



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



The screenshot shows a web browser window at `freedom-to-tinker.com`. The main article is titled "Fast Web-based Attacks to Discover and Control IoT Devices" by Gunes Acar, Danny Y. Huang, Frank Li, Arvind Narayanan, and Nik Feamster, dated June 21, 2016. The article discusses a "DNS rebinding" attack on IoT devices. A video player shows a Google Home device with the text "Attack 3: Detect user's precise location with Google Home".

Overlaid on the right side of the browser is a code editor showing the HTML structure of the page. The code includes a DOCTYPE declaration, a meta charset, and a head section with a profile link. The body contains a wrapper div, a topnav, a header, an inner content area, and a footer. Several JavaScript files are loaded, including `https://freedom-to-tinker.com/wp-content/themes/genesis/lib/js/menu/superfish.min.js` and `https://freedom-to-tinker.com/wp-content/themes/genesis/lib/js/menu/superfish.args.min.js`.



Web and HTTP: An overview



Web page

FREEDOM TO TINKER
research and expert commentary on digital technologies in public life

Fast Web-based Attacks to Discover and Control IoT Devices

JUNE 21, 2016 BY GUNES ACAR

By Gunes Acar, Danny Y. Huang, Frank Li, Arvind Narayanan, and Nick Feamster

Two web-based attacks against IoT devices made the rounds this week. Researchers Craig Young and Brannon Dorsey showed that a well known attack technique called "DNS rebinding" can be used to control your smart thermostat, detect your home address or extract unique identifiers from your IoT devices.

For this type of attack to work, a user needs to visit a web page that contains malicious script and remain on the page while the attack proceeds. The attack simply fails if the user navigates away before the attack completes. According to the demo videos, each of these attacks takes longer than a minute to finish, assuming the attacker already knew the IP address of the targeted IoT device.

According to a study by Chartbeat, however, 55% of typical web users spent fewer than 15 seconds actively on a page. Does it mean that most web users are immune to these attacks?

In a paper to be presented at ACM SIGCOMM 2016 Workshop on IoT Security and Privacy, we developed a much faster version of this attack that takes only around ten seconds to discover and attack local IoT devices. Furthermore, our version assumes that the attacker has no prior knowledge of the targeted IoT device's IP address. Check out our demo video below.

Web-based Attacks to Discover and Control Local IoT Devices

Watch later Add to queue

Attack 3: Detect user's precise location with Google Home

Background

Many Internet-of-Things (IoT) devices are exposed to the Internet. Attackers can scan the Internet, discover vulnerable devices, and exploit vulnerable devices at scale. A notable example to this class of attacks is the 2016 DDoS campaign by the Mirai botnet, which caused massive outages for several popular online platforms.

IoT devices that are not exposed to the Internet — such as those that are connected to your home network — are not safe from attacks either. Many IoT devices, including Google Home and Chromecast, have unauthenticated web interfaces that allow companion smartphone apps, device hubs, or home automation software to discover, query, and control them. This design, which favors openness and interoperability over security, leaves the door open to web-based attacks that use the browser to circumvent firewalls and network address translation (NAT).

Freedom to Tinker is hosted by Princeton's Center for Information Technology Policy, a research center that studies digital technologies in public life. Here you'll find comment and analysis from the digital frontier, written by the Center's faculty, students, and friends.

CITP
CENTER FOR INFORMATION TECHNOLOGY POLICY

Search this website ... Search

What We Discuss

- AACS
- bitcoin
- CD Copy Protection
- Competition
- Copyright
- Cross-Border Issues
- cybersecurity policy
- DMCA
- DRM
- Education
- Events
- Facebook
- FCC
- Government
- Government transparency
- Grokster
- Case
- Humor
- Innovation
- Policy
- Law
- Managing the Internet
- Media
- Misleading Terms
- NSA
- Online Communities
- Patents
- Peer-to-Peer
- Predictions
- Princeton
- Privacy
- Publishing
- Recommended Reading
- Secrecy
- Security
- spam
- Super-DMCA
- surveillance
- Tech/Law/Policy
- Blogs
- Technology and Freedom
- transparency
- Virtual Worlds
- Voting
- Wiretapping
- WPM

Contributors

Select Author...

