Application Layer
Web/HTTP

Prof. Anja Feldmann, Ph.D.

(Based on slide deck of Computer Networking, 7th ed., Jim Kurose and Keith Ross.)

Application Layer

Goals

- Conceptual, implementation aspects
- Communication paradigms
  - Client-server and Peer-to-peer
- Transport-layer service models

- Learn protocols through examples
  - HTTP
  - DNS
  - eMail

Not that Web!

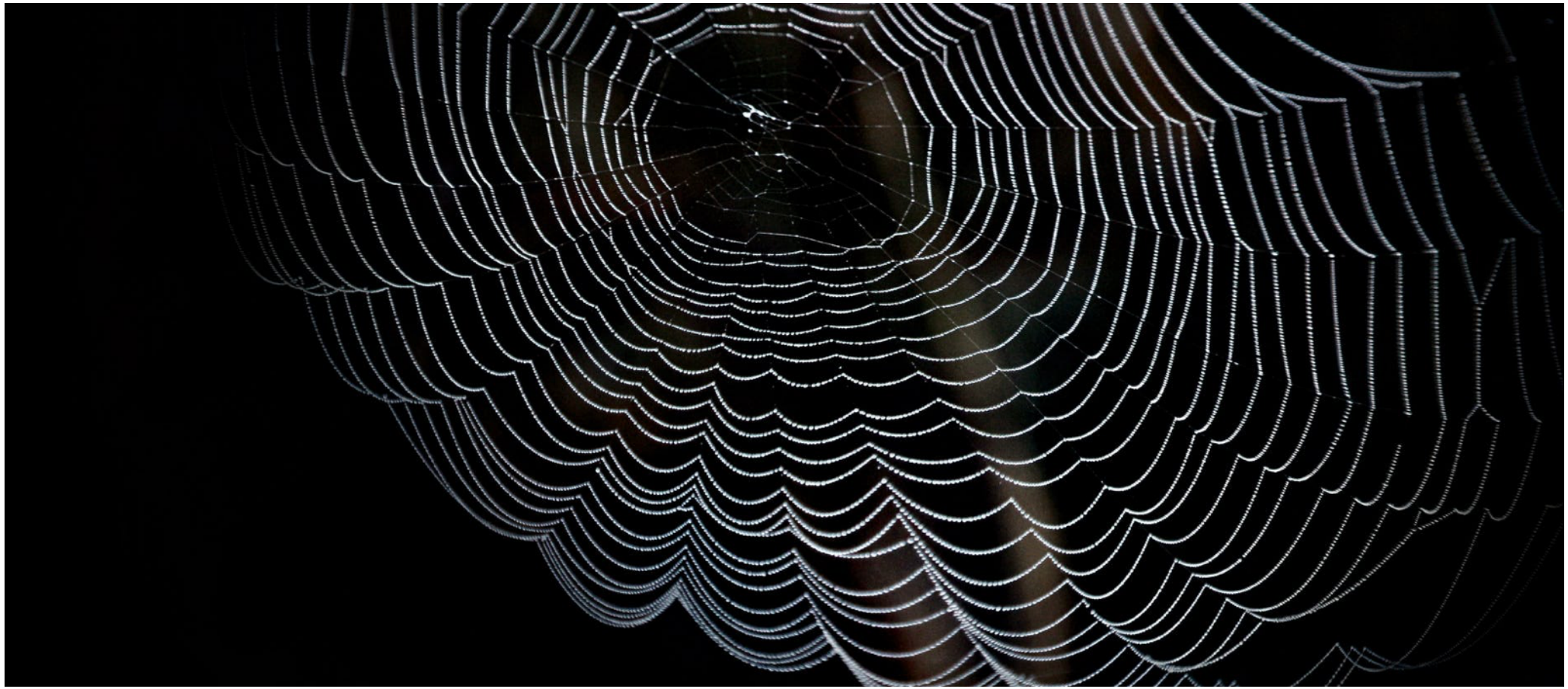
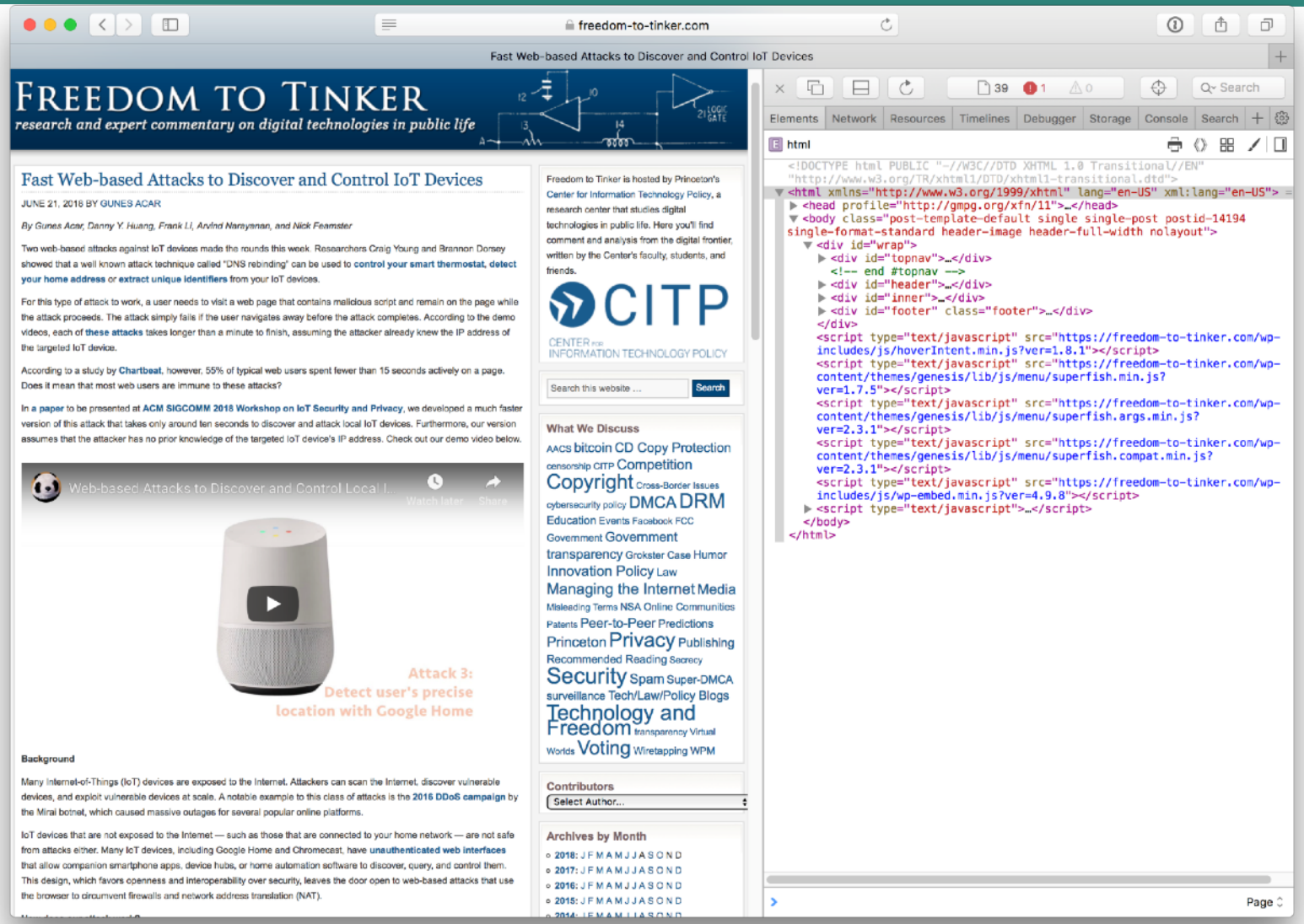


