Web and HTTP

Richiami e estensioni dal corso di Reti, prof. Michele Segata,

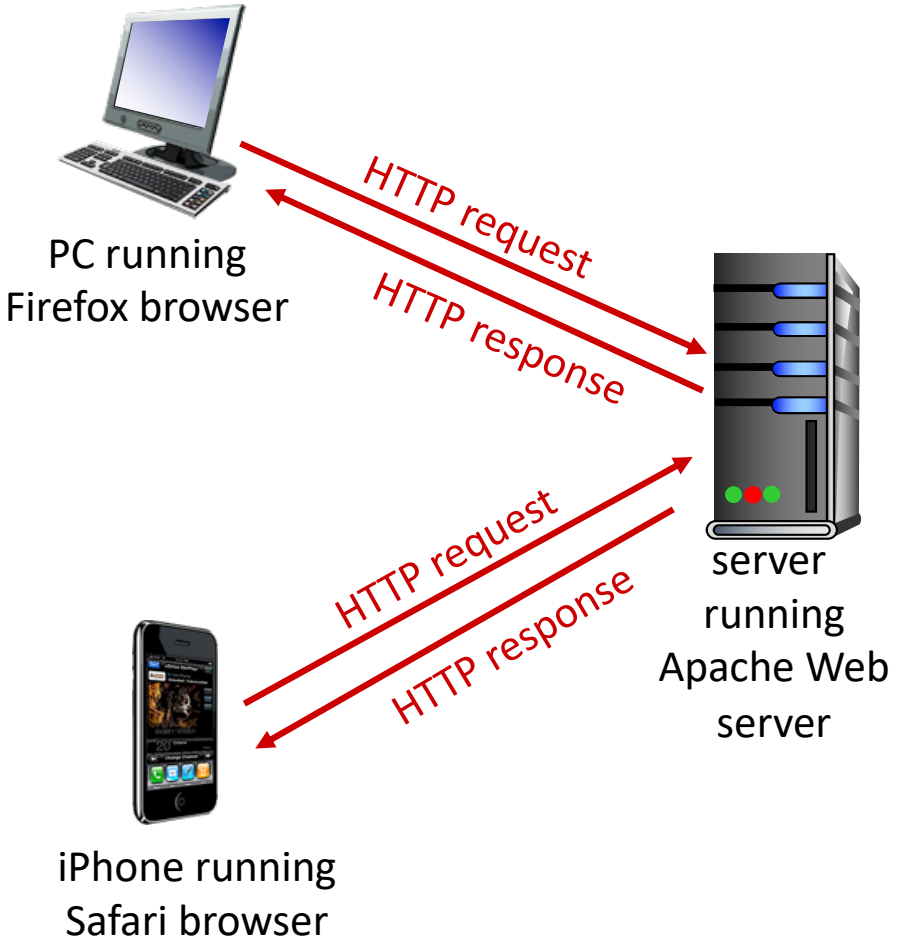
Parzialmente tratto da Computer Networking: A Top-Down Approach, Kurose-Ross

HTTP: basics and connections

HTTP overview

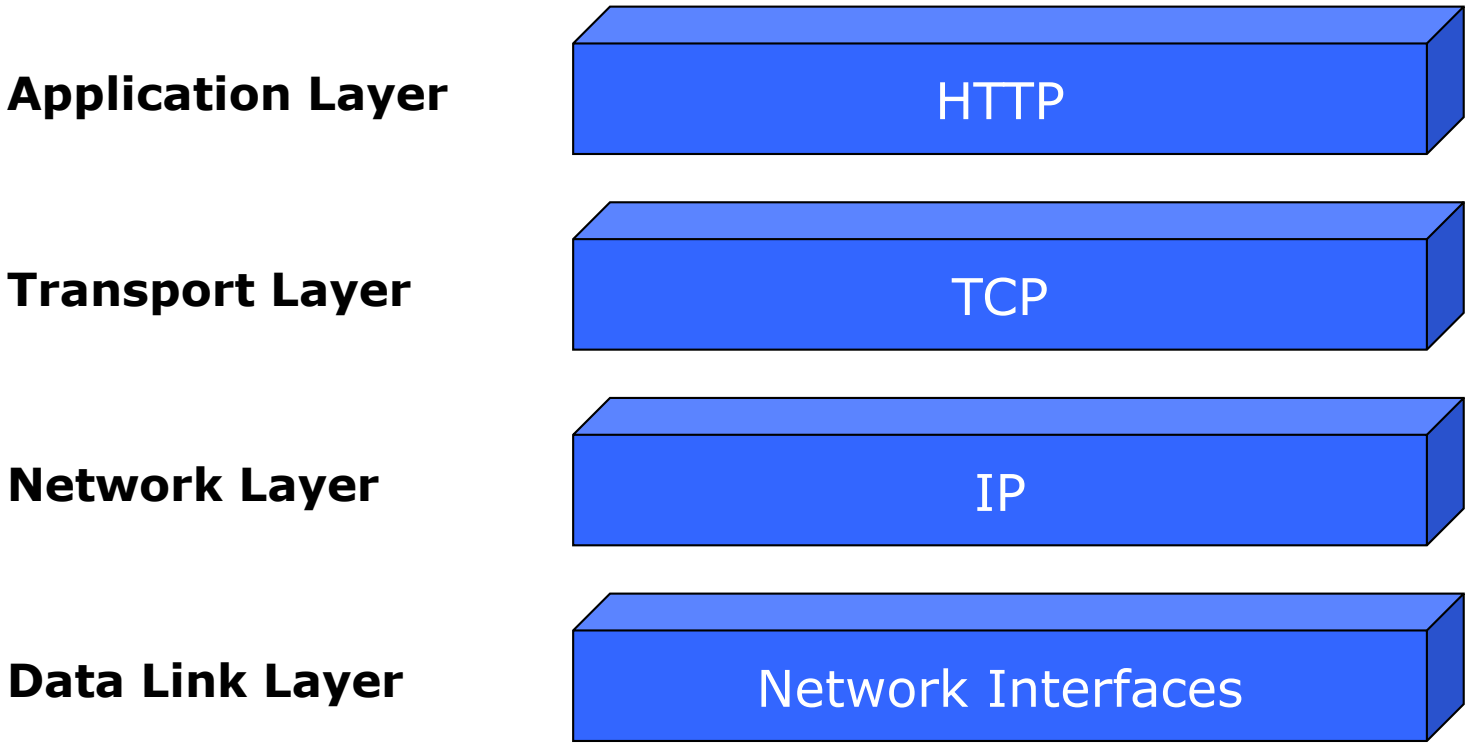
HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - client: browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - server: Web server sends (using HTTP protocol) objects in response to requests



HTTP and TCP/IP

HTTP sits atop the TCP/IP Protocol Stack



HTTP overview (continued)

uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP overview (continued)

HTTP is “stateless”

- server maintains no information about past client requests

protocols that maintain “state” are complex:

- past history (state) must be maintained
- if server/client crashes, their views of “state” may be inconsistent, must be reconciled

HTTP overview (continued)

For comparison: FTP is "stateful"

- server maintains information about past client requests

e.g.:

- you can issue a "cd" command to move into a (remote) directory
- The next commands (e.g. "ls") will be executed with reference to that directory

Q

Where is HTTP defined ?

HTTP

- Three versions have been used, two are in common use and have been specified:

- The Original HTTP as defined in 1991 as HTTP 0.9
- RFC 1945 HTTP 1.0 (1996)
- RFC 2616 HTTP 1.1 (1999)

HTTP 1.1

In June 2014, RFC 2616 was retired and HTTP/1.1 was redefined by

- RFC 7230 - HTTP/1.1: Message Syntax and Routing
- RFC 7231 - HTTP/1.1: Semantics and Content
- RFC 7232 - HTTP/1.1: Conditional Requests
- RFC 7233 - HTTP/1.1: Range Requests
- RFC 7234 - HTTP/1.1: Caching
- RFC 7235 - HTTP/1.1: Authentication

We will stick to HTTP 1.1

HTTP/2

HTTP/2 (originally named HTTP/2.0) is a major revision of the HTTP network protocol used by the World Wide Web. It was derived from the earlier experimental SPDY protocol, originally developed by Google.

The changes do not require any modification to how existing web applications work, but new applications can take advantage of new features for increased speed.

HTTP/2 leaves most of HTTP 1.1's high-level syntax, such as methods, status codes, header fields, and URIs, the same.

What is new is how the data is framed and transported between the client and the server.

HTTP/3

HTTP/3 H3 is the upcoming third major version

HTTP/3 is a draft based on a previous RFC draft, then named "Hypertext Transfer Protocol (HTTP) over QUIC".

QUIC is a transport layer network protocol developed initially by Google where user space congestion control is used over the User Datagram Protocol (UDP).

https://tools.ietf.org/html/draft-ietf-quic-http-23

Q

How does HTTP work ?

Making a simple HTTP request using Telnet

- Open a TCP connection to a host
  - Can borrow telnet protocol to do this, by pointing it at the default HTTP port (80)
  - C:\>telnet www.google.com 80
- Ask for a resource using a minimal request syntax:
  - GET / HTTP/1.1 <CRLF>
  - Host: www.google.ps <CRLF>
  - <CRLF>
- A Host header is required for HTTP 1.1 connections, though not for HTTP 1.0

Making a simple HTTP request using Telnet

ronchet — telnet www.google.com 80 — 80×24

```
[MR-MBP-14955:~ ronchet$ telnet www.google.com 80
Trying 216.58.206.36...
Connected to www.google.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.google.com

HTTP/1.1 200 OK
Date: Sun, 09 Feb 2020 08:54:19 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2020-02-09-08; expires=Tue, 10-Mar-2020 08:54:19 GMT; path=/;
 domain=.google.com; Secure
Set-Cookie: NID=197=BZn2lJOlNIiKeLkhSA3YoTAgo5E0aCh8Sj1ileUG2a8d5Cw_lSQVcZ0j0hH8
3nb18ieVoFlVem5lvbWiB4zH0EHXAoTS_Bc4P2OxmPPjuyFMwyvmPkTX24R2cO9BiRbrbKCirX7C2JrK
fkbpXbJnGWo00zW9Nxp2p0XZOjWLP7o; expires=Mon, 10-Aug-2020 08:54:19 GMT; path=/;
domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
```

