Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 11, 2017

G. Montenegro
Microsoft
S. Cespedes
NIC Labs Chile
S. Loreto
Ericsson
R. Simpson
General Electric

March 10, 2017

HTTP/2 Configuration Profile for the Internet of Things
draft-montenegro-httpbis-h2ot-profile-00

Abstract

This document define an HTTP/2 configuration profile for IoT.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Unlike HTTP/1.X, HTTP/2 is suitable for many IoT applications. Even though IoT was not the primary scenario when HTTP/2 was designed, the protocol is compact, configurable and flexible. The use of header compression as well as the binary encoding of the protocol reduces the size of HTTP/2 flows as compared to HTTP/1.1. HTTP/2's ability to reuse connections for multiple streams reduces connection establishment overhead, such as TCP connection establishment and TLS session establishment. HTTP/2 has been found to be amenable to implementation on class 2 devices, per the constrained device classification in section 3 of [RFC7228]. Furthermore, initial efforts have resulted in successful experiments on class 1 devices.

This document discusses how to configure HTTP/2 so as to adapt it better to IoT scenarios, including those in which the devices running the protocol have constrained resources.

2. Configuration Profile of HTTP/2 for IoT

HTTP/2 has many negotiable settings that can improve its performance for IoT applications by reducing bandwidth, codespace, and RAM requirements. Specifically, the following settings and values have been found to be useful in IoT applications:

- SETTINGS_HEADER_TABLE_SIZE: this setting allows hosts to limit the size of the header compression table used for decoding, reducing required RAM, but potentially increasing bandwidth requirements. Initial value per HTTP/2 is 4096. IoT scenarios might benefit from changing this to a smaller value (e.g., 512), however, to

avoid increased bandwidth usage, IoT scenarios should judiciously use HTTP headers and the dynamic header table [RFC7541].

- SETTINGS_ENABLE_PUSH: This setting allows clients to enable or disable server push. This functionality may not be required in some IoT applications. The initial value per HTTP/2 is 1.

- SETTINGS_MAX_CONCURRENT_STREAMS: this setting allows a sender to limit the number of simultaneous streams that a receiver can create for a connection. HTTP/2 recommends this value be no smaller than 100. IoT scenarios may wish to limit this to a much smaller number, such as 2 or 3.

- SETTINGS_INITIAL_WINDOW_SIZE: this setting allows hosts to limit the flow control window, potentially reducing buffer requirements at the expense of potentially underutilized bandwidth-delay products. Per HTTP/2 the initial value is 2^16-1 (65,535) octets. IoT scenarios may wish to limit this to smaller values in accordance with the node's constraints (e.g., a few kilo-octets).

- SETTINGS_MAX_FRAME_SIZE: this setting allows hosts to specify the largest frame size they are willing to receive. Per HTTP/2 the initial value is 2^14 (16,384) octets. Somewhat counterintuitively, IoT hosts may wish to leave this value large and rely on flow control to avoid unnecessary framing overhead (see: <https://lists.w3.org/Archives/Public/ietf-http-wg/2014JulSep/1626.html>).

- SETTINGS_MAX_HEADER_LIST_SIZE: this setting allows hosts to limit the maximum size of the header list they are willing to receive. Per HTTP/2 the initial value of this setting is unlimited. IoT scenarios may wish to limit this to smaller values in accordance with the node's constraints (e.g., a few kilo-octets).

3. Negotiation of HTTP/2 for IoT

For Constrained and Internet scenarios, it is assumed that HTTP/2 runs over TLS. Accordingly, the ALPN negotiation in section 3.3 of [RFC7540] applies. As seen above, an IoT scenario may wish to depart from the default SETTINGS. To do so, the usual SETTINGS negotiation applies. In this case, the initial SETTINGS negotiation setup is based on the first message exchange initiated by the client. This is simpler than general HTTP/2 case: not having an in-the-clear Upgrade path means the client is always in control of first HTTP/2 message, including any SETTINGS changes it may wish.

Additionally, the use of "prior knowledge" per section 3.4 of [RFC7540] is likely to also work particularly well in IoT scenarios

in which a client and its web service are likely to be closely matched. In such scenarios, prior knowledge may allow for SETTINGS to be set in accordance with some shared state implied by the prior knowledge. In such cases, SETTINGS negotiation may not be necessary in order to depart from the defaults as defined by HTTP/2.