Archives by Month

- 2018: J F M A M J J A S O N D
- 2017: J F M A M J J A S O N D
- 2016: J F M A M J J A S O N D
- 2015: J F M A M J J A S O N D

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US" xml:lang="en-US">
<head profile="http://gmpg.org/xfn/11"></head>
<body class="post-template-default single single-post postid-14194
single-format-standard header-image header-full-width no-layout">
<div id="wrap">
<div id="topnav"></div>
<!-- end #topnav -->
<div id="header"></div>
<div id="inner"></div>
<div id="footer" class="footer"></div>
</div>
<script type="text/javascript" src="https://freedom-to-tinker.com/wp-
includes/js/hoverIntent.min.js?ver=1.8.1"></script>
<script type="text/javascript" src="https://freedom-to-tinker.com/wp-
content/themes/genesis/lib/js/menu/superfish.min.js?
ver=1.7.5"></script>
<script type="text/javascript" src="https://freedom-to-tinker.com/wp-
content/themes/genesis/lib/js/menu/superfish.args.min.js?
ver=2.3.1"></script>
<script type="text/javascript" src="https://freedom-to-tinker.com/wp-
content/themes/genesis/lib/js/menu/superfish.compat.min.js?
ver=2.3.1"></script>
<script type="text/javascript" src="https://freedom-to-tinker.com/wp-
includes/js/wp-embed.min.js?ver=4.9.8"></script>
<script type="text/javascript"></script>
</body>
</html>
```



Web and HTTP: An overview



FREEDOM TO TINKER
research and expert commentary on digital technologies in public life

Fast Web-based Attacks to Discover and Control IoT Devices

JUNE 21, 2016 BY GUNES ACAR

By Gunes Acar, Danny Y. Huang, Frank Li, Arvind Narayanan, and Nik Feamster

Two web-based attacks against IoT devices made the rounds this week. Researchers Craig Young and Brannon Dorsey showed that a well known attack technique called "DNS rebinding" can be used to control your smart thermostat, detect your home address or extract unique identifiers from your IoT devices.

For this type of attack to work, a user needs to visit a web page that contains malicious script and remain on the page while the attack proceeds. The attack simply fails if the user navigates away before the attack completes. According to the demo videos, each of these attacks takes longer than a minute to finish, assuming the attacker already knew the IP address of the targeted IoT device.

According to a study by Chartbeat, however, 55% of typical web users spent fewer than 15 seconds actively on a page. Does it mean that most web users are immune to these attacks?

In a paper to be presented at ACM SIGCOMM 2016 Workshop on IoT Security and Privacy, we developed a much faster version of this attack that takes only around ten seconds to discover and attack local IoT devices. Furthermore, our version assumes that the attacker has no prior knowledge of the targeted IoT device's IP address. Check out our demo video below.

Attack 3: Detect user's precise location with Google Home

Background

Many Internet-of-Things (IoT) devices are exposed to the Internet. Attackers can scan the Internet, discover vulnerable devices, and exploit vulnerable devices at scale. A notable example to this class of attacks is the 2016 DDoS campaign by the Mirai botnet, which caused massive outages for several popular online platforms.

IoT devices that are not exposed to the Internet — such as those that are connected to your home network — are not safe from attacks either. Many IoT devices, including Google Home and Chromecast, have unauthenticated web interfaces that allow companion smartphone apps, device hubs, or home automation software to discover, query, and control them. This design, which favors openness and interoperability over security, leaves the door open to web-based attacks that use the browser to circumvent firewalls and network address translation (NAT).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US" xml:lang="en-US">
  <head profile="http://gmpg.org/xfn/11"></head>
  <body class="post-template-default single single-post postid-14194 single-format-standard header-image header-full-width nlayout">
    <div id="wrap">
      <div id="topnav"></div>
      <!-- end #topnav -->
      <div id="header"></div>
      <div id="inner"></div>
      <div id="footer" class="footer"></div>
      </div>
      <script type="text/javascript" src="https://freedom-to-tinker.com/wp-includes/js/hoverIntent.min.js?ver=1.8.1"></script>
      <script type="text/javascript" src="https://freedom-to-tinker.com/wp-content/themes/genesis/lib/js/menu/superfish.min.js?ver=1.7.5"></script>
      <script type="text/javascript" src="https://freedom-to-tinker.com/wp-content/themes/genesis/lib/js/menu/superfish.args.min.js?ver=2.3.1"></script>
      <script type="text/javascript" src="https://freedom-to-tinker.com/wp-content/themes/genesis/lib/js/menu/superfish.compat.min.js?ver=2.3.1"></script>
      <script type="text/javascript" src="https://freedom-to-tinker.com/wp-includes/js/wp-embed.min.js?ver=4.9.8"></script>
      <script type="text/javascript"></script>
    </body>
  </html>
```