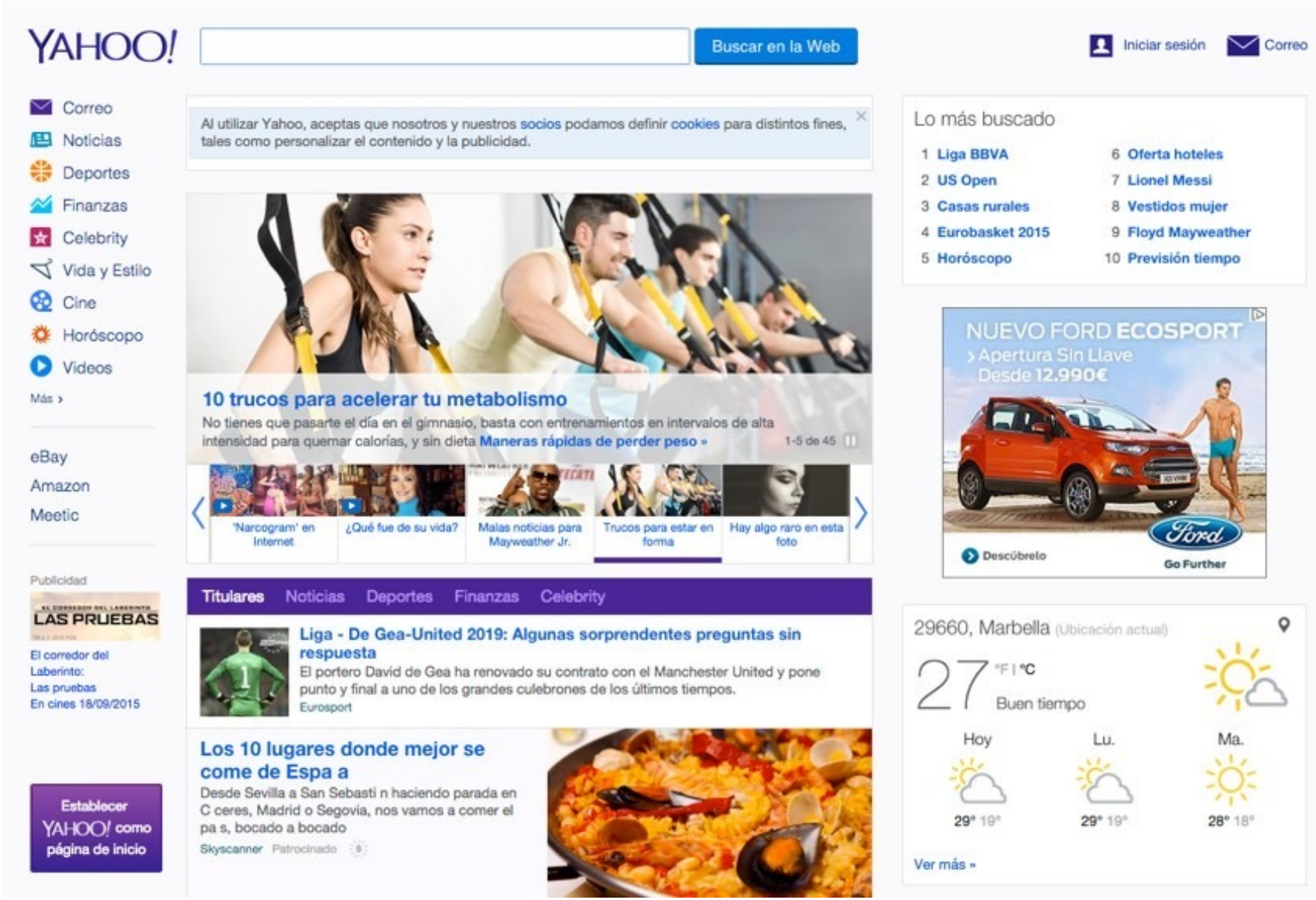
Web and HTTP

- web page consists of objects
- object can be text file, JPEG image, Flash objects, audio file,…
- a web page contains of base HTML-file which includes **several referenced objects**
- each object is addressable by a URL:
  - www.somecompany.com/someDept/pic.gif
  - www. somecompany. com is the host name
  - someDept/pic.gif is the path to the object

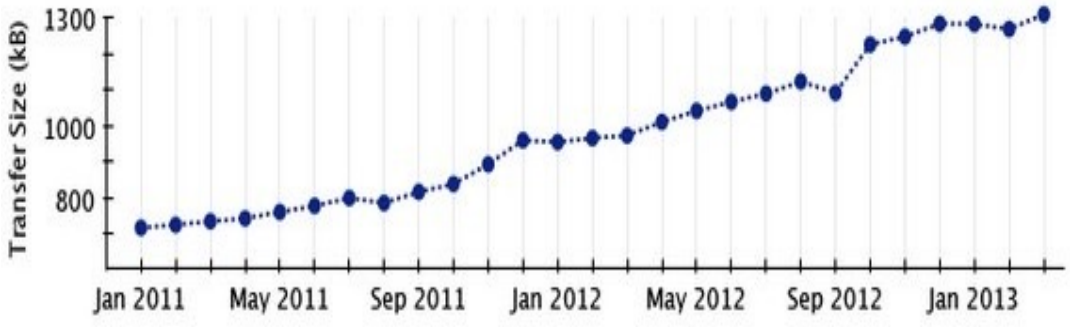
Web in 1996

- **Arts** - - *Humanities*, *Photography*, *Architecture*, ...
- **Business and Economy** [Xtra!] - - *Directory*, *Investments*, *Classifieds*, ...
- **Computers and Internet** [Xtra!] - - *Internet*, *WWW*, *Software*, *Multimedia*, ...
- **Education** - - *Universities*, *K-12*, *Courses*, ...
- **Entertainment** [Xtra!] - - *TV*, *Movies*, *Music*, *Magazines*, ...
- **Government** - - *Politics* [Xtra!], *Agencies*, *Law*, *Military*, ...
- **Health** [Xtra!] - - *Medicine*, *Drugs*, *Diseases*, *Fitness*, ...
- **News** [Xtra!] - - *World* [Xtra!], *Daily*, *Current Events*, ...
- **Recreation and Sports** [Xtra!] - - *Sports*, *Games*, *Travel*, *Autos*, *Outdoors*, ...
- **Reference** - - *Libraries*, *Dictionaries*, *Phone Numbers*, ...
- **Regional** - - *Countries*, *Regions*, *U.S. States*, ...
- **Science** - - *CS*, *Biology*, *Astronomy*, *Engineering*, ...
- **Social Science** - - *Anthropology*, *Sociology*, *Economics*, ...
- **Society and Culture** - - *People*, *Environment*, *Religion*, ...

Web Today

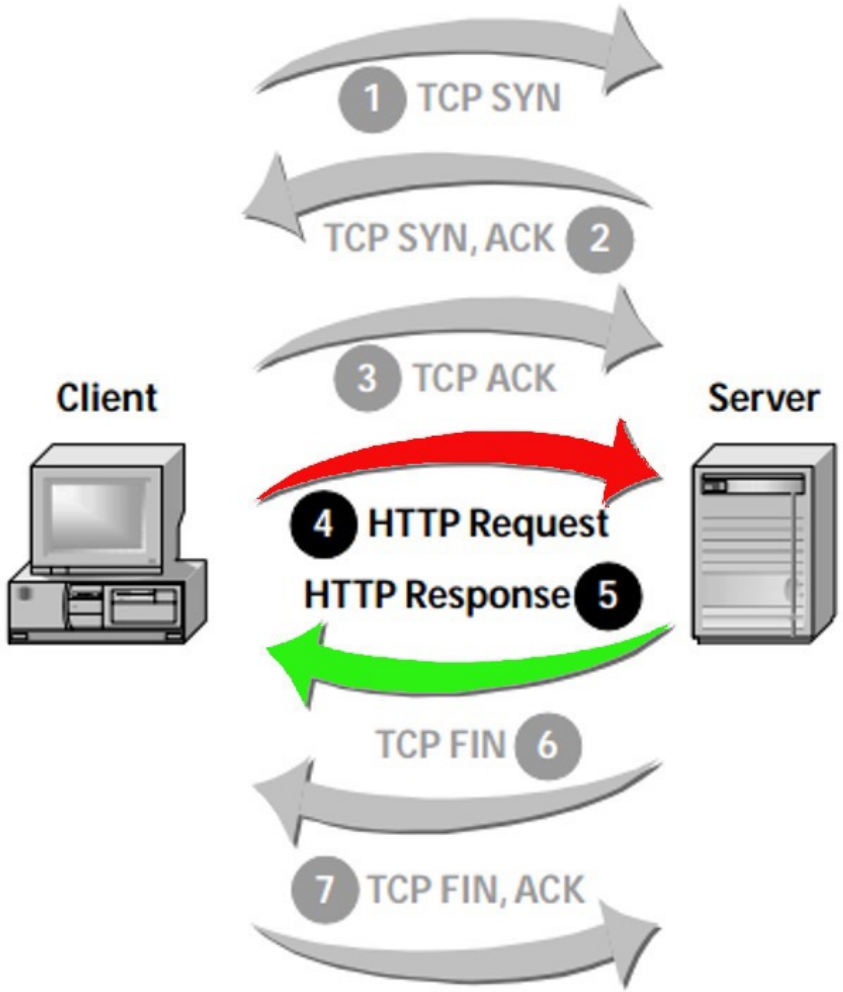

YAHOO!
Buscar en la Web
Iniciar sesión
Correo
Correo
Noticias
Deportes
Finanzas
Celebrity
Vida y Estilo
Cine
Horóscopo
Videos
Más ›
eBay
Amazon
Meetic
Publicidad
LAS PRUEBAS
El corredor del Laberinto: Las pruebas En cines 18/09/2015
Establecer YAHOO! como página de inicio
Al utilizar Yahoo, aceptas que nosotros y nuestros socios podamos definir cookies para distintos fines, tales como personalizar el contenido y la publicidad.
10 trucos para acelerar tu metabolismo
No tienes que pasarte el día en el gimnasio, basta con entrenamientos en intervalos de alta intensidad para quemar calorías, y sin dieta Maneras rápidas de perder peso »
1-5 de 45
'Narcogram' en Internet
¿Qué fue de su vida?
Malas noticias para Mayweather Jr.
Trucos para estar en forma
Hay algo raro en esta foto
Titulares
Noticias
Deportes
Finanzas
Celebrity
Liga - De Gea-United 2019: Algunas sorprendentes preguntas sin respuesta
El portero David de Gea ha renovado su contrato con el Manchester United y pone punto y final a uno de los grandes culebrones de los últimos tiempos.
Eurosport
Los 10 lugares donde mejor se come de Espa a
Desde Sevilla a San Sebasti n haciendo parada en C ceres, Madrid o Segovia, nos vamos a comer el pa s, bocado a bocado
Skyscanner Patrocinado
Lo más buscado
1 Liga BBVA
2 US Open
3 Casas rurales
4 Eurobasket 2015
5 Horóscopo
6 Oferta hoteles
7 Lionel Messi
8 Vestidos mujer
9 Floyd Mayweather
10 Previsión tiempo
NUEVO FORD ECOSPORT
› Apertura Sin Llave
Desde 12.990€
Descúbrelo
Go Further
29660, Marbella (Ubicación actual)
27 °F | °C
Buen tiempo
Hoy
Lu.
Ma.
29° 19°
29° 19°
28° 18°
Ver más »

Our applications are complex, and growing...



Content Type	Desktop		Mobile	
	Avg # of requests	Avg size	Avg # of requests	Avg size
HTML	10	56 KB	6	40 KB
Images	56	856KB	38	498KB
Javascript	15	221KB	10	146KB
CSS	5	36 KB	3	27 KB
Total	86+	1169+KB	57+	711+KB

HTTP requires a TCP connection



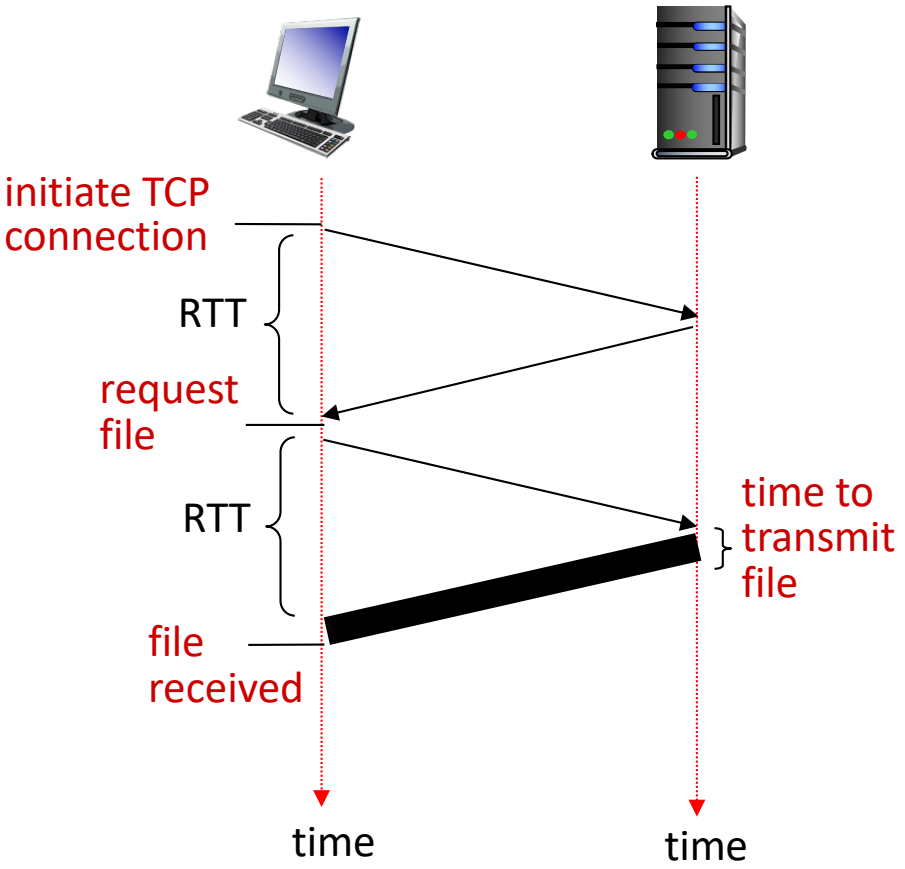
Before systems can exchange HTTP messages, they must establish a TCP connection. Steps 1, 2, and 3 in this example show the connection establishment. Once the TCP connection is available, the client sends the server an HTTP request. The final two steps, 6 and 7, show the closing of the TCP connection.

