UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

EVALUACIÓN DEL PROTOCOLO HTTP/2 PARA INTERNET DE LAS COSAS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERA CIVIL EN COMPUTACIÓN

DANIELA VALENTINA RUZ MIERES

PROFESOR GUÍA:
SANDRA CÉSPEDES UMAÑA

MIEMBROS DE LA COMISIÓN:
JAVIER BUSTOS JIMÉNEZ
JOSÉ PINO URTUBIA

Este trabajo ha sido parcialmente financiado por NIC Chile Research Labs

SANTIAGO DE CHILE
2019

RESUMEN

La presente memoria detalla el proceso de análisis y pruebas sobre el protocolo HTTP/2 en un escenario de Internet de las Cosas para los dispositivos Raspberry Pi 3 Model B y A8 open node.

La Internet de las Cosas, *Internet of Things* o IoT es una plataforma en la que por lo general se tienen máquinas restringidas en procesamiento, almacenamiento y alimentación energética, entre otros. En este modelo, estos dispositivos envían información entre sí, aunque también pueden interactuar con equipos tradicionales. Para el envío de información en este escenario, se hace uso de distintos protocolos de comunicación. Actualmente MQTT es el protocolo más popular, seguido por HTTP, CoAP y HTTP/2.

De los protocolos anteriormente mencionados, tanto MQTT como CoAP son protocolos diseñados para IoT. Sin embargo, la comunicación en Internet ocurre mayoritariamente mediante el protocolo HTTP y la diversidad de protocolos puede generar problemas de interoperabilidad. Además, dada la importancia de la seguridad en Internet de las Cosas, HTTP resulta una mejor opción ya que es un protocolo de aplicación seguro y probado en entornos web.

En 2015 fue presentada la nueva versión del protocolo web HTTP (*Hypertext Transfer Protocol*), llamada HTTP/2. Esta implementa distintos parámetros que pueden optimizar los tiempos de carga de páginas web y ha mostrado ser más efectiva en uso de recursos de red que su predecesora. HTTP/2 mantiene la semántica de HTTP pero mejora la eficiencia, por lo que puede resultar ventajoso hacer uso de HTTP/2 en estos ambientes al mejorar no solo el uso de recursos de red, sino que también al permitir la reutilización de aplicaciones ya existentes diseñadas para HTTP. El problema es que se desconoce una configuración de los parámetros de HTTP/2 adecuada para IoT en términos de uso de recursos de los dispositivos.

En este trabajo se proponen y realizan una serie de pruebas simulando tráfico de tipo IoT y se realizan mediciones sobre el uso de CPU, memoria y consumo energético para distintos valores de parámetros de HTTP/2, como son: el tamaño de la ventana, el tamaño del *frame*, la cantidad de *streams* concurrentes, entre otros.

Se concluye que los parámetros de HTTP/2 no tienen ningún efecto en el uso de estos tres recursos en las plataformas de experimentación utilizadas, incluso para cantidades de tráfico equivalentes al tamaño de varias páginas web actuales. Por lo tanto, la configuración por defecto de HTTP/2 es suficiente para obtener el mejor desempeño posible de este protocolo para un tipo de tráfico IoT.

A mi perra Luna y a Stack Overflow.

Agradecimientos

A la Luna, mi perra favorita del mundo entero y satélite personal, por haberme acompañado durante estos largos años, y perdonarme el tiempo y los paseos que sacrifiqué en pos de terminar este documento. Espero que al titularme te pueda dar la vejez que mereces y pueda acompañarte yo ahora a ti. No sé si sabes cuánto te quiero ni lo importante que has sido.

A mis profesores guía, Sandra y Javier, y a todo NIC Labs, por el cariño, la paciencia y por darme un lugar de trabajo donde pude sentirme cómoda. En especial a Paty, por quedarse en la oficina hasta tarde (aunque no fuera a propósito) y por varias otras cosas.

A toda la gente que de alguna u otra manera me facilitó el camino. Aquí son muchos, pero en particular para este trabajo, agradezco a la Cata por motivarme cada vez que no tenía ganas de trabajar y por poner atención a mis quejas; sin ti quizás me habría rendido antes de tiempo y definitivamente todo habría sido más difícil. También agradezco a Pipe y a la Iva por ayudarme cuando han podido y alivianar la carga de los eventos complicados.

A los que se fueron antes de poder llegar a ver este momento, por el impacto que tuvieron en mi vida y porque fueron y siguen siendo muy importantes. Sin ningún orden en particular: mi abuela, mi abuelo, mi tía Chechi, mi tío Edgardo, mi tía Maruja, mi tío Gustavo y mi tía Cecilia. Gracias por los buenos recuerdos y por enseñarme a vivir, cada uno a su manera. Un "los voy a extrañar siempre" se queda corto.

Finalmente, a mi madre. Porque aunque nunca te haya comprendido ni tú a mí, has sido una constante; compleja y difícil, pero constante al fin y al cabo. Creo que me has hecho enfrentar las dificultades de otra manera y eres inevitablemente parte fundamental de quien soy hoy.

Tabla de Contenido

Índice de Tablas **x**

Índice de Ilustraciones **xiv**

1. Introducción **1**
1.1. Motivación . . . 2
1.2. Definición del problema . . . 2
1.3. Objetivos . . . 2
1.3.1. Objetivo general . . . 2
1.3.2. Objetivos específicos . . . 2
1.4. Metodología . . . 3
1.4.1. Fases del proceso . . . 3
1.4.2. *Hardware* utilizado . . . 5
1.5. Organización del documento . . . 7

2. Marco Teórico **8**
2.1. Internet de las cosas . . . 8
2.1.1. Tráfico en IoT . . . 9
2.1.2. Tecnologías de comunicación para IoT . . . 9
2.2. HTTP/2 . . . 11
2.2.1. Frames . . . 11
2.2.2. Header o encabezado . . . 12
2.2.3. HPACK . . . 13
2.2.4. Streams . . . 13
2.2.5. Server Push . . . 15
2.2.6. Control de flujo . . . 15
2.2.7. TLS . . . 15

3. Estado del arte **16**
3.1. Elección de un protocolo efectivo para IoT . . . 16
3.2. Comparación de HTTP y MQTT en recursos de red para IoT . . . 16
3.3. Evaluación de ventana para Raspberry Pi . . . 17
3.4. Evaluación de ventana y streams para A8 open node . . . 17
3.5. Tiempo de respuesta y rendimiento de red en Raspberry Pi . . . 17
3.6. Perfil de configuración de HTTP/2 para IoT . . . 18

4. Análisis y diseño **19**

4.1. Experimentos anteriores . . . 19
4.2. Nghttp2 . . . 19
4.2.1. nghttp . . . 20
4.2.2. nghttpd . . . 21
4.2.3. libevent-server . . . 22
4.2.4. h2load . . . 22
4.3. Configuración experimental . . . 23
4.3.1. Raspberry . . . 23
4.3.2. A8 open node . . . 24

5. Implementación **25**
5.1. Benchmarking del servidor . . . 25
5.2. Mediciones de CPU, memoria y potencia . . . 25
5.3. Consideraciones para los experimentos . . . 27
5.3.1. Consideraciones para los parámetros . . . 27
5.3.2. Otras consideraciones . . . 27

6. Resultados **29**
6.1. Benchmark del servidor . . . 29
6.1.1. Pruebas con plataforma Rasperry Pi . . . 29
6.1.2. A8 open node . . . 33
6.2. Experimentos con parámetros . . . 34
6.3. Cantidad de medidas en los experimentos . . . 35
6.3.1. Raspberry Pi como servidor . . . 35
6.3.2. Raspberry Pi como cliente . . . 46
6.3.3. A8 open node como servidor . . . 56
6.3.4. A8 open node como cliente . . . 66

7. Discusión **77**

8. Conclusión **79**
8.1. Trabajo futuro . . . 80

Bibliografía **84**

A. Tablas de resultados de Raspberry como servidor **85**
A.1. Max concurrent streams . . . 85
A.2. Initial window size . . . 86
A.3. Max header list size . . . 87
A.4. Header table size . . . 87
A.5. Max frame size . . . 88

B. Tablas de resultados de Raspberry como cliente **89**
B.1. Max concurrent streams . . . 89
B.2. Initial window size . . . 90
B.3. Max header list size . . . 91
B.4. Header table size . . . 91
B.5. Max frame size . . . 92

C. Tablas de resultados de nodo A8 como servidor **93**
C.1. Max concurrent streams 93
C.2. Initial window size 94
C.3. Max header list size 95
C.4. Header table size 95
C.5. Frame size 96

D. Tablas de resultados de nodo A8 como cliente **97**
D.1. Max concurrent streams 97
D.2. Initial window size 98
D.3. Max header list size 99
D.4. Header table size 99
D.5. Max frame size 100

Índice de Tablas

6.1. Valores por defecto de los parámetros relevantes de acuerdo al RFC de HTTP/2 [BPT15] 29
6.2. Resultados para configuración por defecto 36
6.3. Resultados sin Server Push 36
6.4. Resultados para configuración por defecto 46
6.5. Resultados sin Server Push 46
6.6. Resultados para configuración por defecto 56
6.7. Resultados sin Server Push 56
6.8. Resultados para configuración por defecto 66
6.9. Resultados sin Server Push 66

A.1. Medidas para SETTINGS_MAX_CONCURRENT_STREAMS 85
A.2. Medidas para SETTINGS_INITIAL_WINDOW_SIZE 86
A.3. Medidas para SETTINGS_MAX_HEADER_LIST_SIZE 87
A.4. Medidas para SETTINGS_HEADER_TABLE_SIZE 87
A.5. Medidas para SETTINGS_MAX_FRAME_SIZE 88

B.1. Medidas para SETTINGS_MAX_CONCURRENT_STREAMS 89
B.2. Medidas para SETTINGS_INITIAL_WINDOW_SIZE 90
B.3. Medidas para SETTINGS_MAX_HEADER_LIST_SIZE 91
B.4. Medidas para SETTINGS_HEADER_TABLE_SIZE 91
B.5. Medidas para SETTINGS_MAX_FRAME_SIZE 92

C.1. Medidas para SETTINGS_MAX_CONCURRENT_STREAMS 93
C.2. Medidas para SETTINGS_INITIAL_WINDOW_SIZE 94
C.3. Medidas para SETTINGS_MAX_HEADER_LIST_SIZE 95
C.4. Medidas para SETTINGS_HEADER_TABLE_SIZE 95
C.5. Medidas para SETTINGS_MAX_FRAME_SIZE 96

D.1. Medidas para SETTINGS_MAX_CONCURRENT_STREAMS 97
D.2. Medidas para SETTINGS_INITIAL_WINDOW_SIZE 98
D.3. Medidas para SETTINGS_MAX_HEADER_LIST_SIZE 99
D.4. Medidas para SETTINGS_HEADER_TABLE_SIZE 99
D.5. Medidas para SETTINGS_MAX_FRAME_SIZE 100

Índice de Ilustraciones

1.1. Metodología de prototipos incrementales utilizada en este trabajo. 4
1.2. A8 open node. 5
1.3. Raspberry Pi 3 Model B. 6
1.4. Placa Arduino y módulo de corriente. 6

2.1. Frames HEADERS y DATA y su correspondencia con HTTP/1.1. Adaptado de [Gri13]. 12
2.2. Disposición de las partes de un frame. Adaptado de [BPT15]. 12
2.3. Codificación de headers en base a tablas dinámicas y estáticas. Adaptado de [Gri13]. 14
2.4. Multiplexación en una conexión. Adaptado de [Gri13]. 14

4.1. Configuración para medidas de Raspberry como servidor. 24
4.2. Conexión vía SSH con dos nodos A8, cliente y servidor. 24

6.1. Peticiones fallidas para servidor nghttpd en Raspberry Pi. 30
6.2. Peticiones fallidas para servidor libevent-server en Raspberry Pi. 32
6.3. Peticiones fallidas para servidor nghttpd en nodo A8. 33
6.4. Medidas erróneas de uso de CPU para diferentes números de streams ocasionadas por procesos hijos "fantasma" de la librería nghttpd. 34
6.5. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_MAX_CONCURRENT_STREAMS. 37
6.6. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_CONCURRENT_STREAMS. 37
6.7. Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_CONCURRENT_STREAMS. 38
6.8. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_INITIAL_WINDOW_SIZE. 39
6.9. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_INITIAL_WINDOW_SIZE. 39
6.10. Potencia promedio para diferentes valores del parámetro SETTINGS_INITIAL_WINDOW_SIZE. 40
6.11. Porcentaje de uso de CPU para distintos valores del parámetro SETTINGS_MAX_FRAME_SIZE. 40
6.12. Porcentaje de uso de memoria RAM para distintos valores del parámetro SETTINGS_MAX_FRAME_SIZE. 41

6.13. Potencia promedio para distintos valores del parámetro SETTINGS_MAX_-FRAME_SIZE. 41
6.14. Porcentaje de uso de CPU para distintos valores del parámetro SETTINGS_-HEADER_TABLE_SIZE. 42
6.15. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_HEADER_TABLE_SIZE. 43
6.16. Potencia promedio para SETTINGS_HEADER_TABLE_SIZE. 43
6.17. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_HEADER_LIST_SIZE. 44
6.18. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_HEADER_LIST_SIZE. 44
6.19. Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-HEADER_LIST_SIZE. 45
6.20. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_CONCURRENT_STREAMS. 47
6.21. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_CONCURRENT_STREAMS. 47
6.22. Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-CONCURRENT_STREAMS. 48
6.23. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-INITIAL_WINDOW_SIZE. 48
6.24. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_INITIAL_WINDOW_SIZE. 49
6.25. Potencia promedio para diferentes valores del parámetro SETTINGS_INITIAL_-WINDOW_SIZE. 49
6.26. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_FRAME_SIZE. 50
6.27. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_FRAME_SIZE. 50
6.28. Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-FRAME_SIZE. 51
6.29. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-HEADER_TABLE_SIZE. 52
6.30. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_HEADER_TABLE_SIZE. 53
6.31. Potencia promedio para diferentes valores del parámetro SETTINGS_HEA-DER_TABLE_SIZE. 53
6.32. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_HEADER_LIST_SIZE. 54
6.33. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_HEADER_LIST_SIZE. 54
6.34. Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-HEADER_LIST_SIZE. 55
6.35. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_CONCURRENT_STREAMS. 57
6.36. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_CONCURRENT_STREAMS. 57

6.37. Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-CONCURRENT_STREAMS. 58
6.38. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-INITIAL_WINDOW_SIZE. 58
6.39. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_INITIAL_WINDOW_SIZE. 59
6.40. Potencia promedio para diferentes valores del parámetro SETTINGS_INITIAL_-WINDOW_SIZE. 59
6.41. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_FRAME_SIZE. 60
6.42. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_FRAME_SIZE. 61
6.43. Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-FRAME_SIZE. 61
6.44. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_HEADER_LIST_SIZE. 62
6.45. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_HEADER_LIST_SIZE. 63
6.46. Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-HEADER_LIST_SIZE. 63
6.47. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-HEADER_TABLE_SIZE. 64
6.48. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_HEADER_TABLE_SIZE. 65
6.49. Potencia promedio para diferentes valores del parámetro SETTINGS_HEA-DER_TABLE_SIZE. 65
6.50. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_CONCURRENT_STREAMS. 67
6.51. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_CONCURRENT_STREAMS. 67
6.52. Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-CONCURRENT_STREAMS. 68
6.53. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-INITIAL_WINDOW_SIZE. 69
6.54. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_INITIAL_WINDOW_SIZE. 70
6.55. Potencia promedio para diferentes valores del parámetro SETTINGS_INITIAL_-WINDOW_SIZE. 70
6.56. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_FRAME_SIZE. 71
6.57. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_FRAME_SIZE. 71
6.58. Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-FRAME_SIZE. 72
6.59. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_HEADER_LIST_SIZE. 73

6.60. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_HEADER_LIST_SIZE. 73
6.61. Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-HEADER_LIST_SIZE. 74
6.62. Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-HEADER_TABLE_SIZE. 75
6.63. Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_HEADER_TABLE_SIZE. 75
6.64. Potencia promedio para diferentes valores del parámetro SETTINGS_HEA-DER_TABLE_SIZE. 76

Capítulo 1

Introducción

El término Internet de las Cosas, *Internet of Things* o IoT es un término de aparición relativamente reciente en el que, a diferencia del modelo web de conexión entre computadores, hay máquinas restringidas (en términos de procesamiento, almacenamiento, alimentación energética, entre otros) que se comunican entre sí [TATM15]. Estas restricciones han llevado al surgimiento de nuevos protocolos diseñados específicamente para este escenario, como CoAP [SHB14] o MQTT [OAS15].

Sin embargo, estos protocolos pueden permitir la aparición de nuevos tipos de ataques y generar problemas de interoperabilidad. Además, al ser protocolos recientemente introducidos, suele existir menos personal capacitado para trabajar con ellos [MCLS16]. Por otro lado, HTTP/1.1 [FGM+99] es un protocolo seguro, probado y ampliamente conocido, aunque no fue concebido para los entornos restringidos de IoT.

De acuerdo a los resultados de la IoT Developer Survey 2018 [Cab18], en estos ambientes el protocolo más usado es MQTT, seguido de cerca por HTTP; mientras que la principal preocupación de los desarrolladores encuestados es la seguridad. También, respecto al año anterior, se observa un aumento en el uso de HTTP/2 de un 16,8 % a un 24,9 %, en detrimento de HTTP/1.1.

HTTP/2 [BPT15] corresponde a la nueva versión de HTTP o *Hypertext Transfer Protocol*, que es el protocolo de comunicación para la transferencia de información en la *World Wide Web*. Esta nueva versión agrega funcionalidades que permiten agilizar la transferencia de información. El protocolo tiene distintos parámetros ajustables que pueden reducir el uso de ancho de banda, los requerimientos de RAM, entre otros; por lo que se argumenta que podría ser un protocolo adecuado para IoT [MCLS16].

Desde que en 2015 la Internet Engineering Task Force [For86] aprobó la especificación de HTTP/2 [IET15], tanto el WiNet Research Group [Gro] como el NIC Chile Research Labs [Lab] en la Universidad de Chile, han realizado algunos estudios sobre el comportamiento de HTTP/2 en IoT [LGC+16] [LC16] [Lab18], pero aún no se ha determinado un perfil completo y claro que mejore el desempeño en estos escenarios.

1.1. Motivación

La Internet de las Cosas está actualmente presente en nuestras casas y lugares de trabajo o estudio, a través de artículos de línea blanca, de iluminación, de seguridad, entre muchos otros, que pueden ser manipulados a distancia. La cantidad de "cosas" conectadas a Internet, superó el número de habitantes del planeta en el año 2008 y se estima que para el año 2020 alcanzará los 50 mil millones [Eva11]. El potencial de esta industria es enorme; se espera, no solo que produzca grandes ahorros tanto para las empresas como para las personas [New18], sino que sobre todo, signifique mejoras para la calidad de vida de la población a nivel mundial. Hoy en día se continúa buscando nuevas aplicaciones para IoT.

Pese a lo anterior, el desarrollo de IoT presenta algunas dificultades, especialmente en lo que respecta a seguridad e interoperabilidad [MCLS16]. Dado que han surgido distintos protocolos creados para estos entornos, se ha originado una mayor superficie de ataque a la vez que se ha dificultado la comunicación entre dispositivos. La seguridad es crucial en estos entornos, precisamente porque en muchas ocasiones se trata de artículos ubicados al interior de domicilios, que pueden contener información personal o brindar acceso a zonas restringidas.

1.2. Definición del problema

Pese a que HTTP es un protocolo conocido, seguro y probado ampliamente como protocolo de aplicación en entornos web, a la fecha no se ha determinado el valor adecuado de los parámetros de configuración de HTTP/2 para una operación eficiente en entornos restringidos de Internet de las Cosas.

1.3. Objetivos

1.3.1. Objetivo general

El objetivo de esta memoria es proponer una recomendación, en forma de un ajuste de parámetros, del protocolo HTTP/2 que permita operar eficientemente en ambientes restringidos de Internet de las Cosas que actúan sobre plataformas A8 open node y Raspberry Pi.

1.3.2. Objetivos específicos

Para lograr el objetivo general de la memoria, se debe cumplir con varios objetivos específicos detallados a continuación:

- Investigar sobre la operación de HTTP/2, los parámetros relevantes en entornos restringidos, y las implementaciones disponibles para el *hardware* y/o sistema operativo a utilizar para el proyecto.
- Diseñar y realizar experimentos que permitan determinar el efecto del ajuste de los parámetros de HTTP/2 para cada plataforma de implementación.
- Proponer un ajuste común de parámetros para las dos plataformas de implementación con base en los resultados experimentales.
- Evaluar si el ajuste de parámetros encontrado se comporta mejor que los valores por defecto sugeridos en los estándares de HTTP/2 y HPACK.

1.4. Metodología

En esta sección primero se enumeran las fases del proceso, para luego describir el *hardware* utilizado. Una descripción detallada de la configuración experimental se encuentra en el Capítulo 4.

1.4.1. Fases del proceso

Los fases principales del trabajo que aquí se presenta fueron:

- Entender el funcionamiento de HTTP/2 y HPACK.

  Para esto fue necesario investigar y leer algunas publicaciones. En particular, el documento más importante de esta fase fue el RFC de HTTP/2 [BPT15], donde el protocolo se describe en detalle, y el RFC de HPACK [PR15], que especifica el formato de compresión de los encabezados de HTTP/2.
- Determinar los parámetros relevantes para IoT.

  Esto se llevó a cabo por medio de la lectura de distintos documentos [MCLS16] [MCLS17] y la revisión de trabajos anteriores [LGC+16] [LC16] [Lab18]. Así como también en base al funcionamiento del protocolo, abordado en el punto anterior.
- Encontrar una implementación de HTTP/2 que incluyera los parámetros relevantes y pudiera ser usada en los dispositivos en cuestión.

  Se investigó sobre algunas implementaciones de HTTP/2 como: Deuterium [Sim15], H2O [h2o18] y Nghttp2 [Tsu15]. En el caso de Deuterium, la librería no es de código abierto. Por otro lado, la librería H2O corresponde principalmente a un servidor optimizado para HTTP/1.x y HTTP/2, que incluye un cliente de HTTP solo para la versión 1. En cambio, Nghttp2 corresponde a una implementación de HTTP/2 de código abierto con implementación de servidor, cliente y proxy, que además incluye un herramienta de *benchmarking*. Por estas razones, esta última fue la librería escogida para la realización de los experimentos.
- Investigar el tipo de tráfico predominante en IoT.

  En este punto se recurrió a recopilar distintos documentos que trataran el tema en cuestión. Para decidir el tráfico más representativo de un escenario de la Internet de

las Cosas se puso énfasis en que el tipo de tráfico estudiado incluyera distintos tipos de dispositivos IoT, que se tratase de un estudio relativamente largo y que incorporara distintas métricas relevantes para este trabajo.

El estudio elegido fue "Characterizing and Classifying IoT Traffic in Smart Cities and Campuses" [SSG+17] que caracteriza el tráfico IoT para distintos tipos de dispositivos (cámaras, ampolletas, impresoras, detectores de humo, parlantes, etc.) observando su comportamiento durante un período de 3 semanas. Este estudio incluye métricas como: el tamaño promedio de los paquetes enviados, la frecuencia de envío y el tiempo que los dispositivos IoT permanecen activos o inactivos.

- Para cliente y servidor en ambas plataformas: diseñar experimentos, con tipo de tráfico IoT, que permitan observar el efecto del ajuste de cada parámetro.

  Con base en el estudio anterior sobre tráfico en IoT, se determinó que el tamaño de paquete más representativo de IoT es 500 bytes y que los dispositivos IoT no permanecen activos más de 5 segundos. Por lo tanto, los experimentos realizados corresponden a enviar una gran cantidad de solicitudes desde el cliente, recibiendo 500 bytes desde el servidor, mientras se mide el uso de CPU, memoria y consumo energético de ambos.

  La metodología utilizada es un modelo de prototipos incrementales donde, luego de investigar, se analizaba y diseñaba un experimento, para posteriormente implementarlo, probarlo y con ello, obtener algunos resultados. Al analizar estos resultados, se investigaba más de ser necesario, y se realizaba un nuevo diseño. Los pasos seguidos se muestran en la Figura 1.1

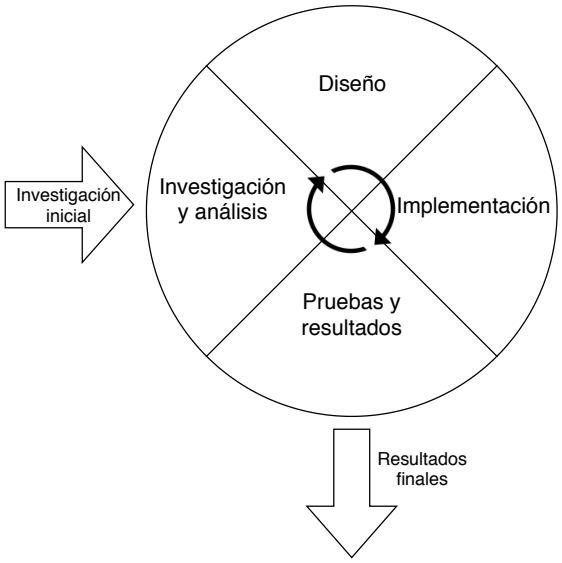


Figura 1.1: Metodología de prototipos incrementales utilizada en este trabajo.