HTML
(source)



Web and HTTP: An overview



Image

The screenshot shows a web browser displaying an article from 'freedom-to-tinker.com'. The article title is 'Fast Web-based Attacks to Discover and Control IoT Devices'. The browser's developer console is open, showing the HTML structure of the page, including a header with the URL 'http://www.w3.org/1999/xhtml' and a body containing a video player and various JavaScript scripts. A yellow box highlights a circuit diagram in the article's header, and a yellow arrow points from the word 'Image' to it. Another yellow arrow points from the word 'Image' to the CITP logo in the sidebar. The sidebar contains a search bar and a list of topics such as 'Copyright', 'Security', and 'Technology and Freedom'.



Web and HTTP: An overview



Video

The screenshot shows a web browser window with the URL `freedom-to-tinker.com`. The page title is "Fast Web-based Attacks to Discover and Control IoT Devices". The article is dated "JUNE 21, 2016 BY GUNES ACAR" and is written by Gunes Acar, Danny Y. Huang, Frank Li, Arvind Narayanan, and Nik Feamster. The article text discusses a "well known attack technique called 'DNS rebinding'" and mentions a "demo video". A video player overlay is visible in the center of the page, titled "Web-based Attacks to Discover and Control Local IoT Devices" and "Attack 3: Detect user's precise location with Google Home". The browser's developer tools are open on the right, showing the HTML structure of the page, including various JavaScript files and a search bar.



Web and HTTP: An overview



The screenshot shows a web browser displaying an article from 'freedom-to-tinker.com'. The article title is 'Fast Web-based Attacks to Discover and Control IoT Devices'. The browser's developer console is open, showing the HTML structure of the page. A yellow highlight in the console points to a JavaScript script tag: `<script type="text/javascript" src="https://freedom-to-tinker.com/wp-content/themes/genesis/lib/js/menu/superfish.min.js?ver=1.7.5"></script>`. A yellow arrow points from the word 'JavaScript' on the left to this script tag. The article content includes a video player for 'Attack 3: Detect user's precise location with Google Home' and a sidebar with various topics like 'Copyright', 'DMCA DRM', and 'Security'.

JavaScript



Web and HTTP: An overview



freedom-to-tinker.com

Fast Web-based Attacks to Discover and Control IoT Devices

research and expert commentary on digital technologies in public life

JUNE 21, 2016 BY GUNES ACAR

By Gunes Acar, Danny Y. Huang, Frank Li, Arvind Narayanan, and Nik Feamster

Two web-based attacks against IoT devices made the rounds this week. Researchers Craig Young and Brannon Dorsey showed that a well known attack technique called "DNS rebinding" can be used to control your smart thermostat, detect your home address or extract unique identifiers from your IoT devices.

For this type of attack to work, a user needs to visit a web page that contains malicious script and remain on the page while the attack proceeds. The attack simply fails if the user navigates away before the attack completes. According to the demo videos, each of these attacks takes longer than a minute to finish, assuming the attacker already knew the IP address of the targeted IoT device.

According to a study by Chartbeat, however, 55% of typical web users spent fewer than 15 seconds actively on a page. Does it mean that most web users are immune to these attacks?

In a paper to be presented at ACM SIGCOMM 2016 Workshop on IoT Security and Privacy, we developed a much faster version of this attack that takes only around ten seconds to discover and attack local IoT devices. Furthermore, our version assumes that the attacker has no prior knowledge of the targeted IoT device's IP address. Check out our demo video below.

Web-based Attacks to Discover and Control Local IoT Devices

Attack 3: Detect user's precise location with Google Home

Background

Many Internet-of-Things (IoT) devices are exposed to the Internet. Attackers can scan the Internet, discover vulnerable devices, and exploit vulnerable devices at scale. A notable example to this class of attacks is the 2016 DDoS campaign by the Mirai botnet, which caused massive outages for several popular online platforms.