Non-persistent HTTP: response time

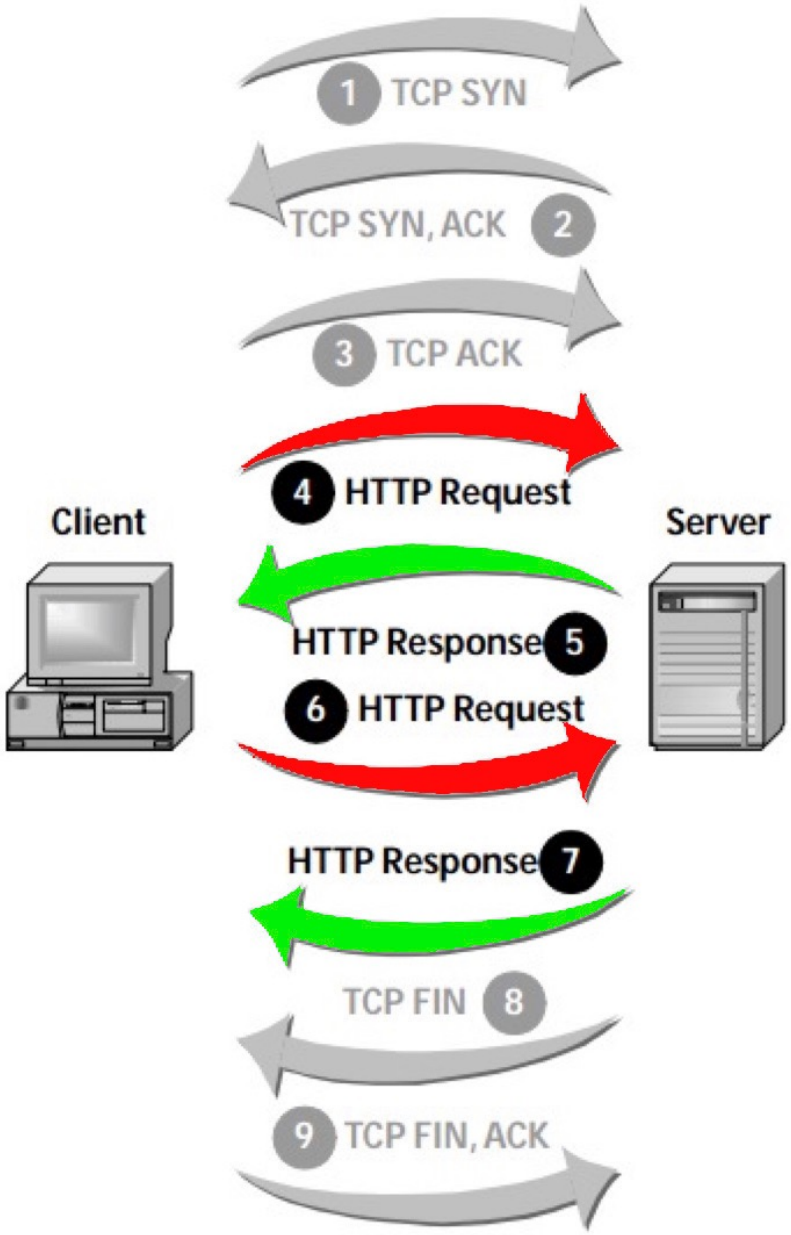
RTT (round trip time): time for a small packet to travel from client to server and back

HTTP response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time
- non-persistent HTTP response time =
  - 2RTT+ file transmission time



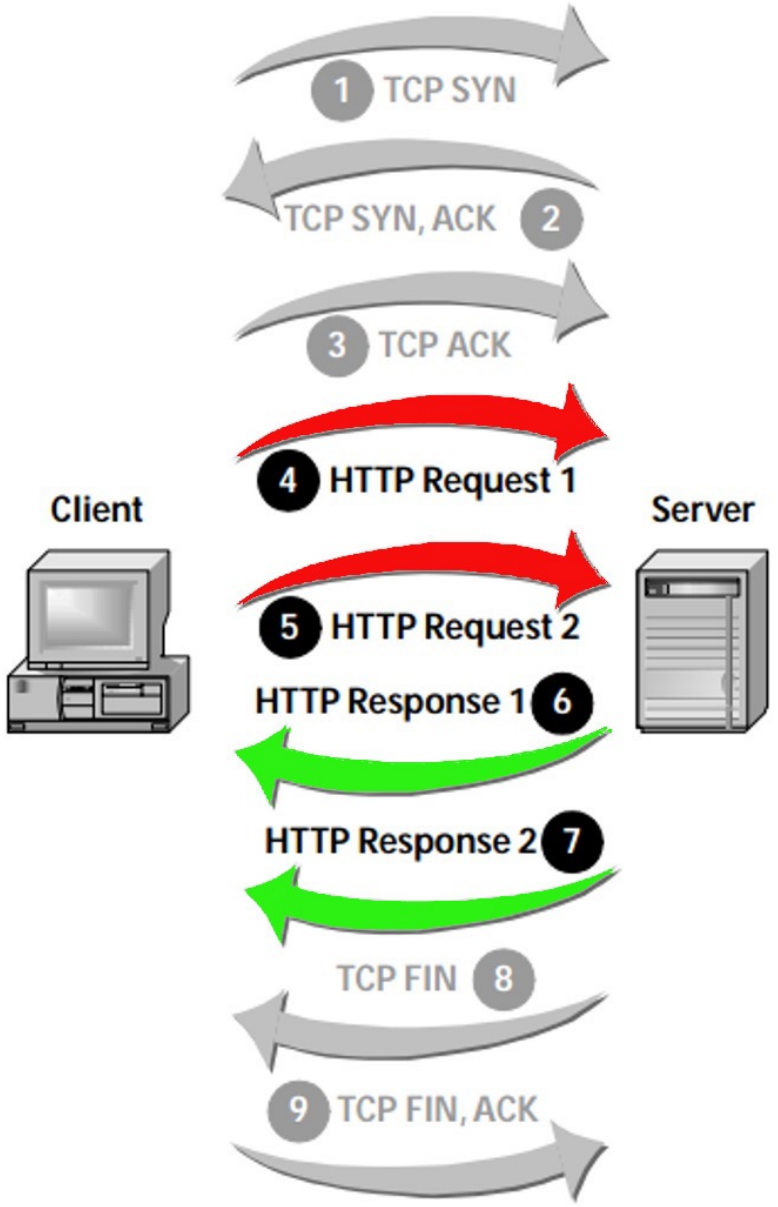
Persistent Connection



With persistent connections, a client can issue many HTTP requests over a single TCP connection. The first request is in step 4, which the server answers in step 5. In step 6 the client continues by sending the server another request on the same TCP connection. The server responds to this request in step 7 and then closes the TCP connection.

Pipelining

Pipelining lets an HTTP client issue new requests without waiting for responses from its previous messages. In the figure, the client sends its first request in step 4. It immediately follows that with a second request in step 5. The client does not wait for the server's response, which arrives in step 6.



HTTP connections

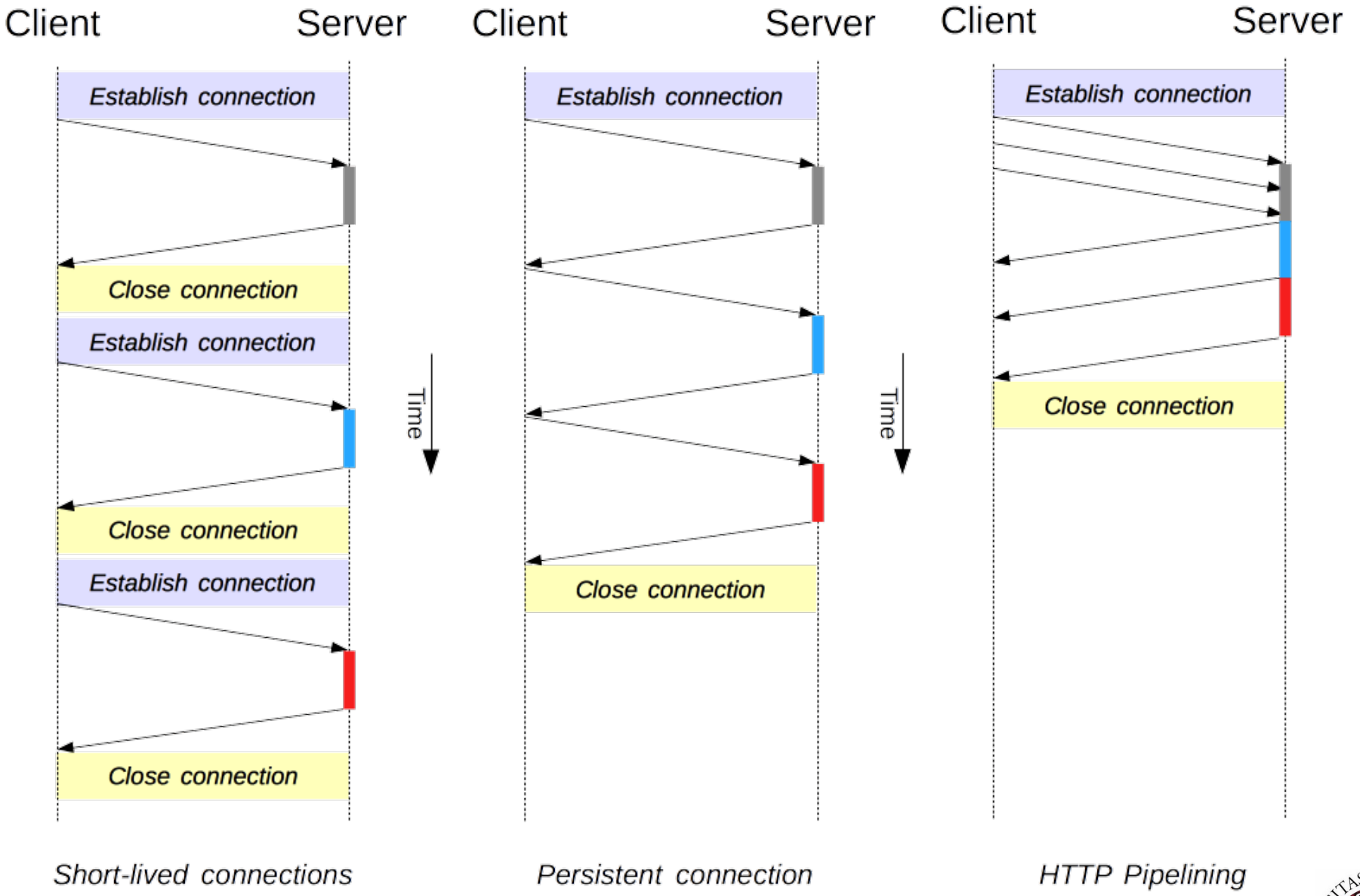
non-persistent HTTP

- at most one object sent over TCP connection
- connection then closed
- downloading multiple objects required multiple connections

persistent HTTP

- multiple objects can be sent over single TCP connection between client and server

HTTP connections



Q

How does the HTTP request composed ?

