu**

mit.edu queries **csail.mit.edu**



TLD servers replicated around the world

- Make queries faster that reach top of tree
- Load shared and gives redundancy

Iterative (non-recursive) DNS Lookups

```
vm-shell1:~ > dig +trace www.csail.mit.edu;
<<>> DiG 9.8.3-P1 <<>> +trace www.csail.mit.edu
;; global options: +cmd
.                       339347  IN      NS      j.root-servers.net.
…
;; Received 228 bytes from 192.168.1.1#53(192.168.1.1) in 25 ms

edu.                    172800  IN      NS      a.edu-servers.net.
…
;; Received 270 bytes from 199.7.83.42#53(199.7.83.42) in 112 ms

mit.edu.                172800  IN      NS      usw2.akam.net.
…
;; Received 414 bytes from 192.31.80.30#53(192.31.80.30) in 104 ms
csail.mit.edu.          1800    IN      NS      auth-ns2.csail.mit.edu.
...
;; Received 191 bytes from 193.108.91.37#53(193.108.91.37) in 11 ms
www.csail.mit.edu.      1800    IN      A       128.30.2.155
;; Received 51 bytes from 18.24.0.120#53(18.24.0.120) in 85 ms
```


DNS Caching

Servers cache responses as names often needed again

- Exploits locality of reference
  - If I look at a web site, my colleagues may look too
  - If **`ns.ic.ac.uk`** resolves **`www.doc.ic.ac.uk`** once, likely to get more requests in future

Cached answers non-authoritative

- May be wrong, out-of-date

DNS entries give TTL based on volatility

- Caches expire records after their TTL has expired
- Stable names can safely be cached for much longer

Types of DNS Records

A records: name ➔ IP address mapping

MX records: mail server address for domain

- Used when delivering email to, e.g., **`@doc.ic.ac.uk`**

NS records: name server address for domain

CNAME records: alias names

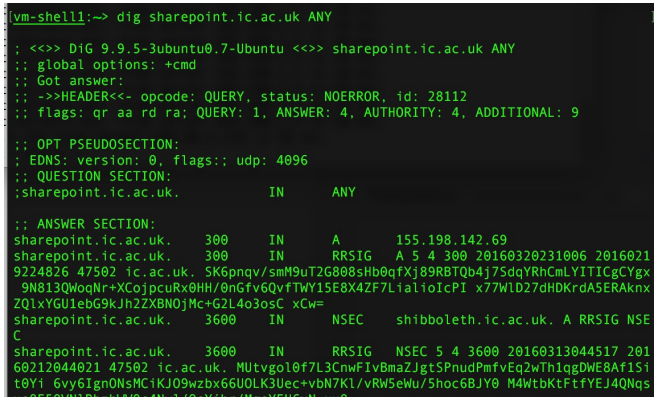
- e.g. **`www.doc.ic.ac.uk`** alias for **`linnet.doc.ic.ac.uk`**

PTR records: name ➔ IP addr (for reverse lookups)

- e.g. **`146.169.7.41`** ➔ **`columbia.doc.ic.ac.uk`**

RRSIG records: DNSSEC Signature

Example: DNS Host Lookup



```
vm-shell1:~> dig sharepoint.ic.ac.uk ANY

; <<>> DiG 9.9.5-3ubuntu0.7-Ubuntu <<>> sharepoint.ic.ac.uk ANY
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28112
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 4, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;sharepoint.ic.ac.uk.           IN      ANY

;; ANSWER SECTION:
sharepoint.ic.ac.uk.    300     IN      A       155.198.142.69
sharepoint.ic.ac.uk.    300     IN      RRSIG   A 5 4 300 20160320231006 2016021
9224826 47502 ic.ac.uk. SK6pnqv/smM9uT2G808sHb0qfXj89RBTQb4j7SdqYRhCmLYITICgCYgx
 9N813QWoqNr+XCojpcuRx0HH/0nGfv6QvfTWY15E8X4ZF7LialioIcPI x77WlD27dHDKrdA5ERAknx
ZQlxYGU1ebG9kJh2ZXBNOjMc+G2L4o3osC xCw=
sharepoint.ic.ac.uk.    3600    IN      NSEC    shibboleth.ic.ac.uk. A RRSIG NSE
C
sharepoint.ic.ac.uk.    3600    IN      RRSIG   NSEC 5 4 3600 20160313044517 201
60212044021 47502 ic.ac.uk. MUtvgol0f7L3CnwFIvBmaZJgtSPnudPmfvEq2wTh1qgDWE8Af1Si
t0Yi 6vy6IgnONsMCiKJO9wzbx66UOLK3Uec+vbN7Kl/vRW5eWu/5hoc6BJY0 M4WtbKtFtfYEJ4QNqs
```