IoT devices that are not exposed to the Internet — such as those that are connected to your home network — are not safe from attacks either. Many IoT devices, including Google Home and Chromecast, have unauthenticated web interfaces that allow companion smartphone apps, device hubs, or home automation software to discover, query, and control them. This design, which favors openness and interoperability over security, leaves the door open to web-based attacks that use the browser to circumvent firewalls and network address translation (NAT).

Freedom to Tinker is hosted by Princeton's Center for Information Technology Policy, a research center that studies digital technologies in public life. Here you'll find comment and analysis from the digital frontier, written by the Center's faculty, students, and friends.

CITP
CENTER FOR INFORMATION TECHNOLOGY POLICY

What We Discuss

- AACS
- bitcoin
- CD Copy Protection
- copyright
- CMP Competition
- Cross-Border Issues
- cybersecurity policy
- DMCA
- DRM
- Education
- Events
- Facebook
- FCC
- Government
- Government transparency
- Grokster
- Case
- Humor
- Innovation
- Policy
- Law
- Managing the Internet
- Media
- Misleading Terms
- NSA
- Online Communities
- Patents
- Peer-to-Peer
- Predictions
- Princeton
- Privacy
- Privacy Publishing
- Recommended Reading
- Secrecy
- Security
- spam
- Super-DMCA
- surveillance
- Tech/Law/Policy
- Blogs
- Technology and Freedom
- transparency
- Virtual Worlds
- Voting
- Wiretapping
- WPM

Contributors

Select Author...

Archives by Month

- 2018: J F M A M J J A S O N D
- 2017: J F M A M J J A S O N D
- 2016: J F M A M J J A S O N D
- 2015: J F M A M J J A S O N D
- 2014: J F M A M J J A S O N D

Base HTML



Web and HTTP: An overview



The screenshot shows a web browser displaying an article from 'freedom-to-tinker.com'. The article title is 'Fast Web-based Attacks to Discover and Control IoT Devices'. The browser's developer console is open, showing the HTML source code. A yellow arrow points from the text 'URL' to a specific line in the code: `<script type="text/javascript" src="https://freedom-to-tinker.com/wp-content/themes/genesis/lib/js/menu/superfish.min.js?ver=1.7.5"></script>`. The article content includes a video player for 'Attack 3: Detect user's precise location with Google Home' and a sidebar with various categories like 'Copyright', 'DMCA DRM', and 'Security'.

URL



The HTTP Protocol



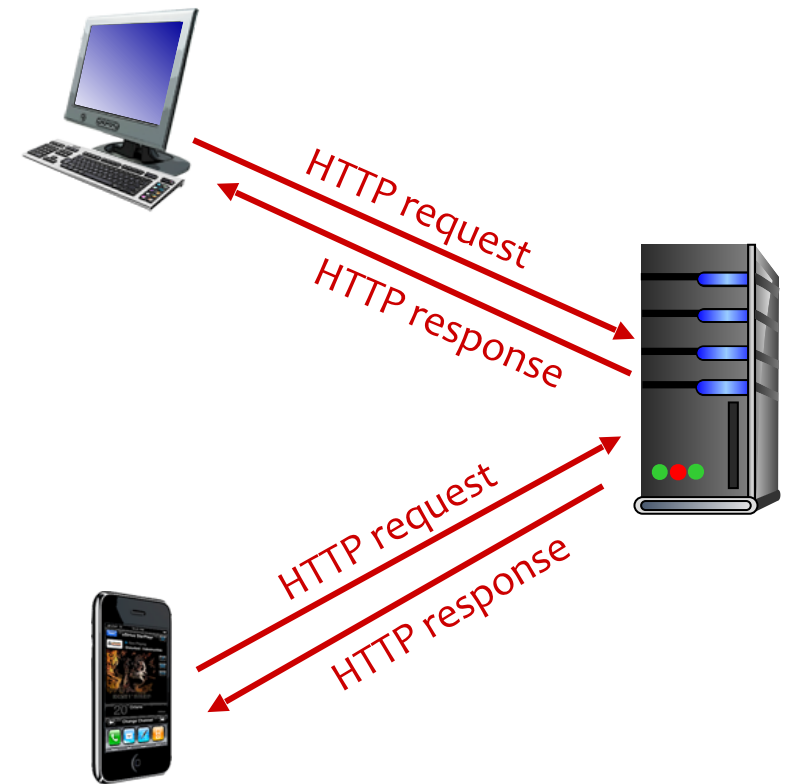
- **HTTP**: HyperText Transfer Protocol
 - Application layer protocol for the **Web**