Making a simple HTTP request using Telnet

ronchet — telnet www.google.com 80 — 80×24

```
[MR-MBP-14955:~ ronchet$ telnet www.google.com 80
Trying 216.58.206.36...
Connected to www.google.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.google.com

HTTP/1.1 200 OK
Date: Sun, 09 Feb 2020 08:54:19 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2020-02-09-08; expires=Tue, 10-Mar-2020 08:54:19 GMT; path=/;
 domain=.google.com; Secure
Set-Cookie: NID=197=BZn2lJOlNIiKeLkhSA3YoTAgo5E0aCh8Sj1ileUG2a8d5Cw_lSQVcZ0j0hH8
3nb18ieVoFlVem5lvbWiB4zH0EHXAoTS_Bc4P2OxmPPjuyFMwyvmPkTX24R2cO9BiRbrbKCirX7C2JrK
fkbpXbJnGWo00zW9Nxp2p0XZOjWLP7o; expires=Mon, 10-Aug-2020 08:54:19 GMT; path=/;
domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
```

HTTP Requests

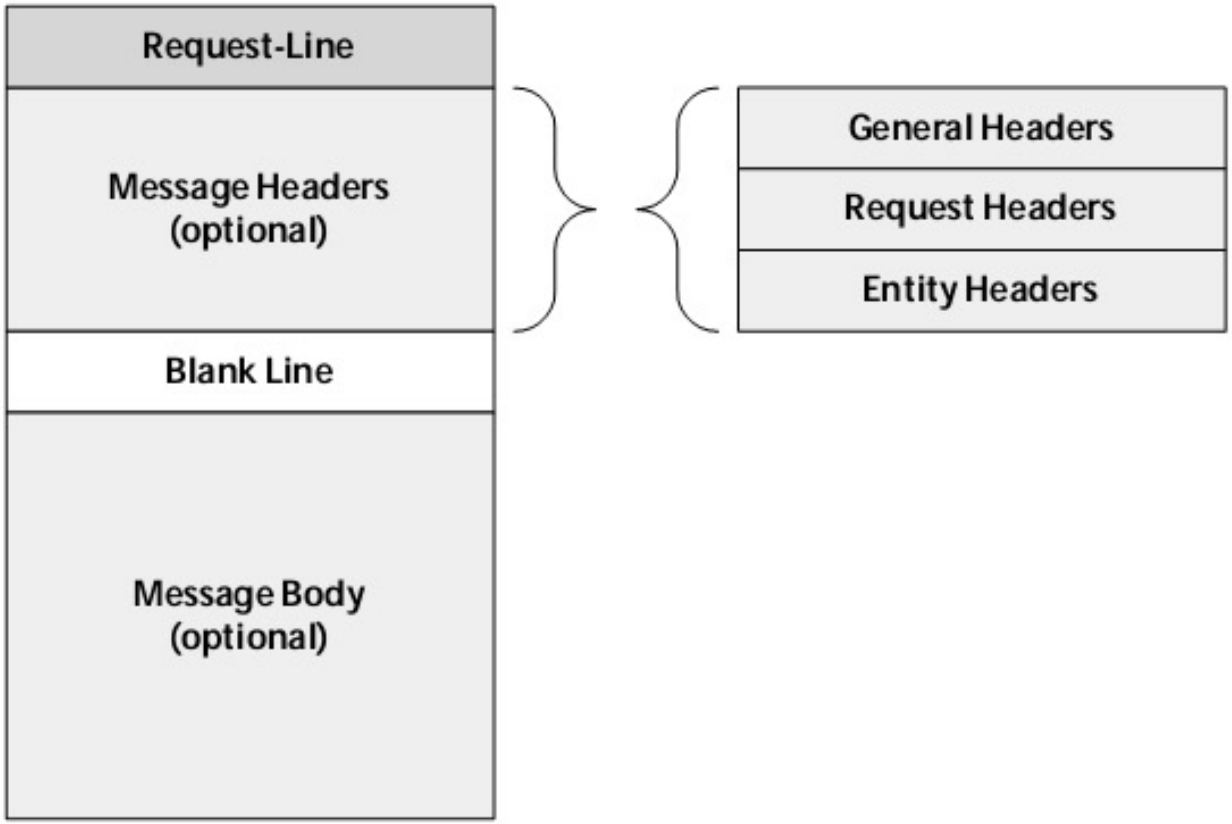
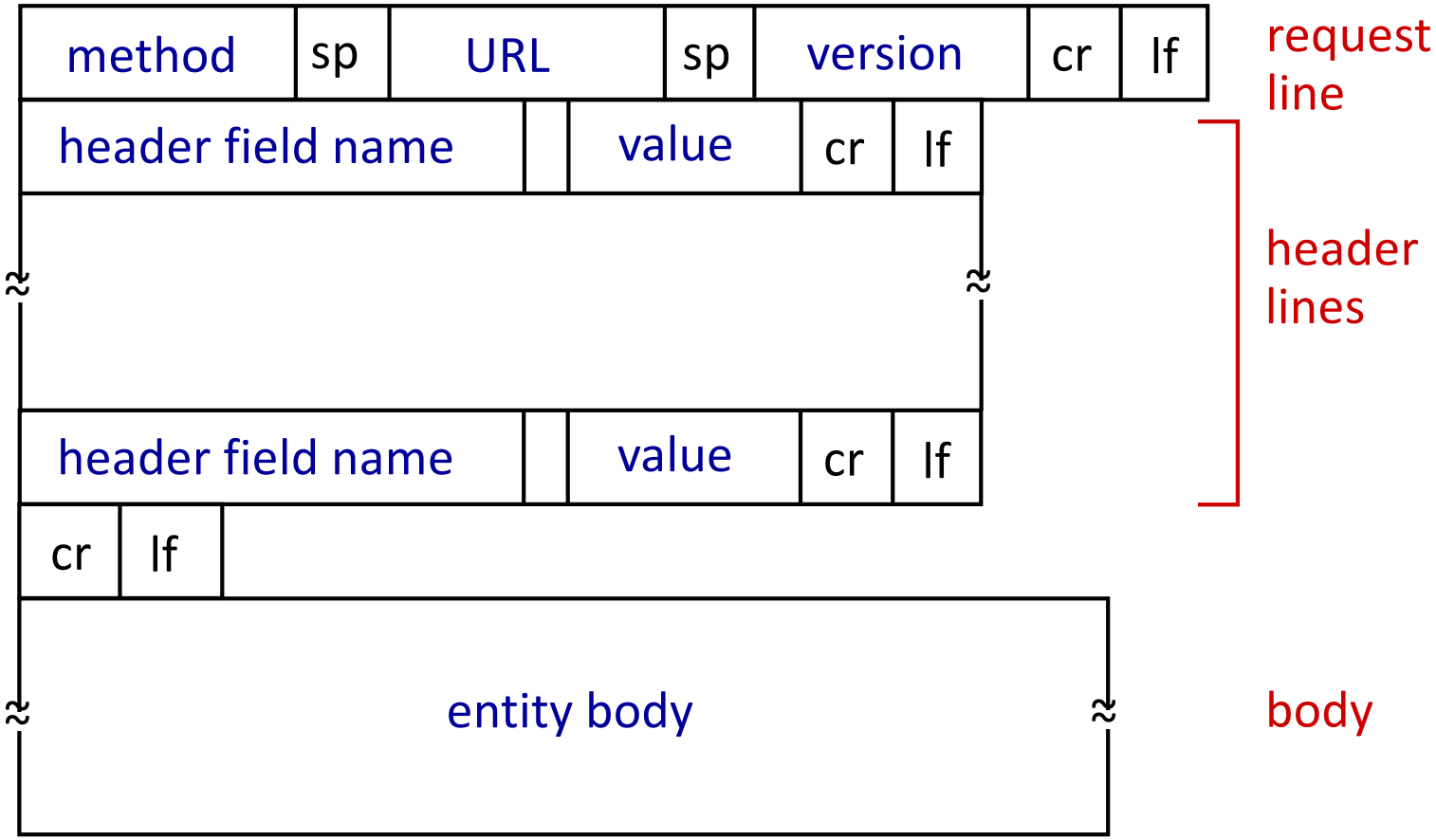


Figure 3.1 ▶
An HTTP request begins with a Request-Line and may include headers and a message body. The headers can describe general communications, the specific request, or the included message body.

HTTP request message: general format

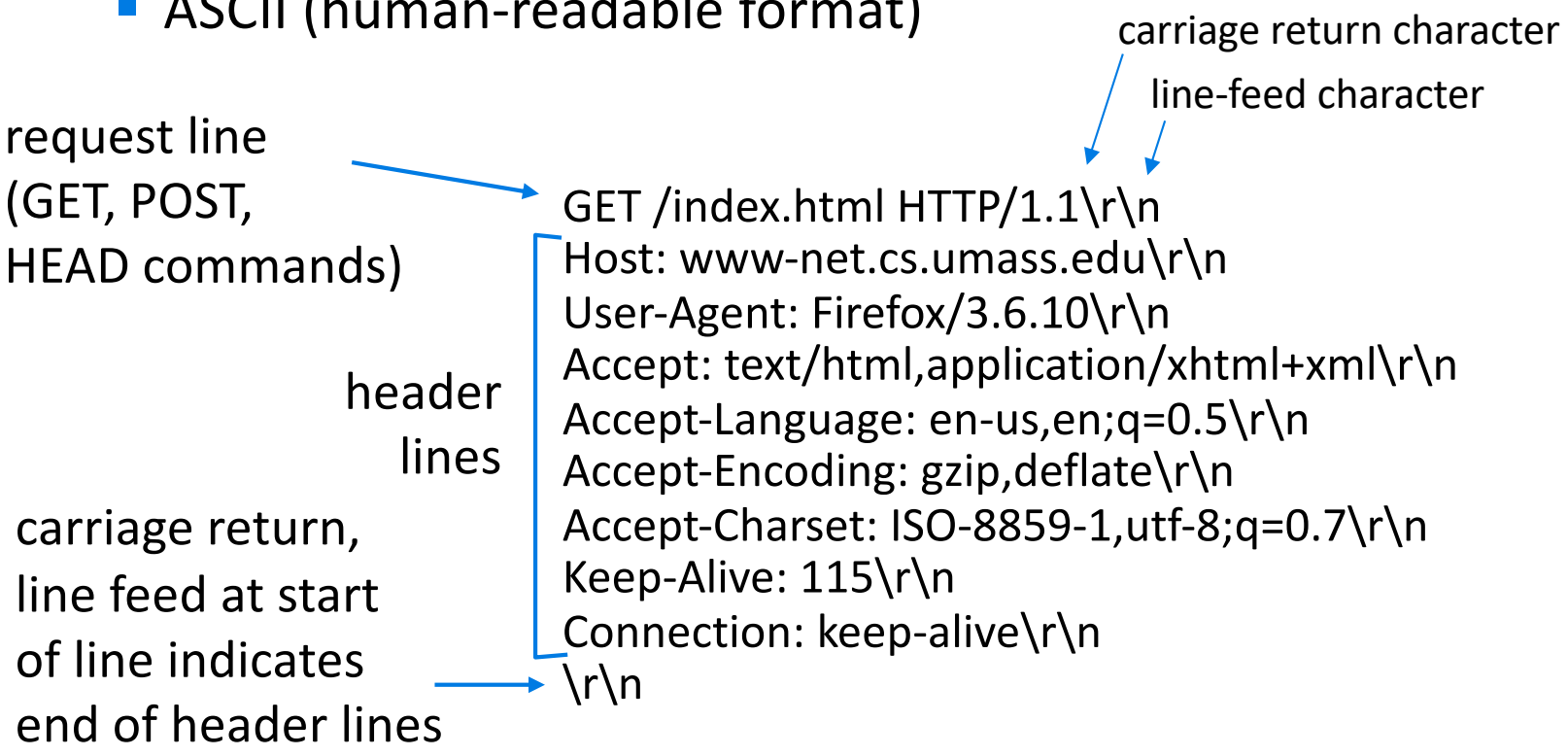


HTTP requests and responses Messages

- HTTP requests and responses are both types of Internet Messages (RFC 822), and share a general format:
  - **A Start Line, followed by a CRLF**
    - Request Line for requests
    - Status Line for responses
  - **Zero or more Message Headers**
    - field-name ":" [field-value] CRLF
  - **An empty line**
    - Two CRLFs mark the end of the Headers
  - **An optional Message Body if there is a payload**
    - All or part of the "Entity Body" or "Entity"

HTTP request message

- two types of HTTP messages: request, response
- HTTP request message:
  - ASCII (human-readable format)



Method types

HTTP/1.0:

- GET, POST
  - asks server to obtain an object
- HEAD
  - asks server to leave requested object out of response

HTTP/1.1:

- GET, POST, HEAD
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field

Uploading form input

POST method:

- web page often includes form input
- input is uploaded to server in entity body

URL method:

- uses GET method
- input is uploaded in URL field of request line:
  - www.somesite.com/animalsearch?monkeys&banana

A Closer Look at the Request Methods

- **GET**
  - By far most common method
  - Retrieves a resource from the server
  - Supports passing of query string arguments
- **HEAD**
  - Retrieves only the Headers associated with a resource but not the entity itself
  - Highly useful for protocol analysis, diagnostics
- **POST**
  - Allows passing of data in entity rather than URL
  - Can transmit of far larger arguments that GET
  - Arguments not displayed on the URL

More Request Methods, cont.

