Perspective on Adopting HTTP/2

2024-05-04

Overview

HTTP/2 offers several performance improvement features including

- Multiplexing - over a single TCP connection
- Stream prioritization - enabling higher priority assets to be delivered first
- Server push - enabling certain assets to be pushed to the client proactively
- Binary encoding - for more efficient parsing and improved response times
- Header compression - reducing header size
- Flow control - ability for the client and server to avoid over burdening each other

Compared to HTTP/1, these can result in reduced network usage, improved user experiences, and can be an enabler for innovation with the potential increase in performance. In addition, as it was released back in 2015, there is good browser adoption and maturity of the protocol.

However, although the promises of HTTP/2 can be realized, there's more that should be taken into consideration before adopting, such as:

- Potentially complex infrastructure implementation
- Determining sufficient ROI and appropriate risk
- Varying real world results, especially with mobile
- Promises of HTTP/3

Below goes a bit into more specifics and potential next steps to take to make an informed decision on moving forward with upgrading the HTTP protocol.

Implementing HTTP/2

On the surface, implementing HTTP/2 is just a server upgrade and it's backward compatible with HTTP/1. Typically applications don't need to make changes.

On the server, HTTP/2 features need to be enabled and set up such as stream prioritization, server push, etc.

Other considerations to account for:

- Preparing for a different type of traffic pattern for clients taking advantage of HTTP/2
    - Eg. setup throttling on the server
    - Eg. applications may introduce a queueing mechanism to process connections

- Updating internal networks for TLS between components as required by HTTP/2
    - Eg. certificates for localhost for local development.

- Removing unnecessary HTTP/1 optimizations
    - Stop file concatenation to allow small files to be cached properly
    - Stop including assets in the HTML page
    - Stop domain sharding

Below include concerns that may need to be addressed when upgrading:

- If the single connection is lost, all ongoing requests and responses are also lost
    - Consider use of multiple connections
- If server push isn't used correctly, it can waste bandwidth or cause other issues.
    - Chrome had removed it - https://developer.chrome.com/blog/removing-push
- Caching of resources at the client side may be a challenge as the client doesn't have control over resources pushed by the server.
- The single connection used for multiple requests can cause head of line blocking. Thus a slow request may hold up processing for other requests.
- Increased strain on servers due to clients requesting more assets at once.
- Infrastructure may need to handle both HTTP/1 and HTTP2 traffic patterns appropriately.
- HTTP/2 requires https, thus all connectivity to the server will need to be encrypted, even inside the network and with development / local machines.
- Monitoring and debugging HTTP/2 traffic can be more challenging than with HTTP/1.
- Varying industry results, including worse performance with mobile in some cases.

Reluctance to adopt

Initially, companies were reluctant to upgrade to HTTP/2 as adoption was low. In 2024 most modern browsers and server technologies support HTTP/2, although not necessarily all features (such as server push).

Companies may still be reluctant to adopt HTTP/2 due to:

- Cost and complexity for organizations with widespread infrastructure
- The performance gain from HTTP/2 may not be sufficient enough to offset costs of infrastructure and resources needed to implement.

- The implementation may not fully support HTTP/2, reducing the expected gains.
- Believe HTTP/1.x is sufficient
- May move to HTTP/3 which promises even more advancements and greater performance gains, but uses a different transport layer

What's next… HTTP/3

HTTP-over-QUIC, is the next major revision to the HTTP protocol providing further performance improvements.

- Released in 2022, thus adoption is fairly low and is still maturing
- Uses the QUIC (Quick UDP Internet Connections) transport layer instead of TCP
- Is not backwards compatible

Anticipated Impacts

Infrastructure

- Consider handling different traffic patterns based on the HTTP version being used
- Setup throttling to handle HTTP/2 traffic patterns
- Potentially avoid server push due to lack of adoption
- Setup stream prioritization algorithm and monitor
    - Avoid relying on browser prioritization as they differ and may be suboptimal
- Monitor / evaluate performance and apply tuning as appropriate

Applications

- POC / Test to evaluate in your environment - both for desktop and mobile
- Stop unnecessary HTTP/1.1 optimizations
- Setup ability to capture performance metrics in production
- Possible updates depending on POC results
    - Avoid overusing / abusing the single connection
    - Recover from a lost connection
    - Establish best practices and reviews to ensure proper HTTP/2 handling

Next steps

Additional research and due diligence

- Discuss options, capabilities, and perspectives with Infrastructure team
- Reach out to CDN partners for their expertise and recommendations
- Determine if should focus upgrade on HTTP/2 or skip straight to HTTP/3
- Further investigate industry results for each protocol (including for mobile)

Conduct POC

Although both protocols promise improvements, diving into networking threads on the internet result in mostly positive, but mixed reviews. Setup a POC to evaluate and make an informed decision for your environment.

Example:

- Setup sandbox environment to handle HTTP/2 or 3
- Load Test to set control metrics on HTTP/1
- Load Test iterations of
    - Server optimizations
    - Application optimizations
- Note: Test mobile apps separately

Document findings

- Benefits, concerns, risk, etc.
- Infrastructure and application impacts (if any)
    - Including impact to local development

Establish implementation plan

Assuming sufficient benefits were realized from the POC, establish a high level implementation plan and timeline.

- Determine efforts, costs, timing to implement, milestones, etc.
- Create roadmap for targeted apps and infrastructure (all environments)

Obtain funding / buy-in

Present upgrade options (including doing nothing) and provide a recommendation.

Example:

- Do nothing
    + Avoids time, cost, and risk
    - Lag behind competition / dated infrastructure
    - Limited ability to address some performance concerns
- Implement HTTP/2
    + Better user experience
    + Potential for innovation leveraging increased performance
    + Good adoption and maturity
    - Time and Cost
- Implement HTTP/3
    + Better user experience
    + Potential for innovation leveraging increased performance
    + Best performance
    - Time and Cost
    - Still early in its adoption and maturity
    - Not backwards compatible
    - Take on more risk

Quantify the benefits to showcase they are worth the effort and associated risks.

- Eg. a “better user experience” can be hard to quantify against time, cost, and risk of implementation. What are the performance issues we have today? Are we losing business because of this? Are we getting a lot of user complaints? What would the user experience change be (5 seconds to 2 seconds)? etc.

Conclusion

HTTP/2 and HTTP/3 can improve performance of applications and reduce network usage. Moving forward with an upgraded infrastructure will allow applications to avoid performance tuning efforts in an outdated infrastructure. Increased performance can bring improved user experiences and improve the potential for new business.

However, the direction and ROI of a protocol upgrade is not clear cut. Proper due diligence and testing should be conducted.

Consider a canary-like implementation where only a select user base or client is moved to an upgraded HTTP environment to validate expected benefits and effort before fulling committing.