The HTTP Protocol



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



iPhone running
Safari browser



The HTTP Protocol



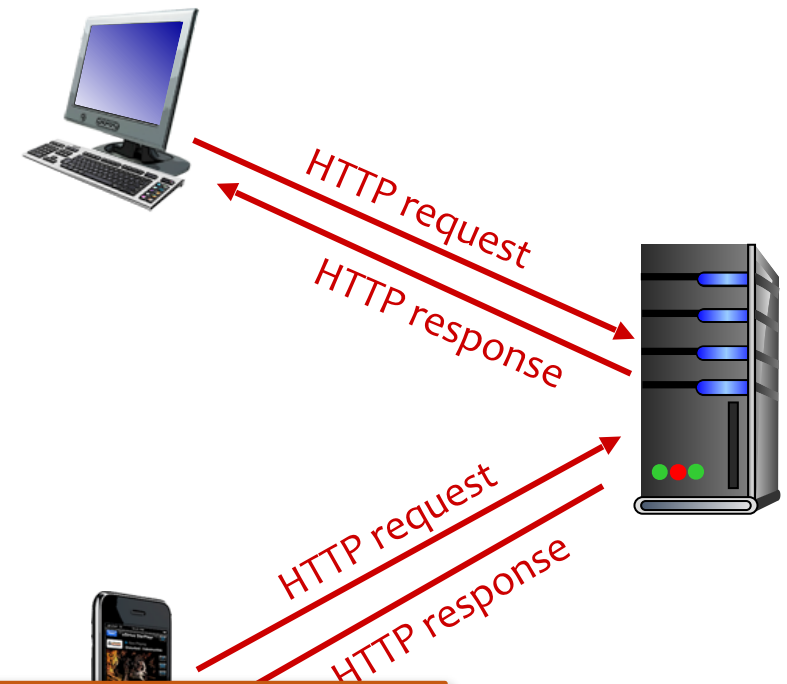
- Client-Server model

- 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 protocols

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., V
- 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\nHost: www.google.com\r\nUser-Agent: Firefox/3.6.10\r\nAccept: text/html,application/xhtml+xml\r\nAccept-Language: en-us,en;q=0.5\r\nAccept-Encoding: gzip,deflate\r\nAccept-Charset: ISO-8859-1,utf-8;q=0.7\r\nKeep-Alive: 115\r\nConnection: keep-alive\r\n\r\n
