

Hypertext Transfer Protocol (HTTP): Advanced Features

Péter Jeszenszky

Faculty of Informatics, University of Debrecen

jeszenszky.peter@inf.unideb.hu

Last modified: December 2, 2024

Contents

- Cookies
- Web tracking
- Origin
- Connection management

Cookies (1)

- **Cookie:**
 - A name/value pair and associated metadata (attributes) sent by an origin server to a user agent using the Set-Cookie header field in a response.
 - An origin server can specify a scope for the cookie using the attributes.
 - In subsequent requests, the user agent returns the name/value pair to the origin server in the Cookie header field.
- The current specification:
 - Adam Barth. *RFC 6265: HTTP State Management Mechanism*. April 2011. <https://www.rfc-editor.org/rfc/rfc6265>
 - Defines the Cookie and Set-Cookie header fields.

Cookies (2)

- Uses:
 - Session management
 - Personalization
 - Web tracking (see the Referer header field)

Cookies (3)

- Information about cookies (cookie databases):
 - *CookieSearch* <https://cookiestearch.org/>
 - Example: YSC
<https://cookiestearch.org/cookies/?cookie-id=YSC>
 - *Open Cookie Database*
<https://github.com/jkwakman/Open-Cookie-Database>
 - Search form:
<https://jkwakman.github.io/Open-Cookie-Database/open-cookie-database.html>

Cookies (4)

- Példa:

- `curl --http1.1 --head https://www.w3.org/`

```
HTTP/1.1 200 OK
Date: Mon, 02 Dec 2024 15:01:45 GMT
Content-Type: text/html; charset=UTF-8
...
Set-Cookie: __cf_bm=VAdE4lj2Jfo8jwNG9tF...f29a9.L7FoS1dQ;
path=/; expires=Mon, 02-Dec-24 15:31:45 GMT; domain=.w3.org;
HttpOnly; Secure; SameSite=None
...
```

Cookies (5)

- An origin server can send multiple cookies in a response:
 - `curl --http1.1 --head https://www.youtube.com/`

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
...
Set-Cookie: YSC=XsDxGlmbJ3k; Domain=.youtube.com; Path=/; Secure;
HttpOnly; SameSite=none
Set-Cookie: __Secure-YEC=CgtQV2ZFckdHWDN5YyiBp...IBAREiEgEQ%3D%3D;
Domain=.youtube.com; Expires=Thu, 01-Jan-2026 15:32:16 GMT; Path=/;
Secure; HttpOnly; SameSite=lax
Set-Cookie: VISITOR_PRIVACY_METADATA=CgJIVRIcE...IBAREiEgEQ%3D%3D;
Domain=.youtube.com; Expires=Thu, 01-Jan-2026 15:32:17 GMT; Path=/;
Secure; HttpOnly; SameSite=none
Set-Cookie: VISITOR_INF01_LIVE=; Domain=.youtube.com;
Expires=Tue, 08-Mar-2022 15:32:17 GMT; Path=/; Secure; HttpOnly;
SameSite=none
...
```

Cookies (6)

- When the user agent receives a `Set-Cookie` header, it stores the cookie together with its attributes.
- Subsequently, when the user agent makes an HTTP request, it includes the applicable, non-expired cookies in the `Cookie` header field.
 - It includes only the name/value pairs without the attributes!
- If the user agent receives a new cookie with the same name, `Domain` attribute, and `Path` attribute as a cookie that it has already stored, the existing cookie is replaced with the new cookie.

Cookies (7)

- Instructing curl to write cookies obtained to a file:
 - `curl --http1.1
https://www.youtube.com/
-c cookies.txt -v -o /dev/null`
 - See: <https://curl.se/docs/http-cookies.html>

Cookies (8)

- Sending back cookies with curl:
 - curl --http1.1
https://www.youtube.com/
-b cookies.txt -v -o /dev/null