Por ejemplo, en un principio, el servidor recibía solicitudes de un solo cliente que enviaba peticiones continuamente, siempre esperando a recibir la respuesta de la anterior antes de enviar la siguiente. Al no apreciarse cambios en las métricas estudiadas, se optó por realizar un *benchmarking* al servidor para establecer la cantidad de solicitudes que podía responder en paralelo sin que hubieran solicitudes perdidas por *timeout*, y tomar las mediciones con el servidor saturado de peticiones. Del lado del cliente, se aumentó la cantidad de solicitudes a ser observadas, manteniendo la restricción de esperar la

respuesta de una solicitud antes de enviar la siguiente, por ser más representativo del tráfico esperado de la Internet de las Cosas.

En cada experimento, se elegía un valor del parámetro de HTTP/2 que se deseaba observar. La cantidad de experimentos correspondía entonces a un cierto número de valores del parámetro, elegidos dentro del rango posible. Para cada experimento del servidor con Rasperry Pi, se simularon 38 clientes enviando en conjunto un total de 32.768 solicitudes (cerca de 862 solicitudes cada uno). Mientras que para las mediciones del cliente, se realizaron 900 solicitudes. Para los nodos A8, cada experimento sobre el servidor consideraba 96 clientes y una cantidad de solicitudes en conjunto de 131.072 (alrededor de 1365 cada uno), y 1400 solicitudes para el caso de las mediciones del nodo A8 como cliente. Las razones para elegir estas cantidades se explican en el Capítulo 6. Además, como validación, cada experimento se repitió 30 veces.

- Con base en los resultados obtenidos del punto anterior, determinar si existe alguna configuración que mejore el desempeño para cada plataforma en particular y para ambas en conjunto.

  Los resultados obtenidos de los experimentos fueron graficados y analizados para llegar a las conclusiones que se detallan en el Capítulo 7.

1.4.2. *Hardware* utilizado

A8 open node

IoT-LAB [otT13] es un laboratorio de IoT provisto por FIT, un consorcio francés formado por cinco instituciones de educación superior e investigación, entre ellas INRIA [INR67]. Este laboratorio dispone de un sitio web en el que es posible solicitar la conexión remota a distintos nodos y correr programas en ellos.

A8 open node [IL] o simplemente nodo A8 (Figura 1.2) es el *hardware* más poderoso de IoT-LAB y permite utilizar sistemas operativos como Linux. Posee un procesador ARM Cortex-A8 de 600 Mhz y 256 MB de RAM. Tiene conectividad Ethernet y USB.

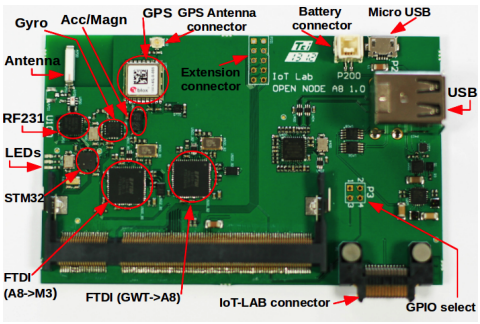


Figura 1.2: A8 open node.

Raspberry Pi 3 Model B

Raspberry Pi (Figura 1.3) es un computador reducido al que se le puede instalar un sistema operativo. Para este trabajo se hace uso de Raspberry Pi 3 Model B [Fou] con Ubuntu Mate. Este dispositivo posee un procesador Quad Core 1.2GHz Broadcom BCM2837 de 64bit y de 1GB RAM, y conectividad USB, Ethernet y Wifi.

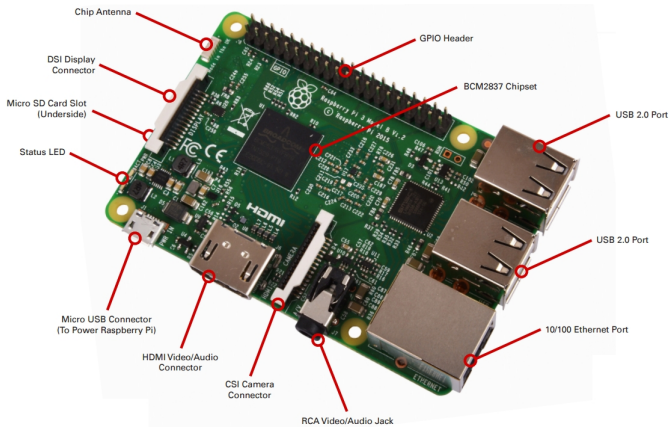
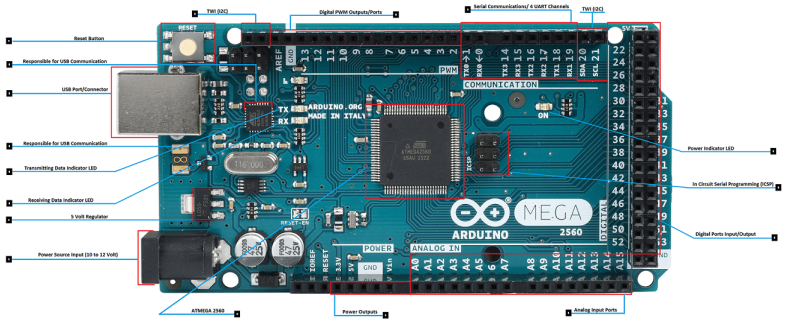


Figura 1.3: Raspberry Pi 3 Model B.

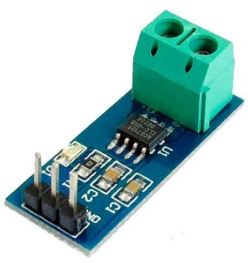
Arduino Mega

La placa Arduino utilizada corresponde a Arduino Mega [Ard], una placa de desarrollo *open-source* con 256 KB de memoria. Es posible adosarle módulos que entregan funcionalidades adicionales. Por ejemplo, no posee conectividad Ethernet ni Wifi pero al agregarle ciertos módulos es posible entregarle esta conectividad.

En este caso, se hizo uso del módulo ACS712 para medición de corriente.



(a) Arduino Mega.



(b) Módulo Arduino ACS712.

Figura 1.4: Placa Arduino y módulo de corriente.

1.5. Organización del documento

En términos generales, este documento pretende detallar el análisis realizado a HTTP/2 para un escenario IoT y las pruebas llevadas a cabo sobre los parámetros de este protocolo que podrían tener alguna incidencia en el uso de recursos de CPU, memoria y consumo energético.

En el capítulo 2 (Marco teórico) se explican los conceptos que es necesario conocer para comprender el trabajo presentado. Luego, en el capítulo 3 (Estado del arte) se muestran trabajos de otros autores relacionados con él. En el capítulo 4 (Análisis y Diseño) se describen las herramientas utilizadas y el análisis realizado para que las pruebas efectivamente permitan medir las distintas configuraciones de los parámetros. También se incluyen las configuraciones experimentales. Posteriormente en el capítulo 5 (Implementación) se describe el funcionamiento del código usado para las distintas pruebas, además de algunas consideraciones de importancia. A continuación, en el capítulo 6 (Resultados) se exponen los resultados de los experimentos realizados y reflexiones al respecto. Finalmente, en el capítulo 7 (Discusión) se presenta una discusión del problema planteado, para terminar con el capítulo 8 (Conclusión) donde se detallan las conclusiones del proceso y se proponen pruebas futuras.

Capítulo 2

Marco Teórico

Para comprender el trabajo que aquí se presenta, se debe estar familiarizado con algunos conceptos de redes, en este caso: Internet de las cosas, tráfico en IoT, tecnologías de comunicación en IoT, HTTP/2 y TLS.

2.1. Internet de las cosas

El término Internet de las Cosas, *Internet of Things* o IoT, es un término propuesto por Kevin Ashton en 1999 [HV18], con una arquitectura típicamente M2M o *machine-to-machine*. Esto quiere decir que distintos dispositivos se comunican entre sí sin necesidad de interacción humana. Los dispositivos que participan en IoT son restringidos en términos de procesamiento, almacenamiento, alimentación energética, entre otros; lo que ha llevado al surgimiento de nuevos protocolos diseñados específicamente para este escenario.

De acuerdo a la IoT Developer Survey 2018 [Cab18], los protocolos que los encuestados afirman usar para sus soluciones IoT, y por lo tanto los más usados son: MQTT (62,6 %), HTTP/1.x (54,1 %), Websockets (34,9 %), HTTP/2 (24,9 %) y CoAP (22,5 %).

MQTT (*Message Queue Telemetry Transport*) [OAS15] fue diseñado para este escenario. Es un protocolo binario que requiere muy poco ancho de banda. Por otro lado, tanto HTTP/1.x como HTTP/2 (descrito en la sección 2.2) son protocolos web. Por su parte, los Websockets corresponden a un protocolo que provee un canal de comunicación bidireccional y *full-duplex* sobre un único socket TCP; al igual que HTTP, presenta un esquema cliente-servidor. Por último, CoAP o Constrained Application Protocol [SHB14] es un protocolo creado para dispositivos restringidos; entre otras cosas, CoAP funciona sobre UDP y está diseñado para ser fácilmente traducido a HTTP y así permitir a estos dispositivos restringidos comunicarse con el resto de la web.

2.1.1. Tráfico en IoT

La caracterización del tráfico en IoT considerada inicialmente en este trabajo, es la descrita en "Characterizing and classifying IoT traffic in smart cities and campuses" [SSG+17]. Este artículo fue elegido por la diversidad de dispositivos IoT incluidos en el estudio, por el largo del tiempo de observación de estos y también por las distintas variables consideradas.

En esta caracterización del tráfico IoT se hizo uso de la herramienta `tcpdump` para analizar el tráfico de la red. Se realizaron observaciones de protocolos utilizados, tamaños de paquete, frecuencia de envío de paquetes, y tiempo activo e inactivo de las máquinas. Estas mediciones fueron realizadas durante el transcurso de tres semanas en un campus universitario en el que existía tráfico IoT y tráfico regular.

Al concluir el estudio del tráfico en cuestión, se creó un algoritmo de clasificación que permitió identificar de forma única a los dispositivos dentro de la red en base a su comportamiento con un 95 % de precisión.

La caracterización descrita en el documento corresponde a lo siguiente:

Tiempo de actividad

El tiempo de actividad, o intervalo en el que un dispositivo intercambia paquetes, suele ser de corta duración. En el estudio mencionado, se encuentra un tiempo máximo de actividad de 250 segundos, donde solo el 5 % de las sesiones duran más de 5 segundos. Mientras que el tiempo que un dispositivo IoT permanece inactivo es inferior a 20 segundos el 85 % de las veces, y superior a un minuto solo el 4 % de ellas. Esto significa que los dispositivos IoT se activan y desactivan con mucha frecuencia.

Tamaño de paquete y volumen de tráfico

Se observa que solo el 10 % de los paquetes transmitidos pesa más de 500 bytes. Respecto al volumen de datos intercambiado, el 75 % de las sesiones transfieren menos de 1 kB, y menos de un 1 % intercambia más de 10 kB. Esto sugiere que la mayoría de los dispositivos IoT generan cantidades de tráfico pequeñas en relación al tráfico web.

2.1.2. Tecnologías de comunicación para IoT

Existen distintas alternativas de comunicación para IoT, que varían en términos de tasa de transmisión, distancia óptima de los dispositivos, entre otros. A continuación se mencionan algunas.

WiFi

WiFi es una tecnología inalámbrica basada en uno de los estándares 802.11 [std16]. Es la manera en que los dispositivos, desde una red de área local, se suelen conectar inalámbricamente a Internet. Tiene una velocidad teórica máxima de transmisión de 600 Mbps, y una distancia de operación óptima de 50 metros en interiores y de 100 en exteriores.

Se eligió WiFi por ser compatible con Raspberry Pi 3 Model B, por ser la tecnología inalámbrica más conocida y porque su velocidad de transmisión es suficiente para no interferir ni ralentizar el tráfico de la configuración experimental.

Ethernet

Ethernet [Ecu] es una tecnología de transmisión cableada definida en el estándar 802.3 y muy comunmente usada con computadores, siendo no tan común en IoT, ya que muchos de estos dispositivos no disponen de un puerto Ethernet. Las velocidades comúnmente desplegadas son de Fast Ethernet, que permite la transmisión de hasta 100 Mbps y GigabitEthernet a 1 Gbps.

La principal razón para el uso de esta tecnología, es que permite aislar el desempeño de los dispositivos del desempeño de la red, al ser esta una red cableada. También es la tecnología disponible para nodos A8 puros, ya que para usar una tecnología inalámbrica con estos nodos es necesario añadir un subsistema adicional llamado M3.

802.15.4

IEEE 802.15.4 [CGH$^+$02] es un estándar, con aplicaciones en redes domésticas, que define tanto el nivel físico, como el control de acceso al medio de redes inalámbricas con tasas bajas de transmisión de datos. Permite la comunicación en un área de 10 metros con una tasa de transferencia de 250 kbps. Entre sus principales características están su bajo costo y su bajo consumo de energía. Además, presenta "flexibilidad de red", lo que significa que es capaz de adaptarse a los cambios en los requerimientos de red, como flujo o topología.

Zigbee

Zigbee es una aplicación basada en el estándar 802.15.4 para crear redes personales o domésticas. Tiene bajo consumo y baja velocidad de transmisión de datos. Es una tecnología más simple y barata que la de otras redes inalámbricas como WiFi. Su bajo consumo limita la distancia de transmisión a entre 10 y 100 metros, dependiendo de las condiciones del ambiente y tiene una tasa de transferencia de 250 kbps, la misma que 802.15.4 [Wor].

LoRa (Long-Range)

LoRa [LA17] es una aplicación con funciones similares a las de Zigbee, que permite cubrir áreas de entre 2 y 5 kilómetros en zonas urbanas, pero que sin embargo, presenta un consumo de energía inferior. Puede transmitir a una tasa de 0,3 a 22 kbps por segundo con LoRa modulation o a 100 kbps con GFSK [Mil16].

Tecnologías elegidas

De las tecnologías mencionadas anteriormente, de acuerdo a la IoT Developer Survey 2018 [Cab18], los protocolos de capa de enlace de datos más populares en la Internet de las Cosas son WiFi con un 55,3 % de los desarrolladores encuestados que afirman utilizarlo en sus soluciones IoT, seguido por Ethernet con un 47,7 %. Por lo tanto, estas fueron las tecnologías utilizadas en los experimentos.

Ya que para los nodos A8 solo se contaba con conectividad Ethernet, y se deseaba probar ambas tecnologías, se optó por realizar los experimentos con Raspberry Pi usando WiFi. La diferencia principal entre ambos protocolos es que al ser Ethernet cableado, permite aislar el protocolo de las interferencias que podrían existir en una red inalámbrica, lo que no necesariamente ocurre en WiFi. Sin embargo, la red WiFi local utilizada en los experimentos era una red estable y solo se conservaron los resultados de experimentos en los que no se presentaron errores de red.

2.2. HTTP/2

HTTP/2 es la nueva versión del protocolo HTTP, el protocolo de uso más común en entornos web. Este protocolo busca solucionar algunas de las falencias de HTTP/1.1, lo que es fundamental para reducir los tiempos de carga de las páginas web actuales que consisten de más objetos que sus predecesoras [Htt].

Este protocolo mantiene el funcionamiento básico de su versión anterior, correspondiente al esquema cliente-servidor (*client-server*) con petición-respuesta (*request-response*), así como también su semántica. Sin embargo, se trata de un protocolo binario y no de texto como su predecesor, además de presentar otras diferencias importantes que se describen a continuación.

2.2.1. Frames

El frame (Figura 2.1) es la unidad básica del protocolo. Existen tipos de frames como HEADERS y DATA, que se usan para enviar encabezados y datos (o mensajes) respectivamente. No obstante, algunas de las características de HTTP/2 requieren de otros tipos de frame, como: SETTINGS, WINDOW_UPDATE y PUSH_PROMISE.

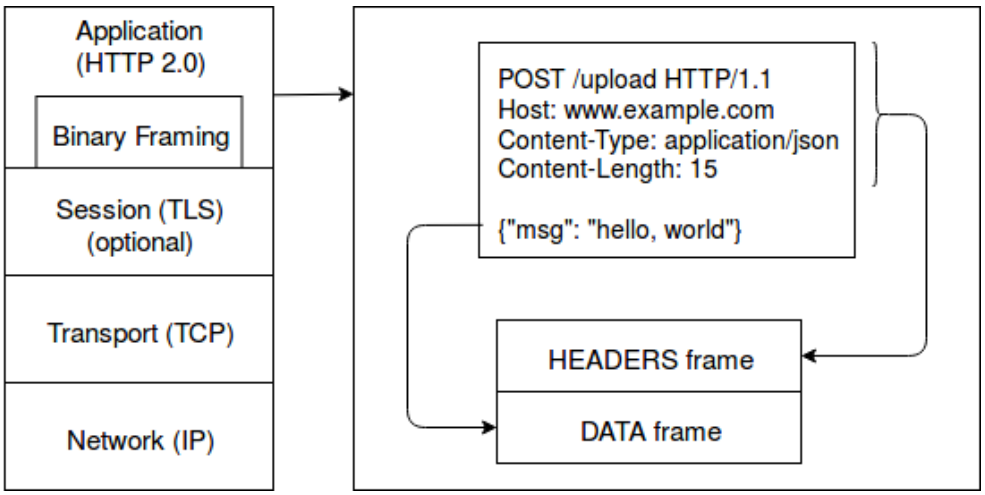


Figura 2.1: Frames HEADERS y DATA y su correspondencia con HTTP/1.1. Adaptado de [Gri13].

El formato de un frame es el siguiente:

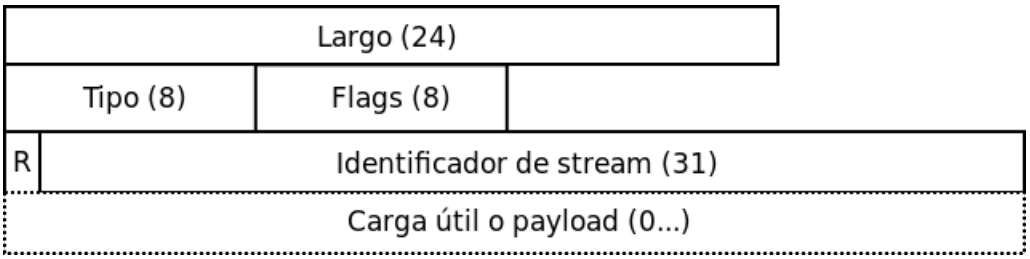


Figura 2.2: Disposición de las partes de un frame. Adaptado de [BPT15].

La carga útil o *payload* del frame corresponde al mensaje que efectivamente se desea enviar. El largo corresponde al largo de este payload, representado como un unsigned integer de 24 bits, que no incluyen los 9 octetos (bytes) del frame del *header*.

HTTP/2 permite definir el tamaño máximo de este payload a través del parámetro SETTINGS_MAX_FRAME_SIZE, pudiendo tomar valores desde 2^{14} (16.384) a $2^{24} - 1$ (16.777.215) octetos (bytes) inclusive. Si se desea enviar un frame mayor a 16.384 bytes, el receptor de este frame debe tener definido un tamaño de frame mayor al establecido por su contraparte. De lo contrario, ocurre un error de tipo FRAME_SIZE. Este error también sucede si el tamaño de frame elegido no está dentro del rango aceptado.

2.2.2. Header o encabezado

HTTP es un protocolo "sin estado", esto quiere decir que los pares petición-respuesta son tratados independientes de los otros pares; y por lo tanto, cada peticion debe incluir todos los detalles que el servidor necesita conocer para poder responderla adecuadamente. Lo anterior implica la necesidad de enviar encabezados con metadatos e información repetitivos con el consecuente mal uso del ancho de banda disponible [Ste].

Para aprovechar mejor los recursos disponibles, HTTP/2 es un protocolo binario, al contrario de HTTP/1.1 que hace uso de texto plano. Pero además, con la misma finalidad, HTTP/2 dispone de compresión para el header llamada HPACK [PR15].

2.2.3. HPACK

En la actualidad una página web promedio tiene un tamaño de 1,8 MB [Htt]. Si se supone que esa página tiene entre 80 y 100 objetos, y que cada request tiene 1.400 bytes de encabezado, se necesitan alrededor de 8 o 9 frames para poder enviar esta información. La idea de HPACK es comprimir los encabezados haciendo uso de la codificación Huffman y de dos tablas, una estática y otra dinámica.

La tabla estática está compuesta de 62 filas con los índices, encabezados y valores de los headers más comúnmente encontrados en HTTP. Como se puede observar en la Figura 2.3, la tercera fila de la tabla estática (índice 2) contiene en sus columnas: 2, :method, GET.

Respecto a la tabla dinámica, inicialmente está vacía. Cuando el cliente manda una solicitud, puede indicar en el bloque del header si un header en particular y su valor deben ser indexados. Con base en esto se crea una entrada en la tabla. Esta tabla, a diferencia de la anterior, puede ser modificada. Por ejemplo, una fila de la tabla podría contener: 64, :header:, text/plain.

En el lado del servidor, al leer los encabezados recibidos, se creará la misma tabla. La siguiente vez que el cliente envíe una petición, si está enviando los mismos encabezados, ya no necesitará enviar la solicitud completa. Podrá simplemente enviar un bloque de encabezados. El servidor verá estos encabezados y los expandirá de acuerdo a los valores de las tablas.

El tamaño máximo de la tabla dinámica se puede modificar mediante el parámetro SETTINGS_HEADER_TABLE_SIZE. Este parámetro tiene por defecto el valor 4.096, pudiendo ser tan pequeño como 0 o tan grande como se desee. Sin embargo, es importante mencionar que la contraparte del *endpoint* que reciba este valor, debe definir para sí mismo un valor igual o menor que el señalado.

Por otro lado, una lista de encabezados es una colección ordenada de nombres de campos con uno o más valores asociados que se envían en una petición, como se ve en la tabla de la izquierda en la Figura 2.3. El tamaño máximo de la lista de encabezados que el emisor está dispuesto a aceptar se puede limitar con el parámetro SETTINGS_MAX_HEADER_LIST_SIZE, considerando el tamaño de los encabezados sin comprimir, incluyendo el largo del nombre y valor en bytes más un espacio adicional de 32 bytes por cada campo.

2.2.4. Streams

Para poder explicar el concepto de *stream*, es importante hacer mención a TCP [Pos81].

TCP, Transmision Control Protocol o Protocolo de Control de Transmisión, es un proto-

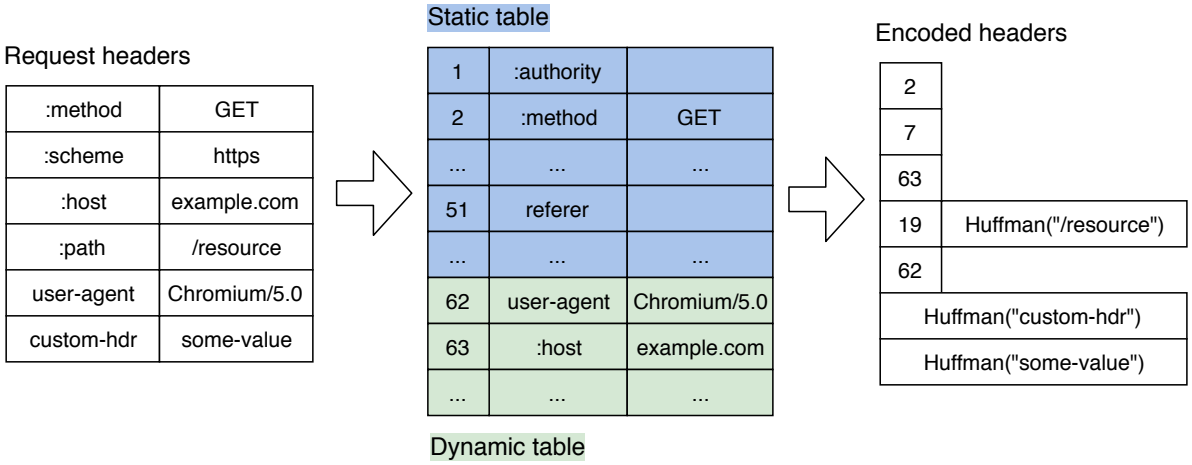


Figura 2.3: Codificación de headers en base a tablas dinámicas y estáticas. Adaptado de [Gri13].

colo del que todas las versiones de HTTP hacen uso. TCP permite crear conexiones entre dos dispositivos y garantiza que los datos sean entregados sin errores y en el mismo orden en el que fueron transmitidos.

Para transmitir datos, HTTP establece una conexión TCP entre dos máquinas. En HTTP/1.1, al establecer una conexión se hace uso de *pipelining*, esto implica enviar todas las peticiones HTTP una después de otra dentro de la misma conexión, y luego esperar las respuestas a cada una de ellas en orden. Si bien esto es una mejora respecto a HTTP/1.0, donde para cada petición debía esperarse una respuesta antes de enviar la siguiente, sigue sin ser óptimo.

Para poder lograr menores tiempos de carga, HTTP/2 hace uso de streams y multiplexación de peticiones (Figura 2.4). Un stream corresponde a un flujo bidireccional de frames dentro de una sola conexión TCP. Cada stream tiene un identificador, como puede apreciarse en la Figura 2.2. Una conexión TCP en HTTP/2 puede tener múltiples streams concurrentes.

Los streams son independientes uno de otro y la multiplexación se logra al asociar cada solicitud con su propio stream. Multiplexar en este caso, significa que los paquetes de distintos streams viajan intercalados en la misma conexión. De este modo, si alguna petición o respuesta se demora más que el resto, no genera un bloqueo.