```



HTTP Messages: Request



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

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

End of header lines



HTTP Messages: Request



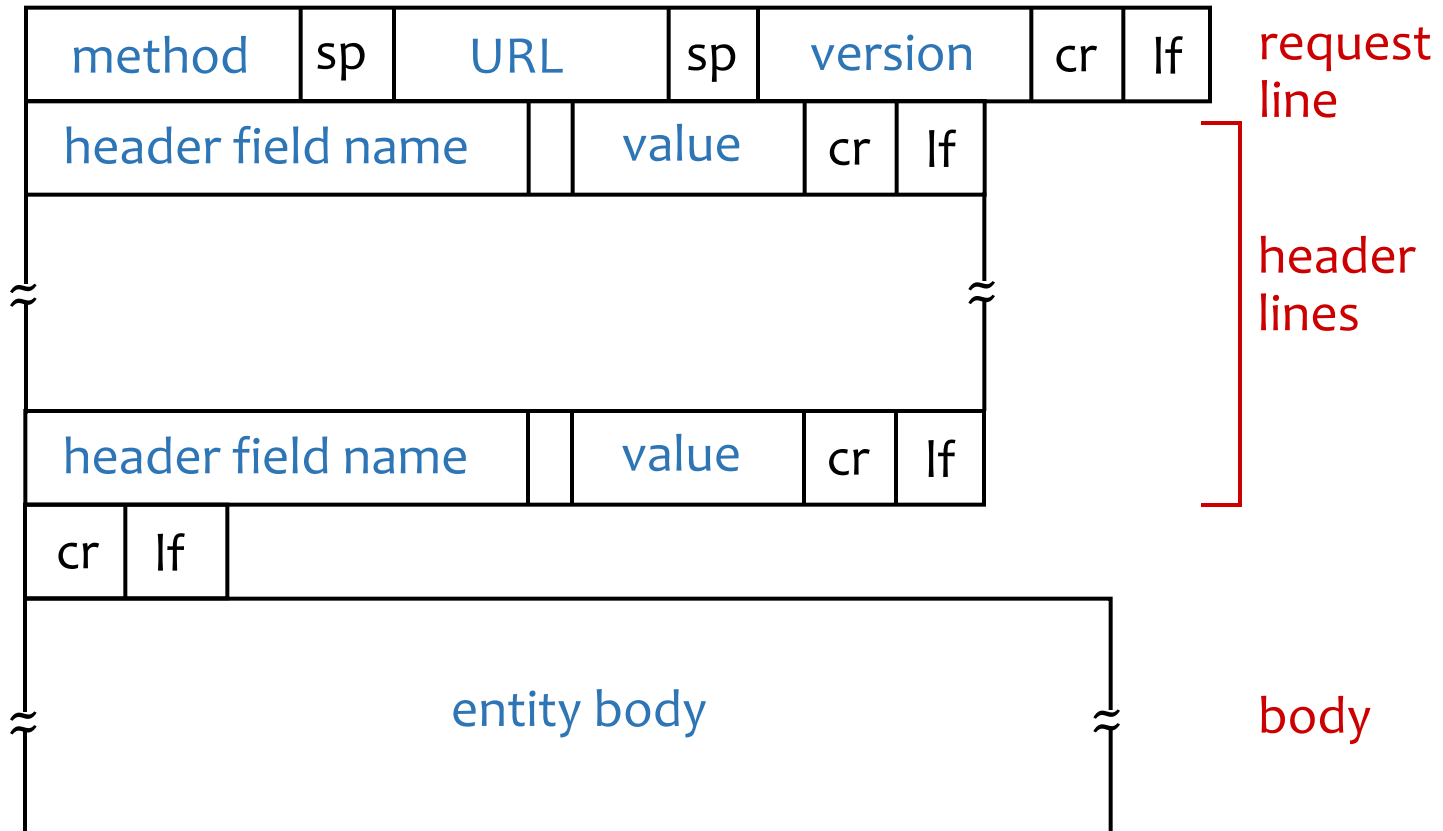
- HTTP Request:
 - ASCII (human-readable format)

```
GET /index.html HTTP/1.1\r\nHost: www.google.com\r\nUser-Agent: Firefox/3.6.10\r\nAccept: text/html,application/xhtml+xml\r\nAccept-Language: en-us,en;q=0.5\r\nAccept-Encoding: gzip,deflate\r\nAccept-Charset: ISO-8859-1,utf-8;q=0.7\r\nKeep-Alive: 115\r\nConnection: 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