```
GET / HTTP/1.1
Host: www.youtube.com
User-Agent: curl/7.81.0
Accept: */*
Cookie: VISITOR_PRIVACY_METADATA=CgJIVRIcE...IBAREiEgHg%3D%3D;
__Secure-YEC=Cgt0Wjc4W...IBAREiEgHg%3D%3D; YSC=J6Cx0kFc5EE
```

Cookie Attributes (1)

- The specification defines the following attributes:
 - Expires
 - Max-Age
 - Domain
 - Path
 - Secure
 - HttpOnly

Cookie Attributes (2)

- Attributes indicating the maximum lifetime of the cookie:
 - **Expires**: specifies the date and time at which the cookie expires.
 - **Max-Age**: specifies the number of seconds until the cookie expires.
- A cookie with a **Max-Age** or **Expires** attribute is called a **persistent cookie**, because a user agent can retain the cookie over multiple sessions.
 - If a cookie has both the **Max-Age** and the **Expires** attribute, **Max-Age** has precedence over the **Expires** attribute.
- If a cookie has neither the **Max-Age** nor the **Expires** attribute, the user agent will retain it until the current session is over.
- Servers can delete cookies by sending the user agent a new cookie with an **Expires** attribute with a value in the past.

Cookie Attributes (3)

- **Domain:**

- Specifies the servers to which the cookie will be sent.
 - For example, if the value of the attribute is `example.com`, the user agent will include the cookie in the `Cookie` header when making HTTP requests to `example.com` or `www.example.com`.
- If the server omits the attribute, the user agent will return the cookie only to the origin server.
- The user agent will reject cookies unless the attribute specifies a scope for the cookie that would include the origin server.
- For security reasons, many user agents are configured to reject `Domain` attributes that correspond to “public suffixes” controlled by a public registry, such as `com`, `co.uk`, ...

- **Path:**

- Limits the scope of the cookie to a set of paths.
- If the server omits the attribute, the user agent will use the “directory” of the request URI's path component as the default value.

Cookie Attributes (4)

- **Secure:**

- Limits the scope of the cookie to secure channels.
- When a cookie has the Secure attribute, the user agent will include the cookie in an HTTP request only if the request is transmitted over a secure channel.
 - Because the cookie may contain sensitive information whose cleartext transmission poses a risk.

- **HttpOnly:**

- Limits the scope of the cookie to HTTP requests.
- Instructs the user agent not to make the cookie available to client-side APIs (e.g., JavaScript).

Cookie Management (1)

- User agents:
 - Must remove all expired cookies from the cookie store.
 - When the current session is over, must remove all non-persistent cookies from the cookie store.
 - Should provide users with a mechanism for managing the cookies stored in the cookie store.
 - For example, let users delete all cookies received during a specified period or all the cookies related to a particular domain.
