CSCI-351
Data communication and Networks

Lecture 13: HTTP

The slide is built with the help of Prof. Alan Mislove, Christo Wilson, and David Choffnes's class

HTTP Basics

- HTTP layered over bidirectional byte stream
- Interaction
  - Client sends **request to server**, followed by **response from server** to client
  - Requests/responses are encoded in text
- Stateless
  - Server maintains no information about past client requests

HTTP Request

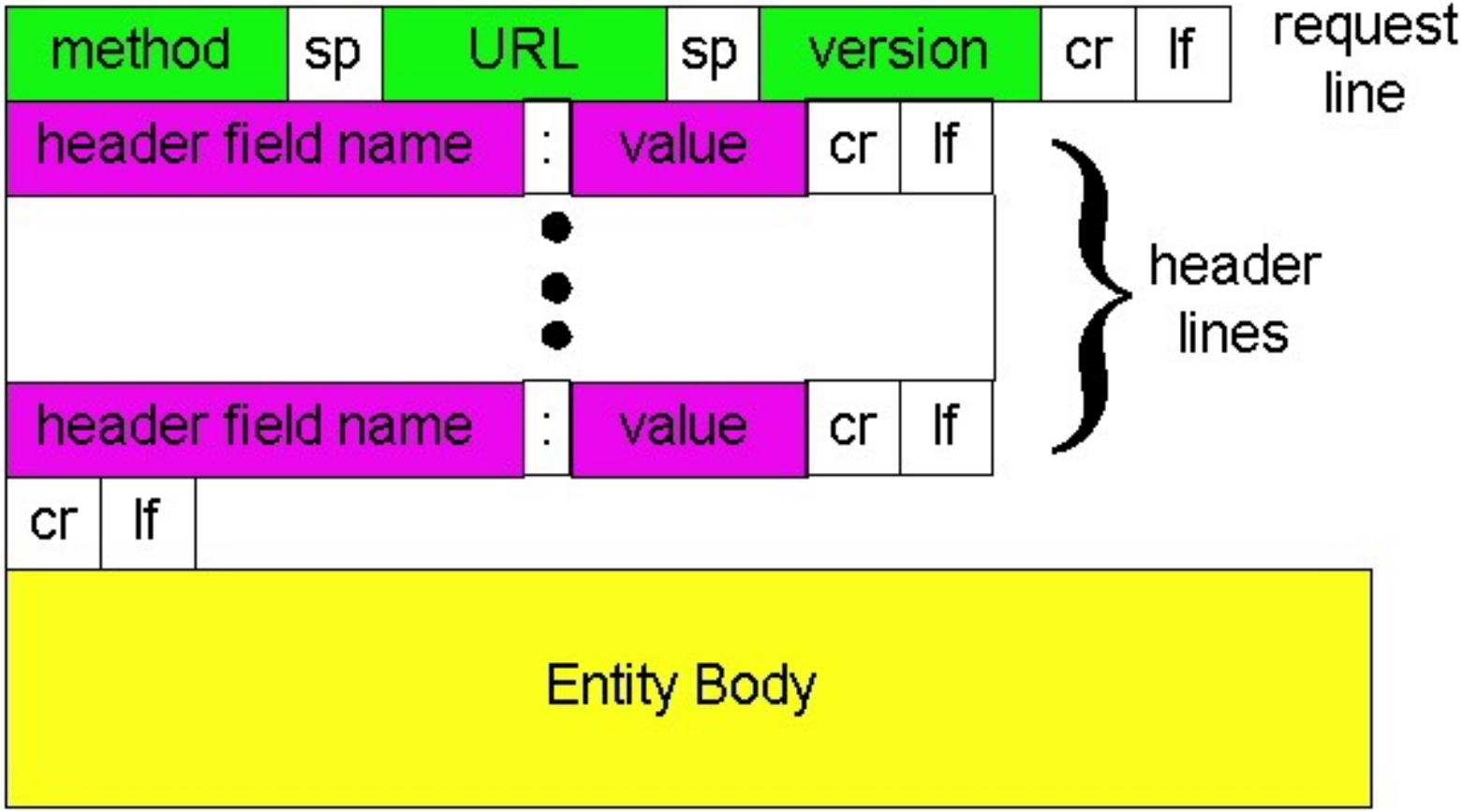
GET /foo/bar.html HTTP/1.1

- Request line
  - Method
    - GET – return URI
    - HEAD – return headers only of GET response
    - POST – send data to the server (forms, etc.)
    - …
  - URL (relative)
    - E.g., /index.html
  - HTTP version