Image credits: Pixabay, www.pexels.com

Web and HTTP: An overview

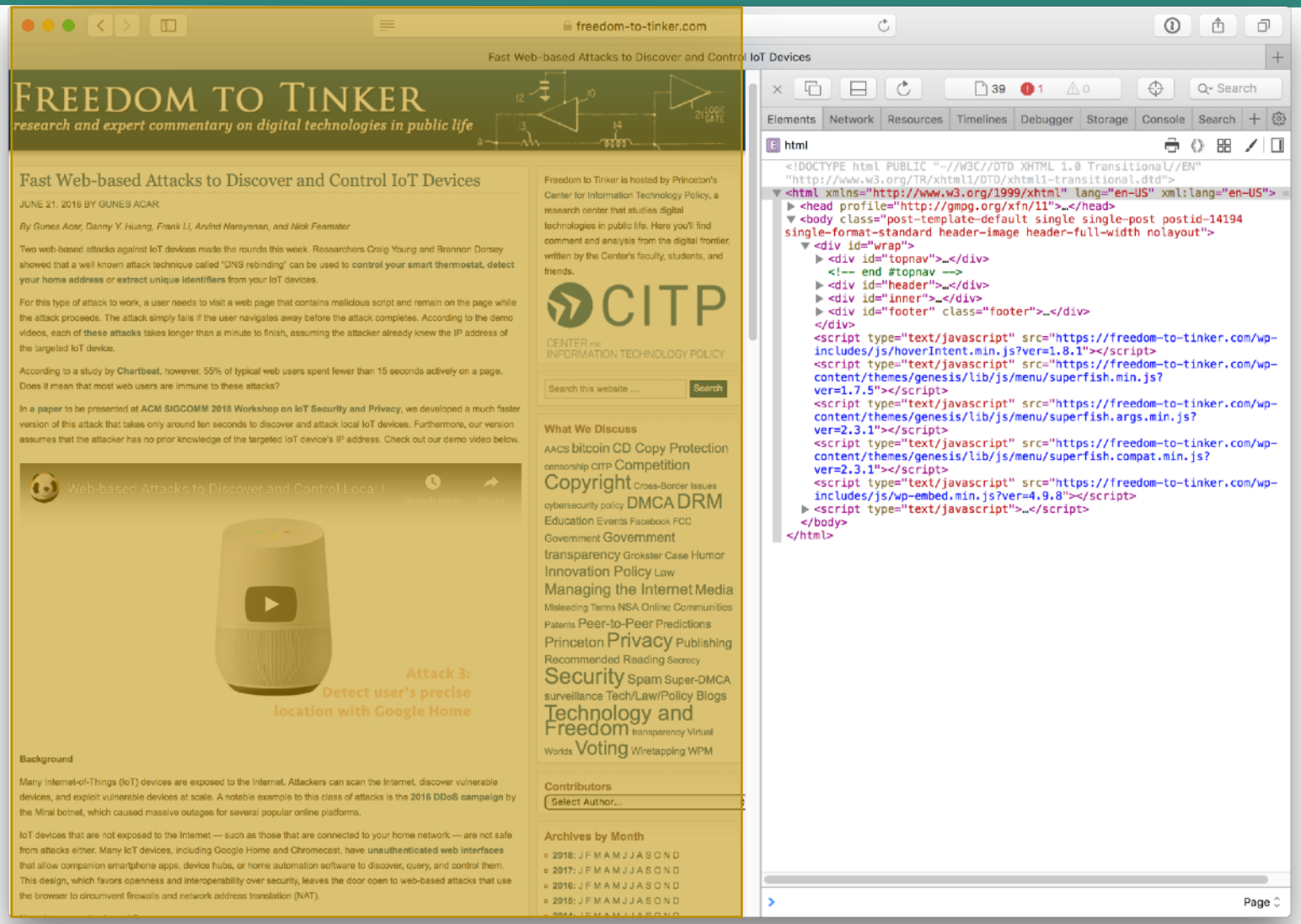
- Web page consists of objects
- Objects?
  - E.g., HTML file, JPEG image, audio file, …
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL
  - E.g., https://www.mpi-inf.mpg.de/inet/