 - Should provide users with a mechanism for disabling cookies.

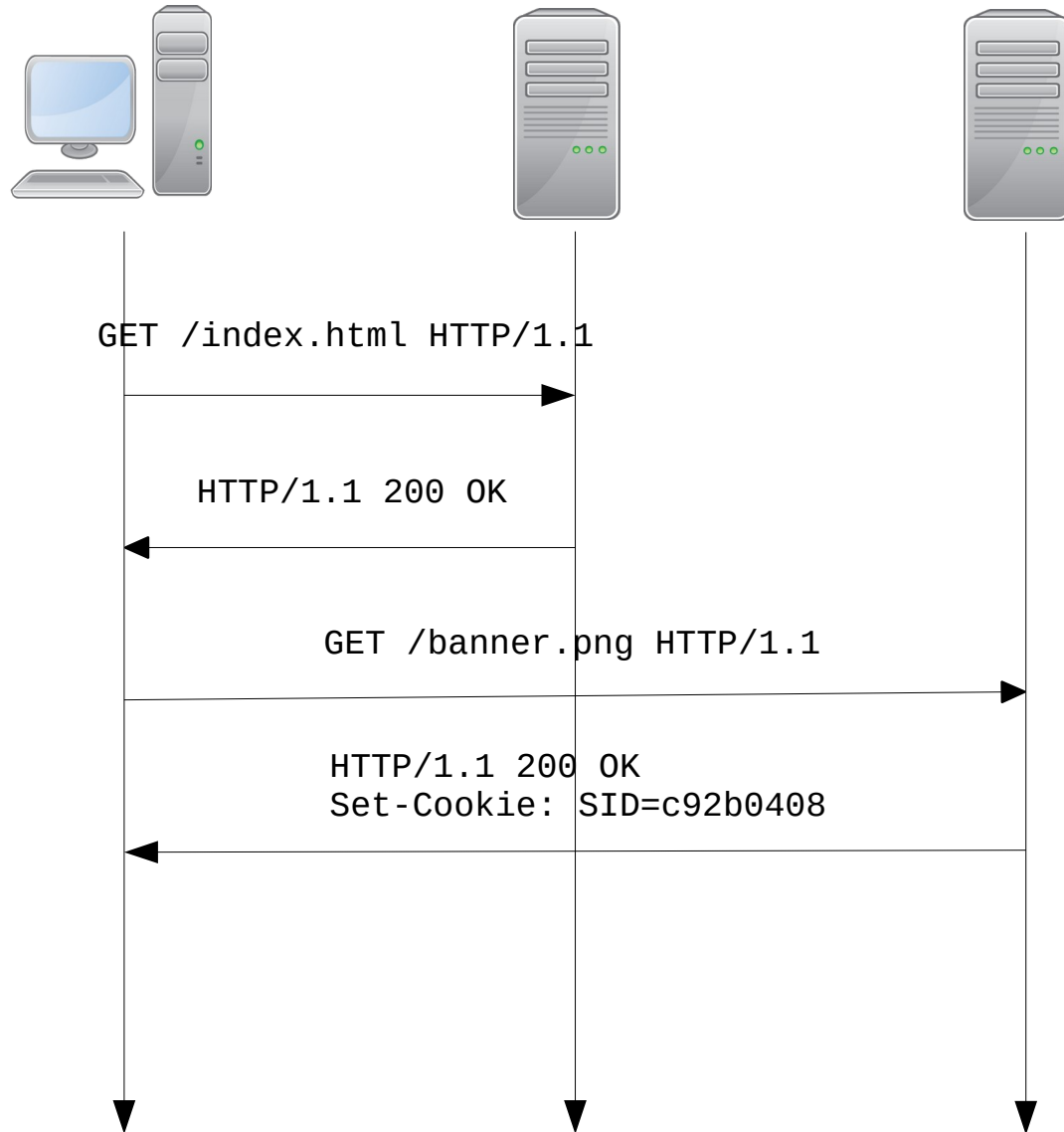
Cookie Management (2)

- User agents have limits on the number and size of cookies that they can store.
- The specification recommends the following minimum capabilities:
 - At least 4096 bytes per cookie.
 - At least 50 cookies per domain.
 - At least 3000 cookies total.

Cookies: Privacy and Security Concerns (1)

- Cookies are often criticized for letting servers track users.
- Particularly problematic are so-called **third-party cookies**.
 - In rendering an HTML document, a user agent often requests resources from other servers.
 - These third-party servers can use cookies to track the user even if the user never visits the server directly.
 - Third-party cookies are also known as cross-site cookies.

Cookies: Privacy and Security Concerns (2)



Cookies: Privacy and Security Concerns (3)

- Unless sent over a secure channel (such as TLS), the information in the Cookie and Set-Cookie headers is transmitted in cleartext.
 - All sensitive information conveyed in these headers is exposed to an eavesdropper and a malicious intermediary could alter it.
- Servers should encrypt and sign the contents of cookies when transmitting them to the user agent (even when sending the cookies over a secure channel).
- When using cookies over a secure channel, servers should set the Secure attribute for every cookie.

Cookies: Privacy and Security Concerns (4)

- The current EU regulations:
 - **2002/58/EC**: *Directive on privacy and electronic communications* (12 July 2002) (“ePrivacy Directive”)
<https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32002L0058>
 - **2009/136/EC**: *Directive 2009/136/EC of the European Parliament and of the Council* (25 November 2009)
<https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32009L0136>
 - See the modification of the paragraph 3 of Article 5 of Directive 2002/58/EC.
 - States that storing cookies on the user's computer requires informed consent from the user.
 - Article 29 Data Protection Working Party: *Opinion 04/2012 on Cookie Consent Exemption* (7 June 2012)
https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2012/wp194_en.pdf

Cookies: Privacy and Security Concerns (5)

- See also:
 - *Cookies, the GDPR, and the ePrivacy Directive*
<https://gdpr.eu/cookies/>
 - *Third-party cookies (MDN)*
https://developer.mozilla.org/en-US/docs/Web/Privacy/Third-party_cookies

Web Tracking (1)

- A possible definition:
 - Tracking is the collection of data regarding a particular user's activity across multiple distinct contexts and the retention, use, or sharing of data derived from that activity outside the context in which it occurred.
 - A context is a set of resources that are controlled by the same party or jointly controlled by a set of parties.
- Source:
<https://www.w3.org/TR/tracking-dnt/#dfn-tracking>

Web Tracking (2)

- Can be based on the following:
 - IP address
 - Cookies
 - The ETag header field