4. Implementation Considerations

This section assumes HTTP/2 over TCP, i.e., as specified in [RFC7540]. Implementors should consider TCP optimizations for IoT, such as [I-D.gomez-core-tcp-constrained-node-networks] as well as LoWPAN-related TCP optimizations such as [I-D.aayadi-6lowpan-tcphc].

In addition to underlying stack issues with respect to IPv4, IPv6, TCP, and TLS, there are implementation considerations for HTTP/2 for IoT.

A primary concern is the number of allowed simultaneous HTTP/2 connections. As each connection has associated overhead, as well as overhead for each of its streams, constrained hosts may wish to limit their number of simultaneous connections. However, implementers should note that some popular browsers require more than one connection to operate (e.g., both Chrome and Firefox have been observed as requiring at least two connections). Given that one of the motivations to use HTTP/2 is to use mainstream technologies, this is important for certain scenarios.

In addition to minimizing the number of simultaneous connections, hosts should consider leaving connections open if there is a possibility of further communication with the remote peer. HTTP/2 contains mechanisms such as PING to periodically check idle connections. Leaving established connections open when there is a possibility of future communication allows connection establishment overhead (and potentially TLS session establishment overhead) to be avoided.

Should TLS be used, implementers may wish to utilize hardware-based encryption to further reduce codespace and RAM requirements.

5. IANA Considerations

This document has no considerations for IANA.

6. Security Considerations

This document suggests a profile for HTTP/2 in IoT applications. All the security considerations for [RFC7540] apply. Given the constraints likely to characterize devices common in IoT scenarios,

issues related to resource exhaustion and denial-of-service attacks are particularly noteworthy. Nevertheless, the suggestions in this document limit the resources used in such a way that any peer exceeding such limits will be in protocol violation making those connection or connection attempts readily droppable. Of course, resource exhaustion attacks are not mitigated, as these will simply obey the limits imposed by the constrained profile specified herein. Hence, it is always imperative to safeguard IoT devices by the usual means (e.g., by using a firewall or a gateway with richer resources to provide some protection).

As for the potential risks to the infrastructure by attacks launched from devices compliant with this specification, each such instance represents less of a threat than usually configured and profiled HTTP/2 clients. Nevertheless, the sheer number of IoT devices means that the overall threat to infrastructure may be formidable, as has been observed in IoT-based DDoS attacks. Accordingly, it is essential for these devices to implement the usual security measures to prevent their hijacking by e.g., requiring strong authentication of the operators, update capabilities, etc.

7. Acknowledgements

This document was produced using the xml2rfc tool [RFC2629][RFC7749].

8. References

8.1. Normative References

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

[RFC7541] Peon, R. and H. Ruellan, "HPACK: Header Compression for HTTP/2", RFC 7541, DOI 10.17487/RFC7541, May 2015, <http://www.rfc-editor.org/info/rfc7541>.

8.2. Informative References

[I-D.aayadi-6lowpan-tcphc]
Ayadi, A., Ros, D., and L. Toutain, "TCP header compression for 6LoWPAN", draft-aayadi-6lowpan-tcphc-01 (work in progress), October 2010.

[I-D.gomez-core-tcp-constrained-node-networks]
Gomez, C. and J. Crowcroft, "TCP over Constrained-Node Networks", draft-gomez-core-tcp-constrained-node-networks-00 (work in progress), June 2016.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <http://www.rfc-editor.org/info/rfc7228>.

[RFC7749] Reschke, J., "The "xml2rfc" Version 2 Vocabulary", RFC 7749, DOI 10.17487/RFC7749, February 2016, <http://www.rfc-editor.org/info/rfc7749>.

Authors' Addresses

Gabriel Montenegro
Microsoft

Email: Gabriel.Montenegro@microsoft.com

Sandra Cespedes
NIC Labs Chile
Universidad de Chile

Email: scespedes@ing.uchile.cl

Salvatore Loreto
Ericsson

Email: salvatore.loreto@ericsson.com

Robby Simpson
General Electric

Email: rsimpson@gmail.com