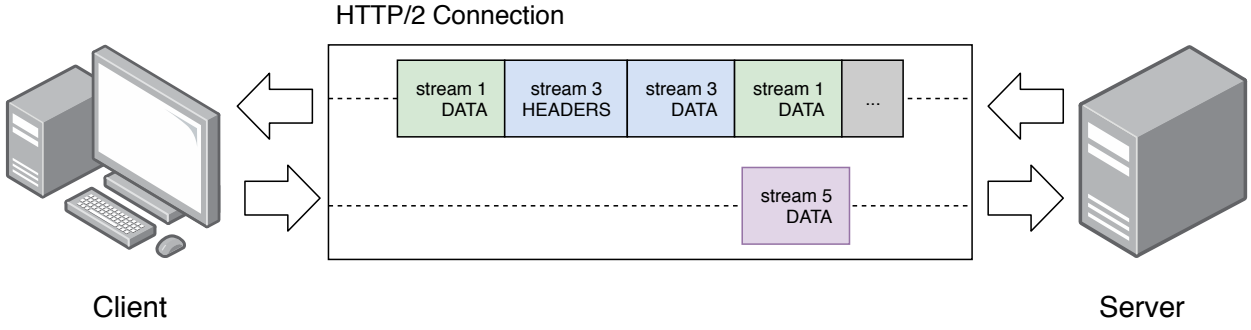


Figura 2.4: Multiplexación en una conexión. Adaptado de [Gri13].

2.2.5. Server Push

Server Push es un mecanismo que permite enviar datos o archivos antes de que sean solicitados por el cliente con la finalidad de mejorar el desempeño en la carga de páginas web. Fue creado para páginas web dinámicas y viene habilitado por defecto. Sin embargo, puede hacer mayor uso del ancho de banda, lo que en el caso de un nodo IoT puede implicar un mayor uso de batería. Además, puede enviar datos o archivos que ya se encuentran en el caché del cliente.

Server Push puede ser habilitado o deshabilitado a través del parámetro SETTINGS_ENABLE_PUSH.

2.2.6. Control de flujo

El control de flujo es un mecanismo que previene que el enviador sature al receptor con información que no es capaz de procesar. Esto podría ocurrir si el receptor se encuentra ocupado, si ya tiene mucha carga o si existen recursos limitados para un determinado stream.

El control de flujo en HTTP/1.x se usa para TCP. Sin embargo, esto no resulta lo suficientemente granular para el caso de HTTP/2 debido a la multiplexación. Se tiene entonces una ventana de control de flujo para cada stream dentro de la conexión. El tamaño inicial de esta ventana puede ser indicado con el parámetro SETTINGS_INITIAL_WINDOW_SIZE que toma valores entre 2^0 y $2^{31} - 1$ bytes, con un valor por defecto de $2^{16} - 1$ (65.535) bytes.

2.2.7. TLS

HTTP no implementa seguridad por sí mismo. Para esto, usa TLS o *Transport Layer Security* [DR08] que corresponde a un protocolo de seguridad de la capa de transporte. En HTTP/2 el uso de este protocolo es opcional. Su uso genera un retardo adicional, ya que requiere de un tiempo de negociación [LG17].

Capítulo 3

Estado del arte

En este capítulo se exponen trabajos de otros autores realizados recientemente y relacionados con el tema de estudio de esta memoria.

3.1. Elección de un protocolo efectivo para IoT

El artículo "Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP" [Nai17] aborda la discusión de la elección de un protocolo estándar y efectivo para la Internet de las Cosas, presentando una evaluación de los protocolos MQTT, CoAP, AMQP y HTTP/1.x, en términos de latencia, confiabilidad, seguridad, interoperabilidad, consumo energético, tamaño de mensaje, etc.

Este estudio afirma que es posible que no exista un protocolo que permita unificar IoT, ya que los escenarios de Internet de las Cosas son muy variados y, por lo tanto, puede que para ciertas aplicaciones sea más adecuado cierto protocolo, pero que ese protocolo no tenga el mejor rendimiento en una situación distinta.

3.2. Comparación de HTTP y MQTT en recursos de red para IoT

El estudio "Comparison with HTTP and MQTT on Required Network Resources for IoT" [YS16] compara los protocolos MQTT y HTTP/1.x, y analiza el sobrecosto (*overhead*) de los mensajes, el uso de ancho de banda, el uso de recursos del servidor, etc.

Como resultado, encuentra que MQTT tiene menor sobrecosto y menor uso de recursos, al ser un protocolo liviano diseñado para Internet de las Cosas. Aunque el uso de recursos del servidor es el mismo para ambos protocolos. Sin embargo, en este artículo se proponen mejoras a MQTT que conllevarían un menor uso de recursos del servidor, eliminando potencialmente esta igualdad.

3.3. Evaluación de ventana para Raspberry Pi

En el artículo "Performance Evaluation of HTTP/2 Window Size in the Internet of Things" [LGC+16] se evalúa el desempeño del parámetro SETTINGS_INITIAL_WINDOW_SIZE de HTTP/2 con la librería Nghttp2 [Tsu15] en términos de uso de CPU, tiempos de respuesta y consumo energético. Los dispositivos usados son dos Raspberry Pi 3 Model B, que actúan como cliente y servidor, y que se encuentran conectados a un switch por medio de *FastEthernet*, formando una red local.

Este es solo uno de los parámetros de HTTP/2 relevantes para IoT y, si bien se analiza el tiempo de respuesta, no se estudia el uso de memoria de los dispositivos. Además, el tipo de tráfico utilizado para las pruebas no corresponde a tráfico IoT, ya que se trata de una página web de alrededor de 7 MB.

3.4. Evaluación de ventana y streams para A8 open node

En el trabajo "TESTHTTP" [Lab18] se evalúan los parámetros SETTINGS_INITIAL_WINDOW_SIZE y SETTINGS_MAX_CONCURRENTS_STREAMS de HTTP/2 para nodos A8 en uso de CPU, memoria, tiempo de respuesta y consumo energético.

Si bien ambos son parámetros relevantes en un entorno IoT, no son todos los parámetros que se debieran evaluar. Además, aunque se analiza el protocolo en un dispositivo restringido, el contenido del servidor corresponde a una página web de aproximadamente 7 MB, muy distinta del tráfico habitual en IoT. Por último, no se realizan pruebas sobre todos los valores posibles de estos parámetros ni se satura el servidor con un gran número de peticiones.

3.5. Tiempo de respuesta y rendimiento de red en Raspberry Pi

En el análisis "Evaluating the Features of HTTP/2 for the Internet of Things" [Elm17] se revisa el protocolo HTTP/2 usando dos Raspberry Pi Model B como clientes, ambas enviando solicitudes a un servidor por medio de una red de área local inalámbrica. El objetivo de este estudio es comparar los protocolos HTTP/1, HTTP/1 con SSL y HTTP/2 con SSL en términos de tiempo de respuesta y rendimiento de la red (velocidad de transmisión).

En este estudio se hace uso de paquetes de tamaño inferior a 1 kB, como suele suceder en IoT; sin embargo la finalidad es otra, ya que se compara el protocolo HTTP/2 con su anterior versión sin considerar su configuración ni establecer el uso de los recursos de los nodos.

3.6. Perfil de configuración de HTTP/2 para IoT

En el internet-draft "HTTP/2 Configuration Profile for the Internet of Things" [MCLS17] se discuten factores importantes en relación a encontrar un perfil completo de los parámetros de HTTP/2 para Internet de las cosas. Se indica que los parámetros relevantes para IoT podrían ser:

- SETTINGS_HEADER_TABLE_SIZE
- SETTINGS_ENABLE_PUSH
- SETTINGS_MAX_CONCURRENT_STREAMS
- SETTINGS_INITIAL_WINDOW_SIZE
- SETTINGS_MAX_FRAME_SIZE
- SETTINGS_MAX_HEADER_LIST_SIZE

También se dan algunas indicaciones sobre cuáles podrían ser los valores apropiados para estos parámetros en Internet de las cosas.

Pese a que esta memoria se basa en este estudio para los parámetros que considera importantes para IoT, las recomendaciones de los valores no son tomadas en cuenta, ya que se analiza el espectro completo de valores para todos los parámetros mencionados.

Capítulo 4

Análisis y diseño

A continuación se describen los experimentos llevados a cabo con anterioridad a este trabajo y luego, los diseñados y realizados en el marco de este estudio.

4.1. Experimentos anteriores

Anterior a este trabajo se habían realizado experimentos con HTTP/2 en nodos A8 y en Raspberry Pi, pero diseñados para aplicaciones web. En estos experimentos, haciendo uso de un servidor web que aloja una página, el cliente realiza una serie de 68 peticiones. Los archivos que componen la página corresponden a un index HTML que contiene un archivo CSS, dos archivos Javascript y cuatro imágenes JPG. Los archivos suman alrededor de 7 MB [LGC+16].

Sin embargo, una página web no es el escenario más común en la Internet de las Cosas. En IoT lo usual es que un nodo actué de sensor, enviando pequeñas cantidades de información con cierta frecuencia a otro dispositivo, pudiendo ser este otro un dispositivo restringido o no. Con base en el estudio mencionado en 2.1.1, se decide hacer uso de peticiones de alrededor de 500 bytes. Respecto a la frecuencia de estas peticiones, esta depende de los datos del *benchmark* del servidor que aparecen en el Capítulo 5.

4.2. Nghttp2

Para el desarrollo de los experimentos recién mencionados, se hizo uso de la librería Nghttp2 [Tsu15], que es una implementación en lenguaje C de HTTP/2 con HPACK. Esta librería incluye un cliente, un servidor, un proxy y una herramienta de *benchmarking* que pueden ser ejecutados como comandos una vez instalada la librería. Además incluye, entre otros, un servidor y cliente de ejemplo que pueden ser ejecutados como *scripts* de C. Las librerías utilizadas que hacen parte de Nghttp2 son: nghttp, nghttpd, libevent-server y h2load;

y se explican a continuación.

4.2.1. nghttp

El cliente HTTP/2 implementado en la librería se llama nghttp. Su uso es:

`nghttp [OPTIONS]... <URI>...`

Donde la URI debe tener el formato `https://URI:port` o `https://URI:port/file.extension` si se desea acceder a una subpágina en particular. Si no se especifica, la librería accede por defecto a `index.html`. Por ejemplo, el comando `nghttp https://nghttp2.org` realiza una solicitud al sitio `nghttp2.org/index.html` con los parámetros de HTTP/2 por defecto.

Cada vez que este comando se ejecuta, se envía una única petición a una única URI y al recibir la respuesta, el programa termina su ejecución. Esto es útil para simular un nodo realizando una solicitud y apagándose al concluir la transacción.

Opciones de la librería

La librería cuenta con diversas opciones, algunas permiten modificar el valor de algunos de los parámetros de HTTP/2. Las relevantes para este trabajo son los que corresponden a los mencionados en el estudio 3.6, y corresponden a:

- `-w, --window-bits=<N>`. Fija la ventana inicial a nivel de stream en el valor $2^{<N>} - 1$. Esta opción corresponde al parámetro `SETTINGS_INITIAL_WINDOW_SIZE`.
- `-W, --connection-window-bits=<N>`. Modifica el tamaño inicial de la ventana a nivel de conexión en $2^{<N>} - 1$. Para hacer un uso adecuado del control de flujo, se recomienda elegir el mismo valor para este parámetro y el anterior.
- `-c, --header-table-size=<SIZE>`. Especifica el tamaño de la tabla dinámica del decodificador (cliente), o sea de `SETTINGS_MAX_HEADER_TABLE_SIZE`. Si la opción se usa múltiples veces, el valor utilizado es el mínimo de los entregados.
- `--encoder-header-table-size=<SIZE>`. Especifica el tamaño de la tabla dinámica del codificador (servidor). La tabla dinámica negociada es el mínimo entre esta opción y la que el servidor señale.
- `--max-concurrent-streams=<N>`. El número de streams concurrentes que este cliente acepta, equivalente a `SETTINGS_MAX_CONCURRENT_STREAMS`.
- `-M, --peer-concurrent-streams`. Envía un mensaje al servidor diciendo cuántos streams debiera él aceptar.
- `--no-push`. Deshabilitar Server Push. Equivalente a `SETTINGS_ENABLE_PUSH` con valor 0.

El argumento `<N>` es un entero, mientras que el argumento `<SIZE>` es un entero que puede incluir opcionalmente una unidad, como K, M o G (potencias de 1024).

Por ejemplo, el comando:

`nghttp -window-bits=16 -connection-window-bits=16 -header-table-size=4K https://nghttp2.org` envía una petición a nghttp2.org con una ventana inicial a nivel de stream y a nivel de conexión de $2^{16} - 1$ y una tabla dinámica con tamaño máximo de 4 kB.

4.2.2. nghttpd

De manera similar a lo anterior, existe nghttpd que es el servidor de Nghttp2. Su uso es:

```
nghttpd [OPTION]... <PORT>[<PRIVATE_KEY><CERT>]
```

Se distinguen aquí las dos maneras de hacer uso del servidor, con y sin TLS:

- Con TLS: Para usar el servidor con TLS se requiere generar un certificado y una llave. Estos son dos archivos necesarios para el funcionamiento de este protocolo, los cuales son generables fácilmente a través de la librería de código libre OpenSSL [Ope].
- Sin TLS: En este caso se especifica la opción `--no-tls`, con lo que solo se tiene HTTP/2 sin seguridad.

Para usar el servidor con TLS el comando sería, por ejemplo:

```
nghttpd 8030 secret.key secret.crt
```

Mientras que para usarlo sin TLS sería:

```
nghttpd --no-tls 8030
```

Donde 8030 es el puerto a usar en este caso.

Opciones

Respecto al resto de opciones relevantes, nghttpd cuenta con las mismas opciones que nghttp. Sin embargo hay aspectos importantes que mencionar.

- `-w, --window-bits=<N>`. Fija la ventana inicial a nivel de stream. Este valor puede ser distinto en ambos lados del stream.
- `-W, --connection-window-bits=<N>`. Modifica el tamaño inicial de la ventana a nivel de conexión. Se recomienda elegir el mismo valor de `--window-bits=<N>`. Es decir, la ventana inicial a nivel de stream y de conexión deben ser la misma; sin embargo, pueden ser distintas a las especificadas en el lado del cliente, pues son independientes de estas.
- `-c, --header-table-size=<SIZE>`. Especifica el tamaño de la tabla dinámica del codificador (servidor), equivalente al parámetro `SETTINGS_MAX_HEADER_TABLE_SIZE` del lado del cliente. Si la opción se usa múltiples veces, el valor utilizado es el mínimo de los entregados.

- `--encoder-header-table-size=<SIZE>`. Especifica el tamaño de la tabla dinámica del decodificador (cliente). La tabla dinámica negociada es el mínimo entre esta opción y la que el cliente señale en su parámetro `SETTINGS_MAX_HEADER_TABLE_SIZE`.
- `--max-concurrent-streams=<N>`. El número de streams concurrentes que este servidor acepta.
- `-M, --peer-concurrent-streams`. Envía un mensaje al cliente diciendo cuántos streams debiera él aceptar. Como los streams son bidireccionales, es útil poder fijar la misma cantidad de streams para ambos lados de la transacción. De otro modo, el valor es el mínimo entre el valor de ambos `SETTINGS_MAX_CONCURRENT_STREAMS`.

Como se puede observar ni el cliente ni el servidor tienen implementadas las opciones para modificar todos los parámetros que se desea evaluar. En particular, los parámetros SETTINGS_MAX_FRAME_SIZE y MAX_HEADER_LIST_SIZE no pueden ser modificados de esta manera.

4.2.3. libevent-server

La librería Nghttp2 cuenta también con un servidor de ejemplo. El resto de funciones de la librería no dependen de este servidor. Este servidor hace uso de la función `submit_settings` que permite ingresar manualmente parámetros al dárselos como argumentos en un arreglo de pares nombre-valor. Por ejemplo, `{NGHTTP2_SETTINGS_MAX_FRAME_SIZE, 32768}`.

Este servidor resulta entonces un buen candidato para poder realizar pruebas con todos los parámetros de importancia.

El servidor nghttpd también cuenta con esta función, pero se utiliza en un archivo distinto, "Httpserver.cc", que es la base para ambos servidores. Al modificar algún parámetro que no pueda ser modificado por medio de las opciones, se debe recompilar este archivo, así como también nghttpd.

Por otro lado, para poder realizar pruebas en el servidor, es necesario hacer uso de varios clientes que envíen una cantidad importante de peticiones, tal que se puedan ver efectivamente los efectos de las diferencias de parámetros en el protocolo.

4.2.4. h2load

h2load es la herramienta de benchmarking de Nghttp2 para servidores, cuyo uso básico es:

```
h2load [OPTIONS]... [URI]...
```

Opciones

Las opciones relevantes para el caso son:

- `-c, --clients=<N>`. Número de clientes concurrentes. Valor por defecto: 1.
- `-n, --requests=<N>`. Número de requests a enviar entre todos los clientes. Las requests no se distribuyen necesariamente de forma equitativa entre los clientes. Valor por defecto: 1.
- `-D, --duration=<N>`. Especifica la duración para las medidas, es decir, por cuánto tiempo se ejecuta h2load.

h2load cuenta también con opciones análogas a nghttp como son: `--max-concurrent-streams` (valor por defecto: 1), `--window-bits`, `--connection-window-bits` (valor por defecto: 30, con la finalidad de no limitar innecesariamente el flujo), `--header-table-size` y `--encoder-header-table-size` (valor por defecto: 4K, el valor por defecto del protocolo).

La herramienta de benchmarking entrega como resultado, entre otros valores, la cantidad de requests exitosas y fallidas, así como también su velocidad en requests por segundo y también la cantidad de tráfico transmitido durante la prueba.

Se realizan pruebas tanto con `--requests` como con `--duration` por separado, usando siempre valores crecientes para estos parámetros, así como también para el número de clientes, con la opción `--clients`. Ya que `--duration` tiende a no sobrecargar el servidor, se decide hacer uso de la opción `--requests` y `--clients` para determinar el máximo valor posible de ambos parámetros para los que el servidor es capaz de procesar todas las peticiones sin errores, y el tiempo promedio que esto demora.

4.3. Configuración experimental

4.3.1. Raspberry

En la configuración utilizada 4.1, se tiene un computador con procesador Intel Core i5 que simula varios clientes enviando peticiones, mientras que la Raspberry actúa como servidor. Los dispositivos se comunican dentro de una red local haciendo uso de WiFi, a una distancia inferior a un metro. La versión de nghttp2 utilizada es la 1.34.0.

El computador cuenta con el sistema operativo Ubuntu 18.04, mientras que la Raspberry corre Ubuntu MATE en su versión para Raspberry 3. Al momento de realizar los experimentos, ambos dispositivos tiene la interfaz gráfica activada.

También, adosado a la Rasperry, se tiene un módulo de corriente Arduino ACS712, que permite medir la potencia usada para cada configuración en la que se pudiera desear medir. Para los experimentos de corriente se desactivó la conectividad Bluetooth y se quitó el monitor conectado a través de HDMI, esto para eliminar posibles distorsiones en las mediciones.

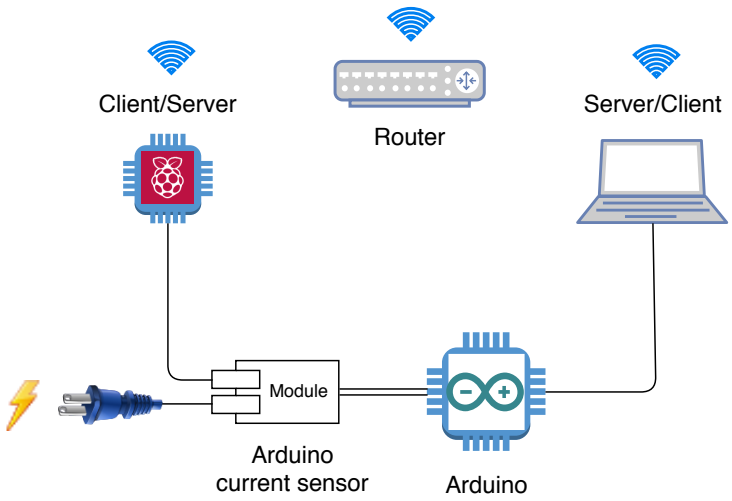


Figura 4.1: Configuración para medidas de Raspberry como servidor.

4.3.2. A8 open node

Acceder a los nodos A8 es posible a través de la plataforma IoT-LAB [otT13], un banco de pruebas que provee de dispositivos en los que pueden realizarse experimentos de forma remota. Es posible, entonces, conectarse a estos nodos mediante SSH e instalar la librería de HTTP/2.

El compilador de C que está disponible en estos nodos es GCC versión 5, no compatible con la más reciente versión de la librería. Sin embargo, es posible hacer uso de la versión 1.28.0 que solo tiene cambios en el proxy de HTTP/2 del que no se hace uso en este trabajo.

Ya que no es posible enviar peticiones directamente de un computador a uno de estos nodos, se tiene un nodo que actúa simulando varios clientes y otro que actúa como servidor. Los dispositivos se comunican a través de Ethernet. Las configuración experimental para nodos A8 se muestra en la Figura 4.2

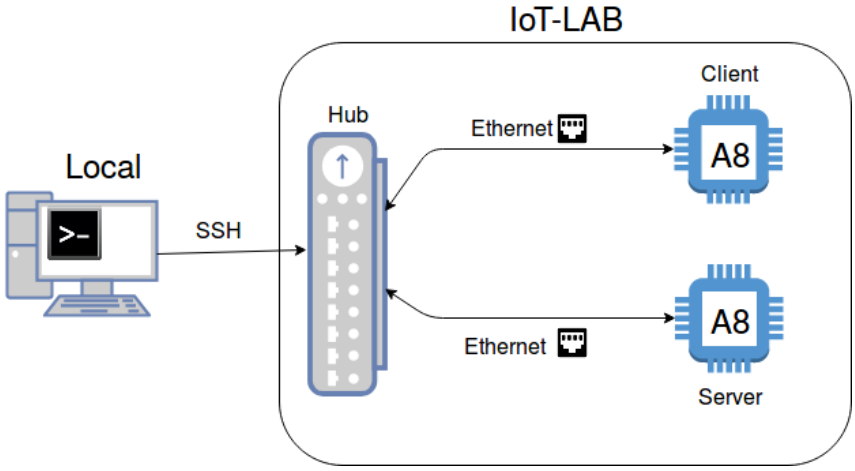


Figura 4.2: Conexión vía SSH con dos nodos A8, cliente y servidor.

Capítulo 5

Implementación

Una vez tomadas las decisiones de diseño, se procede a implementar el código para poder realizar las mediciones.

5.1. Benchmarking del servidor

Para determinar el punto donde el servidor deja de responder a las peticiones adecuadamente, se crea un script que ejecuta h2load para distintas cantidades de clientes y de peticiones. Los resultados se almacenan en archivos con nombre `h2load_clientes_peticiones.log`.

Se realiza una serie de varias repeticiones de estos experimentos ($n \approx 40$) y se procede a procesar los datos mediante otro script. Finalmente, los datos son puestos en un *heatmap*.

A partir de los resultados de esta prueba, se elige una cantidad de clientes y una cantidad de peticiones para el resto de los experimentos. Como resultado de `h2load` se obtiene también el tiempo promedio que estas peticiones tardan en realizarse.

5.2. Mediciones de CPU, memoria y potencia

Una vez elegida la cantidad de clientes y solicitudes a utilizar y determinado el tiempo promedio, se procede a realizar las mediciones de CPU, memoria y potencia.

Para las mediciones se hace uso del comando top. Este muestra los procesos que se ejecutan en la máquina con varios valores, entre ellos los buscados. También muestra de qué tipo es un proceso; por ejemplo, Z significa que se trata de un proceso *zombie*; R corresponde a un proceso que se encuentra corriendo (*running*), etc. Para los experimentos solo se consideran los valores con R, aunque en Raspberry Pi, los identificados con S (*sleep*) y D (sleep ininterrumpible) son una cantidad tan pequeña del total ($< 1\,\%$) que probablemente no conllevan

ninguna diferencia significativa. No se encuentran procesos zombies para este caso. Sin embargo, en los nodos A8 los procesos en estado S son mucho más frecuentes llegando incluso a corresponder hasta al 40 % de los datos. Esto probablemente debido a que la simulación de los clientes se realiza en otro nodo A8 y las restricciones de recursos llevan a que uno o ambos nodos dejen de responder durante ciertos intervalos de tiempo, llevando el uso de CPU a cero. Se usaron hasta 6 nodos A8 para simular 16 clientes cada uno sin observar un cambio en este comportamiento.

Se hace uso del comando con un *delay* de 0,01 segundos. Se eligió este valor por ser el menor valor para el que aún era posible observar cambios en las mediciones (mayor granularidad no otorgaba mayor precisión).

El script del lado del cliente ejecutan los siguientes pasos:

- Se crean tantos procesos pesados como clientes se deseen simular.
- Cada uno de esos procesos envía peticiones al servidor por medio de nghttp en un *loop*.
- Cada cliente espera que el proceso nghttp termine antes de iniciar el siguiente de inmediato.

Pese a que h2load también permite simular clientes, el uso de nghttp permite, a diferencia del anterior, simular conexiones y desconexiones completas lo que representa mejor lo que suele suceder en IoT.