HTTP Request

- Request headers (each ended with CRLF)
  - Acceptable document types/encodings
  - From – user email
  - If-Modified-Since
  - Referrer – what caused this page to be requested
  - User-Agent – client software
  - Cookie - previously stored information
  - Content-Length - Size of data (only on POST)
- Blank-line (CRLF)
- Body

HTTP Request (visual)



HTTP Request Example

GET /blah.html?foo=bar HTTP/1.1

Accept: */*

Accept-Language: en-us

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)

Host: www.intel-iris.net

Connection: Keep-Alive

HTTP Response

- Status-line
  - HTTP version
  - 3 digit response code
    - 1XX – informational
    - 2XX – success
      200 OK
    - 3XX – redirection
      301 Moved Permanently
      303 Moved Temporarily
      304 Not Modified
    - 4XX – client error
      404 Not Found
    - 5XX – server error
      505 HTTP Version Not Supported
  - Reason phrase

HTTP Response (cont.)

- Headers
  - Location – for redirection
  - Server – server software
  - WWW-Authenticate – request for authentication
  - Allow – list of methods supported (get, head, etc)
  - Content-Encoding – E.g x-gzip
  - Content-Length
  - Content-Type
  - Expires
  - Last-Modified
- Blank-line
- Body

HTTP Response Example

HTTP/1.1 200 OK
Date: Tue, 27 Mar 2018 03:49:38 GMT
Server: Apache/1.3.14 (Unix)
Last-Modified: Mon, 29 Jan 2001 17:54:18 GMT
ETag: "7a11f-10ed-3a75ae4a"
Accept-Ranges: bytes
Content-Length: 4333
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
Cache-Control: private

...DATA...

