CSC 524

Computer Networks
Dr. Esam A. Alwagait

29/4/2013

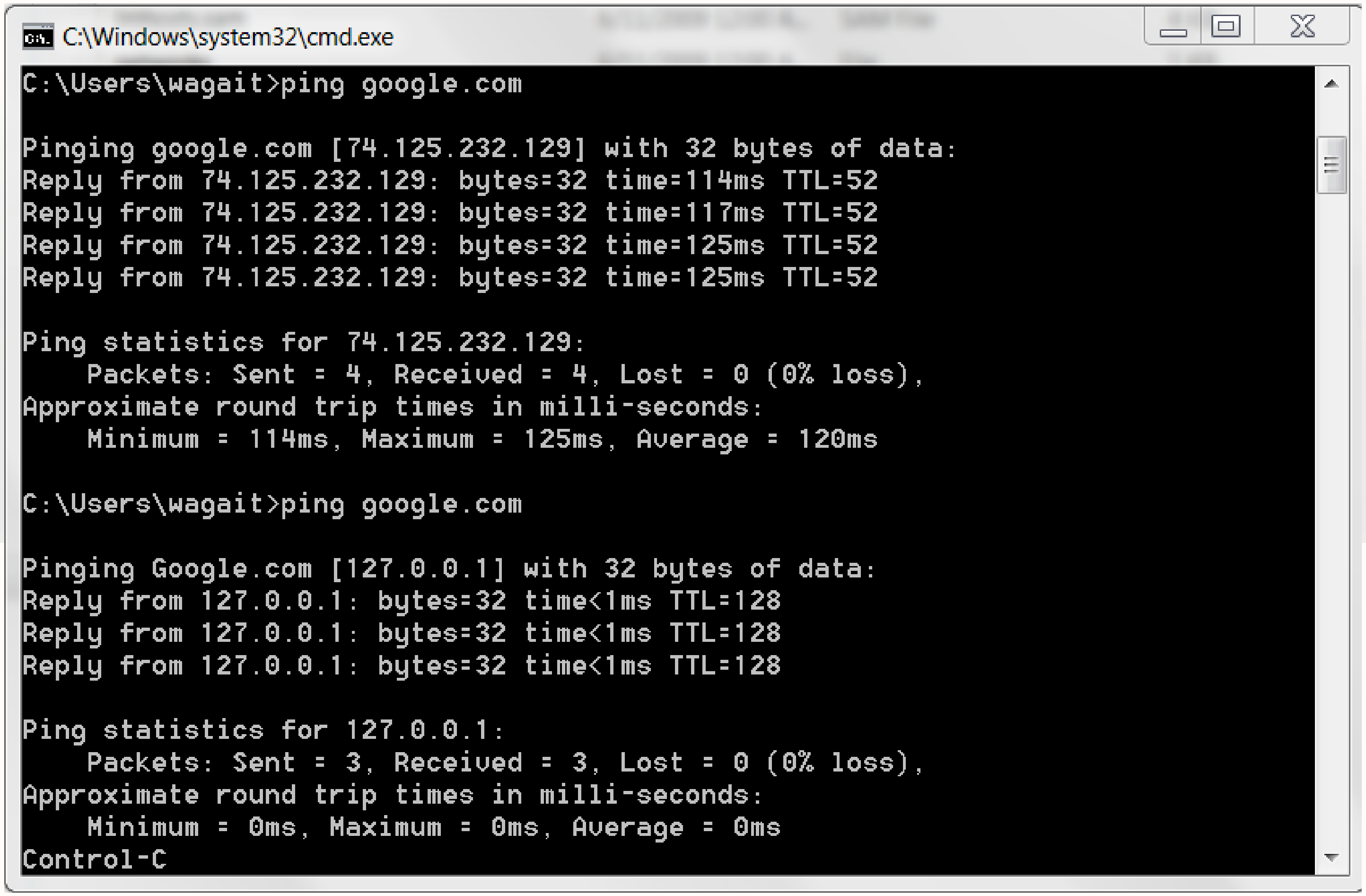
Application Layer

- This is why all the layers below are made !
- Let’s discuss some of the applications
  - DNS
  - MAIL (SMTP, POP, IMAP)