 - Device fingerprint (operating system, screen resolution, installed fonts, ...)
 - See:
 - *Am I Unique?* <https://amiunique.org/>
 - *Cover Your Tracks* <https://coveryourtracks.eff.org/>
 - ...

Web Tracking (3)

- Example for browser fingerprinting:
 - FingerprintJS (written in: TypeScript, license: Business Source License 1.1) <https://github.com/fingerprintjs/fingerprintjs>
 - A source-available (however, not open source), client-side, browser fingerprinting library that queries browser attributes and computes a hashed visitor identifier from them.
 - Unlike cookies and local storage, a fingerprint stays the same in incognito/private mode and even when browser data is purged.
 - Demonstration:
 - Visit the following page in Google Chrome, then also open it in an incognito window: <https://fingerprintjs.github.io/fingerprintjs>
 - Now repeat the experiment in Firefox.

Web Tracking (4)

- Statistics about trackers:
 - *Ghostery WhoTracks.me*
<https://www.ghostery.com/whotracksme>
 - *Trackers Rank*
<https://www.ghostery.com/whotracksme/trackers>
 - *Ghostery Tracker Database*
<https://github.com/ghostery/trackerdb>
- Further information:
 - *BrowserLeaks.com* <https://www.browserleaks.com/>

Origin (1)

- Origins are fundamental to security on the web.
- Two actors on the web that share an origin are assumed to trust each other.
- Actors with differing origins are considered potentially hostile versus each other, and should be isolated from each other.
- See: *HTML Living Standard – Loading web pages – Supporting concepts – Origins*
<https://html.spec.whatwg.org/multipage/browsers.html#origin>

Origin (2)

- The origin of a resource is defined by the scheme, hostname, and port component of the URI used to access it.
- See: *Origin*
<https://developer.mozilla.org/en-US/docs/Glossary/Origin>

Origin (3)

- For example, these are same origin:
 - <http://example.com/index.html>
 - <http://example.com/docs/>
 - <http://example.com:80/>
- For example, these are not same origin:
 - <http://example.com/>
 - <http://www.example.com/>
 - <http://example.com:8080/>
 - <https://example.com/>

Origin (4)

- Same origin policy:
 - A critical security mechanism that restricts how a document or script loaded by one origin can interact with a resource from another origin.
 - In general, it forbids code from one origin to access code from a different origin.
 - See:
https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

Origin (5)

- Cross-Origin Resource Sharing (CORS):
 - A HTTP header-based mechanism that allows to control cross-origin access.
 - It allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources.
 - It is based on the use of the `Origin` and `Access-Control-Allow-Origin` header fields.
 - See:
<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>

The Referer Header Field (1)

- Allows the user agent to specify a URI reference for the resource from which the target URI was obtained.
 - See:
