Will HTTP 2.0 Revolutionise Business and the Web?

Guenter I. Klas
Sept 02, 2012

HTTP 2.0? Ever heard before? It's a new version of the HTTP protocol being worked on by Internet engineers, specifically now with major input in 2012 from Google and Microsoft. Every time you are browsing the Web, the chosen Internet address likely shows http:// in the address bar. Most likely this is HTTP version 1.1 in use. What sort of revolution is to be expected from the new version? When? What's the innovation that's supposed to come along with it? We shed some light on this.

Below we consider a range of aspects, from technical to business. For the ultimate brief summary scroll to **The Short Of It**.

What's HTTP? The hypertext transfer protocol (HTTP) is the main application layer protocol used for shipping web pages from web servers to browsers on desktops and smartphones alike. Every time you browse to a website, HTTP is used.

HTTP 1.0/1.1 design goals. HTTP was designed for a browser to fetch whole web pages and parts of pages from web servers across the Internet. The main version is 1.1, completed in 1999 and widely in use [9]. At the time of HTTP's design, the web was largely a static web. It was exciting enough to request different web pages from a web browser. To this end, an HTTP 1.1 transaction consists of a simple request (for a web resource) and a response (from a web server or a proxy or cache that may sit somewhere between the web server and the web browser). In its basic form, each request from a client to a server sets up a TCP connection (is different with *persistent HTTP connections*). In the olden days, this was perfect for the web. A series of HTTP transactions when reading BBC news would have been: I go to the home page of BBC (request 1 from the browser, response 1 from the server). I click on business news (request 2, response 2), then I move on to the weather forecast – rain again (request 3, response 3) and so on.

The web has grown out of its shoes. HTTP doesn't work that perfectly anymore. In the meantime web pages are dynamic, interactive with a variety of content types. Browsing the Internet is not anymore about loading some static web pages. Think about Google Maps, YouTube and other sites. Web pages have been enhanced by styling and code which make them dynamic and interactive. This entails not only download of the HTML mark-up of a page, but in addition download of style sheets and Javascript files, of video streams, audio and the like. Today's web has largely become too big for the original HTTP for mainly two reasons. First the new type and structure of data. Second the amount of requests to servers given the proliferation of smartphones. As a result the web is far from instant and servers struggle. HTTP 2.0 promises a solution

---

The Short Of It

HTTP 2.0: A new version of HTTP being worked on in IETF.

Key goal: Shall improve speed of web page delivery, reduce latency for HTTP transfers, increase security, reduce waste of bandwidth.

Key feature: Prioritising and multiplexing of multiple HTTP requests (for a web page's subresources) into a single TCP connection between client and server. Compression of HTTP headers. Compression of HTTP body (the payload) optional.

First published on www.yucianga.info at http://yucianga.info/?p=598 on Sept 2, 2012.

Key ideas: Stem from Google and Microsoft [7]. Google's proposal called SPDY, Microsoft's proposal called Microsoft Speed + Mobility.

Key issues: Debates whether some of the suggested HTTP 2.0 features are useful across the spectrum of services that today make use of HTTP and whether more should be done apart from improving speed to make HTTP work better for e.g. mobile/cellular communication and M2M services. Question whether encryption using TLS shall be mandatory.

When ready: The new IETF standard is currently forecast to be completed by end of 2014.

When in use: Non-standard protocols that may be regarded partial, proprietary pre-standards solutions are already deployed by a few players including Google and Amazon. Most prominent: Google's SPDY protocol.

Potential benefit: Say 20 – 60% speed improvement for loading web pages in browsers, depending on server configurations and optimisations. Some of today's highly optimised sites may achieve a speed improvement at the lower end.

Key beneficiaries: Web consumers benefit from better response time, better user experience. Internet application providers in the B2C area benefit as HTTP 2.0 will be an enabler for greater user experience and lower costs on the server side. It allows those companies to sustain current business models, which would otherwise be threatened by deteriorating web performance. If HTTP 2.0 includes new designs that help to relax radio network resource issues and battery lifetime problems encountered from the use of HTTP 1.1 by smartphones, then both mobile phone users as well as mobile/cellular network operators will benefit.

