WIKIPEDIA

Comet (programming)

Comet is a web application model in which a long-held HTTPS request allows a web server to push data to a browser, without the browser explicitly requesting it.[1][2] *Comet* is an umbrella term, encompassing multiple techniques for achieving this interaction. All these methods rely on features included by default in browsers, such as JavaScript, rather than on non-default plugins. The Comet approach differs from the original model of the web, in which a browser requests a complete web page at a time.[3]

The use of Comet techniques in web development predates the use of the word *Comet* as a neologism for the collective techniques. Comet is known by several other names, including *Ajax Push*,[4][5] *Reverse Ajax*,[6] *Two-way-web*,[7] *HTTP Streaming*,[7] and *HTTP server push*[8] among others.[9] The term *Comet* is not an acronym, but was coined by Alex Russell in his 2006 blog post *Comet: Low Latency Data for the Browser*.[10]

In recent years, the standardisation and widespread support of WebSocket and Server-sent events has rendered the Comet model obsolete.

Contents

History
- Early Java applets
- The first browser-to-browser communication framework
- First Comet applications

Implementations
- Streaming
  - Hidden iframe
  - XMLHttpRequest
- Ajax with long polling
  - XMLHttpRequest long polling
  - Script tag long polling

Alternatives

See also

References

External links

History

Early Java applets

The ability to embed Java applets into browsers (starting with Netscape Navigator 2.0 in March 1996[11]) made two-way sustained communications possible, using a raw TCP socket[12] to communicate between the browser and the server. This socket can remain open as long as the browser is at the document hosting the applet. Event notifications can be sent in any format – text or binary – and decoded by the applet.

The first browser-to-browser communication framework

The very first application using browser-to-browser communications was Tango Interactive,[13] implemented in 1996–98 at the Northeast Parallel Architectures Center (NPAC (http://surface.syr.edu/npac/)) at Syracuse University using DARPA funding. TANGO architecture has been patented by Syracuse University.[14] TANGO framework has been extensively used as a distance education tool.[15] The framework has been commercialized by CollabWorx (http://www.collabworx.com) and used in a dozen or so Command&Control and Training applications in the United States Department of Defense.

First Comet applications

The first set of Comet implementations date back to 2000,[16] with the Pushlets, Lightstreamer, and KnowNow projects. Pushlets, a framework created by Just van den Broecke, was one of the first[17] open source implementations. Pushlets were based on server-side Java servlets, and a client-side JavaScript library. Bang Networks – a Silicon Valley start-up backed by Netscape co-founder Marc Andreessen – had a lavishly-financed attempt to create a real-time push standard for the entire web.[18]

In April 2001, Chip Morningstar began developing a Java-based (J2SE) web server which used two HTTP sockets to keep open two communications channels between the custom HTTP server he designed and a client designed by Douglas Crockford; a functioning demo system existed as of June 2001. The server and client used a messaging format that the founders of State Software, Inc. assented to coin as JSON following Crockford's suggestion. The entire system, the client libraries, the messaging format known as JSON and the server, became the State Application Framework, parts of which were sold and used by Sun Microsystems, Amazon.com, EDS and Volkswagen.

In March 2006, software engineer Alex Russell coined the term Comet in a post on his personal blog.[19] The new term was a play on Ajax (Ajax and Comet both being common household cleaners in the USA).[20][21][22]

In 2006, some applications exposed those techniques to a wider audience: Meebo’s multi-protocol web-based chat application enabled users to connect to AOL, Yahoo, and Microsoft chat platforms through the browser; Google added web-based chat to Gmail; JotSpot, a startup since acquired by Google, built Comet-based real-time collaborative document editing.[23] New Comet variants were created, such as the Java-based ICEfaces JSF framework (although they prefer the term "*Ajax Push*"[5]). Others that had previously used Java-applet based transports switched instead to pure-JavaScript implementations.[24]