Para el lado del servidor, el script creado realiza los siguientes pasos:

- Levanta el servidor con una determinada configuración.
- Ejecuta el comando top y almacena las columnas deseadas en un archivo cuyo nombre es el nombre del parámetro modificado en la configuración y el valor correspondiente. Por ejemplo, `SETTINGS_MAX_CONCURRENT_STREAMS_100.log`.
- Una vez transcurrido el tiempo para que se ejecute la cantidad de requests deseada, el servidor se baja.
- Se modifica el archivo necesario con el nuevo valor del parámetro.
- Se recompila el archivo.
- Se vuelve a levantar el servidor.

Este proceso se ejecuta varias veces. Luego los datos de estos archivos son procesados mediante otro script y graficados.

Para realizar las mediciones del lado del cliente, el script realiza solicitudes secuencialmente por medio de nghttp. Se elige en este caso nghttp, ya que corresponde a un único cliente y una única solicitud con HTTP/2, y a diferencia de h2load, abre y cierra completamente la conexión cada vez que se ejecuta, lo que se asemeja más a lo que ocurre en un escenario IoT con dispositivos que se encienden y apagan constantemente.

5.3. Consideraciones para los experimentos

5.3.1. Consideraciones para los parámetros

El RFC de HTTP/2 [BPT15] establece que el contenido de SETTINGS no es negociado. Sin embargo, existen algunas diferencias entre el RFC y la implementación de la librería utilizada, por lo que las mediciones de cada parámetro no pueden realizarse todas de la misma manera. Algunos parámetros requieren de ciertas consideraciones para garantizar que la configuración deseada es efectivamente la que se tiene.

- `SETTINGS_HEADER_TABLE_SIZE`: Para garantizar que el tamaño de la tabla es el deseado, se deben usar las opciones `--header-table-size=<SIZE>` y `--encoder-header-table-size=<SIZE>`, ambas con el mismo valor y en ambos lados de la transacción. De otro modo, el tamaño negociado de la tabla dinámica es el mínimo de los valores entregados.
- `SETTINGS_MAX_HEADER_LIST_SIZE`: En el caso de este parámetro, al tratarse de un valor máximo, no se puede garantizar que el protocolo mantenga el valor indicado en la configuración. Aún así, dado que este parámetro no viene implementado por defecto, se recomienda indicar el mismo valor en ambos lados de la transacción para evitar posibles conflictos.
- `SETTINGS_ENABLE_PUSH`: Este parámetro, pese a habilitar o deshabilitar el Server Push, con el valor 1 o 0 respectivamente, solo puede ser modificado desde el lado del cliente a través de la opción `--no-push`. Si se cambia el valor de este parámetro en el lado del servidor, se obtiene un error.
- `SETTINGS_MAX_CONCURRENT_STREAMS`: Se recomienda siempre modificar este valor desde ambos lados de la transacción, utilizando las opciones `--max-concurrent-streams` y `--peer-max-concurrent-streams`.
- `SETTINGS_INITIAL_WINDOW_SIZE`: Corresponde al tamaño de ventana a nivel de stream. Sin embargo, para funcionar adecuadamente, exige que el parámetro `--connection-window-bits` (en nghttp y nghttpd), que representa el tamaño de ventana a nivel de conexión TCP, lleve su mismo valor.

5.3.2. Otras consideraciones

Seguridad

Como parte importante de la motivación de este trabajo viene del hecho de que HTTP es un protocolo seguro, se decide hacer uso de TLS en todas las mediciones.

Caché

Para evitar que estos valores se vean afectador por la posible existencia de un caché, el contenido de `index.html` se debe modificar para cada configuración probada haciendo uso de `pwgen`, un programa que permite generar contraseñas aleatorias. Con el comando:

```
pwgen 21 23 > path/to/index.html
```

Donde el primer número corresponde al largo de la contraseña, y el segundo a la cantidad de contraseñas que se desea generar. Se usa un largo de contraseña y una cantidad de contraseñas aleatorios entre 21 y 23. Con esto se modifica el archivo y pasa a tener un tamaño de entre 441 y 529 bytes. Es importante mencionar que no existe como exigencia para el archivo que el formato contenido efectivamente corresponda a HTML.

Capítulo 6

Resultados

De acuerdo a lo mencionado anteriormente, se realizan primero las pruebas de saturación del servidor. Todas estas pruebas fueron hechas con los parámetros de HTTP/2 por defecto [BPT15], tanto para el lado del servidor como para el del cliente.

La configuración por defecto corresponde a:

Parámetro	Valor inicial o por defecto
SETTINGS_ENABLE_PUSH	1 (enabled)
SETTINGS_MAX_CONCURRENT_STREAMS	infinito
SETTINGS_INITIAL_WINDOW_SIZE	65535 bytes
SETTINGS_MAX_FRAME_SIZE	16384 bytes
SETTINGS_HEADER_TABLE_SIZE	4096 bytes
SETTINGS_MAX_HEADER_LIST_SIZE	infinito

Tabla 6.1: Valores por defecto de los parámetros relevantes de acuerdo al RFC de HTTP/2 [BPT15]

6.1. Benchmark del servidor

6.1.1. Pruebas con plataforma Rasperry Pi

Para el caso de la Raspberry se realizan pruebas sobre nghttpd y libevent-server. Los resultados son los siguientes:

nghttpd

En la figura 6.1, se tiene el número de clientes para cada prueba, así como también el total de peticiones a repartir entre todos estos clientes. El color muestra la cantidad de peticiones fallidas promedio para todos los clientes en conjunto: un color claro indica que pocas o ninguna petición falló, mientras que un color más oscuro significa que hubo más peticiones fallidas.

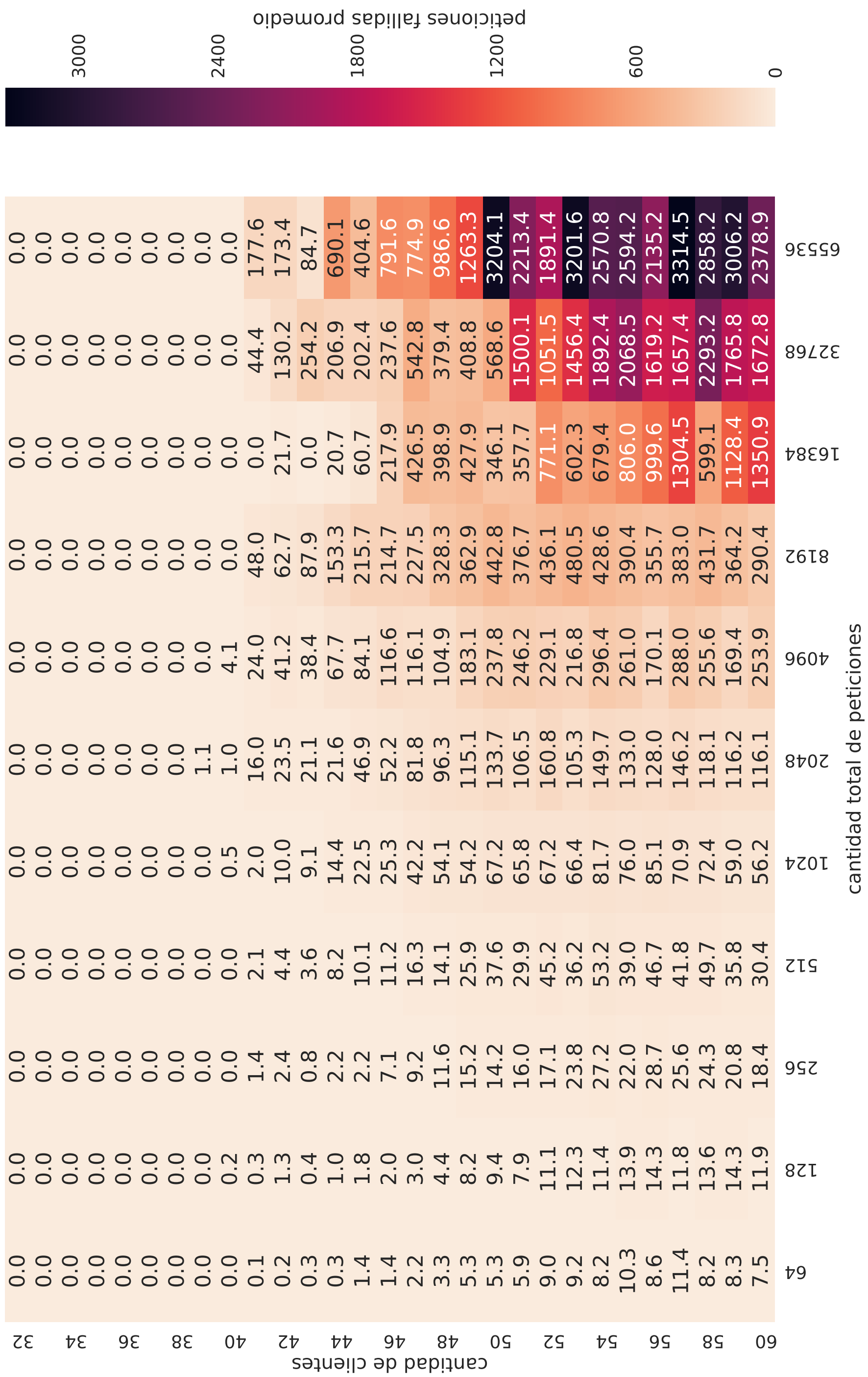


Figura 6.1: Peticiones fallidas para servidor nghttpd en Raspberry Pi.

Se observa un comportamiento del servidor bastante claro y se opta por tomar las pruebas con 38 clientes y 32.768 peticiones. Estas solicitudes se reparten equitativamente, lo que implica alrededor de 860 solicitudes por cliente. Como las peticiones son de alrededor de 500 bytes, la cantidad de tráfico es de alrededor de 16 MB. Este tráfico es teóricamente suficiente para poder realizar observaciones, dado que los efectos de disminución de tiempo de carga con HTTP/2 se pueden observar al cargar solo una página, cuyo tamaño promedio se sabe que es de 1,8 MB [TF16].

Para las mediciones del lado del cliente, se opta por realizar alrededor de 900 solicitudes, por ser un valor cercano a la cantidad de peticiones hechas por cada cliente en las pruebas del servidor.

libevent-server

Se repite el mismo experimento para libevent-server, cuyos resultados se muestran en la Figura 6.2. Sin embargo, el comportamiento encontrado no es lo suficientemente estable como para decidir usar este servidor para pruebas. Los cuadros en blanco se tratan de puntos que nghttp no permite medir, ya que exige que la cantidad de solicitudes sea al menos igual a la cantidad de clientes.

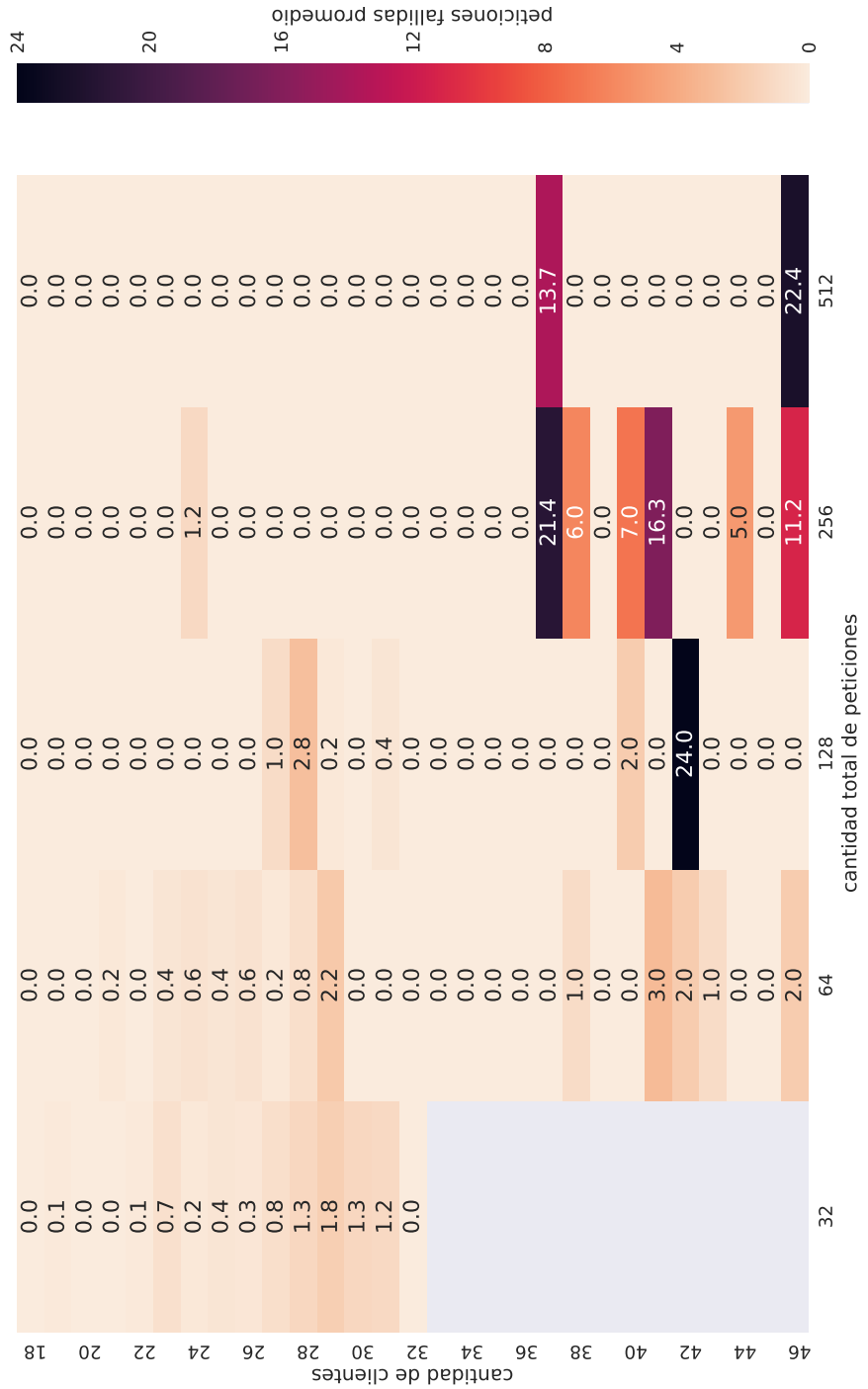


Figura 6.2: Peticiones fallidas para servidor libevent-server en Raspberry Pi.

6.1.2. A8 open node

Dado que el servidor libevent-server muestra un comportamiento poco estable en el caso de Raspberry Pi, se toma la decisión de hacer pruebas de rendimiendo solo con nghttpd en los nodos A8. Esto con la intención de poder comparar resultados con mayor facilidad.

nghttpd

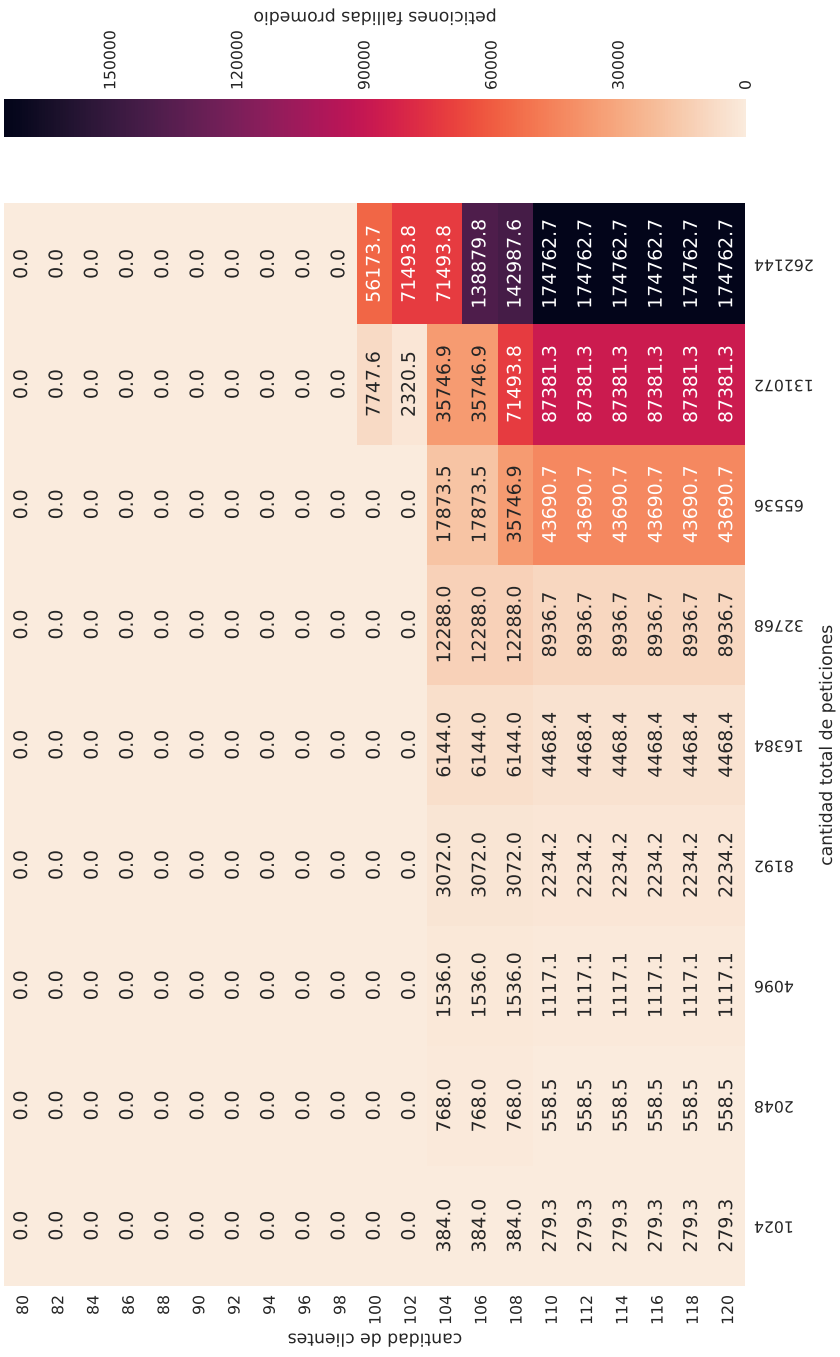


Figura 6.3: Peticiones fallidas para servidor nghttpd en nodo A8.

En la Figura 6.3, se observa que el nodo A8 presenta un comportamiento similar al de la Raspberry Pi. Sin embargo, se logra una mayor cantidad de clientes y peticiones sin fallos. Se elige entonces para este caso una cantidad de clientes igual a 96 y una cantidad de solicitudes total de 131.072.

Resulta quizás paradójico que pese a que el nodo A8 tiene recursos inferiores en términos de procesamiento y memoria que la Raspberry, este sea capaz de manejar mejor un mayor número de clientes y de peticiones. Un posible motivo podría estar relacionado con el sistema operativo que corre cada uno y cuántos recursos exige este a la máquina. Otra alternativa puede estar relacionada con la tecnología usada para la transmisión de datos. Son necesarias más pruebas para determinar con precisión las causas de este comportamiento, aunque estas pruebas se encuentran fuera del alcance de este estudio.

6.2. Experimentos con parámetros

Una consideración de gran importancia que se debe mencionar es la forma en que nghttpd funciona y la manera correcta de terminar el proceso. Al iniciar nghttpd, este genera un proceso hijo llamado lt-nghttpd que es el que efectivamente se encarga de procesar las solicitudes. Si el programa se termina con un SIGINT no hay garantía de que el proceso hijo también se detenga. Luego, al volver a iniciar otro proceso nghttpd, se genera un nuevo proceso hijo, pero el anterior sigue existiendo. Estos procesos hijos se reparten la carga de procesar las requests y pueden llevar a mediciones erróneas como las observadas en la Figura 6.4.

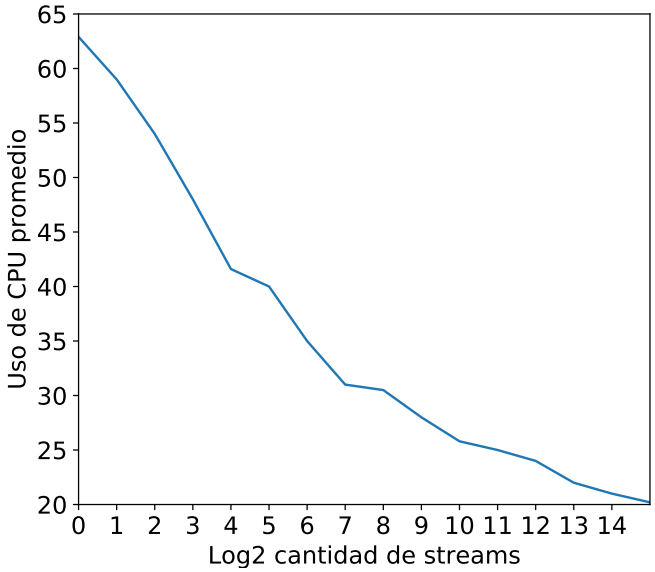


Figura 6.4: Medidas erróneas de uso de CPU para diferentes números de streams ocasionadas por procesos hijos "fantasma" de la librería nghttpd.

En la Figura 6.4 para cada punto del eje X se terminó un proceso nghttpd de forma incorrecta. Puede observarse además el efecto acumulativo de la repartición de carga entre los procesos hijos que siguen corriendo.

Para terminar el programa adecuadamente, se debe enviar una señal SIGTERM, pero ya que esta señal puede ser ignorada, es preferible enviar una señal SIGKILL con el comando `killall -9` y con el nombre de ambos procesos involucrados: nghttpd y lt-nghttpd. Vale la pena mencionar que el servidor libevent-server también genera un proceso hijo, llamado lt-libevent-ser que cumple la misma función.

6.3. Cantidad de medidas en los experimentos

Cada experimento, cuyos resultados se muestran en las secciones siguientes, fue ejecutado 30 veces para el caso de los servidores. Esto significa que para cada punto de los gráficos de las siguientes secciones, se tomaron entre 10.000 y 12.000 muestras de uso de CPU y memoria, y entre 4.000 y 5.000 valores de corriente para el caso del servidor de Raspberry Pi. Para el caso del cliente en la Raspberry Pi, se tomaron entre 25.000 y 30.000 medidas de uso de CPU y memoria, y entre 20.000 y 25.000 medidas de potencia entre los 30 experimentos en conjunto. Para el servidor del nodo A8 se consideraron entre 100.000 y 150.000 muestras de CPU y memoria, y aproximadamente 500.000 de potencia en total. Finalmente para el cliente del nodo A8, se tomaron cerca de 100.000 medidas de uso de CPU y memoria para el cliente, y entre 500.000 y 550.000 para la potencia.

La diferencia en la cantidad de medidas entre cliente de Raspberry y de nodo A8, aparte de la diferencia en la cantidad de solicitudes ejecutadas, se debe a que cada petición se demora más en los nodos A8 y, sin embargo, las medidas de CPU y memoria se toman siempre con la misma frecuencia. Respecto a la potencia en los nodos A8, la plataform FIT IoT-LAB permite realizar mediciones con una frecuencia del orden de microsegundos, mientras que en la Raspberry fue fijado un delay de 300 milisegundos.

En los anexos de este documento, se encuentran las tablas con los valores promedio de uso de CPU, memoria y potencia para todos los experimentos realizados. Los valores promedio mostrados corresponden al promedio de los 30 experimentos ejecutados para cada valor de cada parámetro. Ocurre una excepción en el caso del nodo A8 como servidor, ya que al requerir simular varios clientes desde un nodo, que tiene bastantes restricciones de recursos, puede darse el caso de que este nodo no logre enviar solicitudes durante algunos periodos cortos de tiempo. Por lo tanto, los datos del servidor en los que no se recibían peticiones, fueron descartados.

6.3.1. Raspberry Pi como servidor

Medidas para configuración por defecto [BPT15]

Se mide porcentaje de uso de CPU, porcentaje de uso de memoria y potencia para el servidor nghttpd corriendo con la configuración por defecto, indicada en la Tabla 6.1, y recibiendo 32.768 solicitudes en conjunto de 38 clientes haciendo uso del comando top con un delay de 0,01.

Los resultados obtenidos son presentados en la Tabla 6.2:

%CPU promedio	%MEM promedio	Potencia promedio [W]
97,94	0,62	1,38
Desviación %CPU	Desviación %MEM	Desviación potencia [W]
7,31	0,05	0,08

Tabla 6.2: Resultados para configuración por defecto

Medidas sin Server Push

De manera similar a la anterior se obtiene, dejando todos los parámetros por defecto excepto Server Push, los resultados de la Tabla 6.3:

%CPU promedio	%MEM promedio	Potencia promedio [W]
98,04	0,62	1,38
Desviación %CPU	Desviación %MEM	Desviación potencia [W]
7,21	0,05	0,08

Tabla 6.3: Resultados sin Server Push

No existe, por lo tanto, una diferencia significativa en este caso. Esto puede deberse a la poca cantidad de información que se transmite en cada transacción. Ya que si bien el tráfico total es de alrededor de 16 MB, este se transmite en solicitudes independientes de apenas 500 bytes, por lo que la información es tan poca que no permite que la opción de que Server Push pueda tener algún efecto.

Medidas para diferentes valores de número máximo de streams concurrentes

Si bien para las primeras mediciones de streams, graficadas en la Figura 6.5, se tiene en los resultados un uso ligeramente menor de CPU, incluso para 2^0 (valor por defecto), es necesario recordar el funcionamiento del protocolo.

HTTP/2 establece una única conexión TCP para transmitir datos, dentro de esta conexión pueden ir varios streams con información intercalada. Esto es muy útil en el contexto de una página web donde existen múltiples objetos y tiene sentido llevar la información de esta manera para evitar bloqueos y agilizar la descarga.