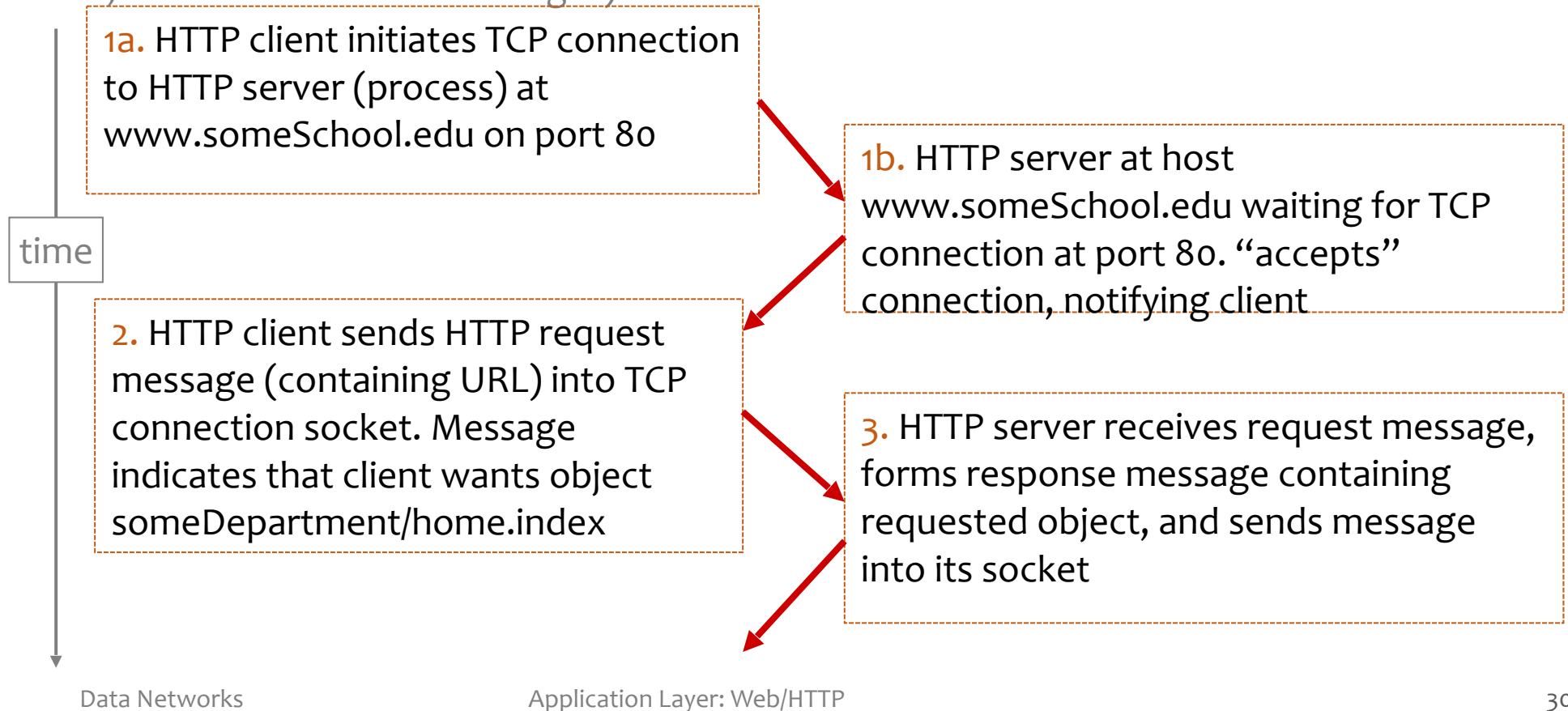
Non-persistent HTTP



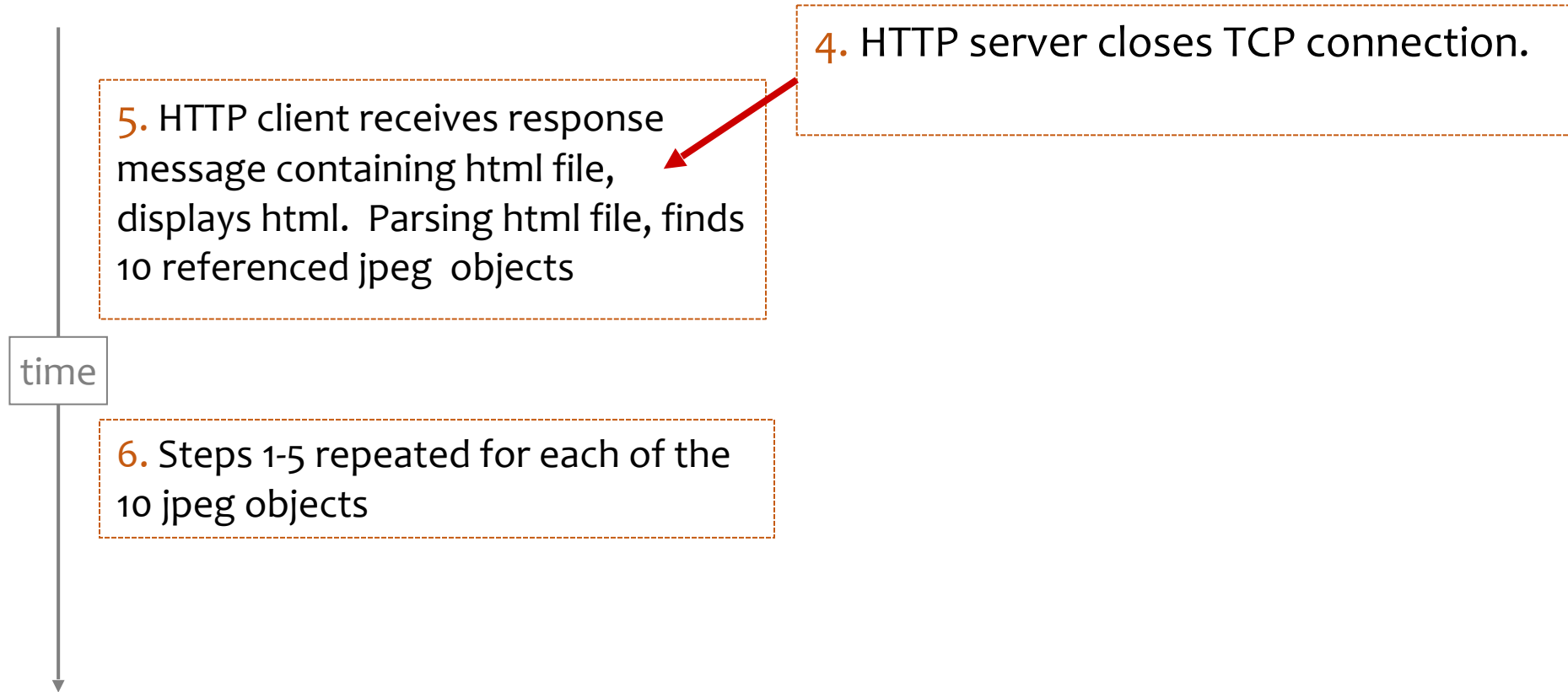
Suppose user enters the following URL

www.someSchool.edu/someDepartment/home.index

(contains text, references to 10 JPEG images)