- **OPTIONS**
  - Shows methods available for use on the resource (if given a path) or the host (if given a "*")
- **TRACE**
  - Diagnostic method for assessing the impact of proxies along the request-response chain
- **CONNECT**
  - A common extension method for Tunneling other protocols through HTTP
- **PUT, DELETE**
  - Used in HTTP publishing (e.g., WebDav)

Web-based Distributed Authoring and Versioning (WebDAV, RFC 4918) is a set of methods based on the Hypertext Transfer Protocol (HTTP) that facilitates collaboration between users in editing and managing documents and files stored on World Wide Web servers.

idempotent methods

https://developer.mozilla.org/en-US/docs/Glossary/idempotent

An HTTP method is **idempotent** if an identical request can be made once or several times in a row with the same effect while leaving the server in the same state.

An idempotent method should not have any side-effects (except for keeping statistics).

Implemented correctly, the GET, HEAD, PUT, and DELETE method are **idempotent**, but not the POST method.

safe methods

https://developer.mozilla.org/en-US/docs/Glossary/safe

An HTTP method is safe if it doesn't alter the state of the server.

A method is safe if it leads to a read-only operation.

Several common HTTP methods are safe: GET, HEAD, or OPTIONS. All safe methods are also idempotent, but not all idempotent methods are safe.

For example, PUT and DELETE are both idempotent but unsafe.

Safe, unsafe and idempotent methods

- GET /pageX HTTP/1.1 is idempotent. Called several times in a row, the client gets the same results:
  - GET /pageX HTTP/1.1
  - GET /pageX HTTP/1.1
  - GET /pageX HTTP/1.1

- POST /add_row HTTP/1.1 is not idempotent; if it is called several times, it adds several rows:
  - POST /add_row HTTP/1.1
  - POST /add_row HTTP/1.1 -> Adds a 2nd row
  - POST /add_row HTTP/1.1 -> Adds a 3rd row

- DELETE /idX/delete HTTP/1.1 is idempotent, even if the returned status code may change between requests:
  - DELETE /idX/delete HTTP/1.1 -> Returns 200 if idX exists
  - DELETE /idX/delete HTTP/1.1 -> Returns 404 as it just got deleted
  - DELETE /idX/delete HTTP/1.1 -> Returns 404

A Closer Look at HTTP Headers

Headers come in four major types, some for requests, some for responses, some for both:

- **General Headers**
  - Provide info about messages of both kinds
- **Request Headers**
  - Provide request-specific info
- **Response Headers**
  - Provide response-specific info
- **Entity Headers**
  - Provide info about request and response entities
- Extension headers are also possible

General Headers

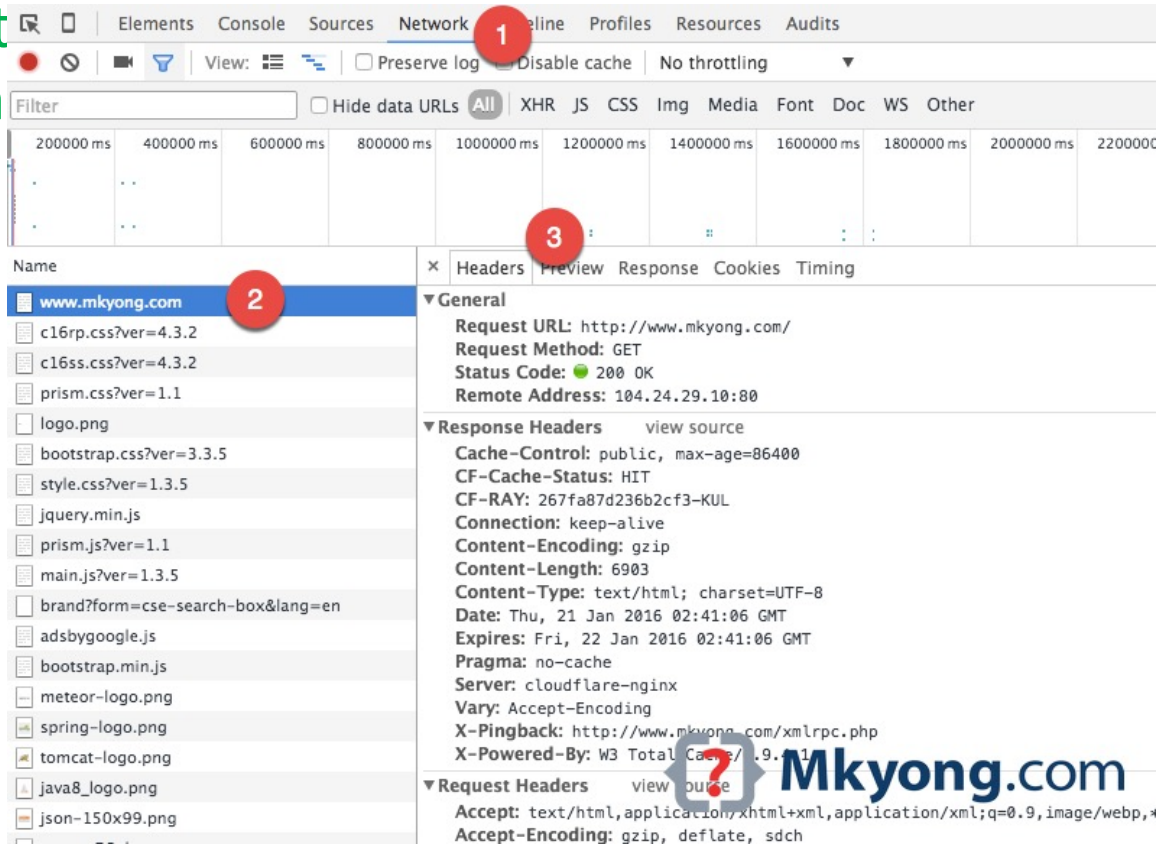
- **Connection** – lets clients and servers manage connection state
  - Connection: Keep-Alive
  - Connection: close
- **Date** – when the message was created
  - Date: Sat, 31-May-03 15:00:00 GMT
- **Via** – shows proxies that handled message
  - Via: 1.1 www.myproxy.com (Squid/1.4)
- **Cache-Control** – Among the most complex of headers, enables caching directives
  - Cache-Control: no-cache

Request Headers

- **Host** – The hostname (and optionally port) of server to which request is being sent
- **Referer** – The URL of the resource from which the current request URI came
  - Referer: http://www.host.com/login.asp
- **User-Agent** – Name of the requesting application, used in browser sensing
  - User-Agent: Mozilla/4.0 (Compatible; MSIE 6.0)
- **Accept** and its variants – Inform servers of client's capabilities and preferences
  - Enables content negotiation
  - Accept: image/gif, image/jpeg;q=0.5
  - Accept- variants for Language, Encoding, Charset
- **Cookie** How clients pass cookies back to the servers that set them
  - Cookie: id=23432;level=3

How to view HTTP headers in Google Chrome?

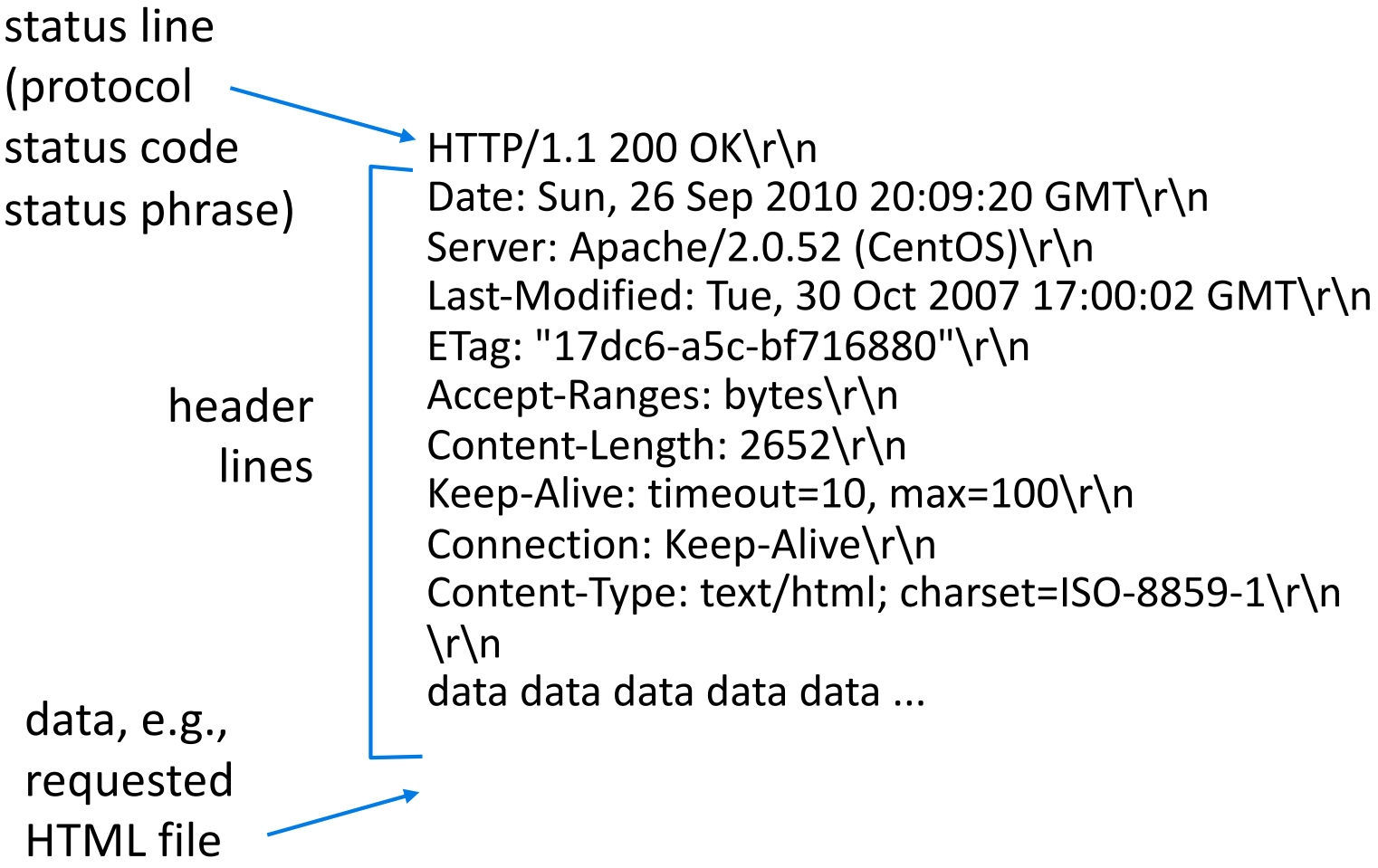
- In Chrome, visit a URL, right click, select Inspect to open the developer tools.
- Select Network tab.
- Reload the page, select any HTTP request on the left panel, and the HTTP headers will be displayed on the right panel.



https://mkyong.com/computer-tips/how-to-view-http-headers-in-google-chrome/

HTTP: response

HTTP response messages



A Closer Look at the Status Line

- **Consists of three major parts:**
- The HTTP Version
  - Just like third part of Request Line
- Status Code
  - **5 groups of 3 digit integers indicating the result of the attempt to satisfy the request**The Reason Phrase followed by the CRLF
  - Short textual description of the status code

Table 3.2 HTTP Status Code Categories

Status Code	Meaning
100-199	Informational; the server received the request but a final result is not yet available.
200-299	Success; the server was able to act on the request successfully.
300-399	Redirection; the client should redirect the request to a different server or resource.
400-499	Client error; the request contained an error that prevented the server from acting on it successfully.
500-599	Server error; the server failed to act on a request even though the request appears to be valid.