Implementations

Comet applications attempt to eliminate the limitations of the page-by-page web model and traditional polling by offering two-way sustained interaction, using a persistent or long-lasting HTTP connection between the server and the client. Since browsers and proxies are not designed with server events in mind, several techniques to achieve this have been developed, each with different benefits and drawbacks. The biggest hurdle is the HTTP 1.1 specification, which states "this specification... encourages clients to be conservative when opening multiple connections".[25] Therefore, holding one connection open for real-time events has a negative impact on browser usability: the browser may be blocked from sending a new request while waiting for the results of a previous request, e.g., a series of images. This can be worked around by creating a distinct hostname for real-time information, which is an alias for the same physical server. This strategy is an application of domain sharding.

Specific methods of implementing Comet fall into two major categories: streaming and long polling.

Streaming

An application using streaming Comet opens a single persistent connection from the client browser to the server for all Comet events. These events are incrementally handled and interpreted on the client side every time the server sends a new event, with neither side closing the connection.[3]

Specific techniques for accomplishing streaming Comet include the following:

Hidden iframe

A basic technique for dynamic web application is to use a hidden iframe HTML element (an *inline frame*, which allows a website to embed one HTML document inside another). This invisible iframe is sent as a chunked block, which implicitly declares it as infinitely long (sometimes called "forever frame"). As events occur, the iframe is gradually filled with `script` tags, containing JavaScript to be executed in the browser. Because browsers render HTML pages incrementally, each `script` tag is executed as it is received. Some browsers require a specific minimum document size before parsing and execution is started, which can be obtained by initially sending 1–2 kB of padding spaces.[26]

One benefit of the iframes method is that it works in every common browser. Two downsides of this technique are the lack of a reliable error handling method, and the impossibility of tracking the state of the request calling process.[26]

XMLHttpRequest

The XMLHttpRequest (XHR) object, a tool used by Ajax applications for browser–server communication, can also be pressed into service for server–browser Comet messaging by generating a custom data format for an XHR response, and parsing out each event using browser-side JavaScript; relying only on the browser firing the `onreadystatechange` callback each time it receives new data.

Ajax with long polling

None of the above streaming transports work across all modern browsers without negative side-effects. This forces Comet developers to implement several complex streaming transports, switching between them depending on the browser. Consequently, many Comet applications use long polling, which is easier to implement on the browser side, and works, at minimum, in every browser that supports XHR. As the name suggests, long polling requires the client to poll the server for an event (or set of events). The browser makes an Ajax-style request to the server, which is kept open until the server has new data to send to the browser, which is sent to the browser in a complete response. The browser initiates a new long polling request in order to obtain subsequent events. IETF RFC 6202 "Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP" (https://tools.ietf.org/html/rfc6202) compares long polling and HTTP streaming. Specific technologies for accomplishing long-polling include the following:

XMLHttpRequest long polling

For the most part, XMLHttpRequest long polling works like any standard use of XHR. The browser makes an asynchronous request of the server, which may wait for data to be available before responding. The response can contain encoded data (typically XML or JSON) or Javascript to be executed by the client. At the end of the processing of the response, the browser creates and sends another XHR, to await the next event. Thus the browser always keeps a request outstanding with the server, to be answered as each event occurs.

Script tag long polling

While any Comet transport can be made to work across subdomains, none of the above transports can be used across different second-level domains (SLDs), due to browser security policies designed to prevent cross-site scripting attacks.[27] That is, if the main web page is served from one SLD, and the Comet server is located at another SLD (which does not have cross-origin resource sharing enabled), Comet events cannot be used to modify the HTML and DOM of the main page, using those transports. This problem can be sidestepped by creating a proxy server in front of one or both sources, making them appear to originate from the same domain. However, this is often undesirable for complexity or performance reasons.

Unlike iframes or XMLHttpRequest objects, `script` tags can be pointed at any URI, and JavaScript code in the response will be executed in the current HTML document. This creates a potential security risk for both servers involved, though the risk to the data provider (in our case, the Comet server) can be avoided using JSONP.