Other impacts: The impact on web service and application developers is zero or minimal, as HTTP 2.0 wont' get rid of the basic HTTP 1.1 constructs. The impact is bigger on vendors who produce or use HTTP protocol stacks. This includes of course browser vendors, and providers of and open source projects for web servers. The impact on mobile network operators needs to be further assessed. This includes impact on radio resources, ability to inspect HTTP traffic etc. It could be positive (fewer set-up and tear-down of TCP connections), or it could be minimal. More in-depth analysis is advisable as long as HTTP 2.0 is still on the white board in IETF.

Key issues with HTTP 1.1. When users of the Web want to enjoy a website, or web-based services like maps, email, social networks, news and others, their experience with HTTP 1.1-based browsers is often impacted by a long page load time. "The Internet is pretty slow today" is the common statement. Even when having laid fibre to the home, or having broadband access with say 70 Mb/s download speed, websites appear to respond slowly. This is due to the large amount of HTTP 1.1 requests which need to be serially and in sequence sent to a web server to request the necessary parts and elements belonging to a web page.

More technically, latency is caused e.g. by repeated TCP connection setups and setup time, waiting time as requests are sent serially, waiting for the server to respond, servers being slowed down when using up to 6 TCP connections to connect with a browser, servers needing to acquire content from helper domains, as a single website has a limitation on possible open TCP connections. Furthermore, concurrent TCP requests generate extra load on Web intermediary boxes, like proxies, firewalls, and NAT which slows them down.

The current aspirations for HTTP 2.0. A key goal is to increase speed and improve network utilization.

The current contenders and concepts under discussion. Several parties have submitted proposals to the Internet Engineering Taskforce IETF. Amongst others, Google has submitted a specification called SPDY (SPeedDY, [2], [3]) whose early days go back to 2009. Microsoft has contributed a draft called Microsoft Speed + Mobility [1]. The IETF working group plans to take SDPY as the basis and we expect them to modify and enhance this proposal with ingredients and recipes from other proposals, including the one from Microsoft.

Major new improvement ideas. Several suggestions have been tabled addressing ways to improve over HTTP 1.1. This includes:

- Multiplexing of multiple HTTP requests concurrently over a single TCP connection.
- Transmitting HTTP messages as independent, bidirectional logical streams, from client to server (HTTP requests) and from server to client (HTTP responses). Means, a client can send multiple HTTP requests to the same server over a single stream without the need to wait for a response after every request.
- Clients (e.g. web browsers) can prioritise amongst the concurrent HTTP requests to the server (e.g. get important text with higher priority than images). Each stream may be marked with a priority by the client. This helps against network congestion.
- Compressing the HTTP request and response headers which typically transmit metadata from sender to receiver. This reduces the amount of HTTP protocol overhead transmitted and saves bandwidth. HTTP 1.1. headers would be stuffed into a header section of a stream frame. That frame's header section is then compressed.
- Removing the need to send some HTTP headers from client to server, when those headers are static and don't change. This saves bandwidth.
- Increasing security by transmitting the multiplexed HTTP request/response stream over a secure connection (TLS/TCP).
- Permitting servers to push data streams to clients without asking them, by anticipating what clients will need. Currently very contentious under debate

Some finer points: HTTP streams can be sent concurrently and can be interleaved. HTTP streams can be independent from each other or a stream can be associated with another stream. This helps browsers to properly assemble e.g. response data. HTTP endpoints can set the configuration of the streaming/framing layer. The server can e.g. tell the client (browser) about the bandwidth the server estimates it has available to fill the stream on its end. This helps the client to prepare. The streaming layer would further use a flow control mechanism.

Let's send two HTTP requests in one go without waiting for the server to respond. On the streaming layer, a client would open a stream via a SYN_STREAM frame, then send a HEADERS frame, a DATA frame, a second HEADERS frame, a DATA frame, and then (half)close the bidirectional stream via a flag. An HTTP 1.1 header including the HTTP command like GET or PUSH is mapped into a HEADERS frame, an HTTP 1.1 body is mapped into DATA frame(s).

A glance at the protocol stack. What is going to change? As of today it's fair to say that the new proposals made for HTTP 2.0 actually don't do away with HTTP 1.1. This would neither make sense nor would it be feasible. A large degree of compatibility with the existing HTTP 1.1 is required because HTTP 1.1 is so pervasive. Users e.g. of the Facebook site may only think of their web browser and the Facebook web server. However, this is only a tiny tip of an iceberg. Between user device (desktop, smartphone) and webserver can sit numerous other Internet devices, like caches, proxies, network address translation boxes, load balancers etc. Many of these intermediary boxes operate on the basis of HTTP 1.1. A radical cleanout of HTTP 1.1 and replacement by something totally new is de facto impossible. Thus, HTTP 2.0 is not a replacement of HTTP 1.1, but an enhancement/extension of HTTP 1.1. Its most significant extension is a new multiplexing layer.