Web pages

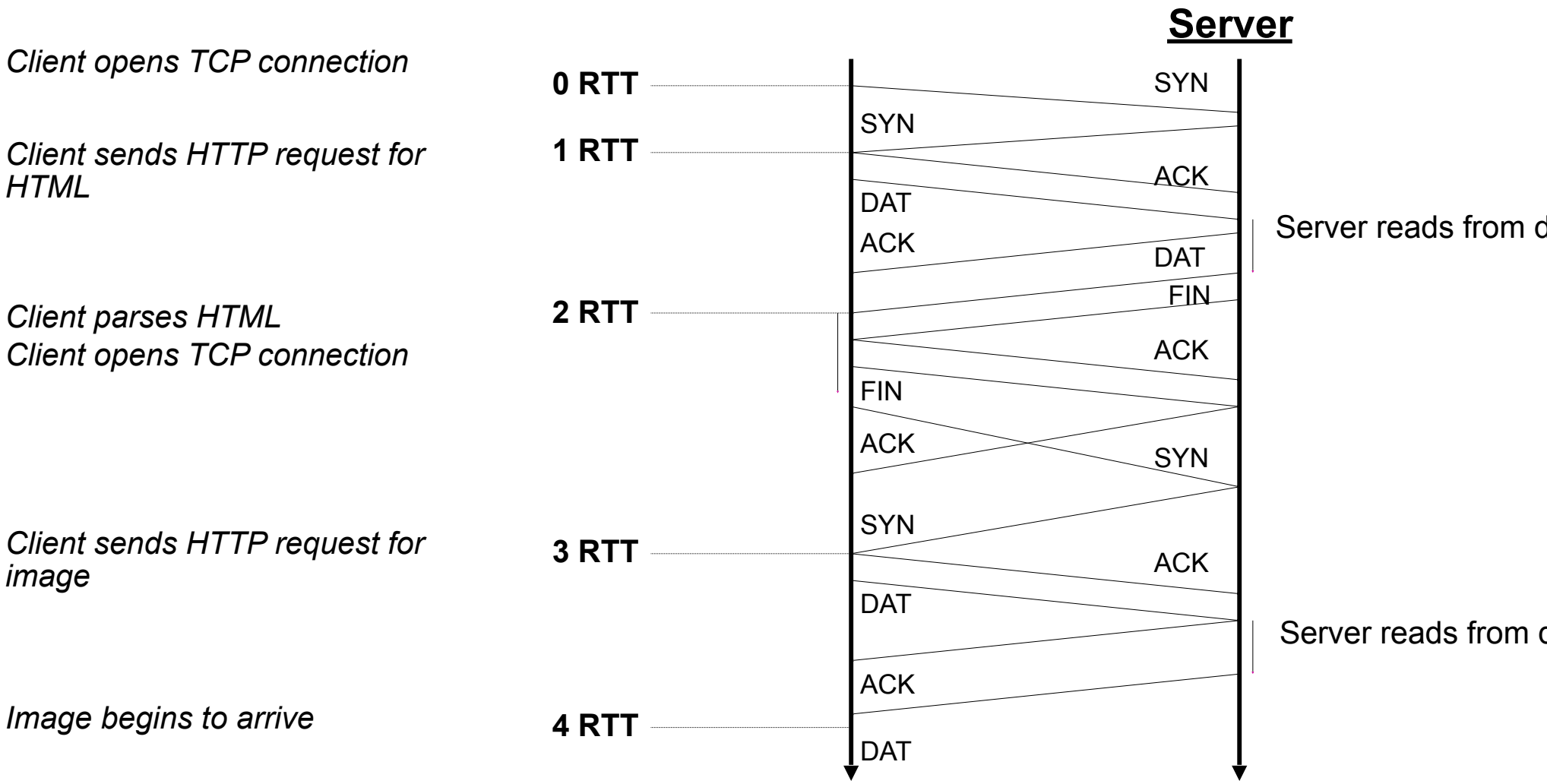
- Multiple (typically small) objects per page
  - E.g., each image, JS, CSS, etc downloaded separately
- Single page can have 100s of HTTP transactions!
- File sizes
  - Heavy-tailed
  - Most transfers/objects very small
- Problem: Browser can't render complete page until all downloaded

HTTP 0.9/1.0

- One request/response per TCP connection
  - Simple to implement

- Disadvantages
  - Multiple connection setups → three-way handshake each time
    - Several extra round trips added to transfer
  - Multiple slow starts