Web and HTTP: An overview

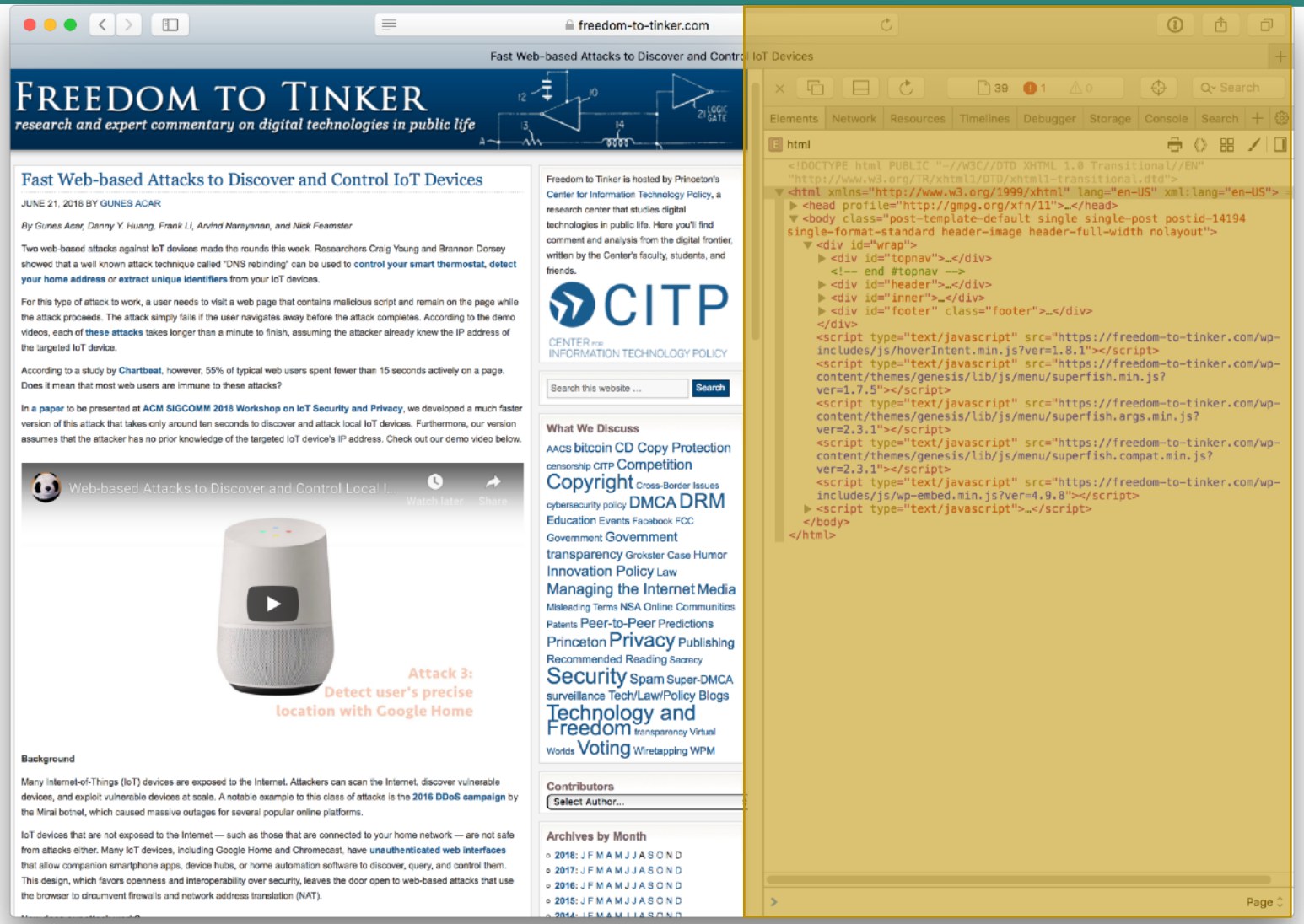


Web and HTTP: An overview

Web page



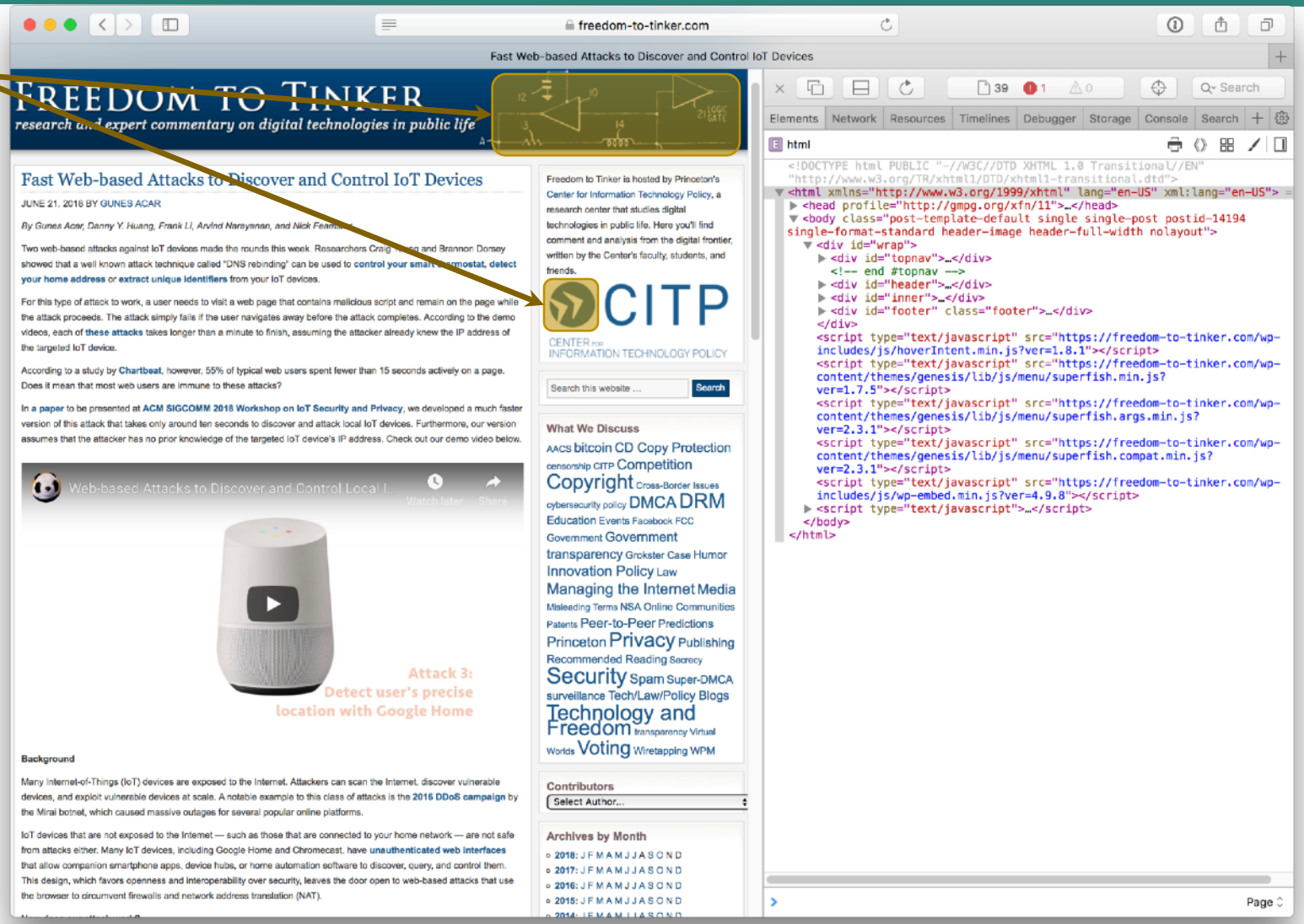
Web and HTTP: An overview



HTML (source)