Sin embargo, en este escenario, en el que se transmite la información en transacciones individuales de 500 bytes, no existen objetos o *assets* que puedan ser enviados en forma independiente. El protocolo envía entonces toda la información contenida en el archivo de 500 bytes en un solo stream. Por lo tanto, independiente de la cantidad de streams que se decida usar, el protocolo no debiera usar más de uno.

Además, nghttp permite hacer solo una solicitud cada vez que el comando es ejecutado, por lo que para forzar el uso de más streams, habría que agregar más datos al archivo index.html.

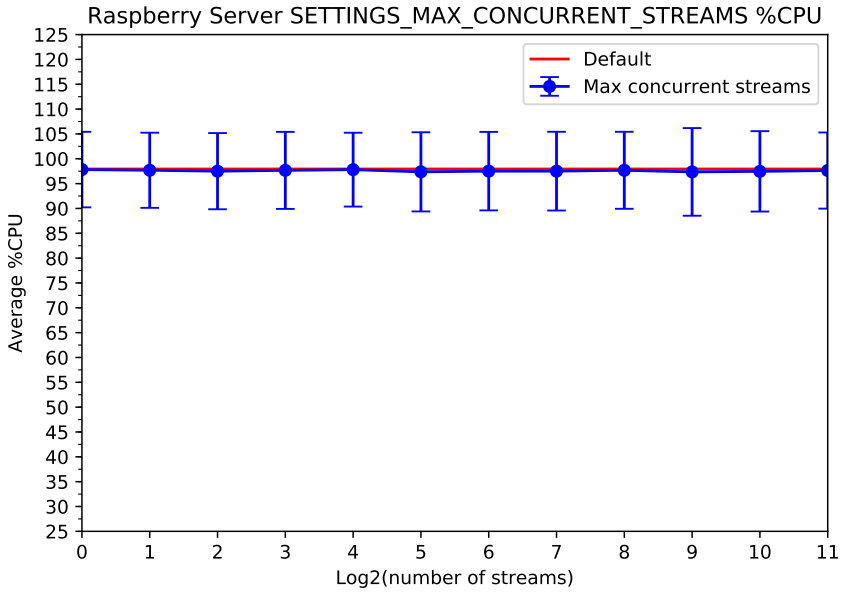


Figura 6.5: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_CONCURRENT_STREAMS.

Esto sería equivalente a probarlo en un contexto web y, por lo tanto, se alejaría del modelo de tráfico bajo estudio.

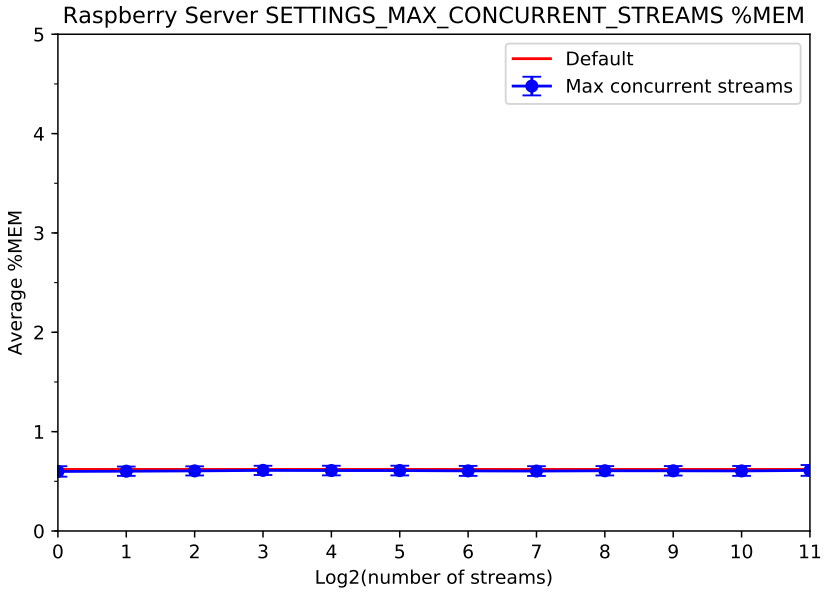


Figura 6.6: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_CONCURRENT_STREAMS.

El hecho de que no se está haciendo uso de streams, también se puede observar en los datos obtenidos para el uso de memoria ilustrados en la Figura 6.6: el valor mínimo es 0.599 %, mientras que el máximo es 0.609 %, con una desviación del orden de un 0,05 %.

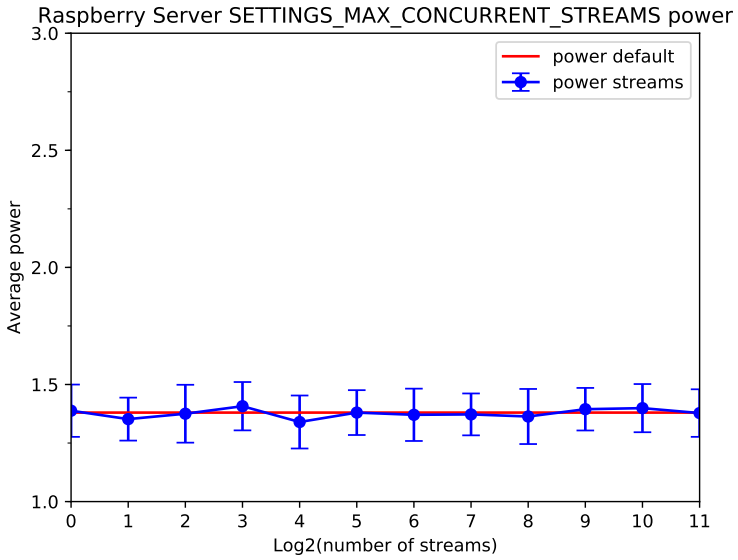


Figura 6.7: Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-CONCURRENT_STREAMS.

En la Figura 6.7 se observa la potencia promedio usada al usar distintas cantidades de streams. Esta potencia es casi siempre muy cercana a la obtenida de la configuración por defecto. Se aprecia una diferencia un poco mayor al realizar la prueba con un número mayor de streams, sin embargo, la desviación estándar de estos datos es de 0,06 watts, por lo que esta diferencia no es significativa.

Medidas para diferentes valores de tamaño inicial de ventana

Los resultados para el uso de CPU según el tamaño inicial de la ventana son los de la Figura 6.8. Para el uso de CPU se tiene una diferencia máxima respecto a la configuración por defecto menor al 1 %. Se tiene además un uso de memoria promedio de 0,6 % con una desviación estándar de 0,04 % (Figura 6.9). Dada la desviación estándar, estas medidas no pueden considerarse diferencias importantes en relación a la configuración por defecto.

Igualmente para las medidas de potencia en la Figura 6.10, no se aprecian grandes diferencias al modificar la configuración.

Similar al caso de los streams, se debe recordar el hecho de que este parámetro establece un valor inicial para el tamaño de la ventana a nivel de stream. Ya que se tiene tan solo un stream, en el mejor de los casos dos, se tiene entonces solo una o dos ventanas por lado. Además, el control de flujo se encarga de disminuir el tamaño de la ventana para ajustarse a lo necesario, que puede o no lograrse dependiendo del timeout fijado.

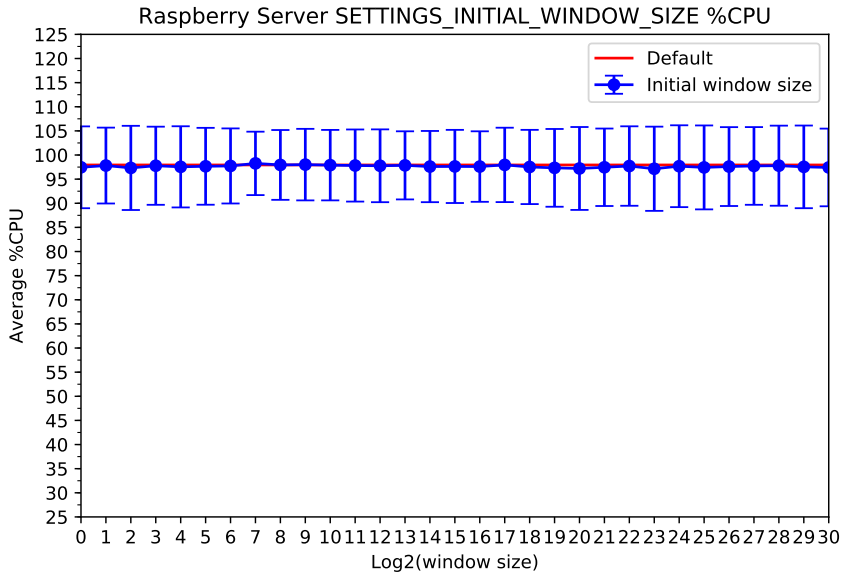


Figura 6.8: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-INITIAL_WINDOW_SIZE.

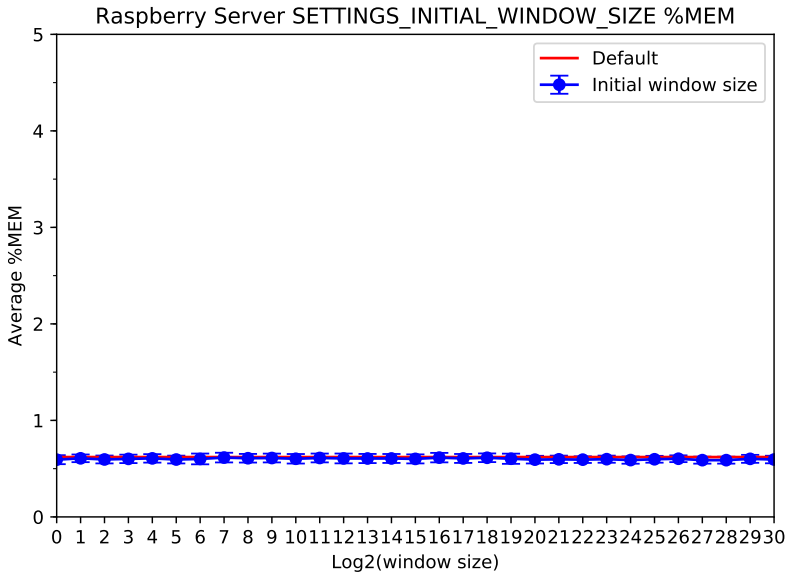


Figura 6.9: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_INITIAL_WINDOW_SIZE.

Por todo lo anterior, como se puede apreciar, el efecto del tamaño de la ventana no conlleva cambios importantes en el rendimiento.

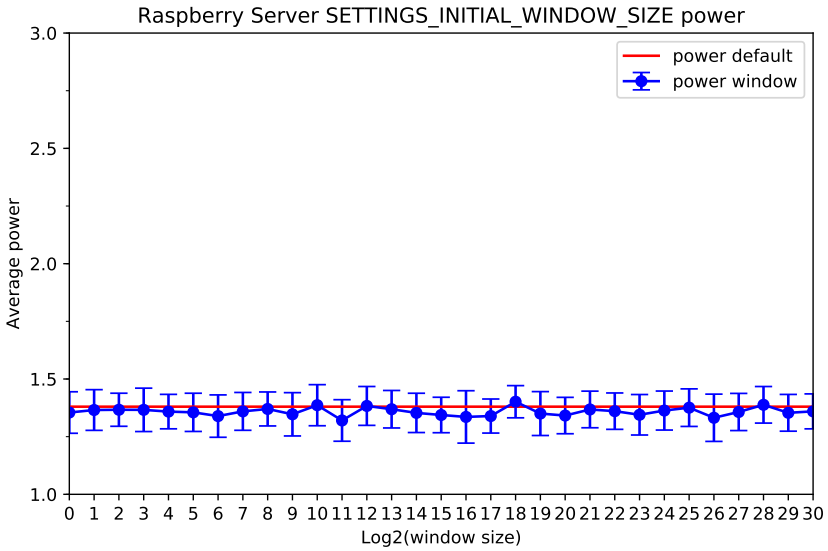


Figura 6.10: Potencia promedio para diferentes valores del parámetro SETTINGS_-INITIAL_WINDOW_SIZE.

Medidas para distintos valores de tamaño máximo de frame

Para el tamaño máximo de frame se observa que la modificación de este parámetro no genera una diferencia importante para ninguna de las métricas de interés.

En la Figura 6.11 se ve que en general la modificación de este parámetro genera un aumento muy ligero del uso de CPU. En los casos, que se logra una disminución, esta es inferior al 0,5 %.

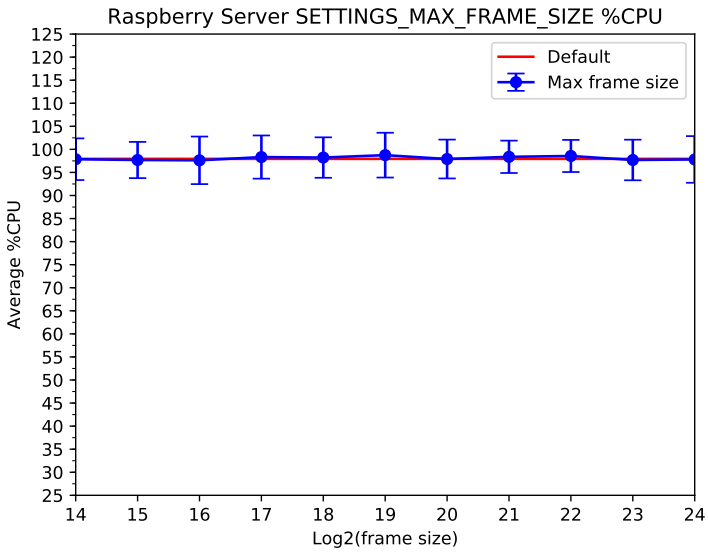


Figura 6.11: Porcentaje de uso de CPU para distintos valores del parámetro SETTINGS_-MAX_FRAME_SIZE.

La Figura 6.12 muestra siempre un uso menor de memoria para todos los valores de frame. Sin embargo, debiera indicar el mismo uso de memoria en el valor por defecto ($x = 14$), por lo tanto por esto y por la desviación estándar de ambas curvas, es posible estimar que estas pequeñas diferencias en el uso de memoria tampoco son de importancia.

En la figura 6.13 se observa que la potencia usada tiende a ser mayor al usar un tamaño de frame más grande, pero se trata de una diferencia del orden de solo 0,05 watts como máximo.

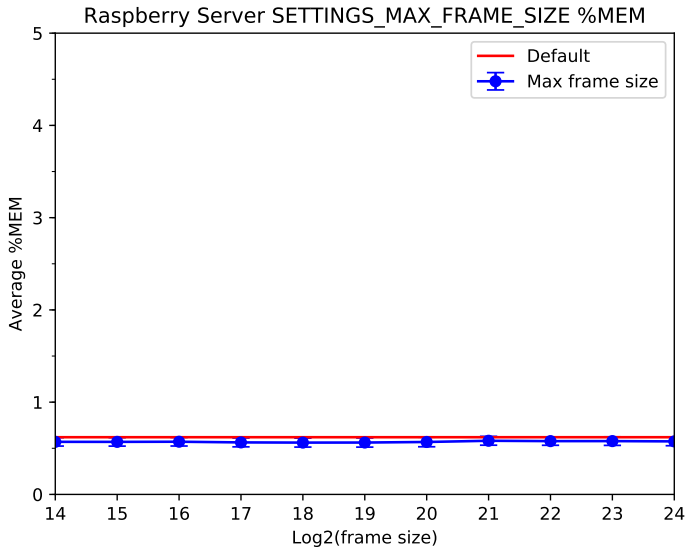


Figura 6.12: Porcentaje de uso de memoria RAM para distintos valores del parámetro SETTINGS_MAX_FRAME_SIZE.

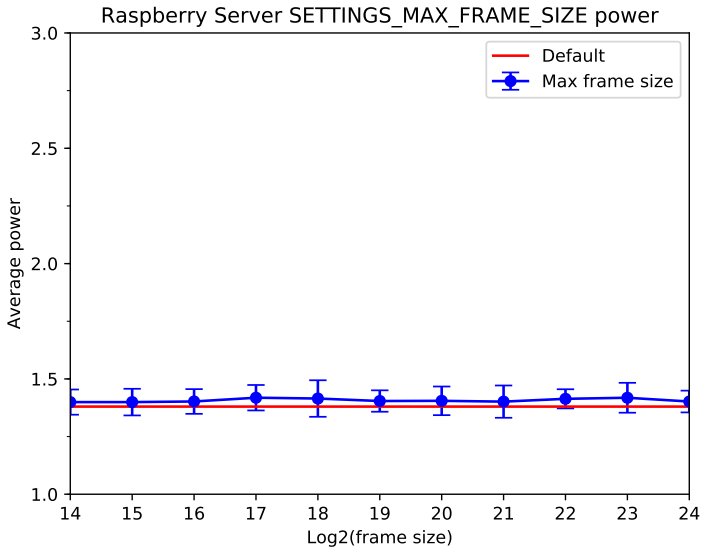


Figura 6.13: Potencia promedio para distintos valores del parámetro SETTINGS_MAX_FRAME_SIZE.

Medidas para distintos valores de tamaño de la tabla de headers

Para este parámetro no se aprecian, en los resultados mostrados en la Figura 6.14 ni el la Figura 6.15, diferencias significativas ni en uso de CPU ni en uso de memoria, que se mantiene en 0,6 % con una desviación promedio de 0,047 %.

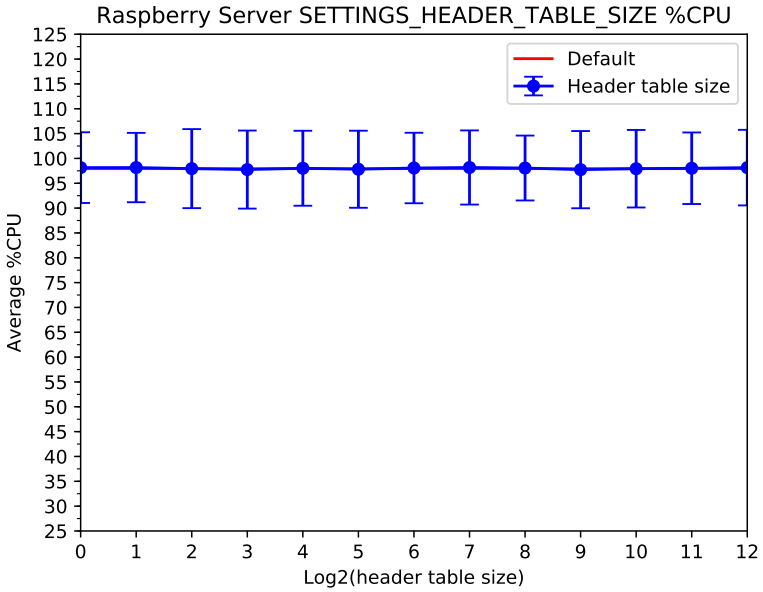


Figura 6.14: Porcentaje de uso de CPU para distintos valores del parámetro SETTINGS_-HEADER_TABLE_SIZE.

Esto nuevamente se puede explicar por la poca información que se transfiere en cada transacción. Las tablas de headers pueden ser muy útiles en el caso de un escenario web, pero en el escenario de IoT, los headers necesarios son muy pocos y, por lo tanto, una tabla pequeña o grande no hace diferencia.

Además este parámetro fija el tamaño máximo de la tabla, que se va construyendo a medida que llegan nuevos headers. Probablemente el tamaño máximo de esta tabla no se alcanza en la mayoría de los experimentos.

En la Figura 6.16 es posible ver que en general la potencia es inferior para todos los valores medidos, incluso para el valor por defecto que se da en $x = 12$, por lo que este resultado tiene probablemente más relación con cierta variación en las mediciones que con un resultado real. Aún así, la mayor diferencia no supera los 0,06 watts.

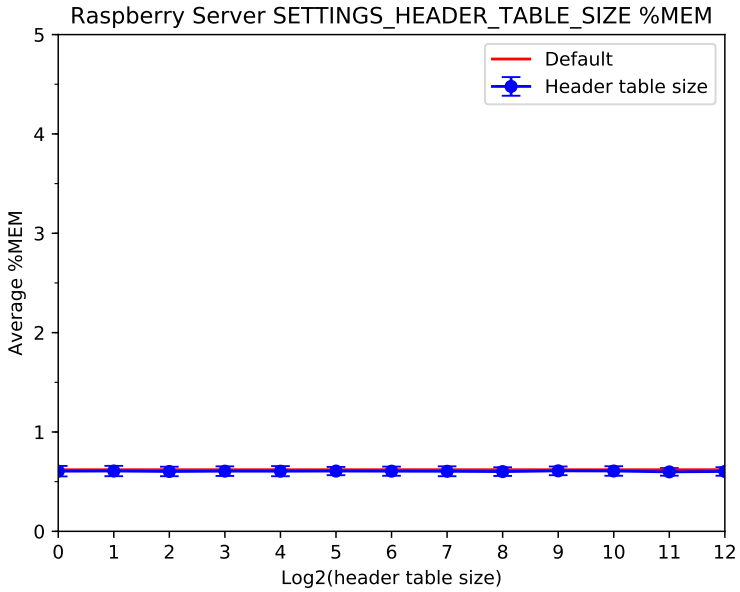


Figura 6.15: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_HEADER_TABLE_SIZE.

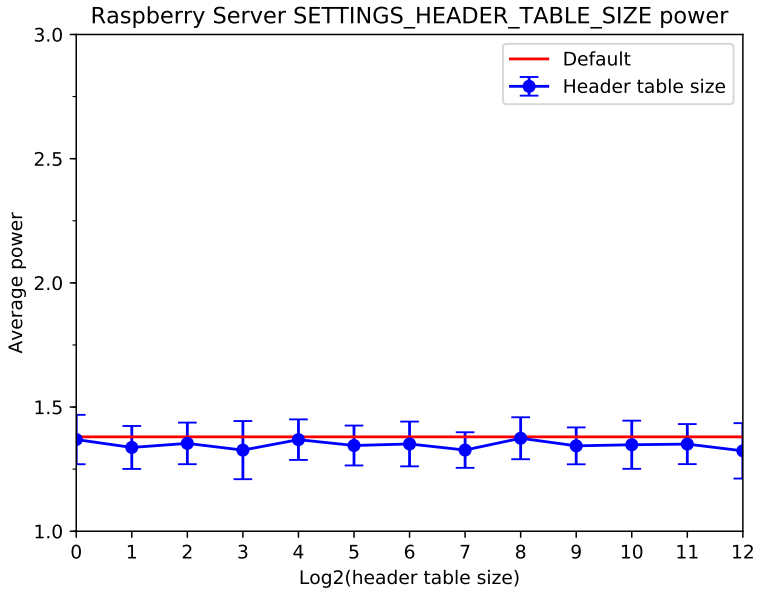


Figura 6.16: Potencia promedio para SETTINGS_HEADER_TABLE_SIZE.

Medidas para distintos valores del tamaño máximo de la lista de headers

Los resultados para este parámetro se muestran en la Figura 6.17. Nuevamente, como en los casos anteriores, la diferencia en el uso de CPU no puede considerarse significativa, por ser inferior a un 1 %. No se observa una configuración que mejore el uso de CPU.

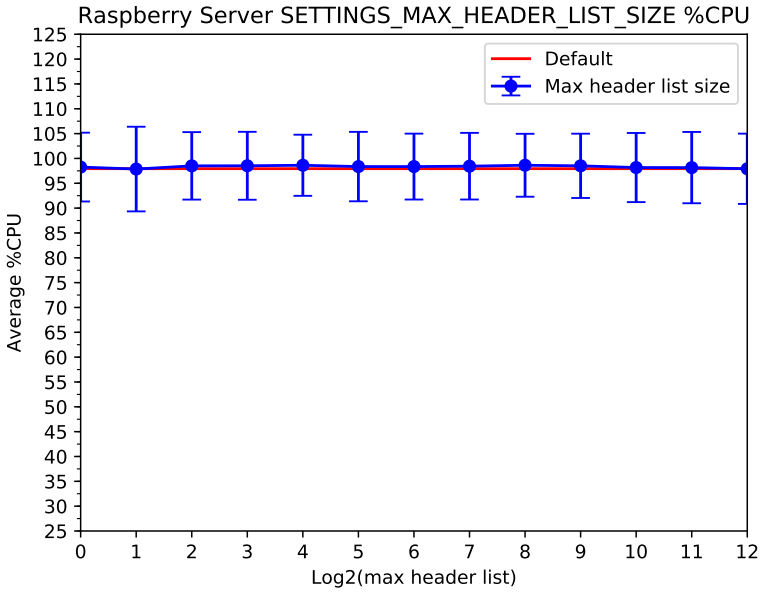


Figura 6.17: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_MAX_HEADER_LIST_SIZE.