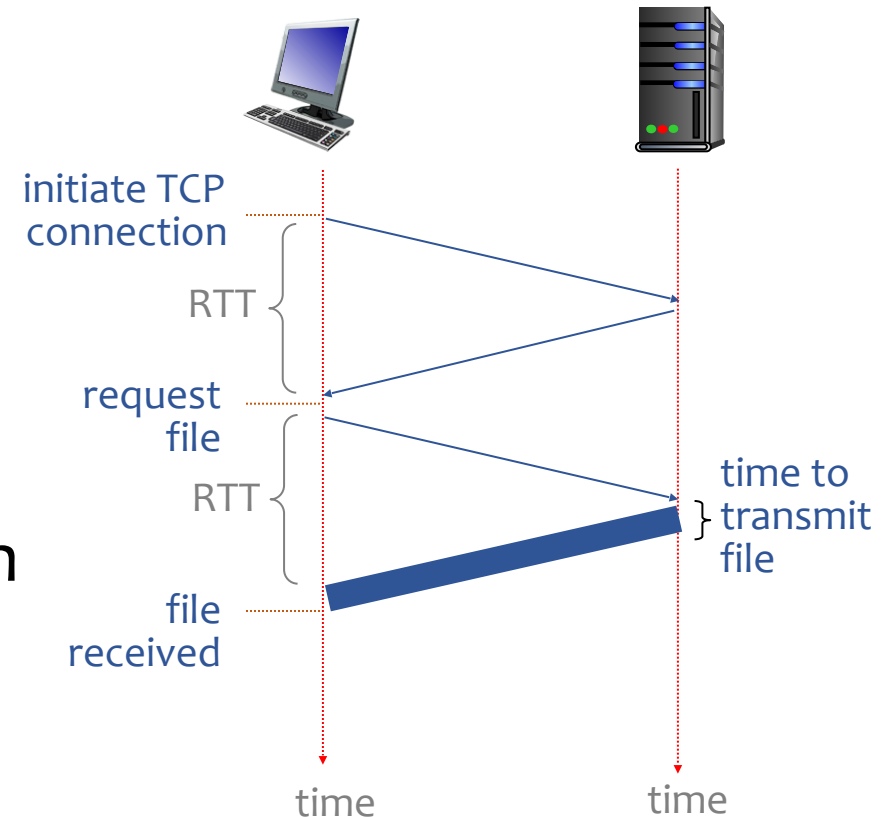
Non-persistent HTTP (continued)



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:
 - $2 * RTT + \text{file transmission time}$



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/>