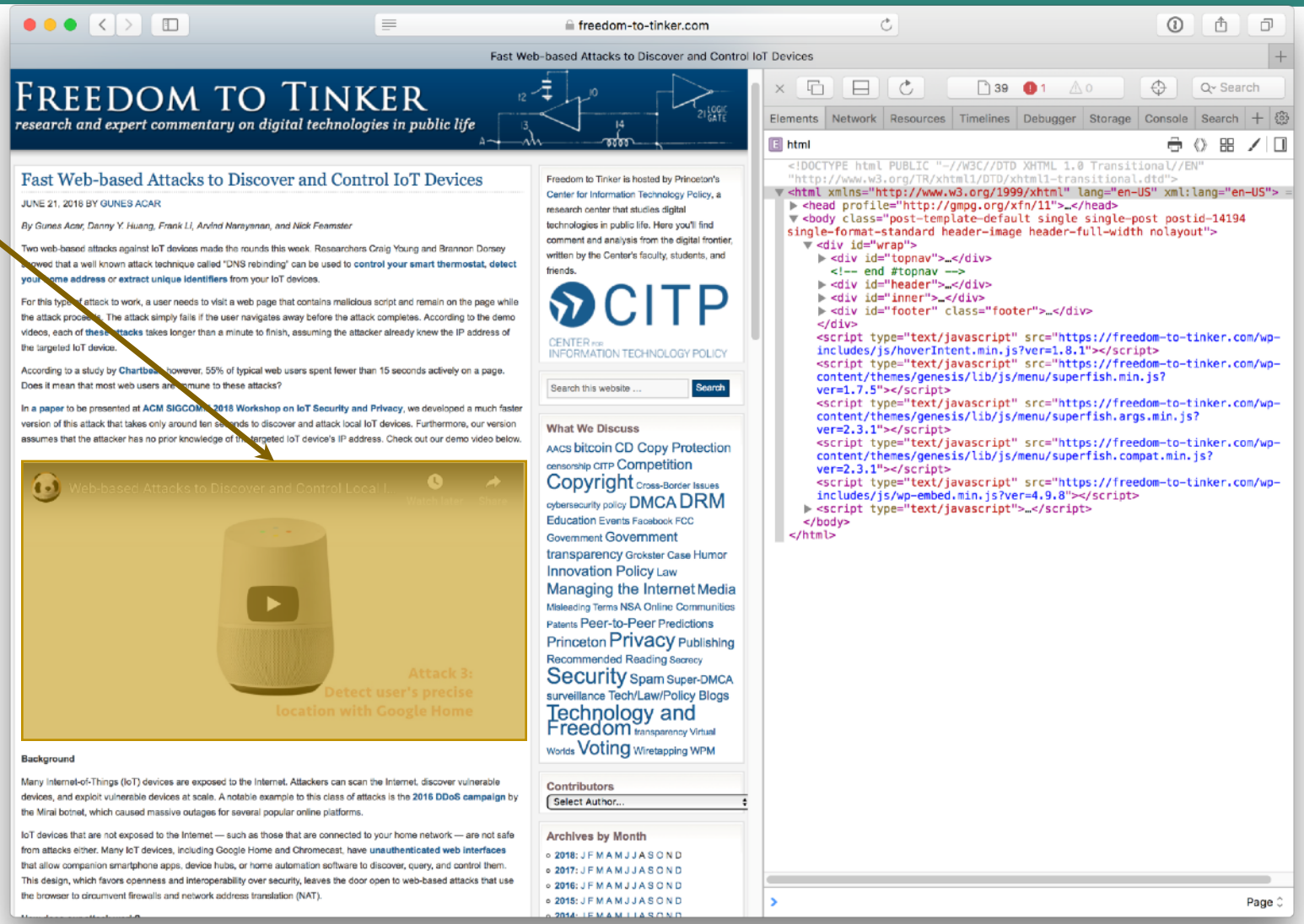
Web and HTTP: An overview

Image

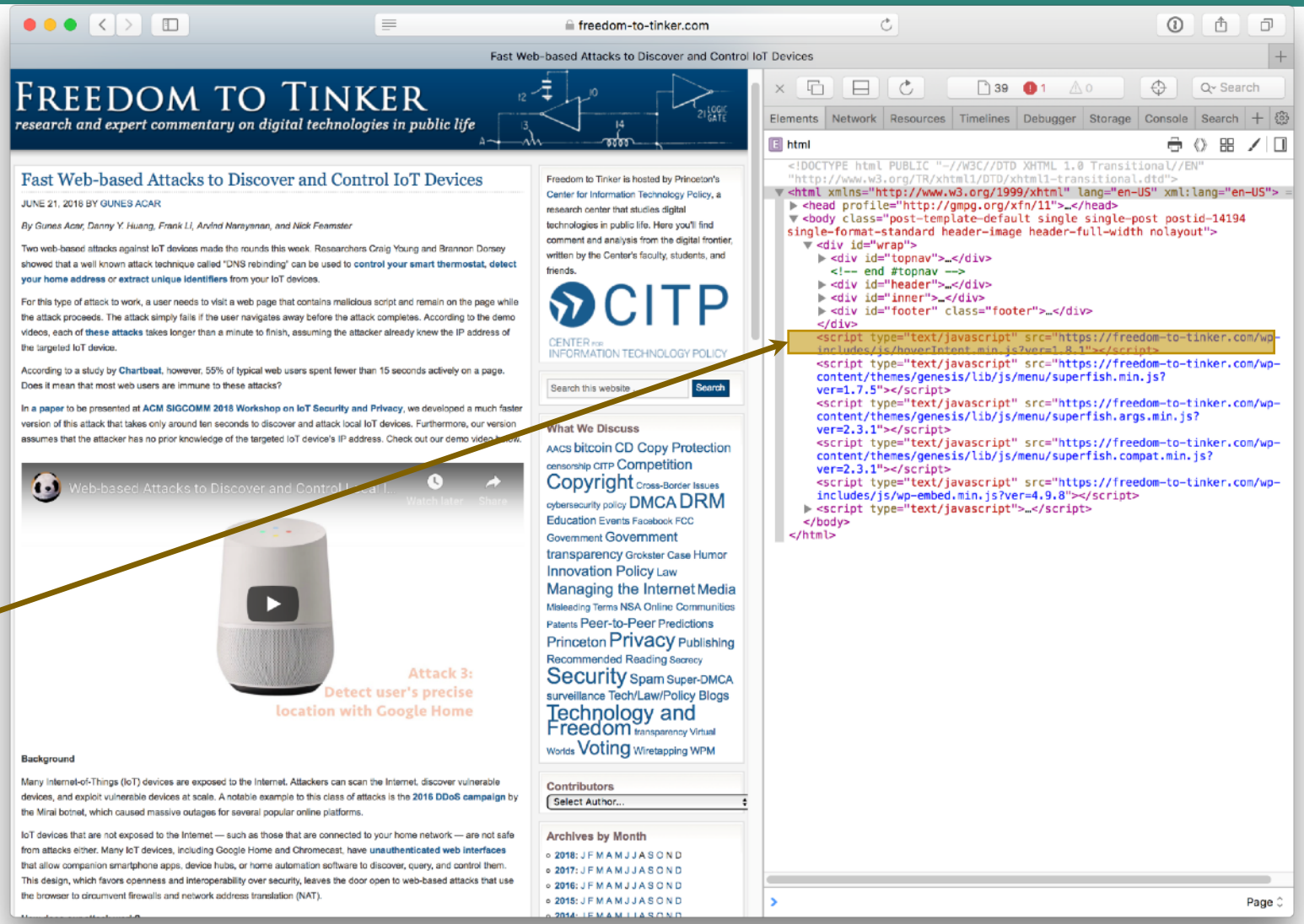


Web and HTTP: An overview

Video

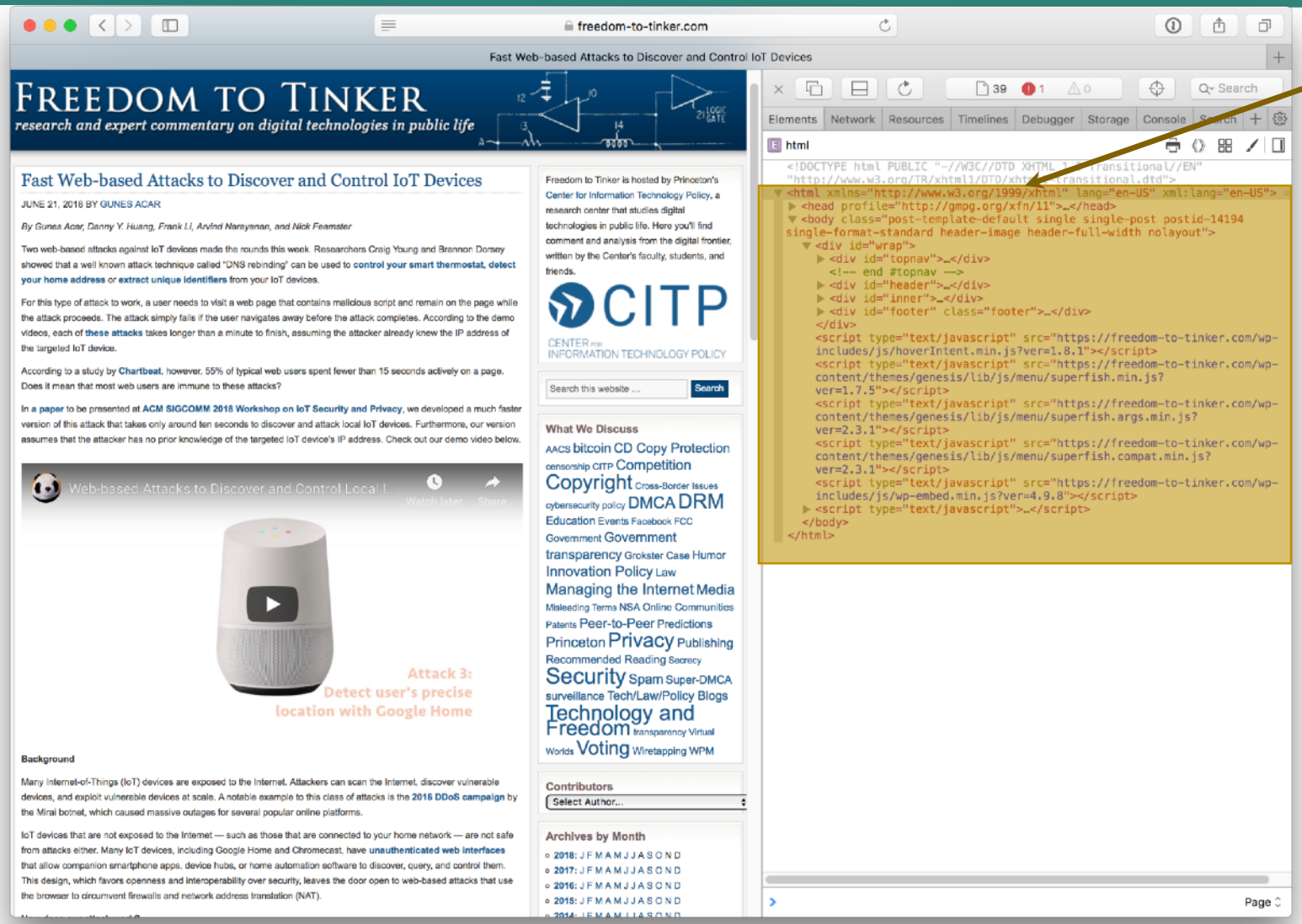


Web and HTTP: An overview



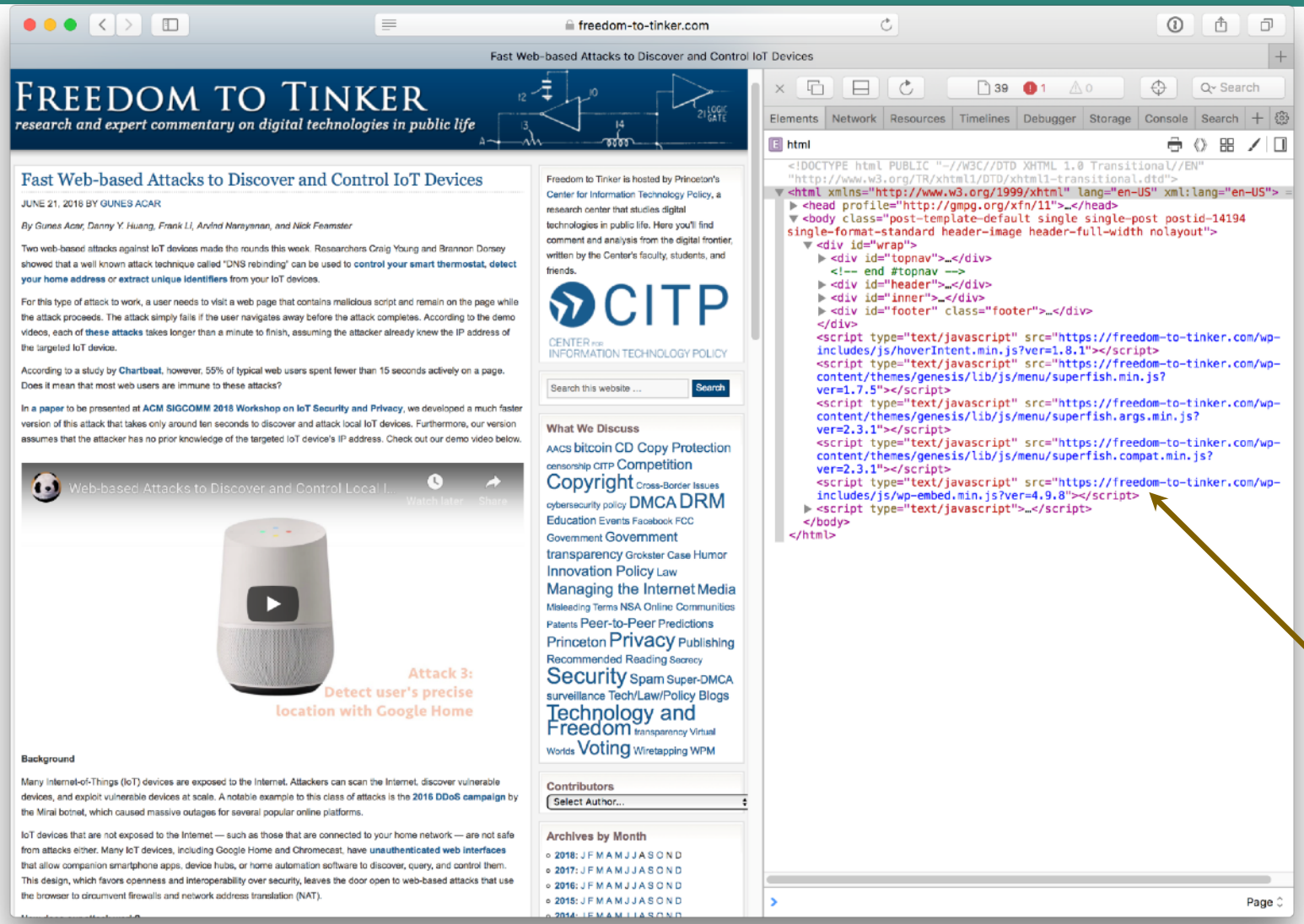
JavaScript

Web and HTTP: An overview



Base HTML

Web and HTTP: An overview

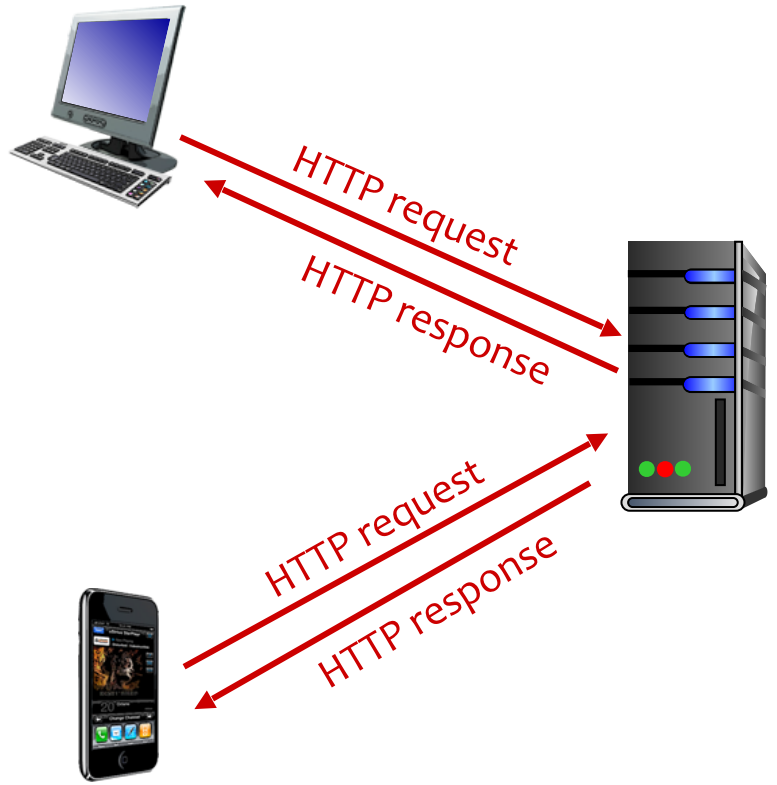


The HTTP Protocol

- HTTP: HyperText Transfer Protocol
  - Application layer protocol for the Web

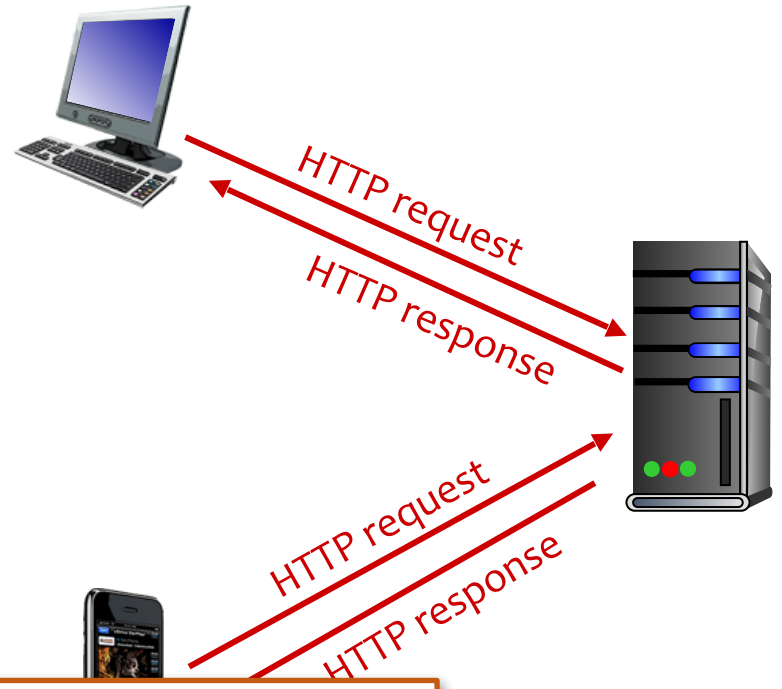
The HTTP Protocol

- Client-Server model
  - Client
    - Browser that requests, receives, "displays" Web objects