<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referer>
- Example:
 - Referer: `https://www.w3.org/`
- It has the potential to reveal information about the browsing history of the user, which is a privacy concern.

The Referer Header Field (2)

- A user agent should not send a Referer header field if the referring resource was accessed with a secure protocol and the request-target has an origin differing from that of the referring resource unless the referring resource explicitly allows Referer to be sent.
- A user agent must not send a Referer header field in an unsecured HTTP request if the referring resource was accessed with a secure protocol.

Controlling the Use of the Referer Header Field (1)

- The mechanism provided by HTML:
 - *HTML Living Standard – Link type "noreferrer"*
<https://html.spec.whatwg.org/multipage/semantics.html#link-type-noreferrer>
 - The `rel="noreferrer"` attribute of the `a` element.
 - Example:
 - `Click here`
 - Browser support: <https://caniuse.com/rel-noreferrer>

Controlling the Use of the Referrer Header Field (2)

- *Referrer Policy* (W3C Candidate Recommendation, 26 January 2017) <https://www.w3.org/TR/referrer-policy/>
 - Provides a mechanism by which document authors can define a policy for clients on the sending of the Referrer header field.
 - Example:
 - Referrer-Policy: no-referrer
 - `<meta name="referrer" content="no-referrer"/>`
 - `Click here`
 - Browser support: <https://caniuse.com/referrer-policy>

Controlling the Use of the Referer Header Field (3)

- *Referrer Policy* (continued):
 - Available options:
<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy>
 - The default is `strict-origin-when-cross-origin`, that is also used by Firefox and Chrome, currently.
 - See:
 - *Firefox 87 trims HTTP Referrers by default to protect user privacy*
<https://blog.mozilla.org/security/2021/03/22/firefox-87-trims-http-referrers-by-default-to-protect-user-privacy/>
 - *A new default Referrer-Policy for Chrome – strict-origin-when-cross-origin*
<https://developer.chrome.com/blog/referrer-policy-new-chrome-default/>
 - Further information: *Referer and Referrer-Policy best practices* <https://web.dev/articles/referrer-best-practices>

Protection Against Tracking (1)

- Disabling the sending of the Referer header field:
 - **Firefox**: see the `network.http.sendRefererHeader` option (`about:config`)
<https://wiki.mozilla.org/Security/Referrer>
 - **Chromium, Google Chrome**: not possible (?)
 - **Opera**: not possible (?)
 - **Chromium-based Microsoft Edge**: not possible (?)

Protection Against Tracking (2)

- Disabling the acceptance of third-party cookies:
 - In major web browsers (i.e., Chromium, Google Chrome, Chromium-based Microsoft Edge, Firefox, Opera) users can choose to block third-party cookies in private browsing/incognito mode or block third-party cookies in general.
 - Since blocking third-party cookies can break sites, users can make exceptions by allowing the use of third-party cookies for specific sites.

