Performance Evaluation of CoAP and HTTP/2 in Web Applications

Diego Londoño
Departamento de Ingeniería Eléctrica
NIC Chile Research Labs
Universidad de Chile
diegolondono@niclabs.cl

Sandra Céspedes
Departamento de Ingeniería Eléctrica
NIC Chile Research Labs
Universidad de Chile
scespedes@ing.uchile.cl

Abstract

Nowadays, different entities are making efforts for standardizing application protocols oriented to Internet of Things (IoT). In this work, the CoAP and HTTP/2 response times in an IoT web environment are evaluated experimentally, varying the channel's traffic and the signal's quality. We explain the motivation to evaluate HTTP2 in IoT environments and describe related works that provide comparisons for IoT protocols. Next, we describe the scenarios employed in our comparisons and analyze the results under variable channel conditions.

1 Introducción

Poco tiempo después de la aparición del término Internet de las Cosas (IoT) en 1999, diferentes entidades y consorcios iniciaron la tarea de crear estándares para este nuevo campo de acción. A diferencia del tradicional modelo de conexión entre computadoras, en IoT las máquinas se comunican entre ellas (*Machine-to-Machine communications* o M2M), dichas máquinas pueden tener limitaciones en cuanto a su alimentación energética, procesamiento, almacenamiento y potencia de transmisión [Ker14]. Las anteriores restricciones han llevado a re-pensar el uso de los protocolos más difundidos en Internet, como lo son HTTP/1.1 y TCP, y a la creación de nuevos protocolos optimizados para estos escenarios [Ler12], tales como *Constrained Application Protocol* (CoAP) y *Message Queue Telemetry Transport* (MQTT).

Por otro lado, siguiendo una lógica de no "reinventar la rueda", existen propuestas para hacer las adaptaciones necesarias a escenarios IoT de protocolos ampliamente difundidos en Internet, como HTTP, o su sucesor HTTP/2 [Mon16]. Los argumentos principales que respaldan esta propuesta son tres: seguridad, interoperabilidad y conocimiento de la tecnología. En cuanto a la seguridad, crear nuevos protocolos y nuevas pilas sugiere también nuevas formas de ataques, con el agravante de que IoT es un escenario de alto riesgo debido a las restricciones de los dispositivos, su amplia distribución geográfica y la posibilidad de fácil acceso físico por parte de los atacantes. Respecto a la interoperabilidad, tener muchas pilas y protocolos puede dificultar este pilar de la Internet. Por último, utilizar la tecnología mejor conocida tiene ventajas en cuanto a que hay más implementaciones hechas, ampliamente probadas y mejoradas con el tiempo, algunas de ellas *Open Source*; también hay más gente con los conocimientos necesarios para trabajar con ella [Mon16].

Se podría esperar que por sencillez y su diseño específico para IoT, CoAP tenga mejor desempeño en cuanto a tiempos de respuesta que HTTP/2, dado que este último fue pensado para la web en general y no necesariamente para ambientes restringidos. Surge, entonces, el interrogante ¿qué tan lejos está HTTP/2 de CoAP? Si HTTP/2 con adaptaciones llegara a tener un desempeño cercano a CoAP, es decir, aceptable en un escenario restringido de IoT, y dadas las ventajas que ofrece, podría ser usado masivamente en este campo. Para responder a dicho interrogante, en este trabajo se realizan dos implementaciones, en la primera se tiene un cliente y un servidor HTTP/2. En la segunda, se tiene un cliente CoAP, un *proxy* CoAP/HTTP y un servidor HTTP1.1. El objetivo es hacer una comparación de desempeño teniendo en cuenta los tiempos de respuesta de ambos casos.

En trabajos previos [Tha14, Col14] se comparó el desempeño de algunos protocolos orientados a IoT, pero no directamente HTTP/2 con CoAP. Los resultados han mostrado como principal conclusión que no hay un protocolo que se desempeñe mejor en todos los escenarios, sino que esto depende de las condiciones de la red. En [Sax15] se demuestra que HTTP/2 tiene mejor desempeño que HTTP/1.1 en cuanto a tiempos de respuesta.