HTTP response status codes examples

- status code appears in 1st line in server-to-client response message.
- some sample codes:
  - 200 OK
    - request succeeded, requested object later in this msg
  - 301 Moved Permanently
    - requested object moved, new location specified later in this msg (Location:)
  - 400 Bad Request
    - request msg not understood by server
  - 404 Not Found
    - requested document not found on this server
  - 505 HTTP Version Not Supported

Response Headers

- **Server** – The server's name and version
  - Server: Microsoft-IIS/5.0
  - Can be problematic for security reasons

- **Set-Cookie** – This is how a server sets a cookie on a client
  - Set-Cookie: id=234; path=/shop; expires=Sat, 31-May-03 15:00:00 GMT; secure

Entity Headers

- **Allow** – Lists the request methods that can be used on the entity
  - Allow: GET, HEAD, POST
- **Location** – Gives the alternate or new location of the entity
  - Used with 3xx response codes (redirects)
  - Location: http://www.iugaza.edu.ps/ar/
- **Content-Encoding** – specifies encoding performed on the body of the response
  - Used with HTTP compression
  - Corresponds to Accept-Encoding request header
  - Content-Encoding: gzip
- **Content-Length** – The size of the entity body in bytes
- **Content-Location** – The actual if different than its request URL
- **Content-Type** – specifies Media (MIME) type of the entity body

HTTPS

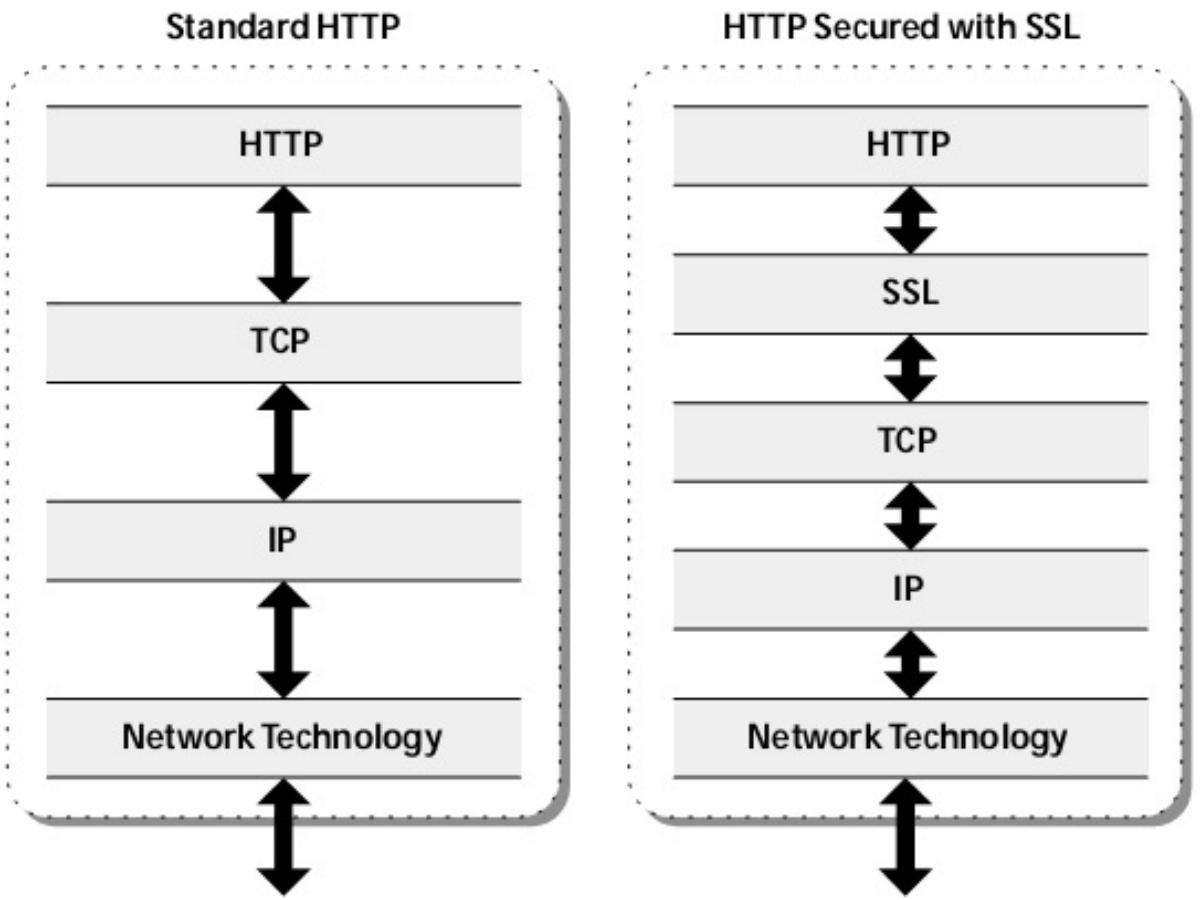
=

HTTP + SSL

Slides from HTTP vs. HTTPS by Eng. T. Aldaldooh

HTTPS

Figure 4.10 ▶
The SSL protocol inserts itself between an application like HTTP and the TCP transport layer. TCP sees SSL as just another application, and HTTP communicates with SSL much the same as it does with TCP.

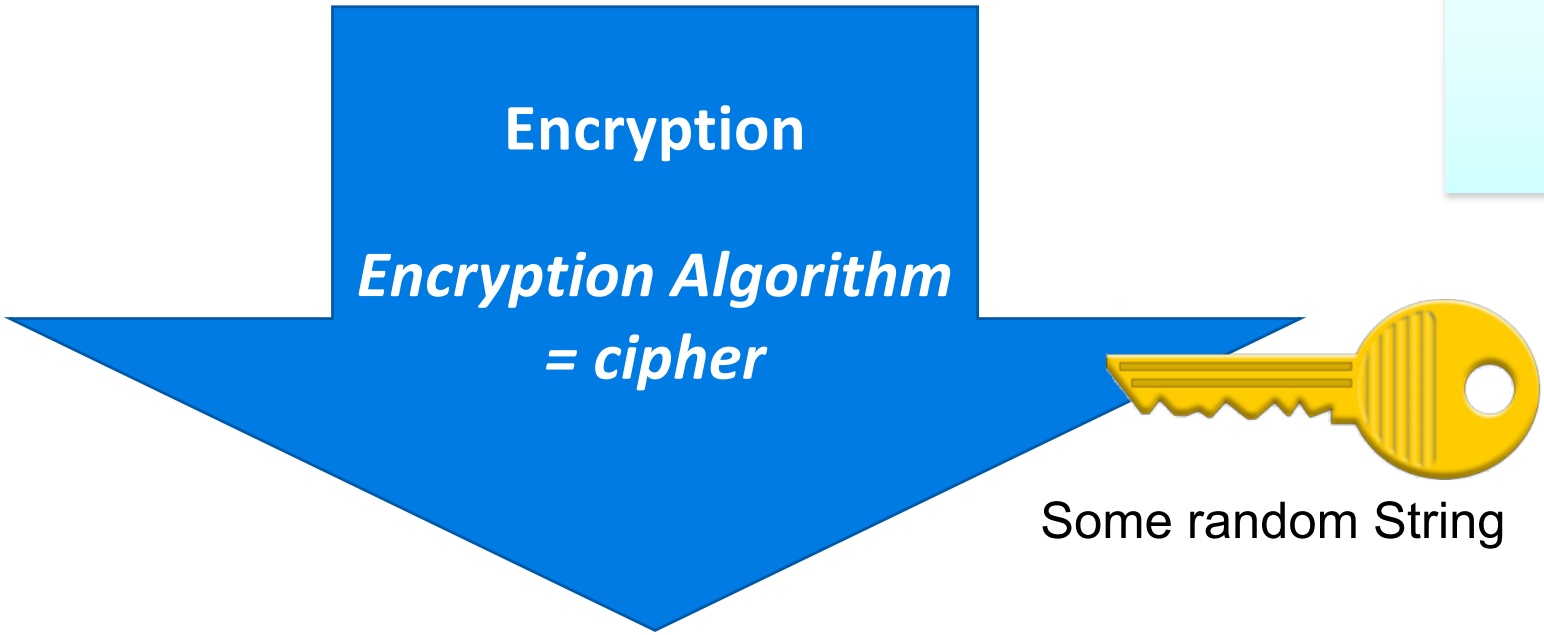


HTTPS only slightly slower than HTTP.

HTTPS is a bit more complex to set up (due to the need of a certificate)

Cryptography for dummies

Important information Data, Data, Data.



Hh2sh!~hH==E#@ns8676%===sdf

Plain Text

Cipher Text

Hyper-basic example

Msg: “Good Morning”

Encription algorithm: shift letters forward by n

Key: n

Encoded msg, with key=1: “Hppe Npsojoh”

Encoded msg, with key=2: “Iqqf Oqtpkpi”

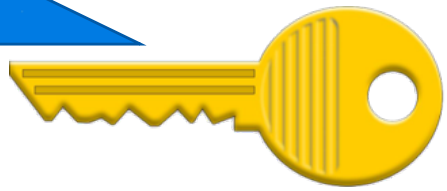
Decription algorithm: shift letters backward by n

Cryptography cont.

Symmetric Key

Important information Data, Data, Data.