Protection Against Tracking (3)

- Disabling the acceptance of third-party cookies:
 - **Firefox:** see the `network.cookie.cookieBehavior` option (`about:config`)
 - See also: *Third-party cookies and Firefox tracking protection – Disable third-party cookies*
https://support.mozilla.org/en-US/kb/third-party-cookies-firefox-tracking-protection#w_disable-third-party-cookies
 - **Chromium, Google Chrome:** see the *Block third-party cookies* option (`chrome://settings/privacy`)
 - **Opera:** see the *Block third-party cookies* option (`opera://settings/cookies`)
 - **Chromium-based Microsoft Edge:** see the *Block third-party cookies* option (`edge://settings/content/cookies`)

Protection Against Tracking (4)

- Private browsing/incognito mode:
 - Many modern web browsers provide a privacy feature called private browsing.
 - In private browsing mode when the current session is over browsing information (such as browsing history, cookies, and cached content) is automatically erased.

Protection Against Tracking (5)

- Private browsing/incognito mode: (continued)
 - **Firefox:**
 - *Private Browsing – Use Firefox without saving history*
<https://support.mozilla.org/en-US/kb/private-browsing-use-firefox-without-history>
 - **Chromium, Google Chrome:**
 - *How private browsing works* <https://support.google.com/chrome/?p=incognito>
 - **Opera:**
 - *Opera Help – Security and privacy – Private window*
<https://help.opera.com/en/latest/security-and-privacy/#privateWindow>
 - **Chromium-based Microsoft Edge:**
 - *Browse InPrivate in Microsoft Edge*
<https://support.microsoft.com/en-us/microsoft-edge/browse-inprivate-in-microsoft-edge-cd2c9a48-0bc4-b98e-5e46-ac40c84e27e2>

Protection Against Tracking (6)

- Built-in tracking protection:
 - **Firefox:**
 - *Enhanced Tracking Protection in Firefox for desktop*
<https://support.mozilla.org/en-US/kb/enhanced-tracking-protection-firefox-desktop>
 - See: `about:protections`
 - **Chromium-based Microsoft Edge:**
 - *Tracking Prevention in Microsoft Edge*
<https://learn.microsoft.com/en-us/microsoft-edge/web-platform/tracking-prevention>
 - See: `edge://settings/privacy`
 - **Safari:**
 - *Tracking Prevention in WebKit* <https://webkit.org/tracking-prevention/>

Protection Against Tracking (7)

- Complex solutions:
 - Adblock Plus (license: GPLv3) <https://adblockplus.org/>
<https://gitlab.com/eyeo/extensions/extensions>
 - Supported web browsers: Chrome, Firefox, Microsoft Edge, Opera, Safari
 - See the EasyPrivacy filter <https://easylist.to/>
 - Ghostery Browser Extension (license: non-free/Mozilla Public License 2.0)
<https://www.ghostery.com/ghostery-ad-blocker>
<https://github.com/ghostery/ghostery-extension>
 - Supported web browsers: Chrome, Firefox, Microsoft Edge, Opera, Safari
 - Privacy Badger (license: GPLv3) <https://privacybadger.org/>
<https://github.com/EFForg/privacybadger>
 - Supported web browsers: Chrome, Firefox, Microsoft Edge, Opera
 - uBlock Origin (license: GPLv3) <https://github.com/gorhill/uBlock>