Este trabajo es un acercamiento a los protocolos de IoT, de cara a una posible adaptación de HTTP/2

This work is partially funded by Project FONDECYT Iniciación No. 11140045. Proceedings of the Spring School of Networks, Santiago, Chile, November 2016.

para escenarios restringidos. A partir de este trabajo se espera sentar las bases para definir los cambios pertinentes que podrán incorporarse a una nueva versión de HTTP/2.

2 Descripción de la implementación

Para la comparación de desempeño entre HTTP/2 y CoAP se hizo un montaje experimental, con el fin de recopilar la información de los tiempos de respuesta promedio de métodos GET y POST en HTTP/2 y CoAP, variando el tráfico en el canal y la calidad de la señal inalámbrica. El número de peticiones a analizar es de 30 para cada caso.

El montaje para el experimento consiste en dos escenarios, uno para probar el desempeño de HTTP/2 y otro para una solución CoAP/HTTP1.1. En el primer caso, el navegador Mozilla Firefox 47.0 se conecta vía WiFi (IEEE 802.15n) a un *Acces Point* (AP) Cisco WRT160NL. El AP se conecta vía FastEthernet a un PC con una máquina virtual Ubuntu 16.04 en el cual se ejecuta un servidor web Apache 2.4.20.

En el caso del escenario para CoAP/HTTP, se usa como cliente CoAP, Copper 0.18.4.1, un complemento de Mozilla Firefox [Cop16]. Este se conecta vía WiFi al AP, que a su vez se conecta vía FastEthernet al PC donde hay dos máquinas virtuales: una de ellas es un *Proxy* CoAP/HTTP1.1 llamado Crosscoap [Ibm16] y la otra es un servidor web Apache HTTP1.1. Para ambos escenarios las capturas de tráfico se realizan en el computador portátil del cliente utilizando Wireshark. En la figura 1 se muestra un diagrama con las implementaciones.

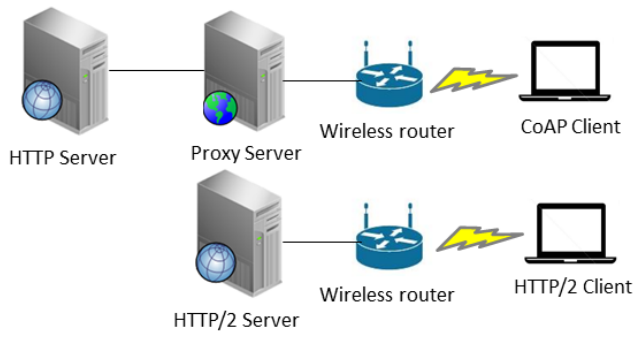

Figure 1: Escenarios implementados para las pruebas

Para crear congestión en el canal inalámbrico se emplean otros dos computadores portátiles que inyectan tráfico TCP a la red, con ayuda de Iperf [Ipe16]. Uno actúa como cliente y otro como servidor. El nivel de congestión se va aumentando gradualmente hasta llegar a 20 solicitudes simultáneas con paquetes de tamaños variables, según la condición del canal, y los mismos tres equipos involucrados generan paquetes PING de 2000 KB cada uno, de forma sostenida hacia el AP. En la figura 2 se ilustra el escenario de congestión.

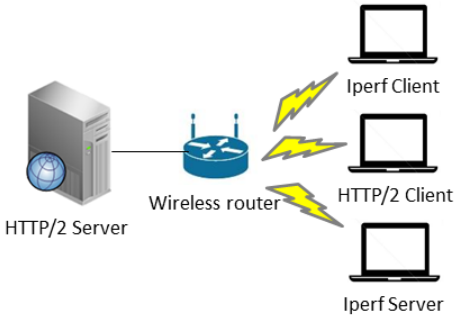

Figure 2: Escenario de HTTP/2 con equipos adicionales generando tráfico

En cuanto a la calidad de la señal, en un primer momento los equipos portátiles tienen línea de vista con el AP y la potencia recibida en el cliente es de alrededor de -27dBm. Para forzar una reducción en la calidad de la señal, los clientes se ubican fuera de línea de vista del AP de tal manera que se reduzca la potencia recibida a aproximadamente -85dBm.