DNS

- Name → IP!
- 13 Root Servers
  - http://en.wikipedia.org/wiki/Root_name_server
- Sequential
  - First ask my computer ..
  - Then ask DNS server
  - Then ask again .. Again
  - ROOT server
- Caching !
  - TTL

DNS


C:\Windows\system32\cmd.exe

```
C:\Users\wagait>ping google.com

Pinging google.com [74.125.232.129] with 32 bytes of data:
Reply from 74.125.232.129: bytes=32 time=114ms TTL=52
Reply from 74.125.232.129: bytes=32 time=117ms TTL=52
Reply from 74.125.232.129: bytes=32 time=125ms TTL=52
Reply from 74.125.232.129: bytes=32 time=125ms TTL=52

Ping statistics for 74.125.232.129:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 114ms, Maximum = 125ms, Average = 120ms

C:\Users\wagait>ping google.com

Pinging Google.com [127.0.0.1] with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
Control-C
```

DNS

- Top-Level Domains (TLDs)
  - Domains
    - When you register a domain, you are asked for a DNS for that domain
  - Sub-Domain
    - Sub-sub domain ☺

Mail

- SMTP (simple Mail Transport Protocol)
  - Sending .. Port 25
  - You could actually send an email through command line!
  - telnet to SMTP server and issue commands
- POP3 (Post Office Protocol)
  - Downloading the mail from servers
- IMAP (Internet Message Access Protocol)

Introduction

- HTTP: Hyper Text Transfer Protocol
  - HTTP 1.0 (RFC 1945), HTTP 1.1 (RFC 2068)
- The transfer protocol for web applications
  - Text documents: HTML, XML, …
  - Multimedia: JPG, GIF, Video, …
- HTTP uses the client/server paradigm
  - HTTP server provide resource
  - HTTP client (usually web browser) get resource
- But not pure client/server communication
  - Proxies, caches, …

Introduction

- HTTP is an application layer protocol
- HTTP assumes reliable communication
  - TCP, default (server) port: 80
- HTTP is stateless
  - Server does not keep history/state of clients
    - If client ask an object 10 times, server will give it back each time
  - High performance & Low complexity
  - Problematic in some applications (sessions)
    - cookies or other solutions

Resources

- Each resource must be identified uniquely
  - URI (Uniform Resource Identifier)
- Common practical URI is URL
  - Uniform Resource Locator
- <protocol(scheme)>://<user>:<pass>@**<host>**:<port>**/<path>?<query>#<frag>**

- http://www.ksu.edu.sa

URL

- Scheme: the application layer protocol
- HTTP: The web protocol
- HTTPS: Secure HTTP
- File: Access to a local file
- …

URL (cont'd)

- Path: the path of the object on the specified host with respect to web server root directory
- E.g. web server root directory: /var/www/
  - http://www.example.com/1.html
    - /var/www/1.html
  - http://www.example.com/1/2/3.jpg
    - /var/www/1/2/3.jpg
- Similar to FS paths
  - Absolute: Path starts from web server root directory
  - Relative: Path starts from current directory

URL (cont’d)

- Query: a mechanism to pass information from client to active pages or forms
  - Fill information in a university registration form
  - Ask Google to search a phrase
- Starts with “?”
- “&” is the border between multiple parameters

- http://www.example.com/submit.php?name=ali&famility=karimi

URL (cont’d)

- Frag: A name for a part of resource
  - A section in a document
- http://www.example.com/paper.html#results
- Handled by browser
  - Browser gets whole resource (doc) from sever
  - In display time, it jumps to the specified part

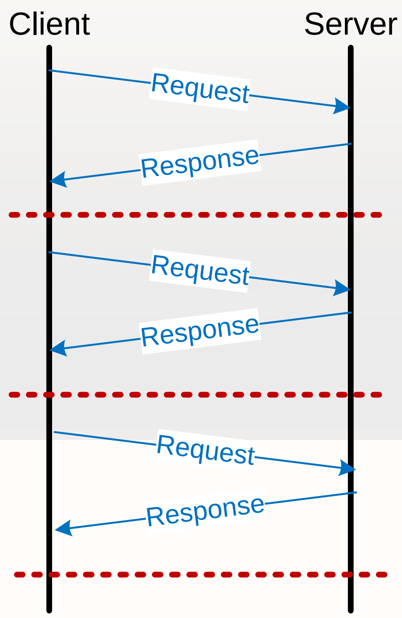
URL (cont’d)