 - Supported web browsers: Chromium, Firefox

Protection Against Tracking (8)

- Privacy-focused web browsers:
 - Brave: a free and open source web browser based on Chromium that focuses on privacy.
 - Developer: Brave Software, Inc.
 - Website: <https://brave.com/>
 - Repository: <https://github.com/brave/brave-browser>
 - Written in: C++, Swift, TypeScript
 - License: Mozilla Public License 2.0
 - Platform: Android, iOS, Linux, macOS, Windows

Protection Against Tracking (9)

- Google Chrome:
 - Google planned to restrict the use of third-party cookies by default for 1% of Chrome users to facilitate testing, and then ramping up to 100% of users from Q3 2024.
 - See:
 - Chris Mills. *Saying goodbye to third-party cookies in 2024*. December 8, 2023. <https://developer.mozilla.org/en-US/blog/goodbye-third-party-cookies/>
 - *Third-party cookies restricted by default for 1% of Chrome users*. January 4, 2024. <https://developers.google.com/privacy-sandbox/blog/cookie-countdown-2024jan>
 - Later Google decided not to do so, instead, to follow a different approach.
 - See: Anthony Chavez. *A new path for Privacy Sandbox on the web*. July 22, 2024. <https://privacysandbox.com/news/privacy-sandbox-update/>

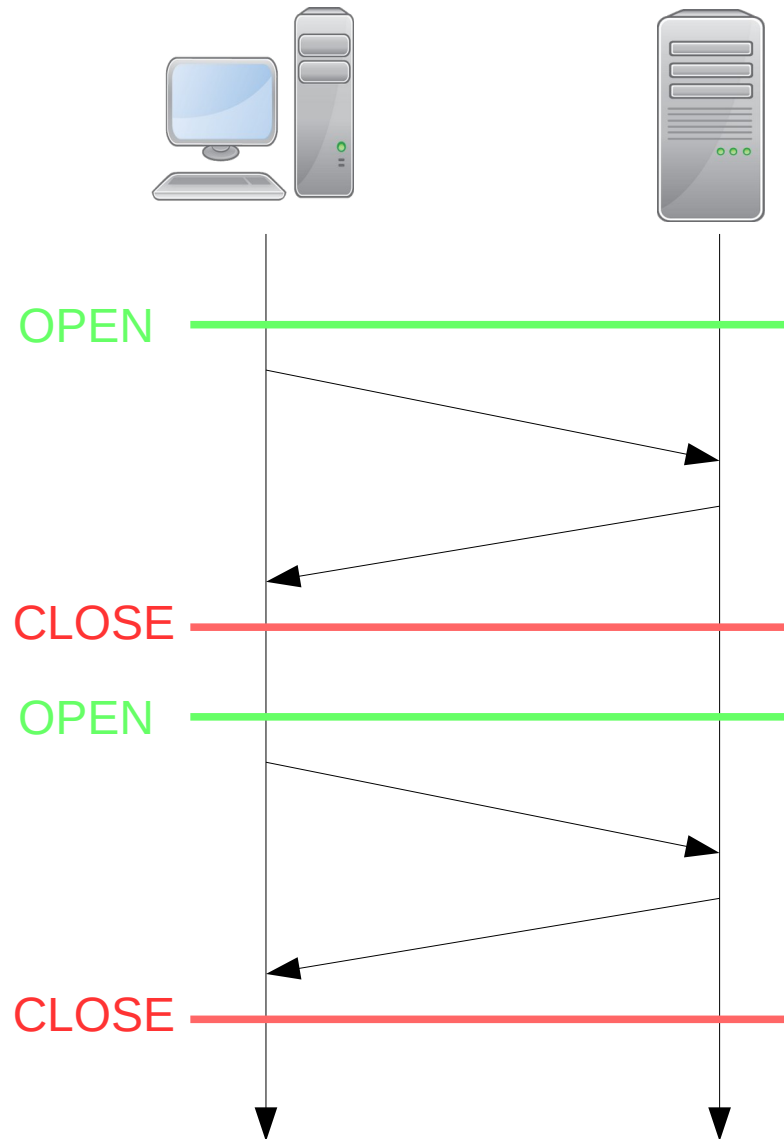
Protection Against Tracking (10)

- Further information:
 - *PrivacyTools* <https://www.privacytools.io/>

Comparison of Web Browsers

- *Can I use...* – *Browser comparison*
<https://caniuse.com/ciu/comparison>
<https://github.com/Fyrd/caniuse>
- *Comparison of Web Browsers*
https://eylenburg.github.io/browser_comparison.htm
- *PrivacyTests.org* <https://privacytests.org/>

Connection Management in HTTP/1.0



Persistent Connections (1)

- Were introduced in HTTP/1.1.
- Allow multiple requests and responses to be transmitted over a single TCP connection.
- HTTP/1.1 defaults to the use of persistent connections.

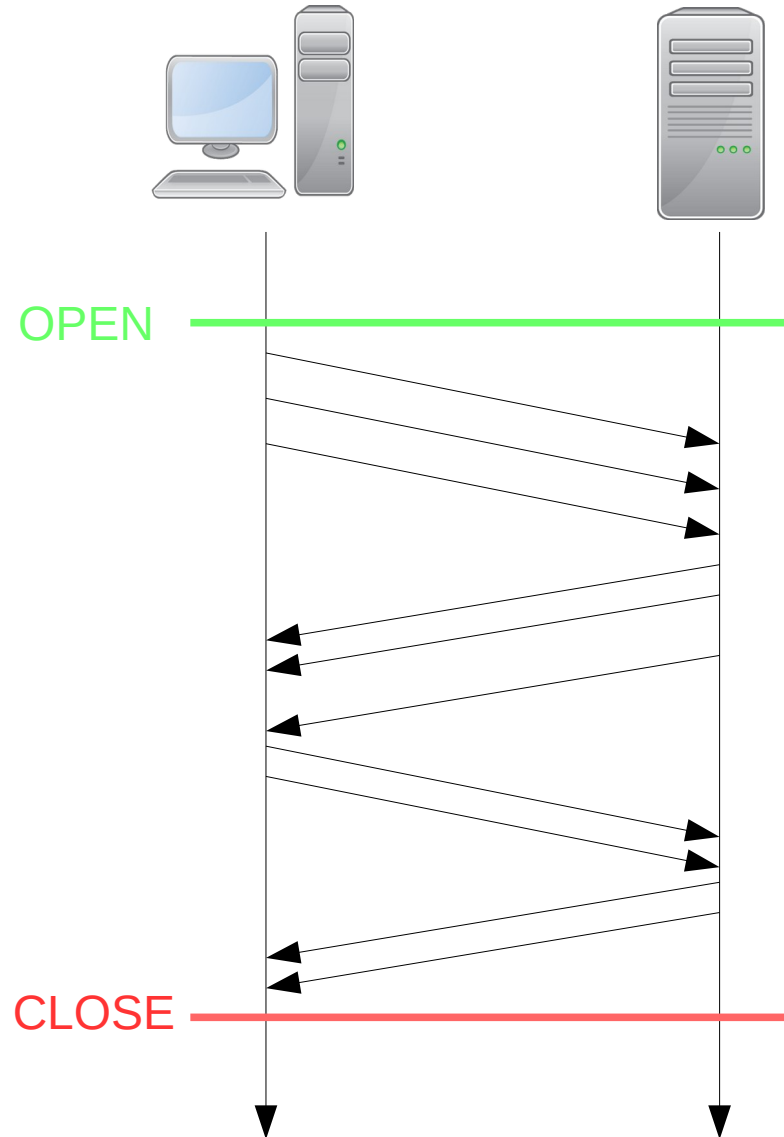
Connection Management

- The `Connection` header field allows the sender to indicate desired control options for the current connection.
 - The field value is a list of case-insensitive options.

Connection Management: Pipelining (1)

- A client that supports persistent connections may “pipeline” its requests, i.e., send multiple requests without waiting for each response.
- A server may process a sequence of pipelined requests in parallel if they all have safe methods, but it must send the corresponding responses in the same order that the requests were received.

Connection Management: Pipelining (2)



Connection Management: Pipelining (3)

- A client that pipelines requests should retry unanswered requests if the connection closes before it receives all of the corresponding responses.
- Idempotent requests can be automatically retried after a connection failure.

Connection Management: Pipelining (4)

- Browser support:
 - **Firefox**: supported, but is disabled by default.
 - See the `network.http.pipelining` option (`about:config`).