A long-polling Comet transport can be created by dynamically creating `script` elements, and setting their source to the location of the Comet server, which then sends back JavaScript (or JSONP) with some event as its payload. Each time the script request is completed, the browser opens a new one, just as in the XHR long polling case. This method has the advantage of being cross-browser while still allowing cross-domain implementations.[27]

Alternatives

Browser-native technologies are inherent in the term Comet. Attempts to improve non-polling HTTP communication have come from multiple sides:

- The HTML 5 draft specification produced by the Web Hypertext Application Technology Working Group (WHATWG) specifies so called server-sent events,[28] which defines a new JavaScript interface `EventSource` and a new MIME type `text/event-stream`. All major browsers except Microsoft Internet Explorer include this technology.
- The HTML 5 WebSocket API working draft specifies a method for creating a persistent connection with a server and receiving messages via an `onmessage` callback.[29]
- The Bayeux protocol by the Dojo Foundation. It leaves browser-specific transports in place, and defines a higher-level protocol for communication between browser and server, with the aim of allowing re-use of client-side JavaScript code with multiple Comet servers, and allowing the same Comet server to communicate with multiple client-side JavaScript implementations. Bayeux is based on a publish/subscribe model, so servers supporting Bayeux have publish/subscribe built-in.[30]
- The BOSH protocol by the XMPP standards foundation. It emulates a bidirectional stream between browser and server by using two synchronous HTTP connections.
- The JSONRequest object, proposed by Douglas Crockford, would be an alternative to the XHR object.[31]
- Use of plugins, such as Java applets or the proprietary Adobe Flash (using RTMP protocol for data streaming to Flash applications). These have the advantage of working identically across all browsers with the appropriate plugin installed and need not rely on HTTP connections, but the disadvantage of requiring the plugin to be installed
- Google announced[32] a new Channel API for Google App Engine,[33] implementing a Comet-like API with the help of a client JavaScript library on the browser. This API has been deprecated. [34]