Answer of form:

`<domain name> <TTL> <class> <type> <value>`

TTL: Time to expiry if cached (in sec)

class: IN = Internet (almost always)

type: Type of record

Example: DNS Domain Lookup

```
vm-shell1:~> dig google.com ANY

; <<>> DiG 9.9.5-3ubuntu0.7-Ubuntu <<>> google.com ANY
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46860
;; flags: qr rd ra; QUERY: 1, ANSWER: 14, AUTHORITY: 4, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;google.com.                    IN      ANY

;; ANSWER SECTION:
google.com.             86400   IN      TYPE257 \# 19 0005697373756573796D616E74
65632E636F6D
google.com.             60      IN      SOA     ns2.google.com. dns-admin.google
.com. 115345670 900 900 1800 60
google.com.             3600    IN      TXT     "v=spf1 include:_spf.google.com
~all"
google.com.             600     IN      MX      50 alt4.aspmx.l.google.com.
google.com.             600     IN      MX      20 alt1.aspmx.l.google.com.
google.com.             600     IN      MX      30 alt2.aspmx.l.google.com.
google.com.             600     IN      MX      40 alt3.aspmx.l.google.com.
google.com.             600     IN      MX      10 aspmx.l.google.com.
google.com.             198     IN      AAAA    2a00:1450:4009:80f::200e
google.com.             150     IN      A       216.58.213.110
google.com.             87838   IN      NS      ns2.google.com.
google.com.             87838   IN      NS      ns1.google.com.
google.com.             87838   IN      NS      ns4.google.com.
google.com.             87838   IN      NS      ns3.google.com.

;; AUTHORITY SECTION:
google.com.             87838   IN      NS      ns1.google.com.
google.com.             87838   IN      NS      ns4.google.com.
google.com.             87838   IN      NS      ns2.google.com.
google.com.             87838   IN      NS      ns3.google.com.

;; ADDITIONAL SECTION:
ns2.google.com.         88517   IN      A       216.239.34.10
ns3.google.com.         88517   IN      A       216.239.36.10
ns1.google.com.         88517   IN      A       216.239.32.10
ns4.google.com.         88517   IN      A       216.239.38.10

;; Query time: 12 msec
;; SERVER: 155.198.142.8#53(155.198.142.8)
;; WHEN: Tue Feb 23 17:54:38 GMT 2016
;; MSG SIZE  rcvd: 508
```

Electronic Mail

Movement of structured text msg between systems

Email message fields:

To:, Cc:, Sender:, Reply-to:	Email addresses
From:	Who sent the message
Received:	List of mail servers that processed mail
Subject:	Summary of message
Message-Id:	Unique id for message

- Detect duplicate messages, responses

Email address: **`<user name>@<mail domain name>`**

- e.g. **`ecl1@doc.ic.ac.uk`**
- Mail server for domain found through DNS MX records

Email Subsystems

Mail User Agents (**MUA**)

- e.g. Thunderbird, Outlook, mutt, …
- Email client for composition, display, filing, …

Message Transfer Agents (**MTA**)

- e.g. sendmail, exim, Exchange, …
- Mail server that handles message transfer

Email often sent from MUA to **relay** (or **gateway**) MTA

- Usually in same organisation as sender
- Delivers to MTA in recipient's domain
  - Possibly via other relays
- Holds (spools) mail if destination unreachable
  - Supposed never to lose mail

Simple Mail Transfer Protocol (SMTP)

Protocol used by MTAs (and MUAs)

- Delivers mail from one system to another

Two parties communicate using TCP and port 25

- Sender (client) & Receiver (server)

Three basic steps

1. Start session
2. Exchange data
3. Complete session

SMTP: Protocol Steps

Open TCP connection to port 25

Server responds with `220` message

Client sends `HELO` command identifying its domain

Client sends 1+ mail messages

- First giving identity information (`MAIL-FROM`, `RCPT-TO`)
- Then sending `DATA` part
- Server responds to each message with 3-digit code

`QUIT` message tells server to close TCP connection

Example: SMTP Session

Often direct connection hosts no longer allowed

```
220 finch.doc.ic.ac.uk ESMTP Exim 4.63 Wed, 03 Jan 2006
  16:18:37 +0000
helo doc.ic.ac.uk
250 finch.doc.ic.ac.uk Hello columbia.doc.ic.ac.uk
  [146.169.7.41]
mail from: prp@doc.ic.ac.uk
250 <prp@doc.ic.ac.uk> is syntactically correct
rcpt to: prp@doc.ic.ac.uk
250 <prp@doc.ic.ac.uk> is syntactically correct
data
354 Enter message, ending with "." on a line by itself
subject: test
1 2 3
.
250 OK id=16l78a-0001cK-00
quit
221 finch.doc.ic.ac.uk closing connection
```