Today's typical HTTP 1.1 protocol stack is as follows:

- HTTP 1.1: on the application layer

- TCP/IP: on transport layer and on routing layer

The HTTP 2.0 proposals suggest the following enhanced protocol stack:

- HTTP 2.0 (command layer): largely compatible with HTTP 1.1

- HTTP 2.0 (multiplexing layer): is a session layer. Here proposals like SPDY (framing) fit in.

- TLS (optional): as presentation layer. Handling secure transfer, encryption etc.

- TCP/IP: on transport layer and routing layer.

Some misunderstandings related to HTTP 2.0. The biggest potential misunderstanding is that HTTP 2.0 replaces HTTP 1.1, which is largely not the case. Today's widely used HTTP message types like GET will likely stay in place. The semantics of the well-known HTTP 1.1 headers will be preserved. HTTP 2.0 is also different from HTTP pipelining. The latter corresponds to a single stream and allows submitting multiple requests into a 'pipeline' without waiting for responses which works strictly like first-in, first-out. It can suffer from head-of-line blocking. Out of order responses from the server even if it were ready to respond are not possible.

WebSocket: What does that mean? The IETF has also specified another protocol which sits on the same level in the protocol stack as the streaming layers in HTTP 2.0 or SDPY, above TCP or TLS/TCP [10]. WebSockets are a bi-directional, full-duplex communication channel over a single TCP connection. WebSocket and HTTP are two very different protocols. This difference is likely to stay also when a standardised version of HTTP 2.0 comes along. WebSocket and HTTP are like a hammer and a screwdriver. They serve two very different purposes, even if they are made of the same metal or wood. HTTP serves the purpose of enabling a client to fetch web resources (like web pages, web forms, Internet media) from a server. WebSocket serves the purpose of a bidirectional communication channel between a client and a server. Bidirectional communication is not needed for all purposes. For example, for reading a text, image and video-clip-based online newspaper, HTTP does the job and no bidirectional communication is needed.

However, bidirectional channels are useful for enabling two-sided audio and video communication for instance between two remote browser windows. Same for real-time shared document editing over the web. A further example is for backend applications that need to extend a TCP-based protocol (say X) to a browser, whilst getting rid of protocol translation from X into HTTP at an app server between client and backend (e.g. JMS, XMPP) or for extending SOA from backend applications over the web to clients on devices.

To get a WebSocket bidirectional communication channel set up, a protocol handshake happens which is delivered using an HTTP 1.1 request/response pair.

The relationship between HTTP 2.0 and WebSockets. As mentioned above, the design goals of HTTP 2.0 have not necessarily much to do with the ones engineers had in mind for WebSockets.

Differences

In SPDY it was all about making the shipment of *HTTP* messages more efficient. In contrast, WebSockets are today kick-started with an HTTP 1.1 handshake, after which they are ready to stream *any kind of fancy protocol data* over the socket. An on-going WebSocket communication could be HTTP-free, whereas an on-going HTTP 2.0 streaming session would ship lots of HTTP messages.

Where WebSocket is an empty pipe to be used by any other higher-layer protocol, HTTP 2.0 (and SPDY) are there to transfer HTTP requests and responses.

HTTP 2.0 is client to server and server to client, whereas WebSockets offer a pipeline beyond that, including client to client, server to server, document to document.

Objectives-wise the key difference is that HTTP 2.0 aims at improving the delivery of web pages, whereas WebSocket is there as a two-way comms channel for browser based web applications which need their own particular, higher level comms protocols (like JMS) for messaging, chat, on-line gaming.

WebSockets are good for real-time gaming (e.g. Farmville HTML 5 app), real-time analytics, real-time control, whereas HTTP 2.0 is mainly there to improve the use of HTTP for shipping fancy web pages.

This means that a new HTTP 2.0 standard could well co-exist with the WebSocket protocol.