See also

- Push technology
- Pull technology

References

1. Krill, Paul (September 24, 2007). "AJAX alliance recognizes mashups" (http://www.infoworld.com/d/developer-world/ajax-alliance-recognizes-mashups-559). InfoWorld. Retrieved 2010-10-20.
2. Crane, Dave; McCarthy, Phil (October 13, 2008). *Comet and Reverse Ajax: The Next-Generation Ajax 2.0*. Apress. ISBN 978-1-59059-998-3.
3. Gravelle, Rob. "Comet Programming: Using Ajax to Simulate Server Push" (https://web.archive.org/web/20101018055530/http://www.webreference.com/programming/javascript/rg28/). Webreference.com. Archived from the original (http://www.webreference.com/programming/javascript/rg28/) on 2010-10-18. Retrieved 2010-10-20.
4. Egloff, Andreas (2007-05-05). *Ajax Push (a.k.a. Comet) with Java Business Integration (JBI)* (http://developers.sun.com/learning/javaoneonline/j1sessn.jsp?sessn=TS-8434&yr=2007&track=7) (Speech). JavaOne 2007, San Francisco, California: Sun Microsystems, Inc. Retrieved 2008-06-10.
5. "Ajax Push" (http://www.icesoft.org/java/projects/ICEfaces/ajax-push.jsf). ICEfaces.org. Retrieved 2014-10-23.
6. Crane, Dave; McCarthy, Phil (July 2008). *Comet and Reverse Ajax: The Next Generation Ajax 2.0*. Apress. ISBN 1-59059-998-5.
7. Mahemoff, Michael (June 2006). "Web Remoting". *Ajax Design Patterns* (https://archive.org/details/ajaxdesignpatter00mahe/page/19). O'Reilly Media. pp. 19, 85 (https://archive.org/details/ajaxdesignpatter00mahe/page/19). ISBN 0-596-10180-5.
8. Double, Chris (2005-11-05). "More on Ajax and server push" (http://www.bluishcoder.co.nz/2005/11/more-on-ajax-and-server-push.html). *Different ways of doing server push*. Retrieved 2008-05-05.
9. Nesbitt, Bryce (2005-11-01). "The Slow Load Technique/Reverse AJAX" (https://web.archive.org/web/20060208041559/http://www.obviously.com/tech_tips/slow_load_technique). *Simulating Server Push in a Standard Web Browser*. Archived from the original (http://www.obviously.com/tech_tips/slow_load_technique) on 2006-02-08. Retrieved 2008-05-06.
10. Russell, Alex (2006-03-04). "Comet: Low Latency Data for the Browser" (http://infrequently.org/2006/03/comet-low-latency-data-for-the-browser/). Retrieved 2014-11-02.
11. "Netscape.com" (https://web.archive.org/web/19961115203505/http://www27.netscape.com/comprod/products/navigator/version_2.0/index.html). Archived from the original on November 15, 1996. Retrieved 2017-08-16.
12. "java.net.Socket (Java 2 Platform SE v1.4.2)" (http://java.sun.com/j2se/1.4.2/docs/api/java/net/Socket.html) Archived (https://web.archive.org/web/20090519063251/http://java.sun.com/j2se/1.4.2/docs/api/java/net/Socket.html) May 19, 2009, at the Wayback Machine
13. Beca, Lukasz (1997). "TANGO - a Collaborative Environment for the World-Wide Web" (http://surface.syr.edu/cgi/viewcontent.cgi?article=1076&context=npac). *Syracuse University SURFACE*. Northeast Parallel Architecture Center, College of Engineering and Computer Science. Retrieved 27 February 2016.
14. Podgorny, Marek; Beca, Lukasz; Cheng, Gang; Fox, Geoffrey C.; Jurga, Tomasz; Olszewski, Konrad; Sokolowski, Piotr; Walczak, Krzysztof; PL (June 20, 2000), *United States Patent: 6078948 - Platform-independent collaboration backbone and framework for forming virtual communities having virtual rooms with collaborative sessions* (http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%252Fnetahtml%252FPTO%252Fsearch-bool.html&r=14&f=G&l=50&co1=AND&d=PTXT&s1=Podgorny.INNM.&OS=IN/Podgorny&RS=IN/Podgorny), retrieved 2016-02-27