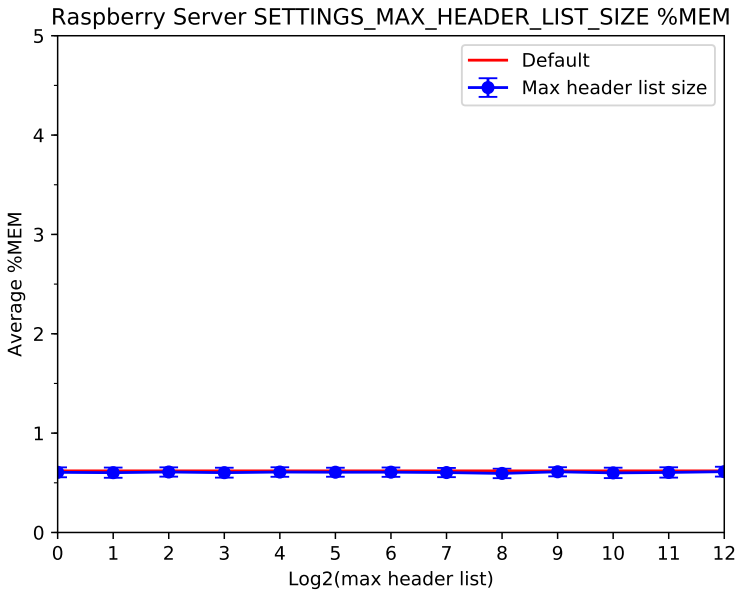


Figura 6.18: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_HEADER_LIST_SIZE.

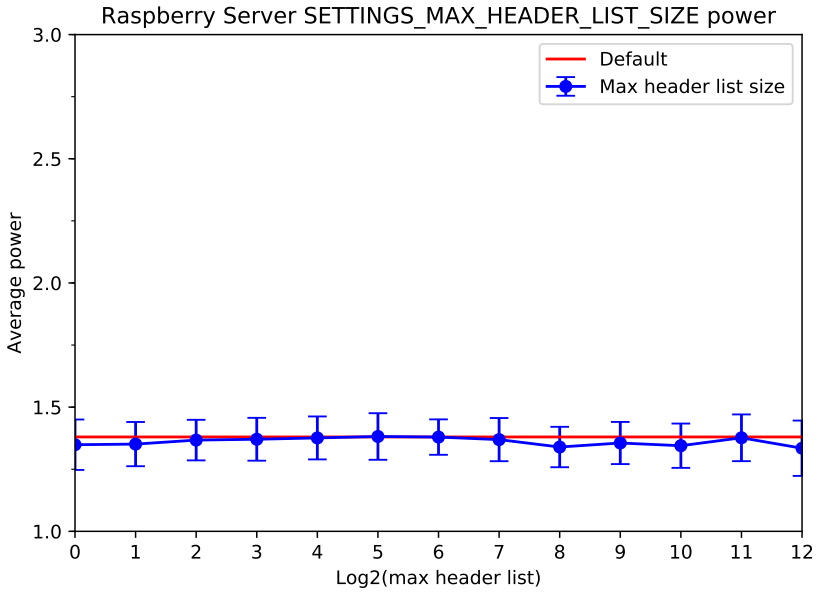


Figura 6.19: Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-HEADER_LIST_SIZE.

La explicación de esta situación es análoga a la de `MAX_HEADER_TABLE_SIZE`. En una muestra del tráfico IoT tomada con h2load con 80 clientes y 4.096 requests, de un tamaño aproximado de 2 MB, solo 55 kB corresponden a encabezados. Esto es apenas un 2,7 % de un total de 500 bytes.

La Figura 6.18 permite ver que el uso de memoria se mantiene casi estable en 0,6 %.

Respecto a la potencia, en la Figura 6.19 se ve que es casi siempre ligeramente inferior sin desviarse más de 0,03 watts de la que corresponde a la configuración por defecto.

6.3.2. Raspberry Pi como cliente

Medidas configuración por defecto

%CPU promedio	%MEM promedio	Potencia promedio [W]
54,77	0,37	1,31
Desviación %CPU	Desviación %MEM	Desviación potencia [W]
2,59	0,02	0,02

Tabla 6.4: Resultados para configuración por defecto

Medidas sin Server Push

De manera similar a la anterior se obtiene, dejando todos los parámetros por defecto excepto Server Push, los resultados de la Tabla 6.3:

%CPU promedio	%MEM promedio	Potencia promedio [W]
54,93	0,37	1,31
Desviación %CPU	Desviación %MEM	Desviación potencia [W]
4,33	0,02	0,02

Tabla 6.5: Resultados sin Server Push

Medidas para diferentes valores de número máximo de streams concurrentes

En la Figura 6.20 se observa que el uso de CPU resulta variable hasta casi un 3 % de esta. Aunque dado que la cantidad de streams por defecto es 1, debiera apreciarse en la figura que el valor de 2^0 tiene el mismo uso de CPU tanto para el stream como para el default. Esta diferencia puede deberse a varias razones, entre ellas que el estado del servidor o de la máquina en la que se corría el cliente no haya sido el mismo mientras se realizaban estas solicitudes o al hecho de que el uso de CPU al realizar solicitudes se comporta de forma oscilatoria con un inicio que requiere poca CPU para luego llegar a un máximo y volver a descender de forma más pronunciada que lo que ocurre con el servidor. Otra posibilidad es que al momento de realizar los experimentos, haya existido alguna diferencia en la red que haya forzado al cliente a necesitar hacer un uso de CPU mayor.

En la Figura 6.21, se puede ver que el uso de memoria es muy estable. Igualmente se aprecia la misma estabilidad en la potencia en la Figura 6.22.

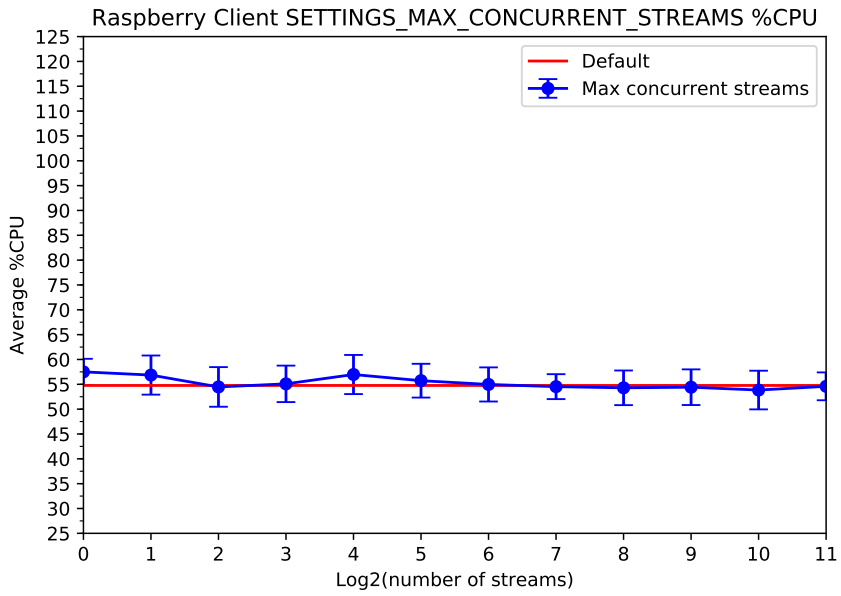


Figura 6.20: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_CONCURRENT_STREAMS.

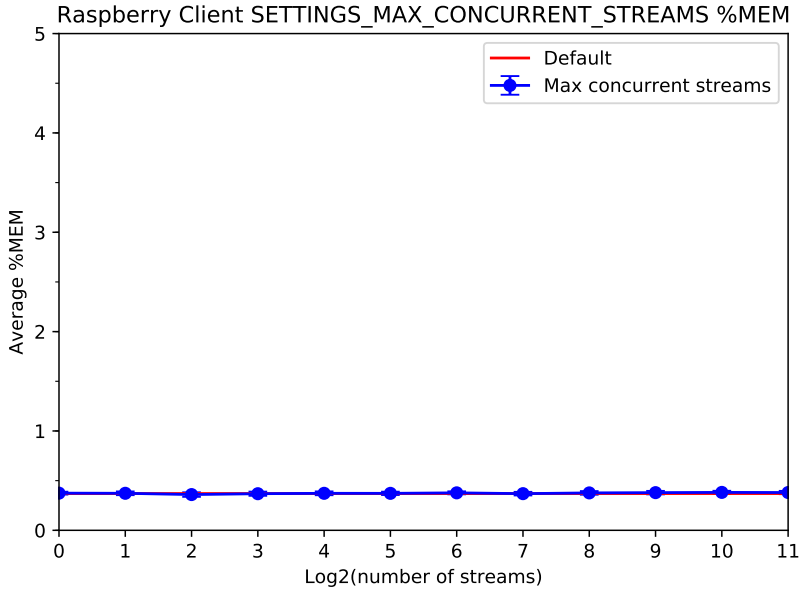


Figura 6.21: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_CONCURRENT_STREAMS.

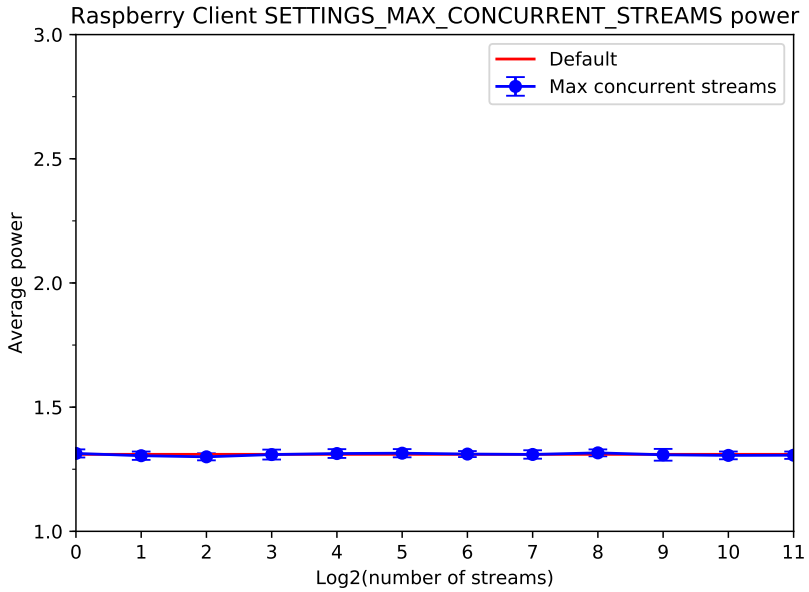


Figura 6.22: Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-CONCURRENT_STREAMS.

Medidas para diferentes valores de tamaño inicial de ventana

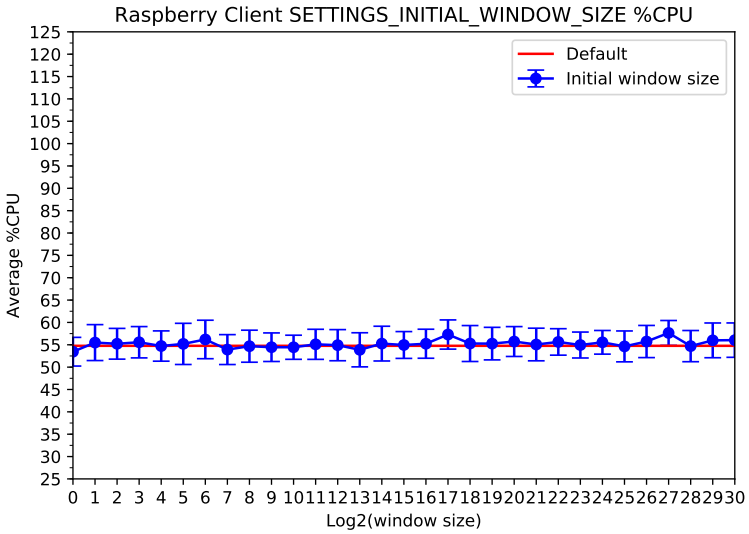


Figura 6.23: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-INITIAL_WINDOW_SIZE.

Nuevamente para el uso de CPU se puede ver en la Figura 6.23 un comportamiento oscilante que no se aleja mucho del valor por defecto. No hay ningún valor del parámetro que mejore de forma importante el uso de CPU del sistema.

Las Figuras 6.24 y 6.25 muestran un comportamiento tanto de uso de memoria como de

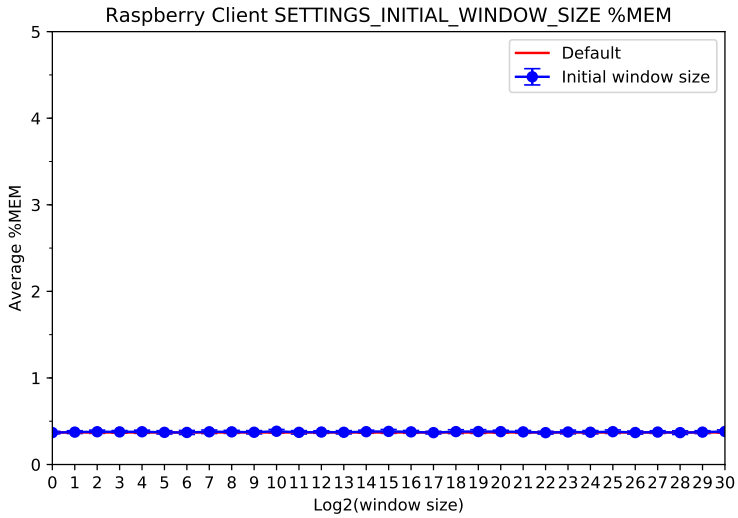


Figura 6.24: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_INITIAL_WINDOW_SIZE.

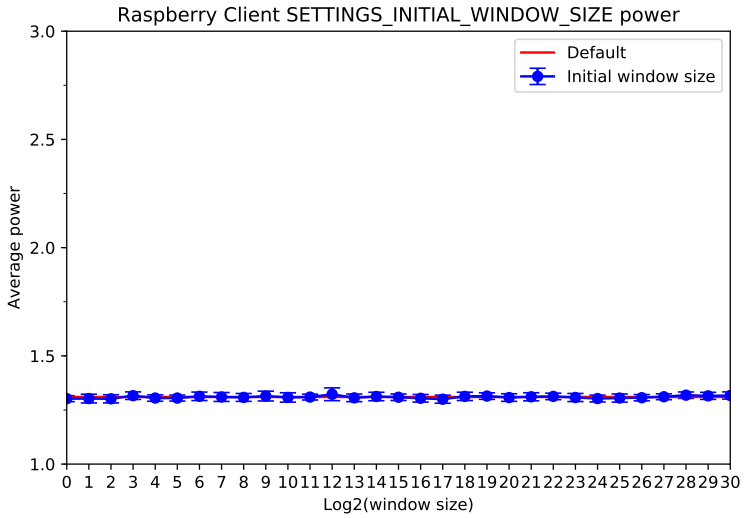


Figura 6.25: Potencia promedio para diferentes valores del parámetro SETTINGS_INITIAL_WINDOW_SIZE.

potencia muy estable y cercano a los valores de la configuración por defecto.

Medidas para distintos valores de tamaño máximo de frame

En la Figura 6.26 se muestra el uso de CPU para el cliente de acuerdo al tamaño de frame. No se aprecian alzas demasiado grandes ni algún valor que implique un descenso importante en el uso de CPU del dispositivo.

De manera similar a los casos anteriores, en las Figuras 6.27 y 6.28 apenas se aprecian

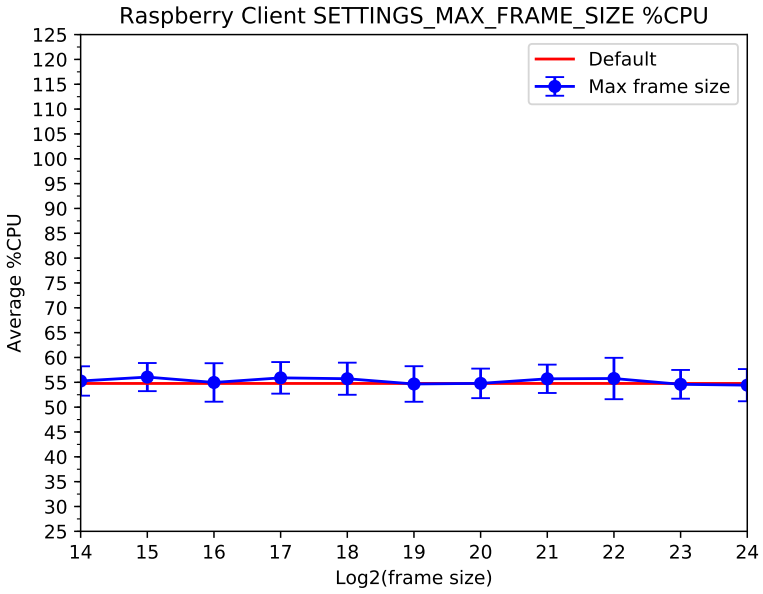


Figura 6.26: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_FRAME_SIZE.

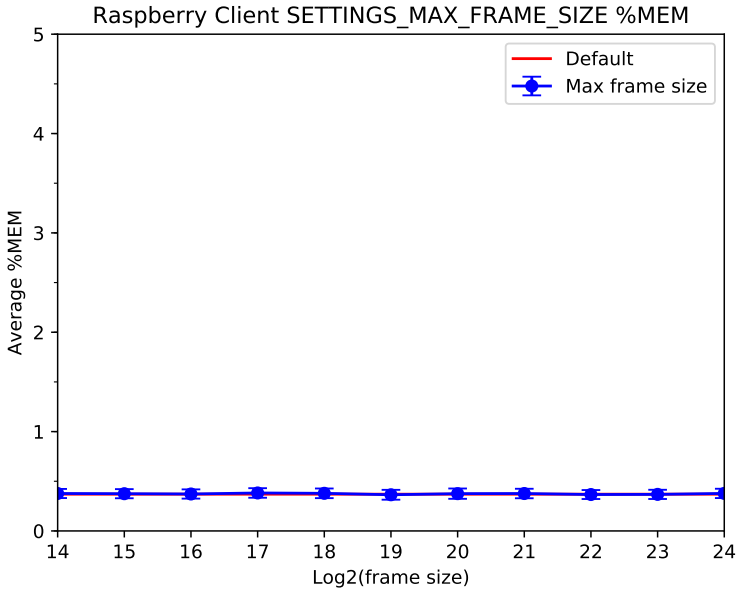


Figura 6.27: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SET-TINGS_MAX_FRAME_SIZE.

diferencias respecto a la configuración por defecto.

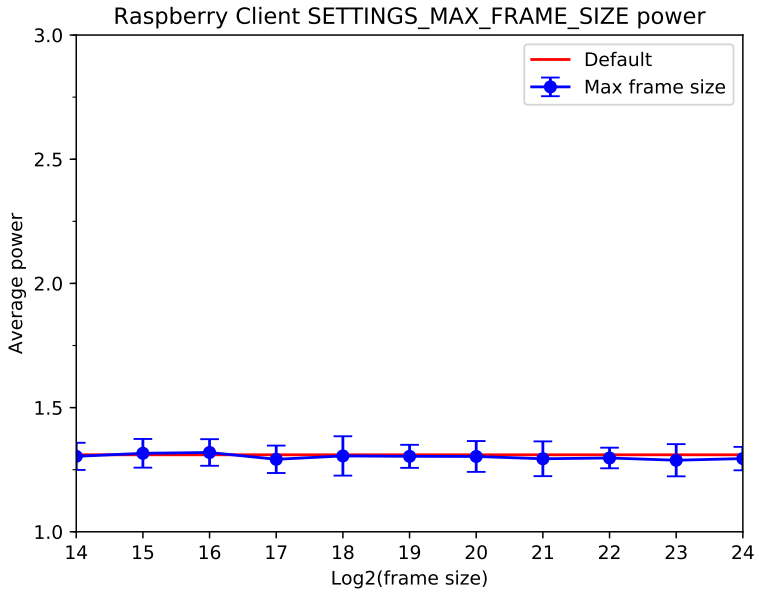


Figura 6.28: Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-FRAME_SIZE.

Medidas para distintos valores de tamaño de tabla de headers

En la Figura 6.29 es posible ver que para el tamaño 2^5 de la tabla de headers, ocurre una bajada en el uso de CPU. Aún así, esta disminución es de solo un 2 % y así como en el caso de la Figura 6.21 donde el uso de CPU no coincidía con el uso por default incluso para el mismo número de streams, es posible que esta disminución que se observa se deba a otras razones. Dado el pequeño tamaño de las transacciones que se tienen en estos experimentos es improbable que un tamaño de tabla mayor genere beneficios, ya que para esto sería necesario que hubiera un cantidad mayor de encabezados o headers, lo que no es el caso ya que estos corresponde apenas a aproximadamente un 2,7 % del total de cerca de 500 bytes de la transacción.

Una vez más para los resultados de uso de memoria RAM y potencia, en las Figuras 6.30 y 6.31, no se encuentran diferencias notorias.

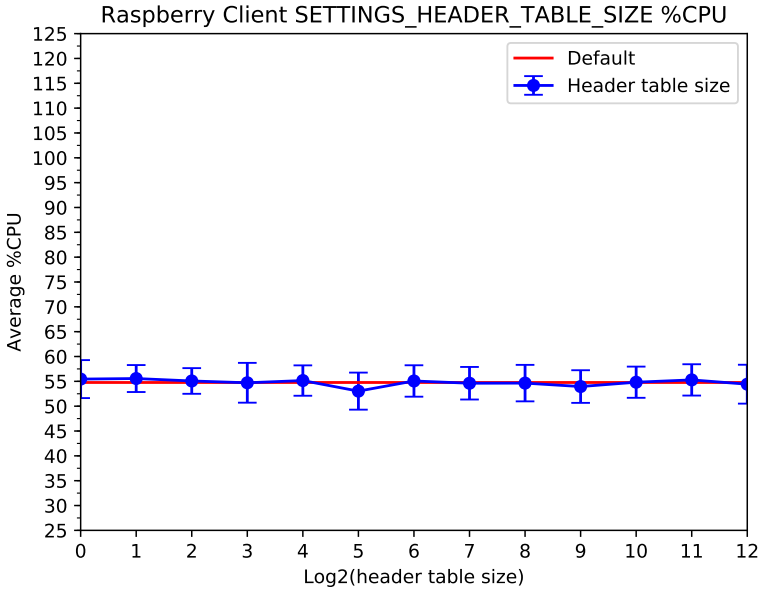


Figura 6.29: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-HEADER_TABLE_SIZE.

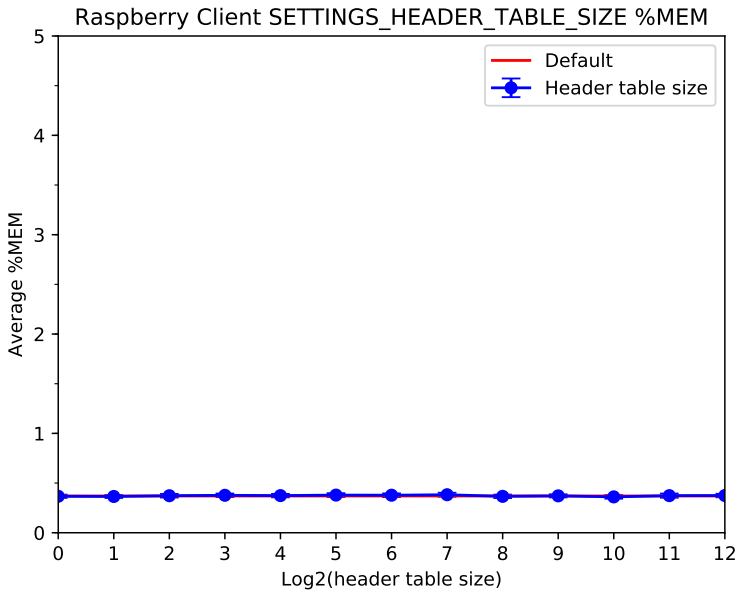


Figura 6.30: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_HEADER_TABLE_SIZE.

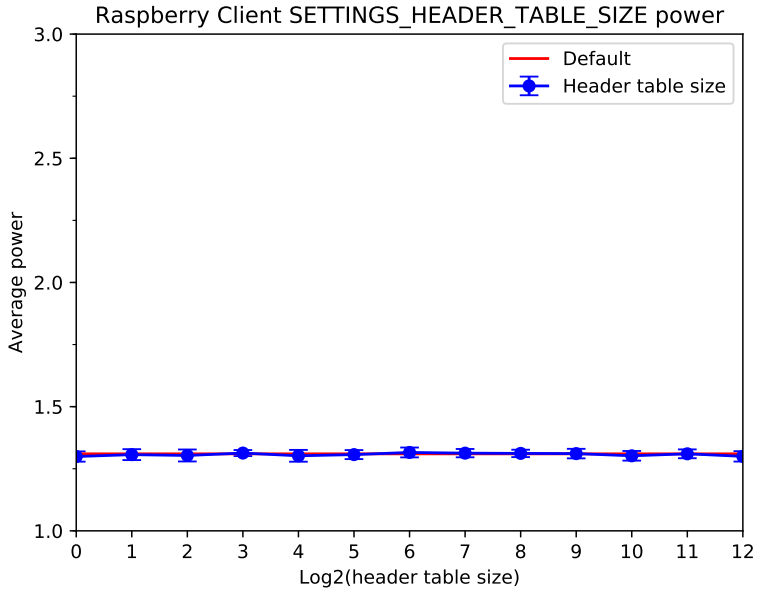


Figura 6.31: Potencia promedio para diferentes valores del parámetro SETTINGS_HEADER_TABLE_SIZE.

Medidas para distintos valores del tamaño máximo de la lista de headers

De manera similar a los parámetros anteriores, se observa un comportamiento oscilante del uso de CPU en la Figura 6.32 y ligeras variaciones para uso de memoria y potencia en las Figuras 6.33 y 6.34.