Similarities and touch points

A similarity exists in that WebSockets provide a full-duplex persistent connection. HTTP 2.0's session layer is also supposed to be full-duplex, bidirectional and persistent.

Objectives-wise there are overlaps between HTTP 2.0 and WebSockets. E.g. both try to save network bandwidth and improve the scaling of server-based applications. Both improve user experience, customer satisfaction, customer retention and loyalty.

There are complex touch points between WebSockets and Microsoft's proposal for HTTP 2.0. Microsoft's HTTP Speed + Mobility proposal suggests using WebSockets handshake and framing mechanism to establish a bidirectional link (Setting up the HTTP streaming session via WebSocket Upgrade from an initial HTTP message, maintaining of the HTTP session (control, error handling) + multiplexing, compression as a WebSocket extension). New operations codes in the WebSocket frame would define streaming session control message types similar to the ones in SPDY.

The relationship between HTML 5 and HTTP 2.0. HTML 5 is a hypertext mark-up language. It serves the purpose of defining how web content shall be structured (e.g. in headlines, paragraphs, canvas areas for graphics and animation). And it helps co-ordinating the submission of requests from browser to the Internet to fetch web resources like images, other media, and script code. HTTP 2.0 is there to ship 'the stuff' across the Internet, as HTTP's name implies. HTTP 2.0 certainly doesn't make HTML 5 superfluous.

A hot debate - Potential future design flaws and things that might be overlooked. Interestingly, the most widely discussed proposal for HTTP 2.0, namely SPDY from Google, praises itself by having an exclusive focus on improving speed (of web page delivery to browsers). This has most prominently been criticised by Microsoft.

They argue that SPDY's speed improvement goal is only a subset of aspects that should be improved in HTTP 1.1. Potential solutions to the speed problem might even be found outside HTTP, namely e.g. with how the DNS system works for translating Internet domain names into IP addresses.

Moreover, Microsoft say that Google's SDPY is characterised by a rather narrow focus, a rather limited use case, namely the delivery of web pages or web-based Internet services (speak Google's services ...) Microsoft rightly point to the fact that today's HTTP 1.1 is not only used for shipping Google maps and documents and email to web browsers over the public Internet. They are correct in pointing out the pervasive use of HTTP in completely different usage scenarios, e.g. for private

Intranets, as a transport in corporate IT infrastructure, as a transport for Machine-to-Machine (M2M) services.

In the latter case, many M2M terminals indeed communicate with the application servers used for fleet management, location and asset tracking, remote monitoring etc. via HTTP.

The debate is currently getting hotter between the main contenders over who is right and wrong, and which arguments are justified. Whilst Microsoft criticise Google's ignorance of other issues related to the now widespread use of HTTP 1.1 from mobile phones (e.g. negative impact on battery life, on CPU power consumption, radio resources of cellular networks), the SPDY promoters hit back by ridiculing Microsoft's use of the term "Internet of things", by which they, as we believe, simply refer to the widespread non-browser usage scenarios of HTTP, including the above-mentioned M2M services. See also [4] and [5] for the arguments in the debate. [6] argues that Google is too indifferent vis-à-vis the needs of enterprise or infrastructure vendors. Henrik Frystyk Nielsen provides good insight into the technical arguments in [8].

What are the benefits? HTTP 2.0 is likely to deliver benefits in several respects.

First, it can be expected to improve *user experience* particularly when consumers make use of web browsers. This stems from increased speed, reduced latency and faster delivery of web content to the browser. No wonder then, that companies whose business model relies on big numbers of users and subscribers, like Google (search, Gmail), Amazon (for Kindle Fire) have already implemented Google's HTTP 2.0 candidate submission, SPDY. Though the more optimised one of these high profile sites is today (mind, the big companies can afford this), the less incremental speed improvement they may achieve from HTTP 2.0. Twitter is also said to be in favour of SPDY.

Second, it will have an impact on *infrastructure vendors*. Very different might be the stance of the plethora of companies that produce the intermediary boxes that help plumbing together the Internet. Vendors of HTTP proxies, HTTP caches, of content delivery networks and the like. They benefit when ISPs and carriers are ready to spend CAPEX on upgrading their infrastructure to make it HTTP 2.0 capable. That CAPEX needs to be earned by those players in turn. The exact (modified?) roles of those intermediary boxes has also to be clarified.