15. Baer, Troy (1999). "Experiences with Using TANGO Interactive in a Distributed Workshop" (https://www.dsc.soic.indiana.edu/sites/default/files/tr_9921.pdf) (PDF). *CEWES Major Shared Resource Center*. CEWES MSRC/PET TR/99-21. Retrieved 27 February 2016.
16. "CometDaily: Comet and Push Technology" (https://web.archive.org/web/20071113174053/http://cometdaily.com/2007/10/19/comet-and-push-technology/). Archived from the original (http://cometdaily.com/2007/10/19/comet-and-push-technology/) on 2007-11-13. Retrieved 2007-12-15.
17. Just van den Broecke (1 March 2000). "Pushlets: Send events from servlets to DHTML client browsers (http://www.javaworld.com/article/2076063/java-web-development/pushlets--send-events-from-servlets-to-dhtml-client-browsers.html)". JavaWorld. Retrieved 1 August 2014.
18. Borland, John (2001-04-01). "Will the "refresh" button become obsolete?" (http://news.cnet.com/2100-1023-255088.html). CNET Networks. Retrieved 2008-07-22.
19. Alex Russell (3 March 2006). "Comet: Low Latency Data for the Browser (http://alex.dojotoolkit.org/?p=545) Archived (https://web.archive.org/web/20080812034003/http://alex.dojotoolkit.org/?p=545) 2008-08-12 at the Wayback Machine". Alex Russell's blog. Retrieved 29 November 2007.
20. K. Taft, Darryl (2006-05-12). "Microsoft Scrubs Comet from AJAX Tool Set" (http://www.eweek.com/c/a/Application-Development/Microsoft-Scrubs-Comet-from-AJAX-Tool-Set/). eWEEK.com. Retrieved 2008-07-21.
21. Orbited: Enabling Comet for the Masses: OSCON 2008 - O'Reilly Conferences, July 21 - 25, 2008, Portland, Oregon (http://en.oreilly.com/oscon2008/public/schedule/detail/3048)
22. Enterprise Comet & Web 2.0 Live Presentation (http://www.web2journal.com/read/457966.htm) Archived (https://web.archive.org/web/20080520222527/http://www.web2journal.com/read/457966.htm) 2008-05-20 at the Wayback Machine
23. Dion Almaer (29 September 2005). "Jotspot Live: Live, group note-taking (http://ajaxian.com/archives/jotspot-live-live-group-note-taking)" (interview with Abe Fettig). Ajaxian. Retrieved 15 December 2007.
Matt Marshall (15 December 2006). "Renkoo launches event service — in time to schedule holiday cocktails (https://venturebeat.com/2006/12/15/renkoo-launches-just-in-time-to-schedule-holiday-cocktails/)". Venture Beat. Retrieved 15 December 2007.
24. Clint Boulton (27 December 2005). "Startups Board the AJAX Bandwagon (http://www.devxnews.com/article.php/3573506/)". DevX News. Retrieved 18 February 2008.
25. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, section 6.4 (http://tools.ietf.org/html/rfc7230#section-6.4). IETF. Retrieved 2014-07-29
26. Holdener III, Anthony T. (January 2008). "Page Layout with Frames that Aren't". *Ajax: The Definitive Guide*. O'Reilly Media. p. 320. ISBN 0-596-52838-8.
27. Flanagan, David (2006-08-17). "13.8.4 Cross-Site Scripting". *JavaScript the Definitive Guide* (https://archive.org/details/javascript00libg_297). The Definitive Guide. O'Reilly Media. p. 994 (https://archive.org/details/javascript00libg_297/page/n981). ISBN 0-596-10199-6.
28. Ian Hickson, ed. (2007-10-27). "6.2 Server-sent DOM events" (https://www.whatwg.org/specs/web-apps/2007-10-26/multipage/section-server-sent-events.html#server-sent-events). *HTML 5 - Call For Comments*. WHATWG. Retrieved 2008-10-07.
29. Hickson, Ian (2009-04-23). "The WebSocket API" (http://www.w3.org/TR/websockets/). W3C. Retrieved 2009-07-21.
30. Alex Russell; et al. (2007). "Bayeux Protocol - Bayeux 1.0draft1" (http://svn.cometd.org/trunk/bayeux/bayeux.html). Dojo Foundation. Retrieved 2007-12-14.
31. Crockford, Douglas (2006-04-17). "JSONRequest Duplex" (http://www.json.org/JSONRequest.html). *An alternative to XMLHttpRequest for long lasting server initiated push of data*. Retrieved 2008-05-05.
32. App, The. (2010-12-02) Google App Engine Blog: Happy Holidays from the App Engine team - 1.4.0 SDK released (http://googleappengine.blogspot.com/2010/12/happy-holidays-from-app-engine-team-140.html). Googleappengine.blogspot.com. Retrieved on 2014-04-12.
33. Paul, Ryan. (2010-12-06) App Engine gets Streaming API and longer background tasks (https://arstechnica.com/web/news/2010/12/app-engine-gets-streaming-api-and-longer-background-tasks.ars). Ars Technica. Retrieved on 2014-04-12.

34. "Package com.google.appengine.api.channel" (https://cloud.google.com/appengine/docs/standard/java/javadoc/com/google/appengine/api/channel/package-summary). Google. 2019-11-16. Retrieved 2020-04-30. "This API has been deprecated."

External links

- "Comet Daily" (https://web.archive.org/web/20080104091304/http://cometdaily.com/). Archived from the original on 2008-01-04. "Comet Daily provides information about Comet techniques."
- Comparison of several comet server implementations (https://web.archive.org/web/20080330033520/http://cometdaily.com/maturity.html)