- URL is encoded by client before transmission
- An encoding “%” is used
- E.g.
  - ~ → %7E
  - Space → %20
  - % → %25

HTTP Transaction

- Transactions are performed as HTTP msg.
  - Client → Server: HTTP Request Message
  - Server → Client: HTTP Response Message
- Requests are identified by Methods
  - Method: The action that client asks from server
- Response are identified by Status codes
  - Status: The result of the asked action

HTTP Transaction (cont'd)



HTTP Transaction (cont'd)

- (Typically) each webpage contain multiple resources
  - The main skeleton HTML page
  - Some figures, videos, …
- Displaying a webpage
  - Get the HTML page (first transaction)
  - Try to display the page
    - Other resources are linked to the page
  - Get the resources (subsequent transactions)

HTTP Transaction (cont'd)

- HTTP Transactions & TCP connections
  - Non-persistent & Persistent
- Non-persistent
  - A new TCP connection per object
    - Processing overhead + Connection establish delay
    - Parallel connections speed up browsing
- Persistent
  - Get multiple objects using single TCP connection

HTTP Messages

- Messages (request/response)

Start line: specifies the type of message
Header: depends on message type
An Empty Line
Message body: Data/payload

HTTP Messages (cont'd)

- Request message format

```
Method<sp>URL<sp>version<CRLF>
<Header field>:<value><CRLF>
…
<Header field>:<value><CRLF>
<CRLF>
<Entity body>
```

HTTP Messages (cont'd)

- E.g. HTTP request message

```
GET /index.html HTTP/1.1
Host: www.ksu.edu.sa
User-Agent: Mozilla/6.0
Accept-Language: en-us
Connection: keep-alive
```

HTTP Messages (cont’d)

- Response message format

```
Version<sp>code<sp>Reason<CRLF>
<Header field>:<value><CRLF>
…
<Header field>:<value><CRLF><CRLF>
<Entity body>
```

HTTP Messages (cont'd)

- E.g. HTTP response message

```
HTTP/1.1  200 OK
Date: Sun, 02 Oct 2011 20:30:40
Server: Apache/2.2.2
Last-Modified: Mon, 03 May 2009 10:20:22
Connection: keep-alive
Content-Length: 3000

(data data data ….)
```

HTTP Methods

- Methods are actions that client asks from server to do on the specified resource
- GET (must be implemented by server): Retrieve resource from server
- HEAD (must be implemented by server): Similar to GET but the resource itself is not retrieved, just the HTTP response header
  - Useful for debugging or some other applications

HTTP Methods (cont'd)

- POST: Submit data to be processed by the specified resource
  - Data itself is enveloped in message body
- DELETE: To remove the resource
- PUT: Add message body as the specified resource to server
- TRACE: Server echo back the received message
  - For troubleshooting & debugging

HTTP Methods (cont’d)

- OPTIONS: Request the list of supported methods by server on the resource
- CONNECT: Create HTTP tunnel
  - Client asks server (which is proxy/gateway) to create TCP connection to the specified destination
  - After TCP connection establishment, all data sent on TCP connection between client & server are copied to the established new TCP connection

HTTP Responses

- 2xx: Successful operation
  - 200: OK
  - 201: Created
- 3xx: Resource has been moved, Redirection
  - Location header → the new location of resource
  - 301: Moved Permanently
  - 303: Mainly used to redirect after POST
  - 304: Not modified
  - 307: Moved Temporarily

HTTP Responses (cont’d)

- 4xx: Client error
  - 400: Bad request
  - 401: Unauthorized (Authorization required)
  - 403: Forbidden
  - 404: Not found
  - 405: Not allowed method
- 5xx: Server error
  - 500: Internal server error
  - 501: Not implemented
  - 503: Service unavailable

HTTP Headers

- General headers
  - Appear both on request & response messages
- Request headers
  - Information about request
- Response headers
  - Information about response
- Entity headers
  - Information about body (size, …)
- Extension headers
  - New headers (not standard)

General Headers

- Date: Date & Time the message is created
- Connection: close or keep-alive
  - Close: Non-persistent connection
  - Keep-alive: Persistent connection
- Via: Intermediate nodes between two sides
  - Proxy servers