  - Server
    - Server sends objects in response to requests



The HTTP Protocol

- Client-Server model
  - Client
    - Browser that requests, receives, "displays" Web objects
  - Server
    - Server sends objects in response to requests



Also used as part of many other application layer protocol

HTTP Development Timeline

- Mar. 1990 — CERN labs document proposing Web
- Jan. 1992 — HTTP/0.9 specification
- Dec. 1992 — Proposal to add MIME to HTTP
- Feb. 1993 — UDI (Universal Document Identifier) Network
- Mar. 1993 — HTTP/1.0 first draft
- Jun. 1993 — HTML (1.0 Specification)
- Oct. 1993 — URL specification
- Nov. 1993 — HTTP/1.0 second draft
- Mar. 1994 — URI in WWW
- May. 1996 — HTTP/1.0 Informational, RFC 1945
- Jan. 1997 — HTTP/1.1 Proposed Standard, RFC 2068
- Jun. 1999 — HTTP/1.1 Draft Standard, RFC 2616
- 2001 — HTTP/1.1 Formal Standard
- … ongoing — HTTP/2 Drafts and Standardization

The HTTP Protocol: Basics

- Uses TCP as transport service
- Client
  - E.g., Web Browser
  - Initiates TCP connection (creates socket) to server (on port 80)
- Server
  - Accepts TCP connection from client
- HTTP messages exchanged between client and server
  - TCP connection closed after exchange