SMTP: Basic Sender Commands

Command	Argument	Meaning
HELLO/HELO	Sender's domain	I'm in this domain (must be first command)
MAIL FROM:	User id	Identify sender
RCPT TO: (1 or more)	User id	Identify recipient(s) (first must follow MAIL (or other starting command))
DATA		Email data follows (must follow RCPT)
<CRLF>.<CRLF>		End of email text (line with just full-stop)
RESET/RSET		Abort current mail
VERIFY/VRFY	User id	Is user ID valid?
QUIT		Sender signing off (last command)

SMTP: Basic Receiver Replies

Reply Code	Meaning
220	Service ready
221	OK, I too am closing connection
250	OK, requested mail action completed
354	Start sending me email text, end with <CRLF>.<CRLF>
500	Syntax error – command unrecognised
501	Syntax error in parameters or arguments
503	Bad sequence of commands
550	Requested action not taken: mailbox unavailable
552	Requested mail action aborted: exceeded storage allocation

Example: Delivered Message

```
Return-path: <prp@doc.ic.ac.uk>
Envelope-to: prp@doc.ic.ac.uk
Delivery-date: Wed, 02 Jan 2006 11:42:00 +0000
Received: from [146.169.7.46] (helo=finch.doc.ic.ac.uk
 ident=exim) by falcon.doc.ic.ac.uk with esmtp (Exim
 3.13 #8) id 16l78q-0004M7-00 for prp@doc.ic.ac.uk; Wed,
 02 Jan 2006 11:42:00 +0000
Received: from columbia.doc.ic.ac.uk ([146.169.7.41]
 helo=doc.ic.ac.uk) by finch.doc.ic.ac.uk with smtp
 (Exim 3.16 #7) id 16l78a-0001cK-00 for
 prp@doc.ic.ac.uk; Wed, 02 Jan 2006 11:41:59 +0000
Subject: test
Message-ID: <E16l78a-0001cK-00@finch.doc.ic.ac.uk>
From: prp@doc.ic.ac.uk
BCC:
Date: Wed, 02 Jan 2006 11:41:59 +0000

1 2 3
```


Post Office Protocol (POP)

SMTP designed for permanently available hosts

- e.g. Internet mail servers
- SMTP delivers mail to mailbox at ISP
- Not usually used to deliver mail to user's desktop

Post Office Protocol allows user access to mailbox

- POP client (MUA) connects to mailbox (POP) server
- Connection to port 110 when "get mail" is requested
- POP authenticates user (with password)
- Mailbox downloaded for processing

Internet Message Access Protocol (IMAP)

POP only supports downloads of Inbox

IMAP handles multiple folders

- Possible to leave emails on mail server
- Useful when using multiple MUAs or machines

Other features

- Partial downloads of emails/attachments
- Offline mode when unconnected
- Server-side searches

But IMAP complex and adds server load

File Transfer Protocol (FTP)

Used to exchange files between hosts

- Optimised for efficient transfer of large files
- Separate control and data connections
  - Uses port 21 to initiate control connection
- Data can be
  - fetched by client (GET)
  - sent to server (PUT)

Typical FTP client exposes messages exchanged

- Some command capabilities but not designed to be terminal program

```
vm-shell1:~> sftp ecl1@shell2.doc.ic.ac.uk
WARNING : Unauthorized access to this system is forbidden and will be
prosecuted by law. By accessing this system, you agree that your actions
may be monitored if unauthorized usage is suspected.

   ***DO NOT RUN RESOURCE-INTENSIVE PROCESSES ON THE SHELL SERVERS***
Connected to shell2.doc.ic.ac.uk.
sftp> !ls
Desktop      IMAP_backup  Microsoft    Templates
desktop.ini  KDesktop     Pictures     testftp.txt
Documents    lib          Public       WINDOWS
GNUstep      Macromedia   public_html  WindowsDocuments
sftp> get testftp.txt
Fetching /homes/ecl1/testftp.txt to testftp.txt
/homes/ecl1/testftp.txt                       0%    0     0.0KB/s   --:-- ETA
sftp> put testftp.txt
Uploading testftp.txt to /homes/ecl1/testftp.txt
testftp.txt                                 100%    0     0.0KB/s   00:00
sftp> bye
vm-shell1:~>
```


SMTP vs. FTP

Information transfer from client to server

Information transfer in either direction

Single connection for control/data

Separate control and data connections

Message transfer protocol

- Server file system invisible

File transfer protocol

- File system (partly) visible

ASCII/MIME encoded data

- Data typed with MIME

ASCII/binary data

- Data un-typed

Multiple "hops" to perform end-to-end transaction

- Spooling when server unavailable

Direct client/server interaction

- Interactive protocol - fails if server unavailable

World Wide Web (WWW)