Request Headers

- Host: The name of the server (required)
- Referer: URL that contains requested URL
- User-Agent: The client program
- UA-OS: The OS of client program
- UA-Disp: Information about display of client
- Accept: The acceptable media types
- Accept-Encoding: Acceptable encodings

Request Headers (cont’d)

- Accept-Language: What language are acceptable
- If-Modified-Since: Request is processed if the objected is modified since the specified time
- If-Unmodified-Since: Request is processed if the objected is not modified since the specified time

Response Headers

- Server: Information about server
- WWW-Authenticate: Used to specify authentication parameters by server
- Proxy-Authenticate: Used to specify authentication parameters by proxy
- Authenticate headers are replied by Authorization header from client side
- Set-Cookie: To send a cookie to client

Entity Headers

- Content-Length: The length of body (in byte)
- Content-Type: The type of entity, similar to MIME types
  - text/html, image/gif
- Allow: The allowed request method can be performed on the entity
  - This is in response of OPTIONS method
- Location: The new location of entity to redirect client

Entity Headers (cont'd)

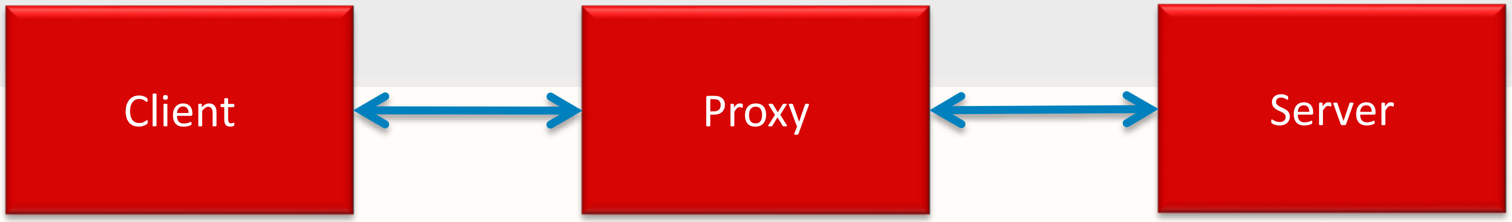
- Content-Range: Range of this entity in the entire resource
- Expire: The date and time at which the entity will expire
- Last-Modified: The date and time of last modification of entity
- Cache-Control: To control entity caching

Hands-on

- Lets see it in action
- http://web-sniffer.net/

Proxy

- Proxies sit between client and server
- Act as server for client
- Act as client for server



Proxy Applications

- Authentication
  - Client side: Authenticate clients before they access web
  - Server side: Authenticate clients before they access the server
- Accounting: Log client activities
- Security: Analyze request before send it to server
  - Integrated in modern firewalls
- Filtering: Limit access to specified contents
- Anonymizer: Anonymous web browsing
- Caching

Proxy in Action

- How to redirect traffic to proxy
- Client configuration
  - Manual configuration
  - Automatic (WPAD protocol) & Scripting
- L4 switches
  - Redirect traffic according to destination port
- DNS mechanisms
  - Return proxy server’s IP address instead of server’s address

Caching

- Caching: save a copy of resource and use it instead of requesting server
- Browser has its own local caches
- Cache server is special proxy for caching
- Benefits
  - Reduce redundant data transfer
  - Reduce network bottleneck
  - Reduce load on server
  - Reduce delay

Caching (cont'd)

- Two possible cases for requested objects
  - Hit: Cache has a fresh copy of the object
  - Miss: Otherwise
- Cache performance index
  - Hit ratio (0~1)
    - Object hit ratio vs. Byte hit ratio
      - Higher Object hit ratio → More responsive browsing
      - Higher Byte hit ratio → More bandwidth saving
    - Depends on Cache size, Similar activities, Objects size, Objects type, Caching control headers, …

Caching (cont'd)

- Cache server must return only fresh objects
  - Freshness check
- Objects life-time specified by server
  - Expire header: Absolute expiration time
  - Cache-Control: max-age: Relative expiration time
- If requested object is not expired
  - Cache server gives it to client
- If requested object is expired
  - Its freshness must be checked

Caching (cont'd)

- Freshness is checked by conditional request
  - If-Modified-Since: current last-modified time

- Server responses
  - 304 Not modified response + new expire time
    - Cached copy is valid until the specified time
  - 200 OK
    - Server provides a new version of the object
    - Cache server updates cached copy

THANK YOU!

Your Logo