Single Transfer, One Image



More Problems

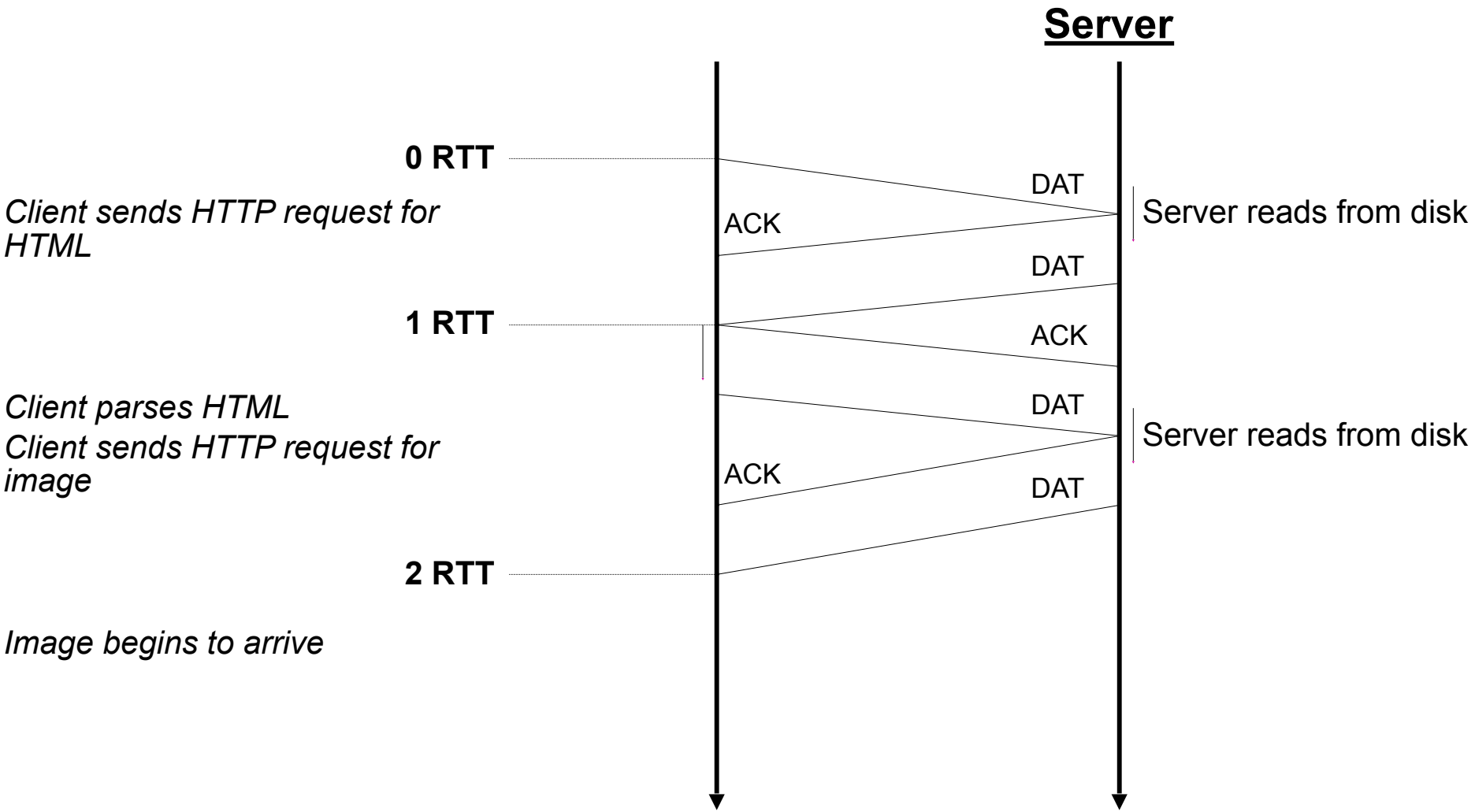
- Short transfers are hard on TCP
  - Stuck in slow start
  - Loss recovery is poor when windows are small
  - SYN/ACK overhead is highest
- Lots of extra connections
  - Increases server state/processing
- Server also forced to keep TIME_WAIT connection state
  - Why must server keep these?
  - Tends to be an order of magnitude greater than # of active connections, why?

Persistent Connections

- Multiplex multiple transfers onto one TCP connection

- Client keeps connection open
  - Can send another request after the first completes
  - Must announce intention via a header
    - Connection: keepalive
  - Server must say how long response is, so client knows when done
    - Content-Length: XXX

Persistent Connection Example



HTTP Caching

- Clients often cache documents
  - Challenge: update of documents
  - If-Modified-Since requests to check
    - HTTP 0.9/1.0 used just date
    - HTTP 1.1 has an opaque “etag” (could be a file signature, etc.) as well

- When/how often should the original be checked for changes?
  - Check every time?
  - Check each session? Day? Etc?
  - Use Expires header
    - If no Expires, often use Last-Modified as estimate

Example Cache Check Request

17

GET / HTTP/1.1

Accept: */*

Accept-Language: en-us

Accept-Encoding: gzip, deflate

If-Modified-Since: Mon, 29 Jan 2018 17:54:18 GMT

If-None-Match: "7a11f-10ed-3a75ae4a"

User-Agent: Mozilla/4.0 (compatible)

Host: www.intel-iris.net

Connection: Keep-Alive

Example Cache Check Response

HTTP/1.1 304 Not Modified

Date: Tue, 27 Mar 2018 03:50:51 GMT

Server: Apache/1.3.14 (Unix)

Connection: Keep-Alive

Keep-Alive: timeout=15, max=100

ETag: "7a11f-10ed-3a75ae4a”

Content in today's Internet

- Most flows are HTTP
  - Web is at least 52% of traffic
  - Median object size is 2.7K, average is 85K (as of 2007)
- HTTP uses TCP, so it will
  - Be ACK clocked
  - For Web, likely never leave slow start
- Is the Internet designed for this common case?
  - Why?