COMPSCI 314 S1 C

Sockets,
Email (SMTP, POP),
HTTP, DNS

Sockets

- Sockets are software objects that are implemented in some way or another for all languages that need to access a machine's TCP/IP stack
- The applications 'plug into the socket'
- We may distinguish between client sockets and server sockets, although in many cases the same object will handle both socket types

Server socket particulars

- A server application's socket **binds** to a particular port using a particular protocol (TCP or UDP). This is usually done by setting the port and protocol properties of the socket object
- The port number/protocol pair are unique to the server application on the server host – no other application has the same port number *and* uses the same protocol
- When the server application has been loaded and is ready to receive connections (or UDP packets), the server socket is told to **listen**. This is usually accomplished by means of a method call on the socket object
- For each TCP connection (or UDP client port), the server socket either causes the application to fork a child process, or sets up a connection ID that uniquely identifies the connection or the UDP client
- Note that the server socket is not given an IP address – it can read the server host's own address from the configuration and learns the client's IP address from the incoming datagrams

Client socket particulars

- A client application must tell its socket which IP address it wishes to connect to. This is usually done by setting a property of the socket object
- The client application must tell the socket which protocol it wishes to use. This is usually also done by setting a property of the socket object
- A client application's socket usually doesn't specify the port it wishes to use on the client machine
- Rather, it asks the TCP/IP stack for a port from a pool of available ports
- The application then tells the client socket to connect. This is usually done by invoking a method on the client object
- Once the connection is established, the socket generates either an event or calls a callback function in the application in order to notify it of the successful connection

Common mechanisms for both client and server sockets

- Once the connection is established, the socket sets up two buffers: a send buffer, and a receive buffer
- The application writes data into the send buffer by invoking a method on the socket. The socket then tries to dispatch the data to the other end
- If data is received, the socket writes it to the receive buffer. It then notifies the application via an event or a callback function that data has arrived
- The application then invokes a method on the socket object that reads the data out of the receive buffer
- If the application wishes to disconnect, it invokes the associated method on the socket object
- If the other party disconnects, the socket object notifies the application by means of an event or by invoking a callback function

SMTP – Simple Mail Transfer Protocol

- SMTP is used to send e-mail
- Very widely used
- Uses port 25
- An SMTP connection is known as a ‘session’
- All commands in clear text*!*
- Unauthenticated*!*

SMTP at work

- Client connects to *server* port 25 using TCP
- HELO
- *250 mail.compsci314.com Hello a-client.com [130.216.197.66], pleased to meet you*
- MAIL From: jane@bloggs.com
- *250 jane@bloggs.com … Sender OK*
- RCPT To: joe@bloggs.com
- *250 joe@bloggs.com ... Recipient OK*
- DATA
- *354 Enter mail, end with '.' on a line by itself*
- Subject: SMTP is soooo cool!
  Howzit goin, Joe?
  .
- *250 VBB09405 Message accepted for delivery*
- QUIT
- *221 mail.compsci314.com closing connection*

SMTP problems

- SMTP does not authenticate its user!
- SMTP does not ask for username and password before accepting a mail for delivery
- Anyone can connect to an SMTP server and send spam e-mails from fake senders!
- This is a major problem on the Internet now – most of the e-mail traffic is now spam
- SMTP traffic can be eavesdropped on – not encrypted!

Anti-spam strategies used by network operators

- Source filtering: Accept connections only from within own network or with secondary authentication (e.g. NetLogin) – no ‘open relay’
- Log connections with IP address, time, etc.
- Optional: Don’t accept mails with unknown user/domain names for delivery

Aside: Anti-spam strategies you can use

- Don’t put your e-mail address on the web or into a newsgroup
- Don’t run an SMTP server (sendmail) on your computer – if you do, make sure it won’t accept outside connections
- Never reply to spam e-mail
- Don’t ‘unsubscribe’ yourself from spam e-mails. This only confirms that your address is live.

Email: User Interface

- Network Administrator runs an email server, i.e. a Mail Transfer Agent (MTA)
- User runs an email client, i.e. a Mail User Agent (MUA)
- User composes email and sends it to her MTA, MTA sends it on to other MTAs using SMTP
- Incoming mail arrives at user's MTA. She retrieves it from there to her MUA using a protocol such as IMAP
- Web-based email systems now provide a common, easily-accessible MUA

POP – Post Office Protocol

- Client connects to *POP3 server* on port 110
- *+OK POP3 server pop3.foobar.com ready*
- USER joebloggs
  PASS iluvjane
- *+OK 1 message(s)*
- LIST
- *+OK 1 message(s)*
  1 4578
  .
- RETR 1
- *+OK 4578 octets*
  From: jinx@foo.com
  To: joe@bloggs.com
  …
- QUIT
- *+OK Goodbye!*

POP3 shortcomings

- Plain TCP connection is not secure enough in many cases – can use secure POP to deal with this
- No compression of mail data (=long downloads)
- Not very flexible if we wish to receive mail on multiple machines – IMAP addresses this problem

HTTP – HyperText Transfer Protocol

- Used for the communication between web clients (often browsers) and web servers
- Client connects to web server using TCP, generally on port 80. (Other ports are also used, especially for secondary servers on a machine)
- Client sends an HTTP request and receives an HTTP response
  - HTTP 1.0 closes the TCP connection
  - HTTP 1.1 keeps connections open, allowing further objects to be transferred through them

DNS – Domain Name System

- The Domain Name System maps host names (such as www.cs.auckland.ac.nz) to IP addresses (such as 130.216.33.106)
- DNS is a distributed (tree-structured) database system
- A client sends a lookup query to a DNS server (can do this e.g., with nslookup or dig)
- DNS server will try to answer query from its own records. If it can't, it will start either a recursive or a non-recursive query
- Recursive query: DNS server will query an 'upstream' DNS server on behalf of the client and present the result to the client
- Non-recursive query: DNS server will point the client at the 'upstream' server, client queries upstream server itself
- DNS servers may cache (temporarily store) records retrieved from other DNS servers – this reduces lookup traffic
- Cached records are 'non-authoritative,' genuine original records are 'authoritative'

DNS: Nameserver Hierarchy