 - **Chromium, Google Chrome**: the option to enable pipelining has been removed from Chrome.
 - See:
<https://www.chromium.org/developers/design-documents/network-stack/http-pipelining>
 - **Opera**: not supported
 - **Chromium-based Microsoft Edge**: not supported

Connection Management: Pipelining (5)

- Theoretically, performance could also be improved if multiple HTTP requests were to be packed into the same TCP message.
- Has been superseded by a better mechanism, multiplexing, which is used by HTTP/2.
- See: *Connection management in HTTP/1.x – HTTP pipelining*
https://developer.mozilla.org/en-US/docs/Web/HTTP/Connection_management_in_HTTP_1.x#http_pipelining

Connection Management: Connection Closing

- Explicit connection closing:
 - The `Connection` header field provides a `close` connection option for a sender to signal that the connection will be closed after the current request/response is complete.
 - Can be used in both requests and responses.
 - Example:
 - `Connection: close`
- Connection timeout:
 - Servers usually have some timeout value beyond which they will no longer maintain an inactive connection.

Connection Management: Concurrent Connections (1)

- Most servers are designed to be able to maintain thousands of concurrent connections.
- Most clients maintain multiple connections in parallel, including more than one connection per server.
 - Multiple connections are typically used to avoid the **head-of-line blocking** problem.

Connection Management: Concurrent Connections (2)

- Previous revisions of HTTP specified a limit for the number of simultaneous open connections that a client can maintain to a given server.
 - RFC 2616: the maximum number of connections is 2.
- This was found to be impractical for many applications, thus, HTTP/1.1 does not mandate a particular limit, but instead, encourages clients to be conservative when opening multiple connections.
 - Each connection consumes server resources, and a server might deny requests when a client opens an excessive number of connections.

Connection Management: Concurrent Connections (3)

- Limits for the number of connections:
 - **Firefox:** see the following options (`about:config`):
 - `network.http.max-connections` (default value: 900)
 - `network.http.max-persistent-connections-per-proxy` (default value: 32)
 - `network.http.max-persistent-connections-per-server` (default value: 6)
 - **Chromium, Google Chrome:** uses the default values of the above Firefox options as hard-coded settings.
 - See:
<https://www.chromium.org/developers/design-documents/network-stack/#connection-management>
 - **Chromium-based Microsoft Edge:**
<https://docs.microsoft.com/en-us/microsoft-edge/devtools-guide-chromium/network/issues#queued-or-stalled-requests>

Further Recommended Reading

- *MDN Web Docs – HTTP*
<https://developer.mozilla.org/docs/Web/HTTP>
- Ilya Grigorik, *High Performance Browser Networking*. O'Reilly, 2013. <https://hpbn.co/>