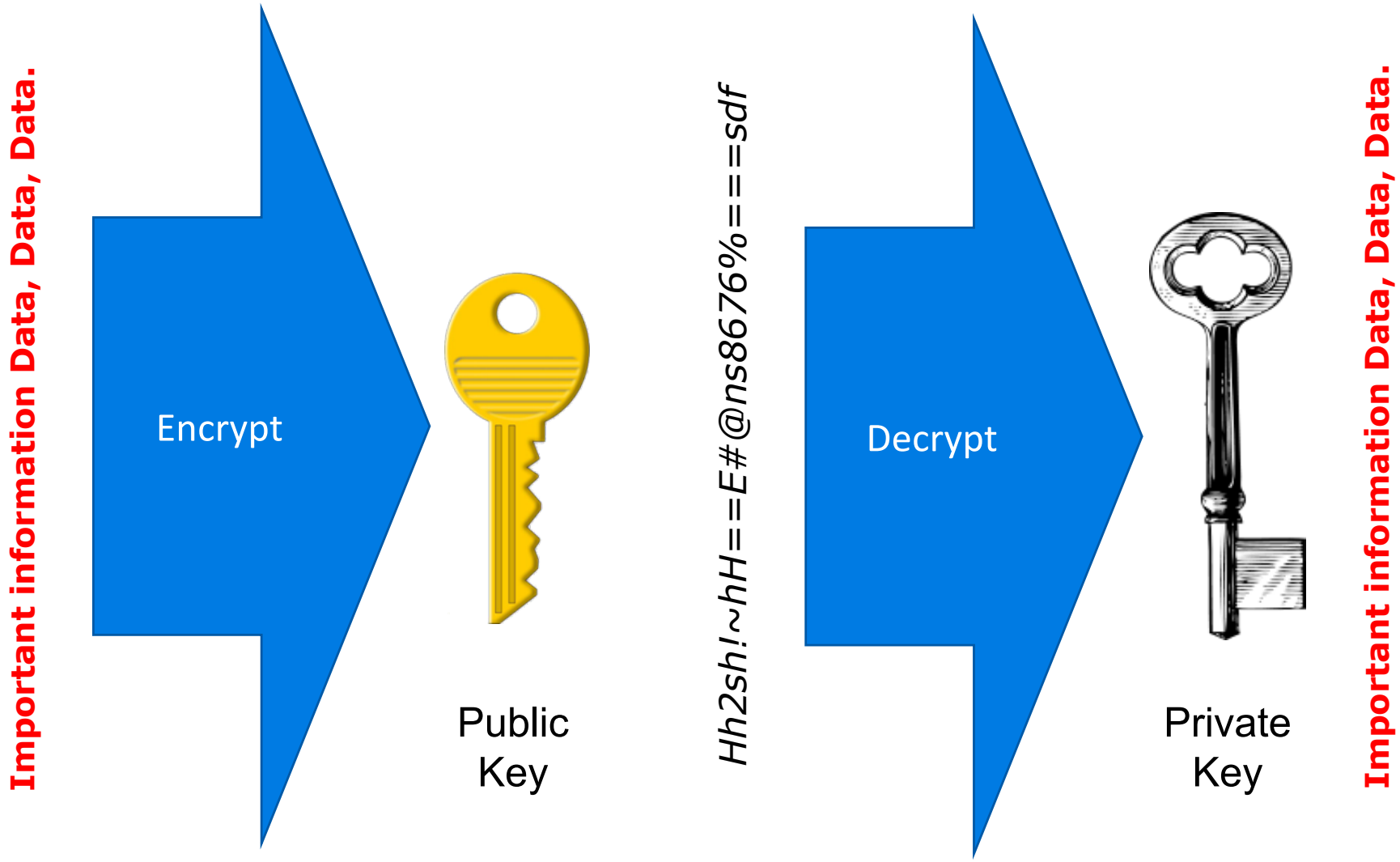
Decryption Algorithm



Some random String

Hh2sh!~hH==E#@ns8676%===sdf

Asymmetric (public-key) encryption

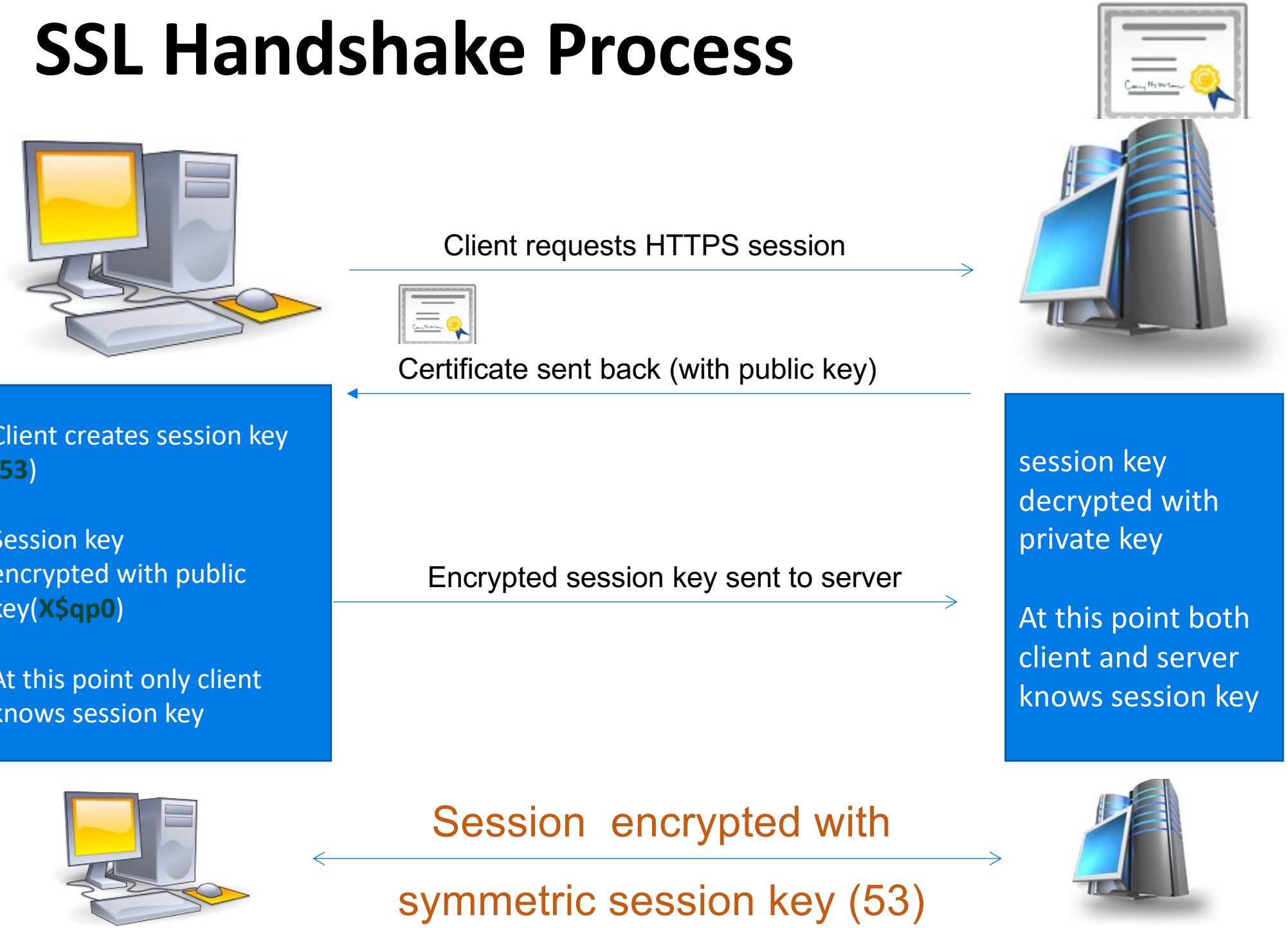


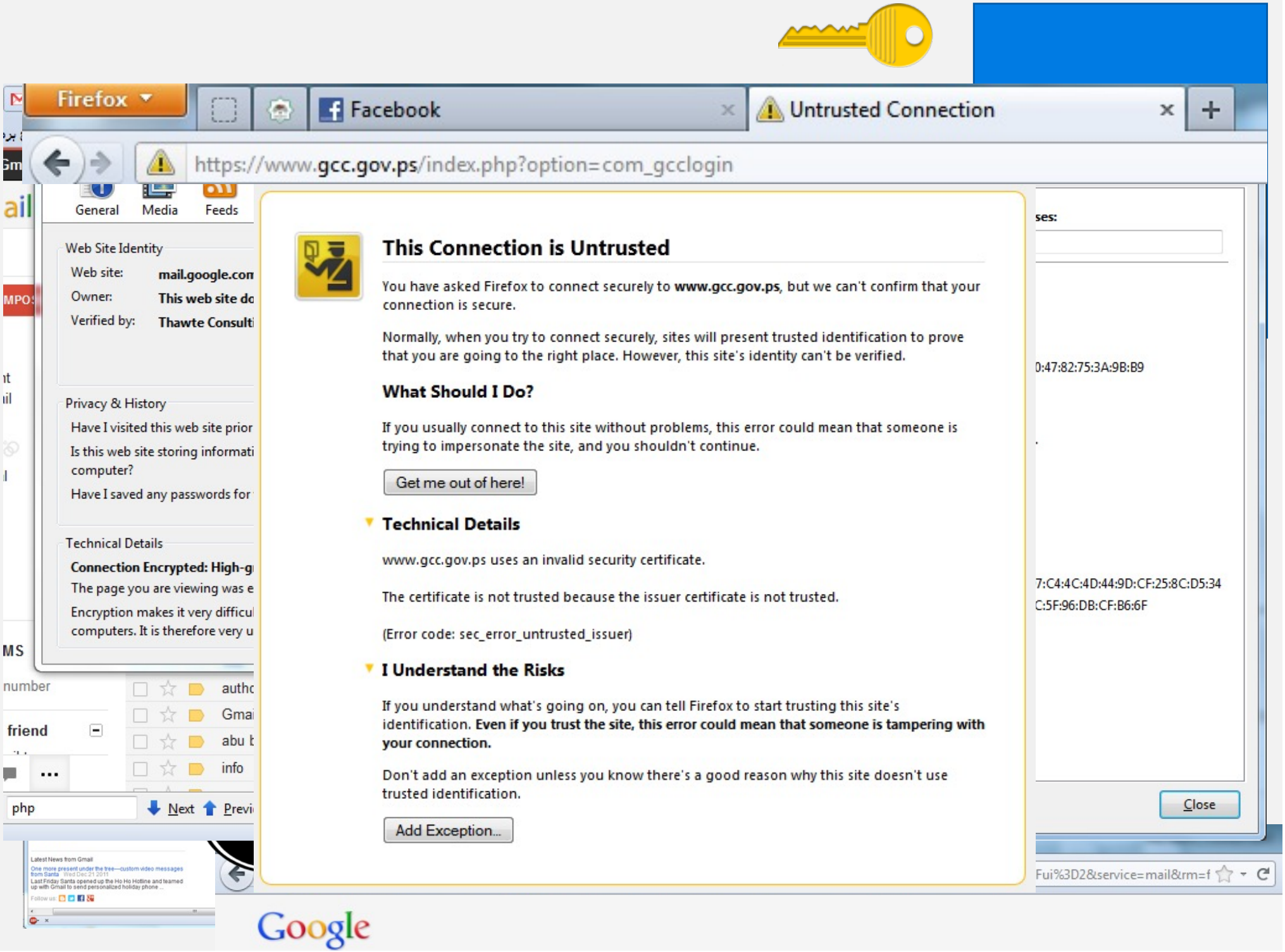
Asymmetric key decryption is slower than symmetric key decryption

SSL Session

- Uses asymmetric encryption to privately share the session key
  - Asymmetric has a lot of overhead
- Uses symmetric encryption to encrypt data
  - Symmetric encryption is quicker and uses less resource

SSL Handshake Process




Firefox
Facebook
Untrusted Connection
https://www.gcc.gov.ps/index.php?option=com_gcclogin
General
Media
Feeds
Web Site Identity
Web site: mail.google.com
Owner: This web site do
Verified by: Thawte Consulti
Privacy & History
Have I visited this web site prior
Is this web site storing informati
computer?
Have I saved any passwords for
Technical Details
Connection Encrypted: High-g
The page you are viewing was e
Encryption makes it very difficul
computers. It is therefore very u
This Connection is Untrusted
You have asked Firefox to connect securely to www.gcc.gov.ps, but we can't confirm that your connection is secure.
Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.
What Should I Do?
If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.
Get me out of here!
Technical Details
www.gcc.gov.ps uses an invalid security certificate.
The certificate is not trusted because the issuer certificate is not trusted.
(Error code: sec_error_untrusted_issuer)
I Understand the Risks
If you understand what's going on, you can tell Firefox to start trusting this site's identification. Even if you trust the site, this error could mean that someone is tampering with your connection.
Don't add an exception unless you know there's a good reason why this site doesn't use trusted identification.
Add Exception...
ses:
0:47:82:75:3A:9B:B9
7:C4:4C:4D:44:9D:CF:25:8C:D5:34
C:5F:96:DB:CF:B6:6F
Close
friend
php
Next
Google

Other security issues

- Who guarantees that the certificate is authentic?
  - “Chain of trust”
- More in the security courses

Conclusion

- HTTPS only slightly slower than HTTP.