Third, it will benefit *companies that host their services in the cloud.* They may achieve lower costs or be able to limit the rise of costs when services are scaled up with elastic computing. HTTP 2.0 will use fewer TCP network connections between client and server compared to today's practice. This would relax today's emerging scalability problems on web servers, thus produce CAPEX/OPEX savings on the server side.

Finally, HTTP 2.0 might help *mobile network operators* to save network resources as fewer TCP connections are set up and torn down across a radio and core network.

Completion of HTTP 2.0 may not enable reuse of early pre-standards implementations, as HTTP 2.0 may well not be backward compatible to current candidate implementations like SPDY. See the statement made by Microsoft in [8].

When will it be ready? Improvements over HTTP 1.1 have been attempted before. The last attempt under the label HTTP-NG (for Next Generation) stalled and was suspended back in 1998. Complimentary additions were made so to say outside HTTP, e.g. by introducing WebSockets. The current attempt to define a next version as HTTP 2.0 is forecast to be completed by Nov 2014 only. Thus, still some time to shape the proposals in IETF. Also time to underpin the protocol design suggestions by evidence from analyses and performance assessments that get executed in a well-defined and controlled lab environment.

Will HTTP 2.0 revolutionise the world? HTTP 1.0/1.1 was in fact a major catalyst. Without HTTP 1.1, the web and Internet wouldn't exist and flourish as it does today. Although as a user, we only notice a miniscule tip of a gigantic HTTP iceberg when we type in a Web URI like http://www.twitter.com, it's important to recognise that the already existing version of HTTP is both very complex and very powerful at the same time. Using a Web browser and the HTTP protocol many times during the day, many people use HTTP similar to driving a small car (say a nice Ford Fiesta) without realising that it has a mighty 12 cylinder engine under its hood. More different to the car, that V12 engine is not concentrated in a single spot, but distributed as HTTP 1.1 Internet equipment in many flavours around the globe. Today's HTTP 1.1 and its request/response paradigm, by some people now declared outdated (in contrast we believe it is still ingeniously elegant), is also the basis of hundreds of RESTful APIs and service implementations in different industries. Because of this, most likely, HTTP 2.0 won't replace the core HTTP 1.1 mechanisms, but will add to it and deprecate some of the HTTP 1.1 features which won't make sense in the presence of those additions.

Forecast. Unfortunately, this implies that HTTP 2.0 won't revolutionise the world at all compared to what HTTP 1.1 has achieved. It's unlikely to bring about a new hype (to make one up say mCommerce[2]), though it has the potential to help sustain some of the currently employed business models of web players who rely on vast follow-ship and happy web users (not complaining about sluggish websites and web applications). If the participating IETF engineers succumbed to the view of the promoters of SPDY, then HTTP 2.0 would 'only' deliver a speed improvement for users. Not bad, of course, but others say, HTTP deserves more attention in further respects when it comes to improving it, due to its massive, widespread use beyond the simple web browsing.

References

[1] Microsoft's proposal for HTTP 2.0: https://datatracker.ietf.org/doc/draft-montenegro-httpbis-speed-mobility/?include_text=1

[2] Google's SPDY description: http://www.chromium.org/spdy/spdy-protocol

[3] Google SPDY whitepaper: http://dev.chromium.org/spdy/spdy-whitepaper

[4] Article addressing Microsoft versus Google on HTTP 2.0: http://www.extremetech.com/computing/124153-sm-vs-spdy-microsoft-and-google-battle-over-the-future-of-http-2-0

[5] Article addressing Microsoft's challenge in face of SPDY: http://www.readwriteweb.com/enterprise/2012/03/microsoft-sees-googles-hand-fo.php

[6] Article about HTTP 2.0 proposals: https://devcentral.f5.com/weblogs/macvittie/archive/2012/04/11/the-http-2.0-war-has-just-begun.aspx

[7] Article referring to SPDY as starting point for HTTP 2.0: http://www.engadget.com/2012/08/07/http-standards-group-looks-to-spdy-protocol/

[8] Microsoft blog covering HTTP 2.0: http://blogs.msdn.com/b/interoperability/archive/2012/08/05/http-2-0-makes-a-great-step-forward-in-vancouver.aspx

[9] HTTP 1.1 specification RFC2616: http://tools.ietf.org/html/rfc2616

[10] WebSocket protocol RFC6455: http://tools.ietf.org/html/rfc6455