- HTTP is "stateless"
  - Server maintains no information about past client requests

The HTTP Protocol: Basics

- Uses TCP as transport service
- Client
  - E.g., W
  - Initiat
    to ser
- Server
  - Accep
- HTTP me
  client and server
  - TCP connection closed after exchange

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 Messages

Two types: Request and response

HTTP Messages: Request

- HTTP Request:
  - ASCII (human-readable format)

```
GET /index.html HTTP/1.1\r\n
Host: www.google.com\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

HTTP Messages: Request

- HTTP Request:
  - ASCII (human-readable format)

```
GET /index.html HTTP/1.1\r\n
Host: www.google.com\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

Request line
(GET, POST, HEAD, …)

HTTP Messages: Request

- HTTP Request:
  - ASCII (human-readable format)

```
GET /index.html HTTP/1.1\r\n
Host: www.google.com\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

Header lines
(key-value pairs)

HTTP Messages: Request

- HTTP Request:
  - ASCII (human-readable format)

```
GET /index.html HTTP/1.1\r\n
Host: www.google.com\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

\r: Carriage return
\n: Line feed

HTTP Messages: Request

- HTTP Request:
  - ASCII (human-readable format)

```
GET /index.html HTTP/1.1\r\n
Host: www.google.com\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

End of header lines

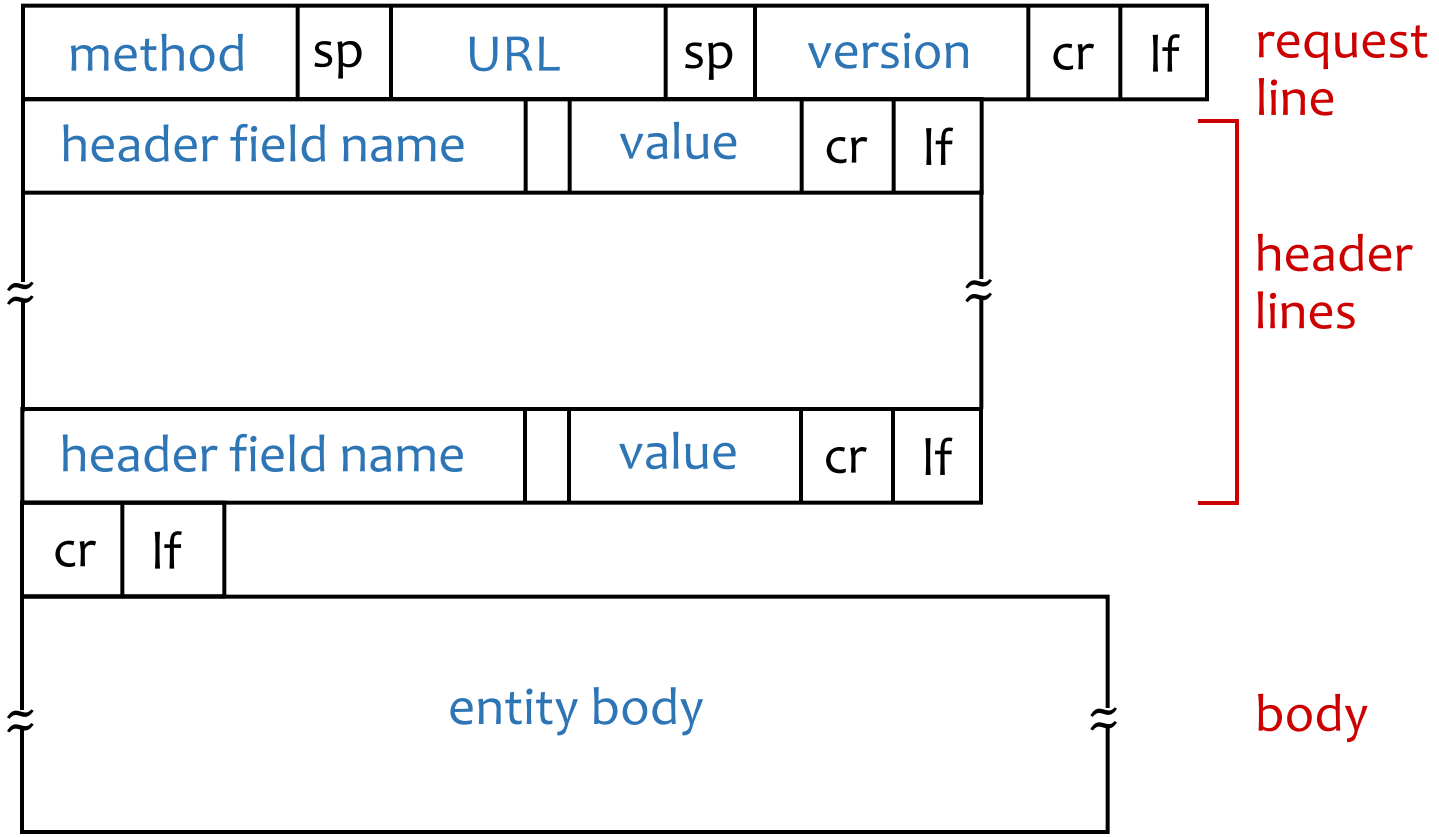
HTTP Messages: Request

- HTTP Request:
  - ASCII (human-readable format)

```
GET /index.html HTTP/1.1\r\n
Host: www.google.com\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

HTTP Messages: Format



HTTP Methods

HTTP/1.0:

- GET
- POST
- HEAD
  - Only the meta data
  - 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

HTTP Messages: Response

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
<data> ...
```

HTTP Messages: Response

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
<data> ...
```

Status line

HTTP Messages: Response

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
<data> ...
```

Status line

Protocol

HTTP Messages: Response

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
<data> ...
```

Status line

Status code

HTTP Messages: Response

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
<data> ...
```

Status line

Status phrase

HTTP Messages: Response

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
<data> ...
```

Header lines
(key-value pairs)