Reference

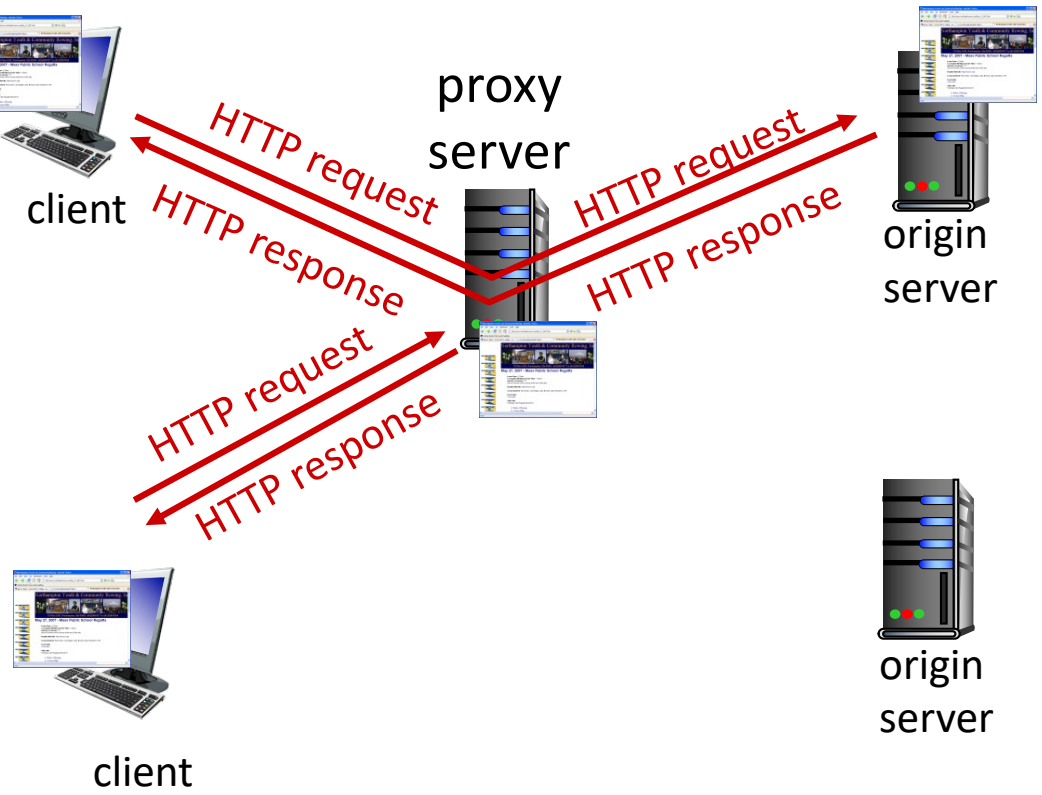
- HTTP Essentials Protocols for Secure, Scaleable Web Sites by Stephen Thomas .
- HTTP The Definitive Guide.
- View HTTP Request and Response Header < http://web-sniffer.net/ >

Traffic optimization 1: HTTP: proxy

Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client



More about Web caching

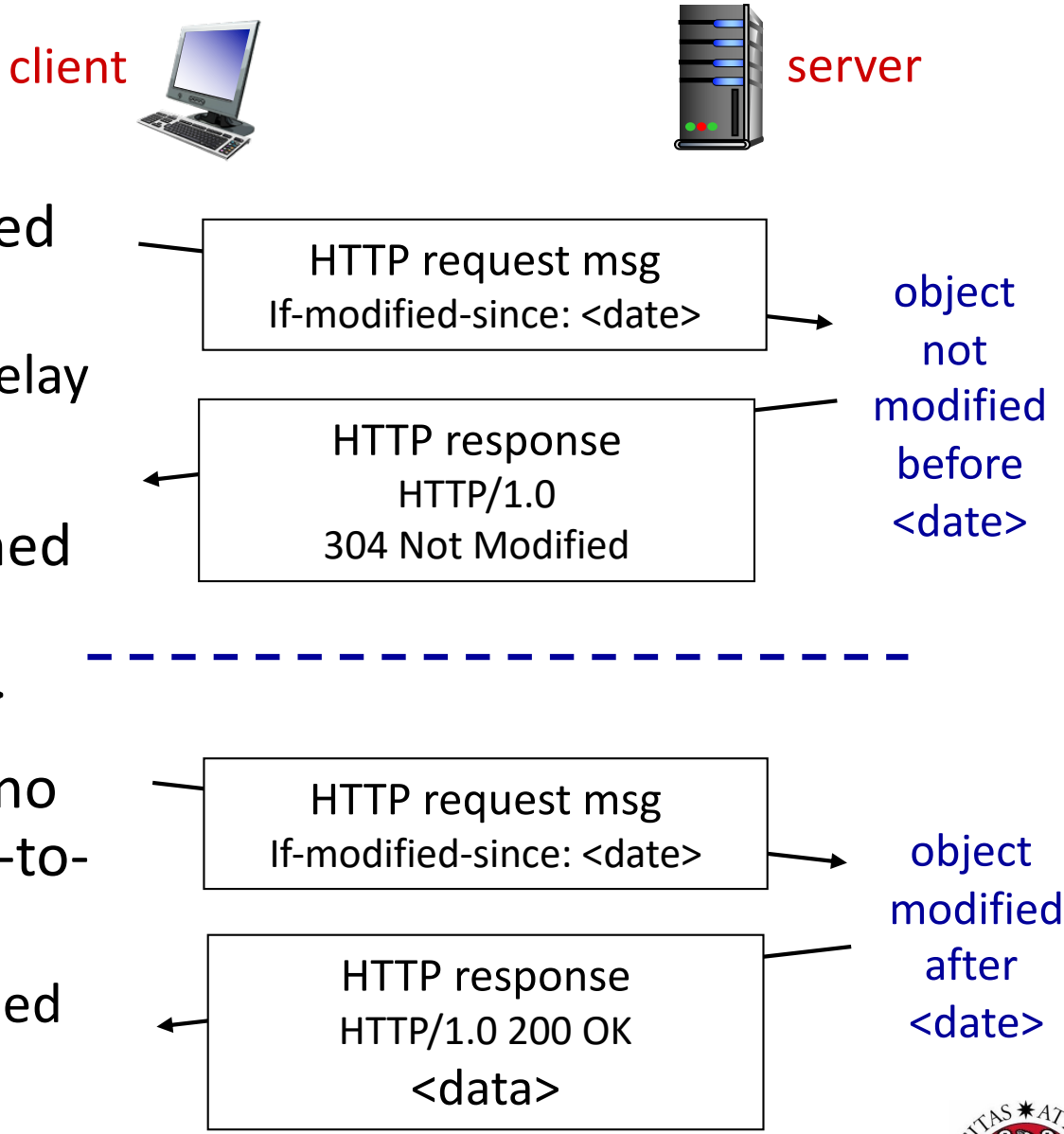
- cache acts as both client and server
  - server for original requesting client
  - client to origin server
- typically cache is installed by ISP (university, company, residential ISP)

why Web caching?

- reduce response time for client request
- reduce traffic on an institution's access link
- Internet dense with caches: enables "poor" content providers to effectively deliver content (so too does P2P file sharing)

Conditional GET

- Goal: don't send object if cache has up-to-date cached version
  - no object transmission delay
  - lower link utilization
- cache: specify date of cached copy in HTTP request
  - If-modified-since: <date>
- server: response contains no object if cached copy is up-to-date:
  - HTTP/1.0 304 Not Modified



Traffic optimization 2: Video streaming and CDNs

Content Distribution Networks

Video Streaming and CDNs: context

- video traffic: major consumer of Internet bandwidth
  - Netflix, YouTube: 37%, 16% of downstream residential ISP traffic
  - ~1B YouTube users, ~75M Netflix users
- challenge: scale - how to reach ~1B users?
  - single mega-video server won't work (why?)
- challenge: heterogeneity
  - different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)
- solution: distributed, application-level infrastructure

Multimedia: video

- video: sequence of images displayed at constant rate
  - e.g., 24 images/sec
- digital image: array of pixels
  - each pixel represented by bits
- coding: use redundancy within and between images to decrease # bits used to encode image
  - spatial (within image)
  - temporal (from one image to next)

spatial coding example: instead of sending N values of same color (all purple), send only two values: color value (purple) and number of repeated values (N)

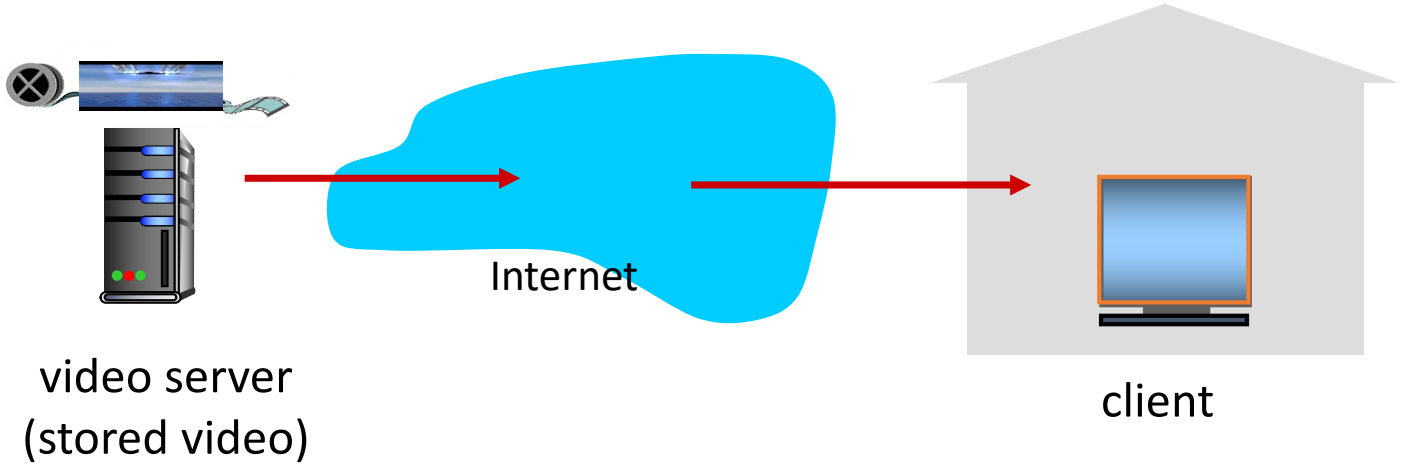
frame i

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

frame i+1

Streaming stored video

- Simple scenario:



Streaming multimedia: DASH

DASH: Dynamic, Adaptive Streaming over HTTP

- server:
  - divides video file into multiple chunks
  - each chunk stored, encoded at different rates
  - manifest file: provides URLs for different chunks
- client:
  - periodically measures server-to-client bandwidth
  - consulting manifest, requests one chunk at a time
    - chooses maximum coding rate sustainable given current bandwidth
    - can choose different coding rates at different points in time (depending on available bandwidth at time)

Streaming multimedia: DASH

DASH: Dynamic, Adaptive Streaming over HTTP

- "intelligence" at client: client determines
  - when to request chunk (so that buffer starvation, or overflow does not occur)
  - what encoding rate to request (higher quality when more bandwidth available)
  - where to request chunk (can request from URL server that is "close" to client or has high available bandwidth)

Content distribution networks

- challenge: how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

- option 1: single, large “mega-server”
  - single point of failure
  - point of network congestion
  - long path to distant clients
  - multiple copies of video sent over outgoing link

- quite simply: this solution doesn’t scale

Content distribution networks

- challenge: how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

- option 2: store/serve multiple copies of videos at multiple geographically distributed sites (CDN)
  - enter deep: push CDN servers deep into many access networks
    - close to users
    - used by Akamai, 1700 locations
  - bring home: smaller number (10's) of larger clusters in POPs near (but not within) access networks
    - used by Limelight

Content Distribution Networks (CDNs)

- CDN: stores copies of content at CDN nodes
  - e.g. Netflix stores copies of MadMen
- subscriber requests content from CDN
  - directed to nearby copy, retrieves content
  - may choose different copy if network path congested