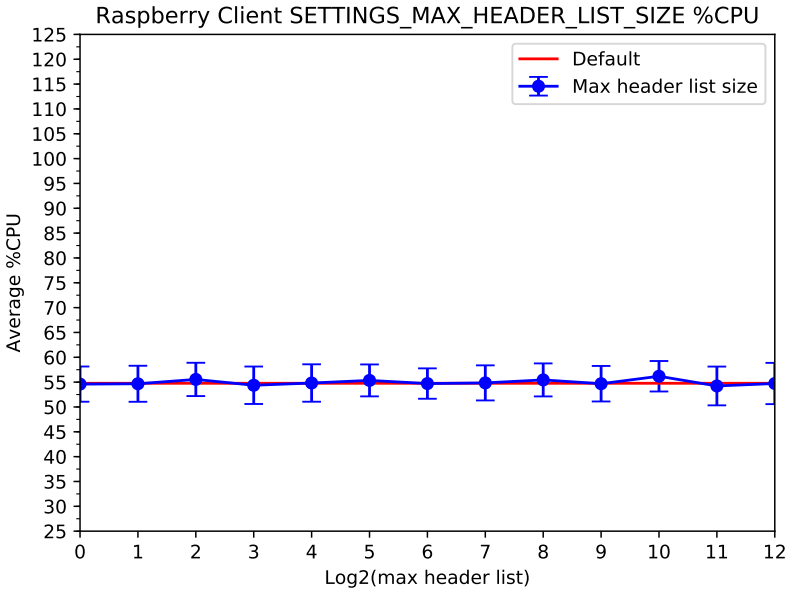


Figura 6.32: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_HEADER_LIST_SIZE.

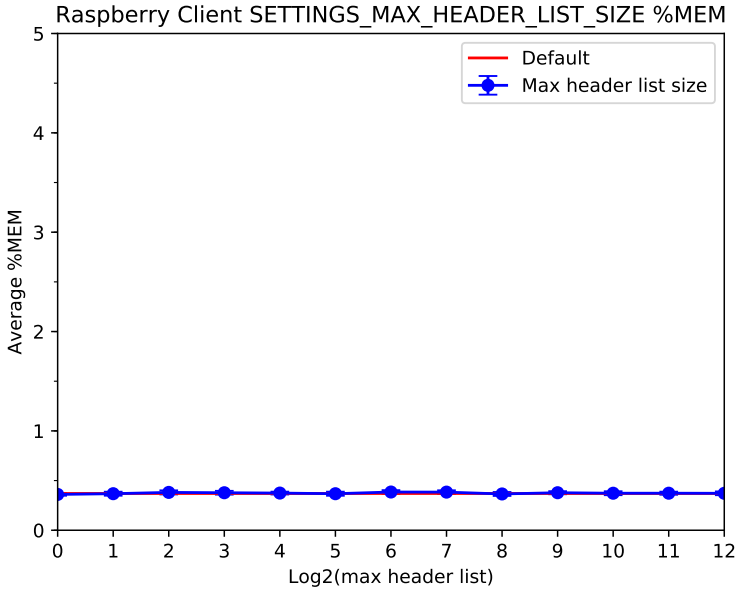


Figura 6.33: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_HEADER_LIST_SIZE.

Este comportamiento puede nuevamente explicarse por la escasez de headers en las transacciones realizadas. Como este parámetro se refiere a un tamaño máximo de lista, puede no tener ningún efecto si la cantidad de encabezados no es nunca suficiente para alcanzar o superar ese tamaño de lista.

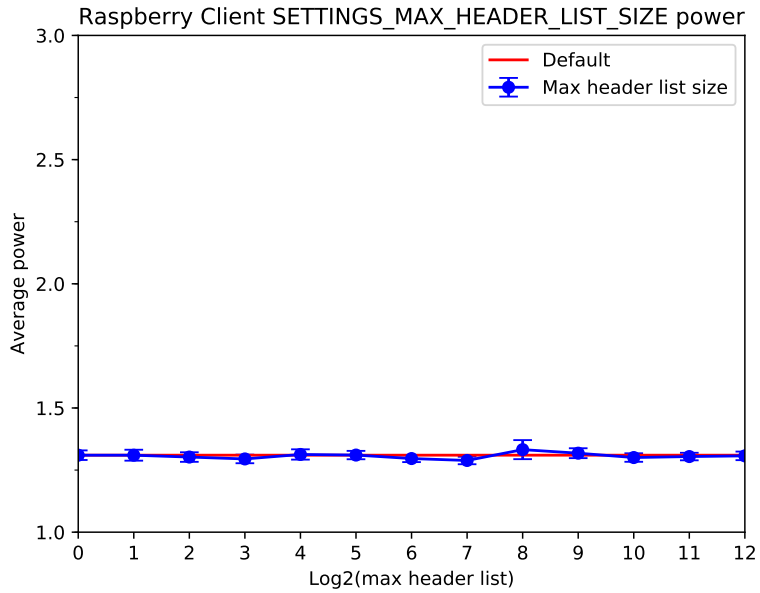


Figura 6.34: Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-HEADER_LIST_SIZE.

6.3.3. A8 open node como servidor

Medidas configuración por defecto

%CPU promedio	%MEM promedio	Potencia promedio [W]
78,34	3,70	2,45
Desviación %CPU	Desviación %MEM	Desviación potencia [W]
7,68	0,21	0,05

Tabla 6.6: Resultados para configuración por defecto

Medidas sin Server Push

De manera similar a la anterior se obtiene, dejando todos los parámetros por defecto excepto Server Push, los resultados de la Tabla 6.3:

%CPU promedio	%MEM promedio	Potencia promedio [W]
79,29	3,57	2,47
Desviación %CPU	Desviación %MEM	Desviación potencia [W]
6,67	0,22	0,07

Tabla 6.7: Resultados sin Server Push

Medidas para diferentes valores de número máximo de streams concurrentes

Para las medidas de distintos valores máximos de streams concurrentes, se puede ver en la Figura 6.35 un uso ligeramente menor de CPU para casi todas las medidas tomadas. La mayor diferencia ocurre con 4 streams, aunque el comportamiento es oscilante y no se aprecia un comportamiento claro. La desviación estándar del uso de CPU promedio para 4 streams es 4,25 %, aproximadamente la misma diferencia que se observa entre la medida de uso de CPU por defecto y esta.

En la Figura 6.36, el uso de memoria tiende a oscilar en torno al valor por defecto. Aunque sin embargo, la diferencia es siempre inferior a un 1 % respecto a la configuración por defecto, de un stream. El uso de memoria de un stream debiera corresponder entonces al mismo uso de la configuración por defecto y aún así, se aprecia una ligera diferencia.

La potencia en la Figura 6.37 es, sin embargo, muy estable y cercana a la configuración por defecto para todos los valores medidos.

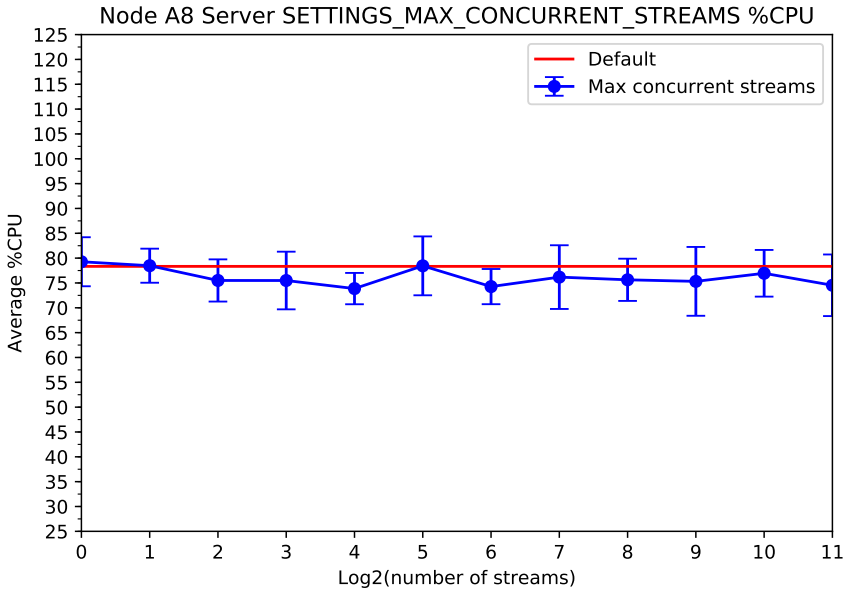


Figura 6.35: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_CONCURRENT_STREAMS.

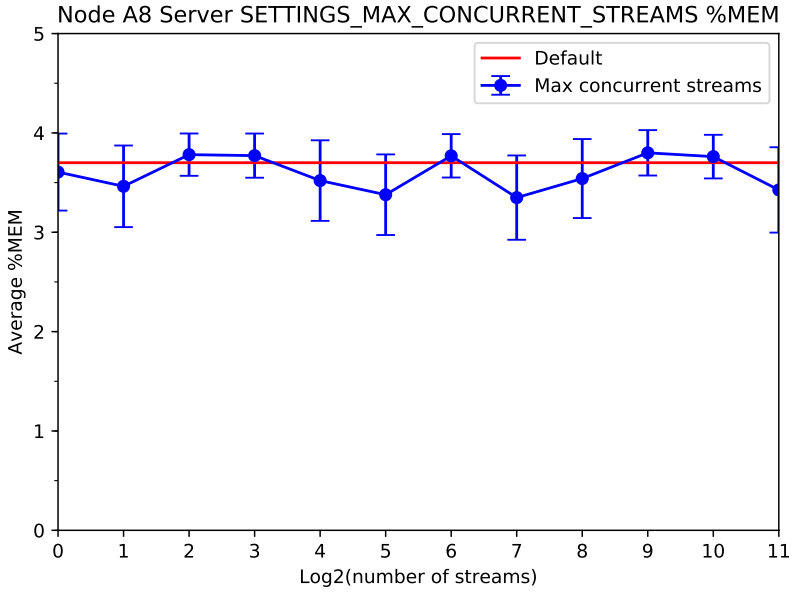


Figura 6.36: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_CONCURRENT_STREAMS.

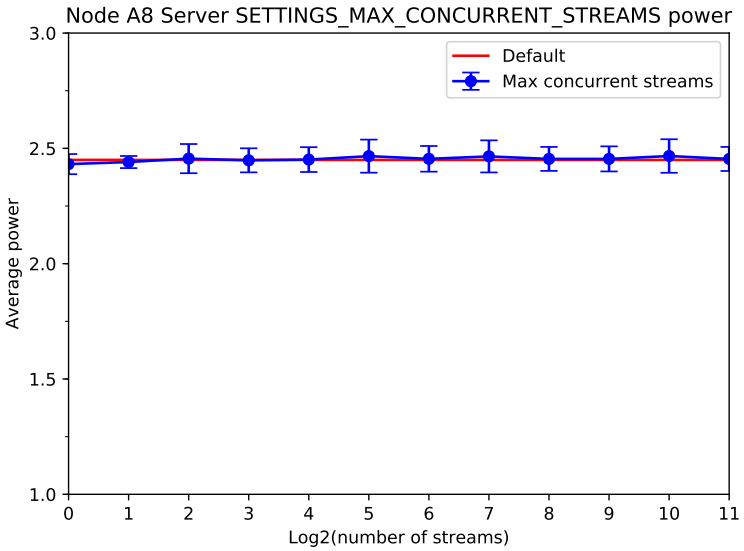


Figura 6.37: Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-CONCURRENT_STREAMS.

Medidas para diferentes valores de tamaño inicial de ventana

En la Figura 6.38 se puede ver un uso de CPU oscilante que no se aleja más de un 2 % del valor de referencia que se da en $x = 16$. En las Figuras 6.39 y 6.40 se observa un comportamiento similar para uso de memoria y potencia. Aunque en algunos valores de memoria el uso parece ser ligeramente inferior, sigue manteniendo un comportamiento poco claro.

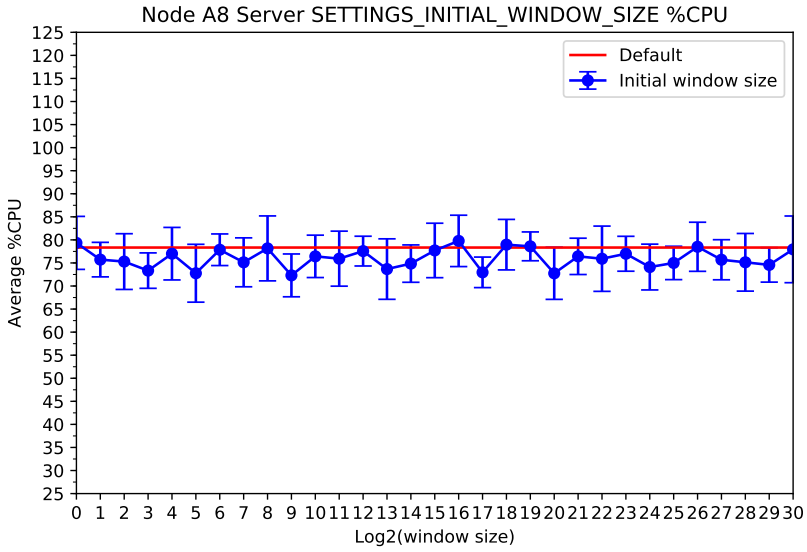


Figura 6.38: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-INITIAL_WINDOW_SIZE.

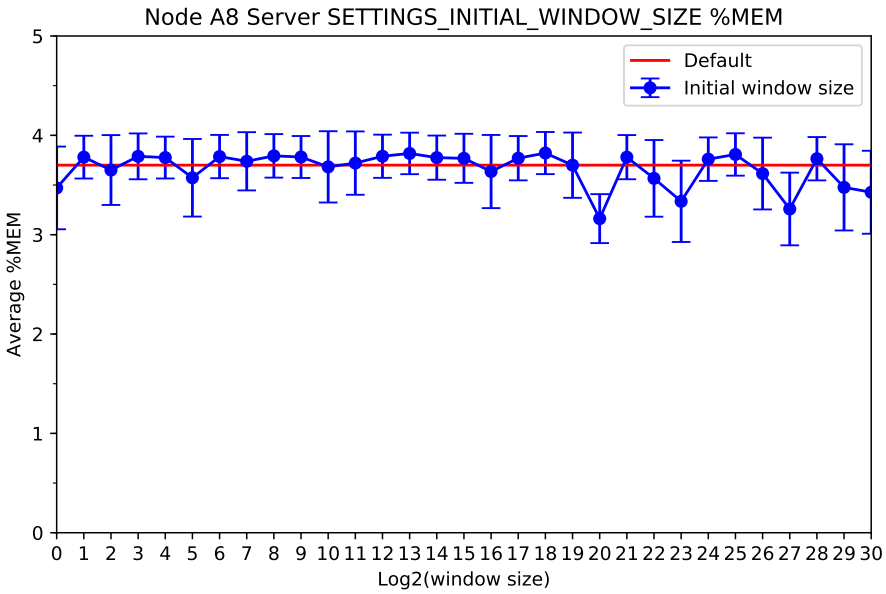


Figura 6.39: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_INITIAL_WINDOW_SIZE.

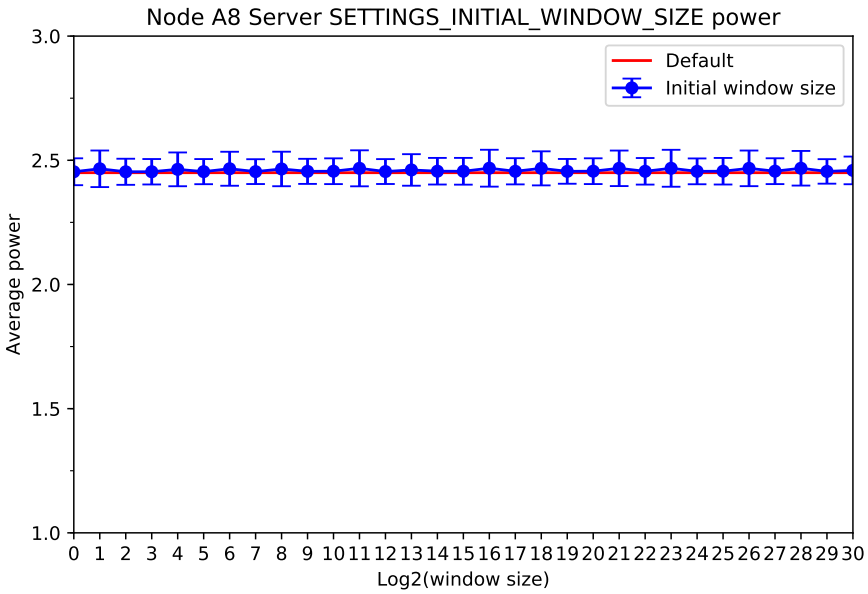


Figura 6.40: Potencia promedio para diferentes valores del parámetro SETTINGS_INITIAL_WINDOW_SIZE.

Medidas para distintos valores de tamaño máximo de frame

En la Figura 6.41 se observa para la mayoría de los valores del tamaño de frame un uso mayor de CPU que la configuración por defecto, aún así es de menos de un 4 %. Desde $x = 22$ en adelante, se observa un descenso brusco que teóricamente se aleja de lo esperado. Es probable, dado los recursos limitados del nodo, que a partir de cierto punto los recursos hayan sido insuficientes para procesar las solicitudes adecuadamente, o que el nodo que simulaba los distintos clientes haya disminuido temporalmente la frecuencia de envío de peticiones, quizás por sus propias limitaciones de recursos.

Por otro lado, en la Figura 6.42 nuevamente aparece un comportamiento oscilante del uso de memoria, aunque superior hasta un 1 %. No se observa, como en el caso de la CPU, un descenso brusco en el uso que pudiera sustentar la hipótesis de que en esos valores los recursos de los nodos hayan sido insuficientes, pese a esto, se observó que incluso al procesar la mitad de las solicitudes, el uso de memoria es aproximadamente el mismo, por lo que este resultado tampoco es un indicador de lo contrario. Los datos obtenidos en esta curva son siempre superiores a los de la configuración por defecto hasta para $x = 14$, el valor por defecto. Este resultado que fue observado también en otros experimentos. El uso de memoria parece menos estable en el nodo A8 que en la Raspberry Pi, probablemente por que tiene una limitación de recursos mayor.

Por último en la Figura 6.43, el comportamiento de la potencia es muy estable y no presenta cambios a considerar.

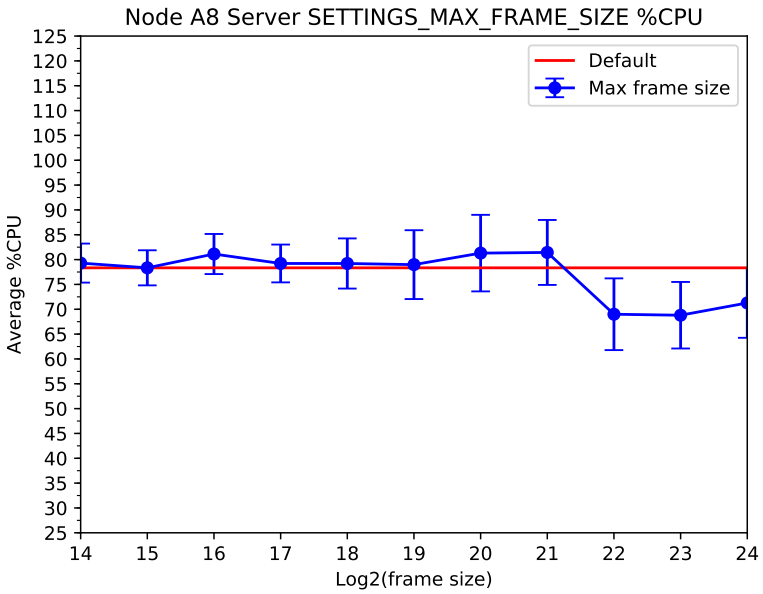


Figura 6.41: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_FRAME_SIZE.

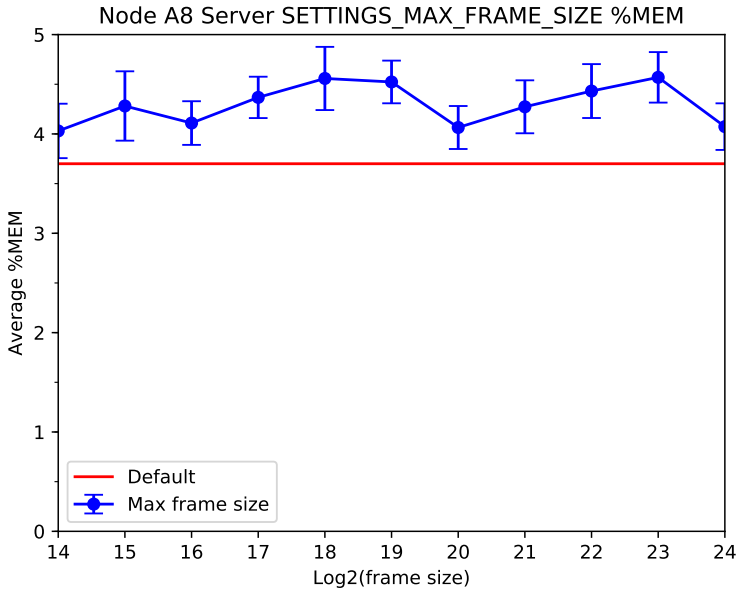


Figura 6.42: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_FRAME_SIZE.

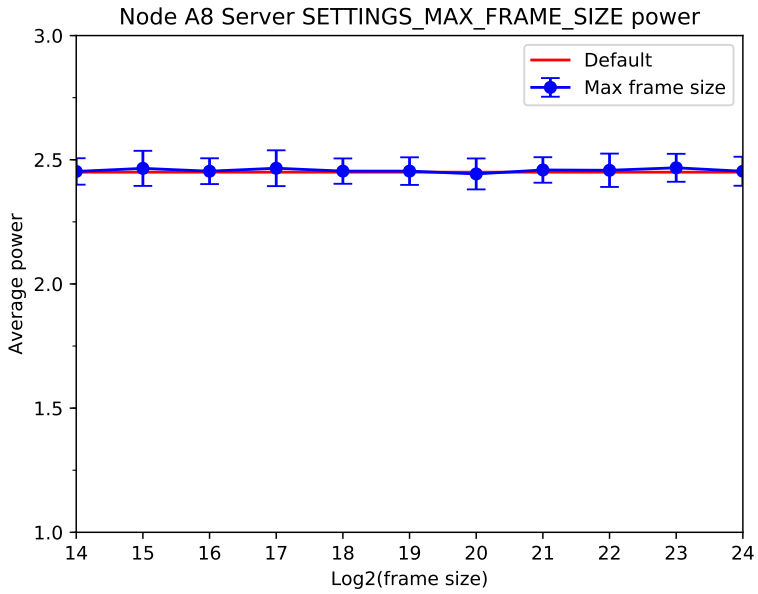


Figura 6.43: Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_FRAME_SIZE.

Medidas para distintos valores de tamaño máximo de lista de headers

En la Figura 6.44 se aprecian algunos puntos donde se tiene un uso menor de CPU de hasta un 7 %. Este cambio no se observa en el servidor de Raspberry Pi, y no hay razón para suponer que un uso de una lista mayor de headers pudiera implicar un uso inferior de CPU. Como ya se mencionó, los headers constituyen menos de un 3 % de la transacción de 500 bytes que se realiza, y el tamaño de la lista de headers se mide en bytes. Esto puede nuevamente deberse a las dificultades para simular los clientes y a la cantidad de conexiones que están constantemente abriéndose y cerrándose.

En cambio, en la Figura 6.45 existen algunos puntos en los que se logra un uso menor de memoria de hasta un 0,4 % respecto al valor de referencia. Por último, en la Figura 6.46 una vez más la potencia se mantiene estable.

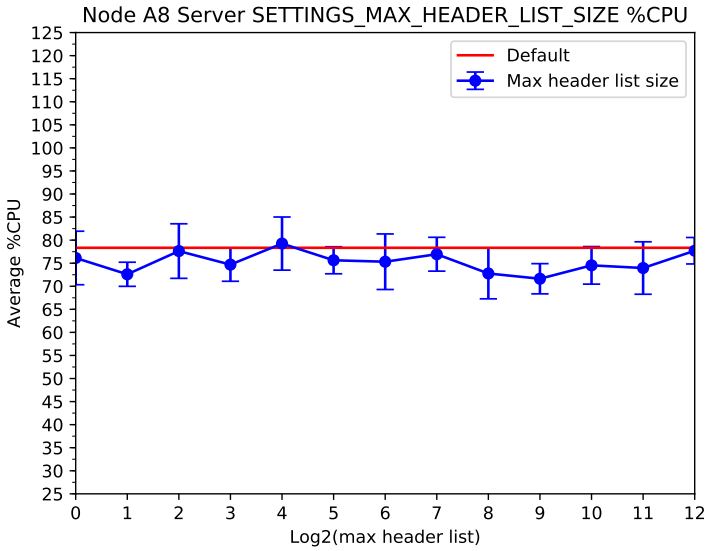


Figura 6.44: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_HEADER_LIST_SIZE.