3 Resultados y discusión

Los tiempos de respuesta obtenidos del método GET para todos los escenarios se muestran en la tabla 1 y los del método POST en la tabla 2.

Table 1: Tiempos de respuesta promedio - método GET

Potencia recibida	Nivel de congestión	HTTP/2 (s) [% perdido]	CoAP (s) [% perdido]
-27dBm	Sin congestión	0,21039 [0%]	0,00956 [0%]
	Congestión x1	0,28495 [0%]	0,04969 [0%]
	Congestión x2	0,26536 [0%]	0,06115 [0%]
	Congestión x3	0,31693 [0%]	0,09064 [0%]
-85dBm	Sin congestion	1,06149 [7%]	2,03934 [0%]
	Congestión x3	2,85344 [24%]	6,44652 [16%]

Table 2: Tiempos de respuesta promedio - método POST

Calidad de la señal	Nivel de congestión	HTTP/2 (s)	CoAP (s)
-27 dBm	Sin congestión	0,07112	0,0168
	Congestión x3	0,66474	0,1393
-85 dBm	Sin congestión	0,37146	0,3795
	Congestión x3	0,4567	2,1253

Para ambos métodos cuando la señal es buena, CoAP tiene significativamente mejores tiempos de respuesta respecto a HTTP/2, independiente del nivel de congestión; esto se debe principalmente a dos razones, la primera es la sencillez de la estructura de los paquetes CoAP frente a los de HTTP/2, lo que agiliza la comunicación. La segunda razón es porque al utilizar UDP no necesita establecer una conexión de un *stream*, ni hacer uso de ACKs en la capa de transporte, lo que agiliza el envío de la información de capa de aplicación.

Se debe tener en cuenta que parte del tiempo de respuesta de las peticiones CoAP corresponden al

tiempo que se demora el servidor *Proxy* en hacer la respectiva petición al servidor Web y que este a su vez responda. En los escenarios implementados, la comunicación entre estos dos servidores no se ve afectada por la congestión en el canal inalámbrico. El servidor *Proxy* no está haciendo *caching*.

Cuando la calidad de la señal se deteriora, en todos los casos HTTP/2 tiene mejores tiempos de respuesta, sin embargo hay peticiones GET no respondidas. La razón por la que CoAP reduce significativamente su desempeño es porque la implementación utilizada de CoAP envía las peticiones GET y POST sobre un mensaje confirmable (CON) con un *back-off* exponencial que inicia en 2s. El cliente hará hasta cuatro retransmisiones, esperando 2, 4, 8 y 16 segundos respectivamente. Estos tiempos de *back-off*, si bien aumentan la probabilidad de que el canal se recupere, también aumentan considerablemente los tiempos de respuesta de las solicitudes, mientras que las retransmisiones de HTTP/2 son más rápidas.

La principal afectación en el desempeño de HTTP/2 cuando el canal presenta fallas, se da porque se pierden las conexiones TCP, obligando a que se haga el *handshaking* varias veces, y a que finalmente los tiempos de respuesta aumenten. Esta es una desventaja de HTTP/2 frente a CoAP, en cuanto a procesamiento y tiempos de respuesta, debido a que en este último el *handshaking* no existe. Una posible solución a este punto es hacer que HTTP/2 también corra sobre UDP [Qui16]. Cabe anotar que Mozilla Firefox tiene implementado HTTP/2 sólo sobre TLS, lo cual añade una capa de complejidad adicional que impacta en el tiempo de respuesta.

En los experimentos, el tipo de información que se transmite tanto del servidor al cliente como del cliente al servidor, simula de manera sencilla la información arrojada en texto plano por un nodo-sensor. Debido a esto, en un primer momento no se explotan todas las funcionalidades y ventajas que ofrece HTTP/2. Esto lleva a pensar que si se quiere adaptar este protocolo a un escenario de IoT, se deberá tener en cuenta que la información manejada puede ser más sencilla y esporádica que la de las páginas web tradicionales de la Internet. Por otro lado, los tiempos de respuesta de CoAP podrían mejorar si los tiempos de espera para retransmitir se hacen más pequeños, en cuyo caso depende de la implementación del cliente que se utilice.