Developed by Tim Berners-Lee (CERN in 1989)

Killer application for the Internet

- Supports transfer and display of documents
- Documents include multimedia content and hyperlinks
- Important lesson for HCI…

Client/server based

- Client (browser): Firefox, Explorer, Opera, lynx, …
- Web servers: Apache, IIS, …

Web Standards

World Wide Web Consortium (W3C) manages standards:

HyperText Transfer Protocol (**HTTP**)

- Used by browsers to get resources from servers

HyperText Mark-up Language (**HTML**)

- Encoding of data for web pages
- Supports text with images, formatting and hyperlinks

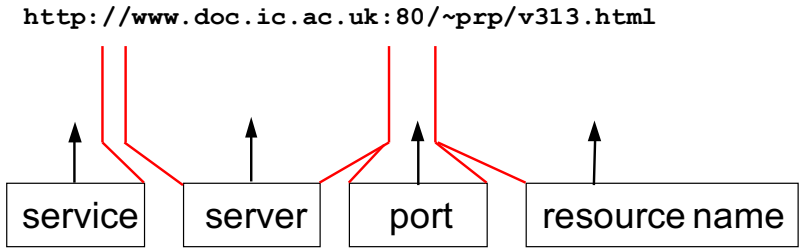
Uniform Resource Locator (**URL**)

- Used to identify links to resources on servers

Uniform Resource Identifier (**URI**)

- More general, newer version of URLs
- URIs can also be location independent (pure names)

Uniform Resource Locators (URLs)



URLs can indicate:

- other protocols, e.g. ftp, gopher, telnet, mail, news, file, …
- other resource types, e.g. image, program, service …

URLs encode location of resource

Hypertext Transfer Protocol (HTTP)

HTTP server usually listens at TCP port 80

Stateless, transaction-oriented protocol

1. Client opens connection to server
2. Request sent from client to server
3. Server responds
4. Connection closed

HTTP Request Message

Request format:

Request line (method, identifier, version)
Header (additional info)
Body (data)

Retrieve web page from server:

```
GET /index.html HTTP/1.0
User-Agent: Mozilla/2.01 (X11; I; IRIX 5.2 IP7)
Accept: image/gif, image/x-bitmap, image/jpeg
/* a blank line */
```


HTTP/1.0 Methods

HTTP 1.0 has **methods** rather than commands

GET: Client requests resource from server

- No permanent action on server is implied
- Most common method

HEAD: Requests only header of web page

- Useful when deciding if changed

POST: Append/send data to named resource

- Used to submit client data from web forms

Others often not implemented:

PUT, DELETE, LINK, UNLINK

HTTP Reply Messages

Reply format:

Status line (version, code, optional message)

Header (additional info)

Body (data, MIME compatible)

Sample response:

```
HTTP/1.0 200 OK
MIME-Version: 1.0
Server: CERN/3.0
Date: Wednesday 10-Apr-96 03:59:47 GMT
Content-type: text/html
Content-length: 2168
Last-Modified: Friday 06-Oct-95 07:16:52 GMT
/* a blank line */
/* HTML text of the Web page */
```


HTTP Reply Codes

200	OK
204	No response
301	Moved (permanently)
400	Bad request (e.g. syntax error)
401	Unauthorised
402	Payment required
404	Not found
503	Service unavailable
505	HTTP version unsupported

Stateless and Non-Persistent

HTTP protocol is stateless

- Server doesn't keep track of client requests
  - Simplifies server design
- But websites would like to keep track of customers

HTTP/1.0 uses non-persistent connections

- New TCP connection opened for every request
  - Page with 10 images will open 11 TCP connections
- Adds server load and setup costs and delays
- TCP congestion control inefficient for short transfers

Maintaining State: Cookies

Websites want to identify users

- IP addrs not practical ids for users due to NAT, sharing

Persistent information through **cookies**

- Allows server to maintain state between HTTP requests
- Sent by web server and stored by browser
  - Name/value pair with expiry time
  - Set-Cookie header in HTTP response
  - Javascript (setCookie/getCookie)

Persistent Connections: HTTP/1.1

Deals with some of HTTP/1.0's performance issues

- **Persistent connections**
  - Client opens TCP connection
  - HTTP requests pipelined through this connection
  - Multiple requests with one TCP open/close overhead
- Other features:
  Better proxy handling; more methods; improved encoding and authentication

Now in widespread use

- Easy transition because backwards compatible

Further Information On The Web

World Wide Web Consortium (W3C)

- `www.w3.org` website
- Much information, tutorials, standards definitions for HTTP, HTML, XML, DOM, CSS, RDF etc.

Practical guides in the O'Reilley books