HTTP Messages: Response

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
<data> ...
```

Data

e.g., requested HTML file

HTTP Response: Status codes

- 200 OK
  - Request succeeded
- 301 Moved Permanently
  - Requested object moved, new location specified later in the message
- 400 Bad Request
  - Request message not understood by server
- 404 Not Found
  - Requested document not found on this server
- 505 HTTP Version Not Supported

Trying out HTTP client-side

1. Telnet to your favorite Web server:

telnet gaia.cs.umass.edu 80

Opens TCP connection to port 80 (default HTTP server port) at gaia.cs.umass.edu. anything typed in will be sent to port 80 at gaia.cs.umass.edu

2. Type in a GET HTTP request:

GET /kurose_ross/interactive/index.php HTTP/1.1
Host: gaia.cs.umass.edu

By typing this in (hit carriage return twice), you send this minimal (but complete)
GET request to HTTP server

3. Look at response message sent by HTTP server!
(or use Wireshark to look at captured HTTP request/response)

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

Non-persistent HTTP

Suppose user enters the following URL

www.someSchool.edu/someDepartment/home.index

(contains text, references to 10 JPEG images)

1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

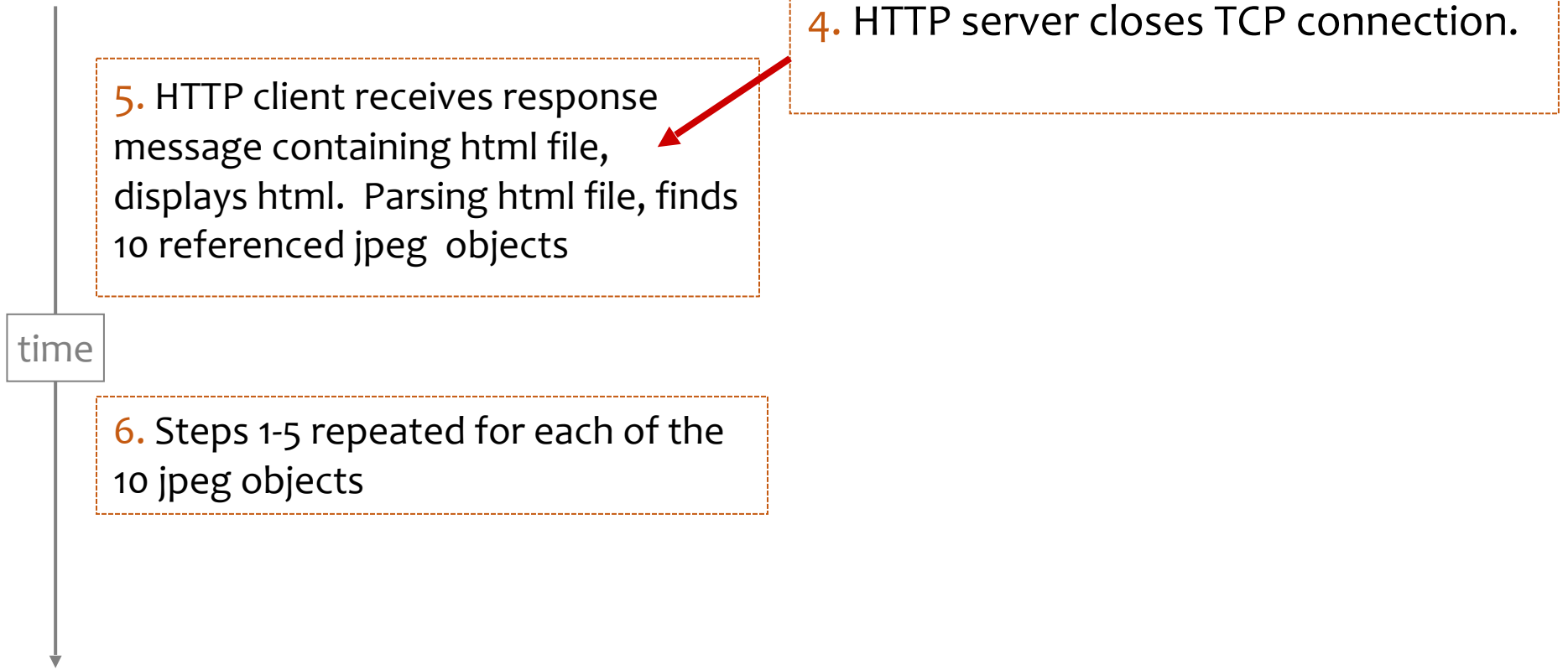
1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

time

2. HTTP client sends HTTP request message (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

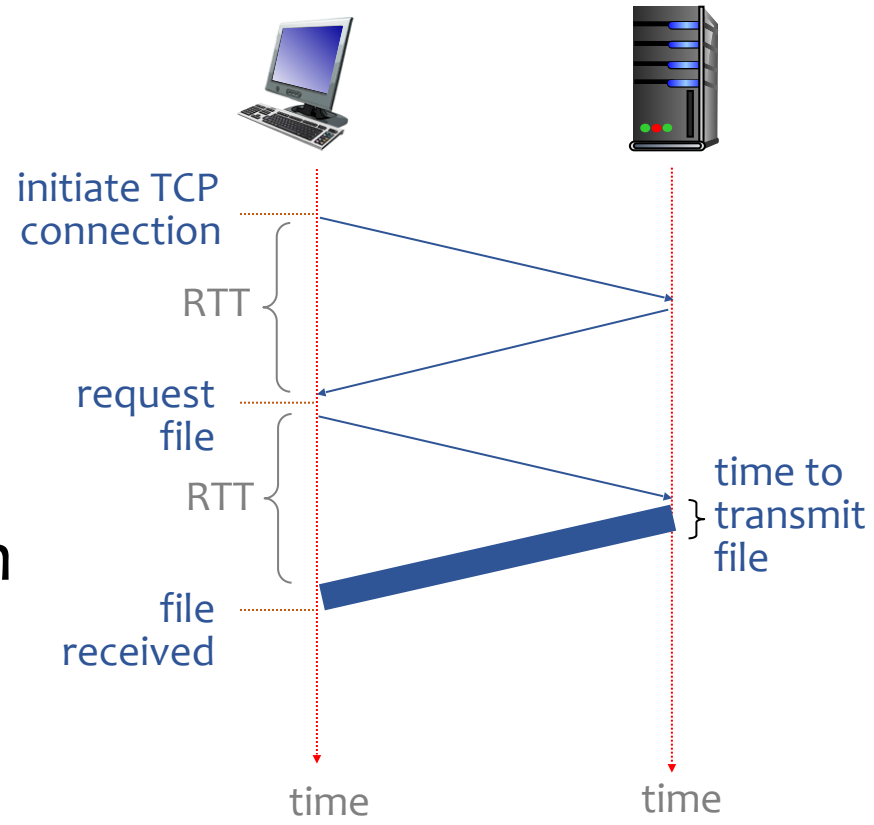
3. HTTP server receives request message, forms response message containing requested object, and sends message into its socket

Non-persistent HTTP (continued)



Non-persistent HTTP: Response time

- Round-trip time (RTT)
  - 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
- HTTP Response time:
  - 2*RTT+ file transmission time



Persistent HTTP

Non-persistent HTTP:

- 2 RTTs per object
- OS overhead for each TCP connection
- Browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP:

- Server leaves connection open after sending response
- Subsequent HTTP messages between same client/server sent over open connection
- Client sends requests as soon as possible
- ~ 1 RTT per object

Web and HTTP: An overview

- Web page consists of objects
- Objects?
  - E.g., HTML file, JPEG image, audio file, …
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL
  - E.g., https://www.mpi-inf.mpg.de/inet/