4 Conclusiones y trabajo futuro

Los resultados muestran que no hay un protocolo que sea el mejor en todos los escenarios, la elección del más conveniente dependerá de las condiciones de los equipos y de la red. En el caso del presente estudio, se obtiene que si el canal es confiable, CoAP tiene mejores tiempos de respuesta y cuando el canal tiene pérdidas, los tiempos de respuesta son mejores con HTTP/2.

Adicionalmente se determinó que el *handshaking* de TCP aumenta los tiempos de respuesta. Si la señal es mala y las conexiones se pierden, el desempeño de HTTP/2 empeora. Respecto a este tema, hay soluciones propuestas y probadas como QUIC [Qui16], donde HTTP/2 corre sobre UDP. Corresponde a trabajos futuros determinar si esta opción se desempeña bien en un ambiente restringido. Está claro es que la capa de transporte deberá tener cambios en IoT.

También se determinó que el desempeño de CoAP se ve deteriorado en redes con pérdidas debido al uso de mensajes confirmables (CON) que generan tiempos de espera prolongados para hacer la retransmisión.

Si bien el objetivo de este trabajo es evidenciar el desempeño de CoAP y HTTP/2 de manera experimental frente a escenarios restringidos de IoT, en el escenario web descrito aún no se han utilizado escenarios restringidos. Como trabajo futuro se deberán hacer pruebas de desempeño similares, pero con equipos con capacidades limitadas y utilizando tecnologías típicas de redes de sensores y dispositivos IoT, como IEEE 802.15.4. Adicionalmente, en este trabajo se utilizó un *Proxy* CoAP/HTTP1.1. En un trabajo futuro se espera poder evaluar el desempeño cuando todo el escenario es CoAP, es decir, sin la existencia de un Proxy.

Los trabajos futuros servirán para poder establecer qué adaptaciones o mejoras se le pueden hacer a HTTP/2 de cara a una implementación en una red con nodos restringidos en sus capacidades, como es de esperarse en un escenario típico de dispositivos IoT.

References

[Ker14] C. Bormann, M. Ersue, and A. Keranen, *Terminology for Constrained-Node Networks*, RFC 7228, May 2014. Disponible: http://www.rfc-editor.org/info/rfc7228.

[Ler12] C. Lerche, N. Laum, F. Golatowski, D. Timmermann, and C. Niedermeier, *Connecting the web with the web of things: lessons learned from implementing a CoAP-HTTP proxy*, in 2012 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012), pp. 1–8, 2012.

[Mon16] G. Montenegro, S. Céspedes, S. Loreto, and R. Simpson, *H2oT: HTTP/2 for the Internet of Things*, IETF Internet-draft (work in progress), July 2016. Online: https://tools.ietf.org/html/draft-montenegro-httpbis-h2ot-00

[Tha14] D. Thangavel, X. Ma, A. Valera, H. X. Tan, and C. K. Y. Tan, *Performance evaluation of MQTT and CoAP via a common middleware*, in 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 1–6, 2014.

[Col14] M. Collina, M. Bartolucci, A. Vanelli-Coralli, and G. E. Corazza, *Internet of Things application layer protocol analysis over error and delay prone links*, in 2014 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC), pp. 398–404, 2014.

[Sax15] H. de Saxcé, I. Oprescu, and Y. Chen, *Is HTTP/2 really faster than HTTP/1.1?*, in 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 293–299, 2015.

[Cop16] Copper (Cu). [Online]. Available in: https://addons.mozilla.org/en-US/firefox/addon/copper-270430/. [Accessed: 06-jul-2016].

[Ibm16] Ibm-security-innovation/crosscoap, GitHub. [Online]. Available in: https://github.com/ibm-security-innovation/crosscoap. [Accessed: 06-jul-2016].

[Ipe16] iPerf - The TCP, UDP and SCTP network bandwidth measurement tool. [Online]. Available in: https://iperf.fr/. [Accessed: 06-jul-2016].

[Qui16] QUIC, a multiplexed stream transport over UDP - The Chromium Projects. [Online]. Disponible en: https://www.chromium.org/quic. [Accessed: 06-jul-2016]