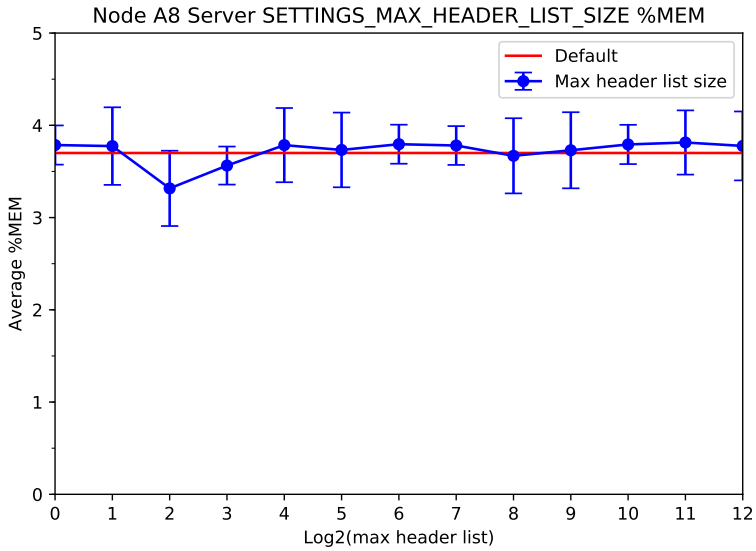


Figura 6.45: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_HEADER_LIST_SIZE.

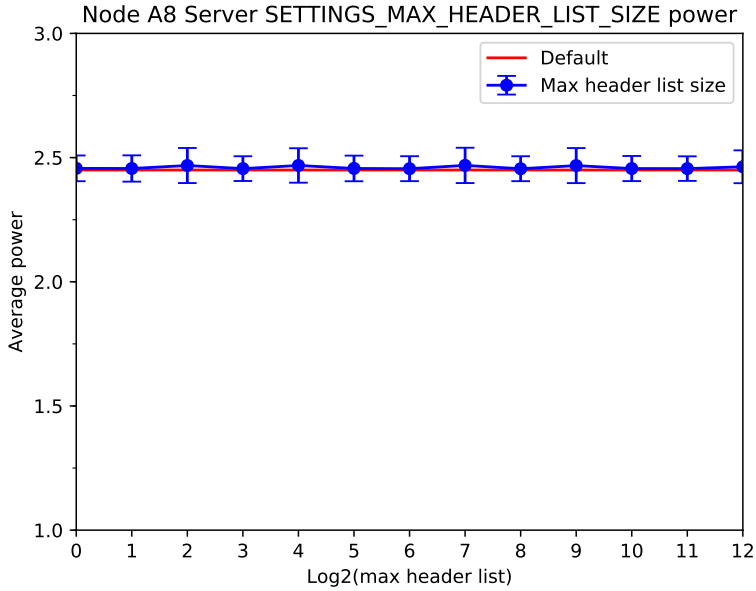


Figura 6.46: Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_HEADER_LIST_SIZE.

Medidas para distintos valores de tamaño de tabla de headers

Para el caso del tamaño de la tabla de headers en la Figura 6.47 en casi todas las zonas de la Figura se puede observar un uso de CPU inferior al de la configuración por defecto del orden de hasta un 5 %. Sin embargo, el valor por defecto se da en $x = 12$ donde se aprecia en cambio, una diferencia respecto a lo esperado. Además, la desviación estándar de los datos en este punto es de un 4,31 %. En la Figura 6.48 se pueden ver algunas zonas en las que el uso de memoria es hasta un 0,4 % inferior, aunque con un comportamiento poco claro centrado en la configuración por defecto. La potencia en la Figura 6.49 se mantiene estable.

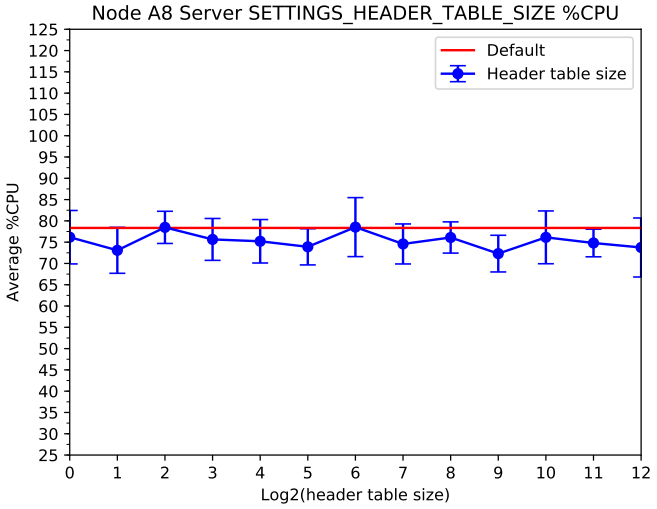


Figura 6.47: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-HEADER_TABLE_SIZE.

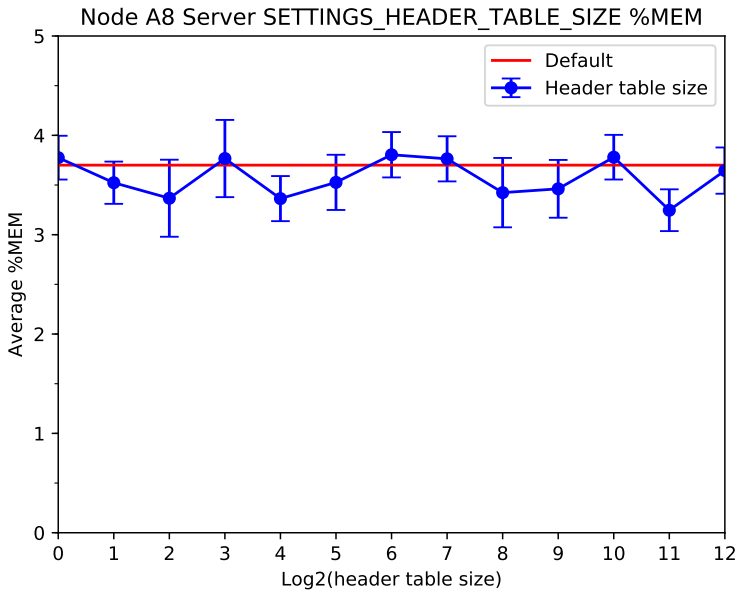


Figura 6.48: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_HEADER_TABLE_SIZE.

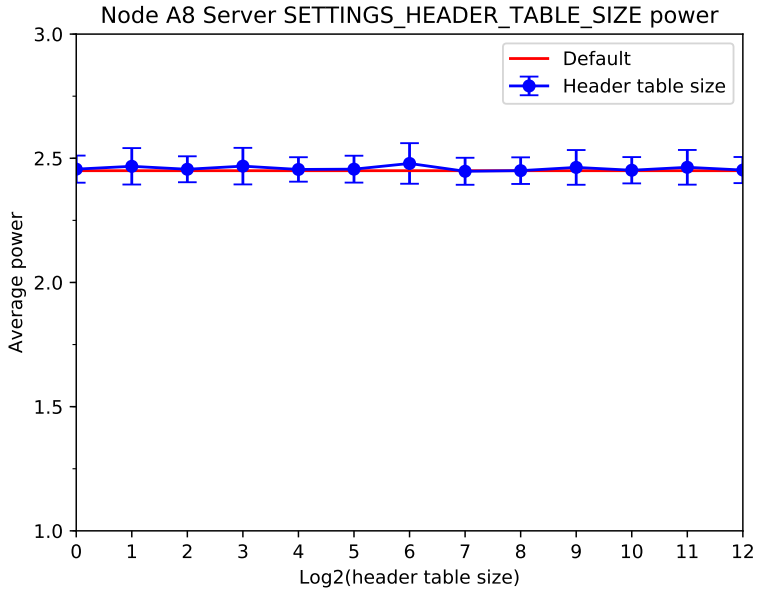


Figura 6.49: Potencia promedio para diferentes valores del parámetro SETTINGS_HEADER_TABLE_SIZE.

6.3.4. A8 open node como cliente

Medidas configuración por defecto

%CPU promedio	%MEM promedio	Potencia promedio [W]
29,2	1,36	2,38
Desviación %CPU	Desviación %MEM	Desviación potencia [W]
1,50	0,24	0,02

Tabla 6.8: Resultados para configuración por defecto

Medidas sin Server Push

De manera similar a la anterior se obtiene, dejando todos los parámetros por defecto excepto Server Push, los resultados de la Tabla 6.3:

%CPU promedio	%MEM promedio	Potencia promedio [W]
30,51	1,32	2,38
Desviación %CPU	Desviación %MEM	Desviación potencia [W]
3,26	0,20	0,02

Tabla 6.9: Resultados sin Server Push

Medidas para diferentes valores de número máximo de streams concurrentes

Aquí tanto la Figura 6.50 como la Figura 6.52, mantienen un patrón muy similar al observado en la mayoría de los experimentos expuestos anteriormente. Sin embargo en la Figura 6.51 se observa que se alcanzan valores de uso de memoria hasta un 0,1 % inferiores. Pese a que se usa una escala similar a la usada para los gráficos de memoria de la Raspberry Pi, el impacto parece ser muy superior, esto probablemente debido a que al tener los nodos A8 menos recursos de memoria, cualquier cambio pequeño es porcentualmente más grande y, por lo tanto, la memoria tiende a ser menos estable que en el caso de la Raspberry Pi.

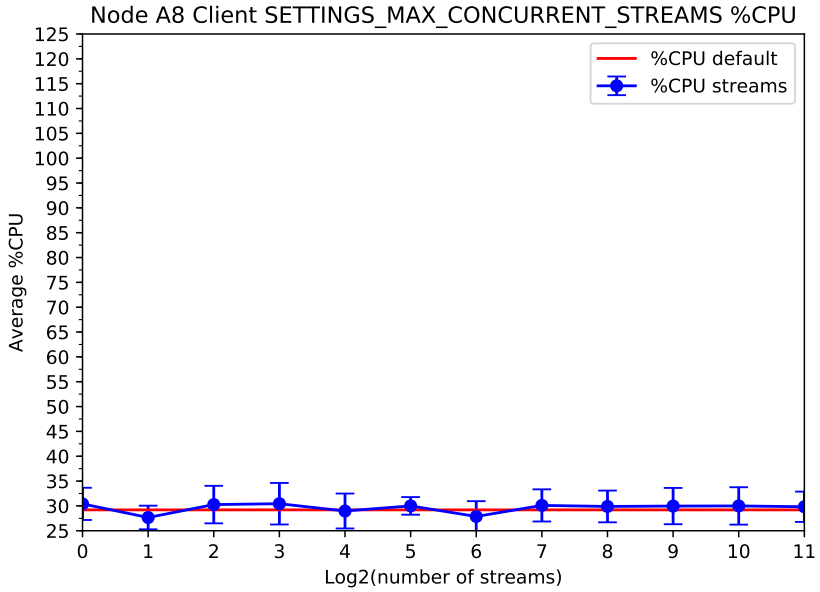


Figura 6.50: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_CONCURRENT_STREAMS.

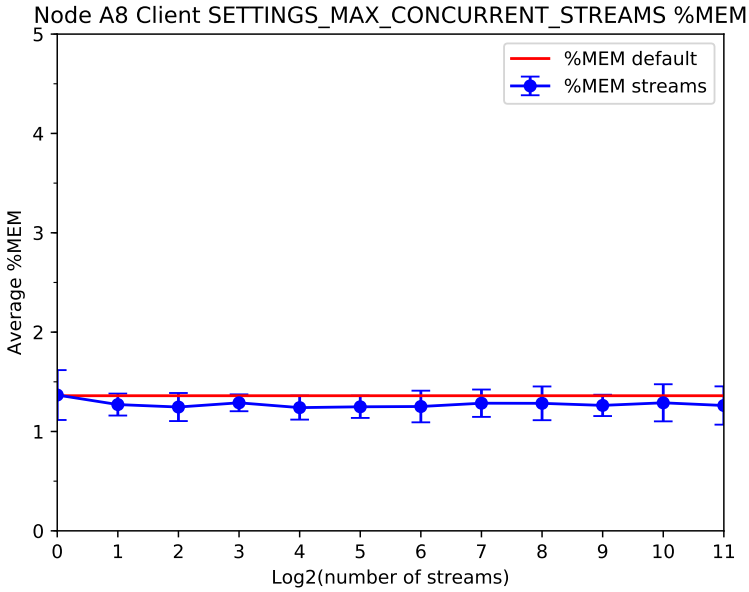


Figura 6.51: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_CONCURRENT_STREAMS.

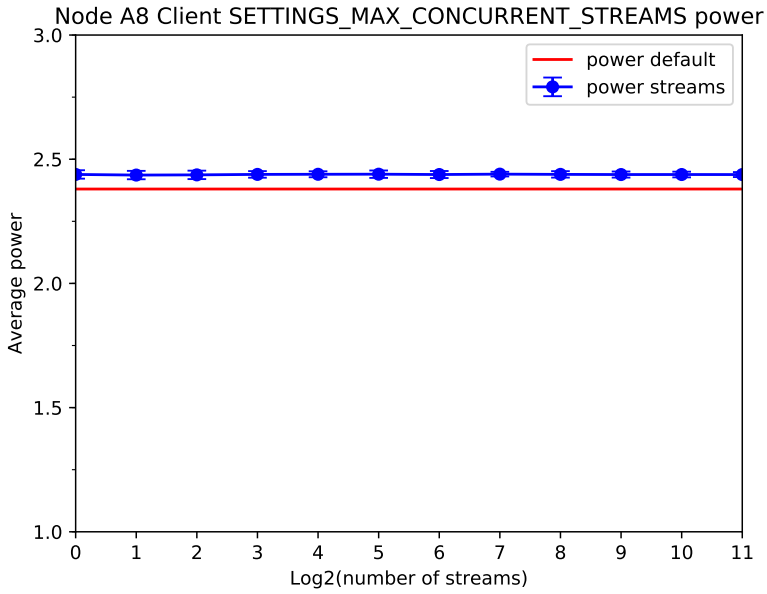


Figura 6.52: Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-CONCURRENT_STREAMS.

Medidas para diferentes valores de tamaño inicial de ventana

En la Figura 6.53 es posible ver puntos en los que el uso de CPU es ligeramente menor que el uso por defecto, sin embargo, el comportamiento general no es claro. Para la Figura 6.54 el uso de memoria es siempre inferior, lo que habla principalmente de la variabialidad del uso de memoria promedio ya que el uso de memoria debiera ser igual en $x = 16$, ya que 2^{16} es el valor por defecto del tamaño de ventana inicial. Similar a lo anterior, en la Figura 6.55 la potencia se mantiene siempre por sobre la de la configuración por defecto, lo que también habla de cierta variabilidad o inestabilidad en esta.

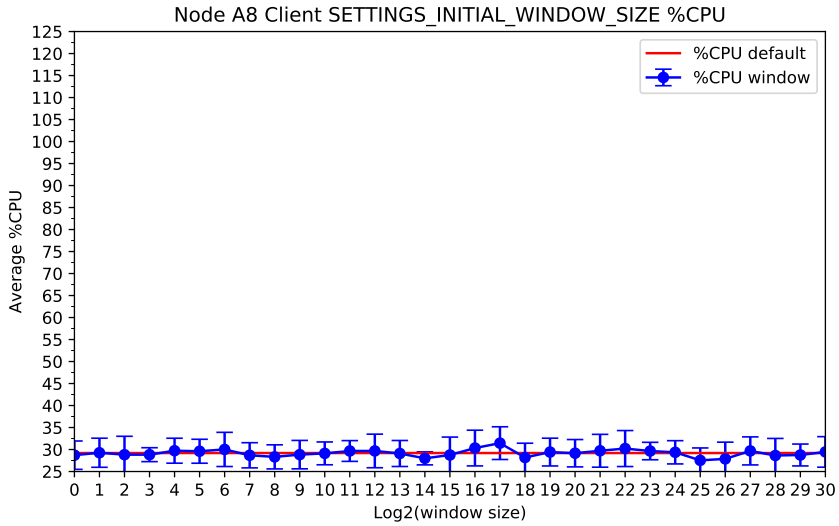


Figura 6.53: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-INITIAL_WINDOW_SIZE.

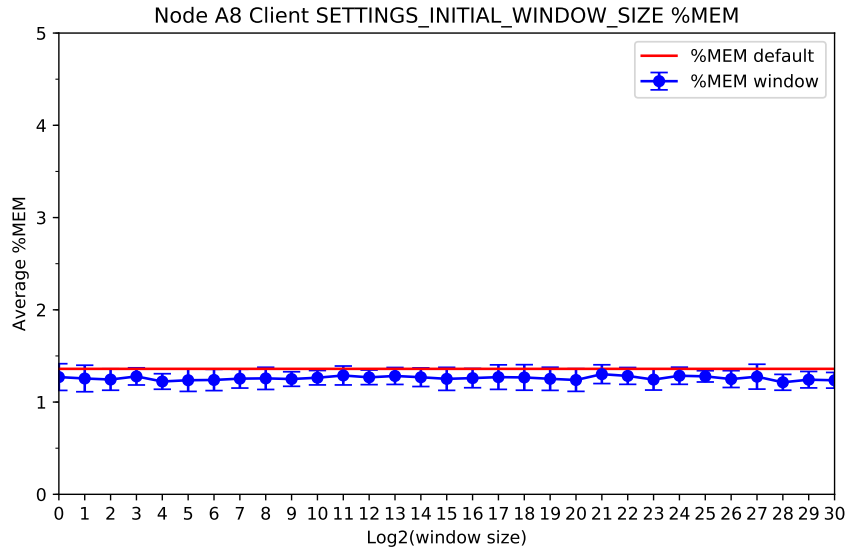


Figura 6.54: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_INITIAL_WINDOW_SIZE.

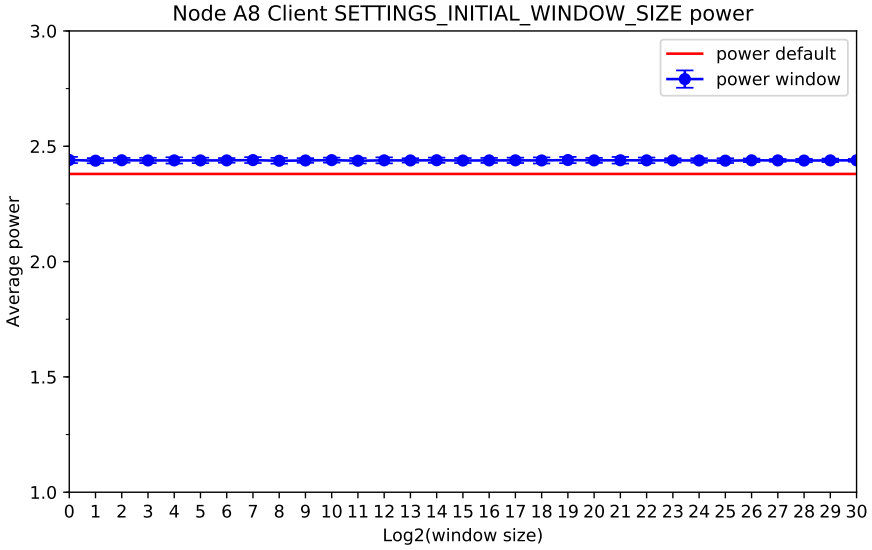


Figura 6.55: Potencia promedio para diferentes valores del parámetro SETTINGS_INITIAL_WINDOW_SIZE.

Medidas para distintos valores de tamaño máximo de frame

Para las Figuras 6.56, 6.57 y 6.58 se aprecia un comportamiento del mismo estilo del descrito en la subsección inmediatamente anterior: un comportamiento de CPU oscilante, uso de memoria inferior al uso por defecto y potencia estable.

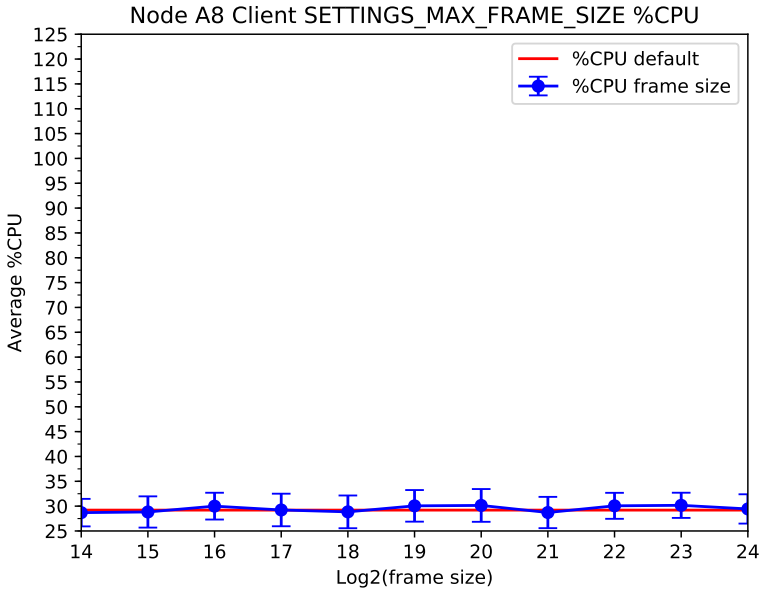


Figura 6.56: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_FRAME_SIZE.

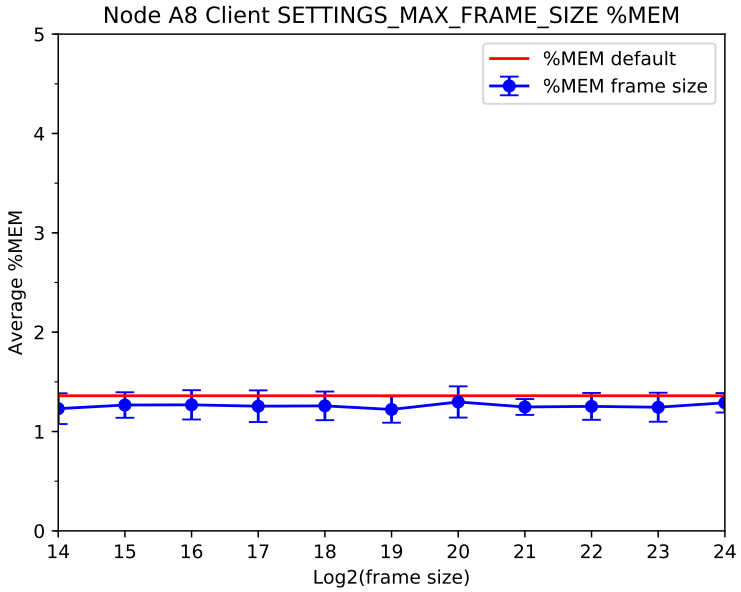


Figura 6.57: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_FRAME_SIZE.

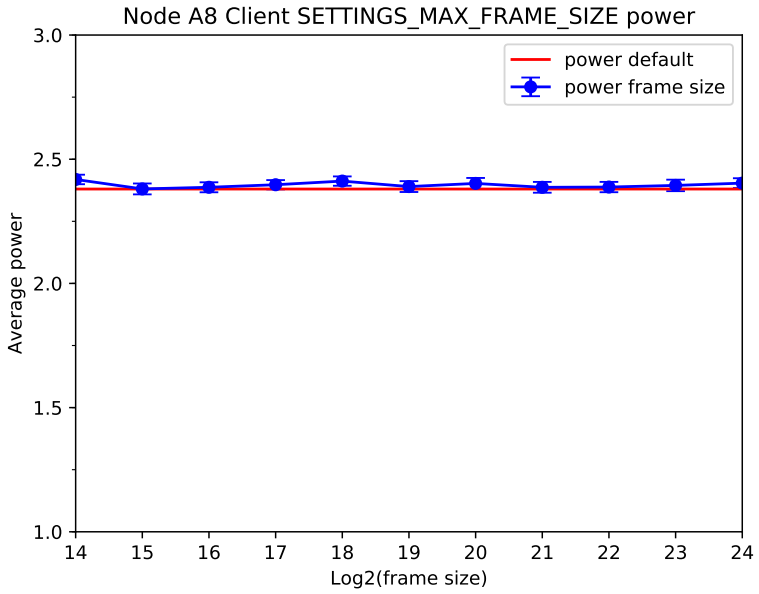


Figura 6.58: Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-FRAME_SIZE.

Medidas para distintos valores de tamaño máximo de lista de headers

En la Figura 6.59 se observa el comportamiento ya habitual, así como también en la Figura 6.61. En cambio, en la figura 6.60 se ve que la curva descrita por la variación del uso de memoria de SETTINGS_MAX_HEADER_LIST_SIZE se centra alrededor de 1,25 con algunas disminuciones bruscas de hasta 0,23 % en relación al uso de memoria por defecto. El hecho de que la muestra se encuentre centrada en torno a un valor distinto del por defecto, probablemente obedece a esta cierta inestabilidad observada a veces en el uso de memoria.

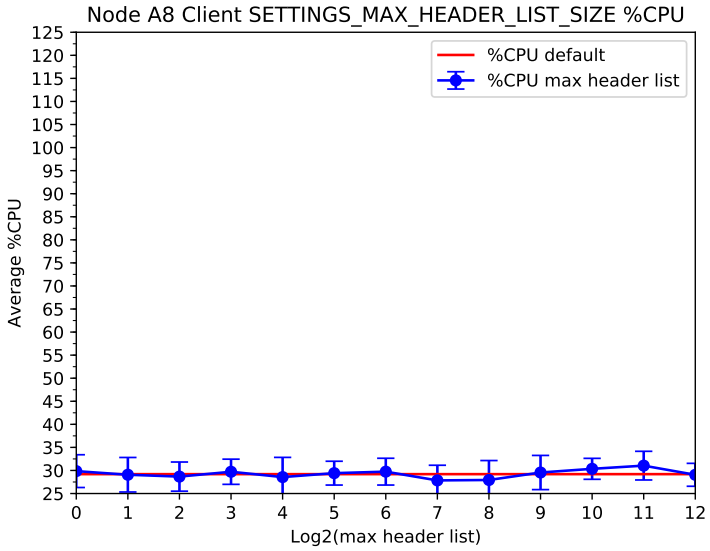


Figura 6.59: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-MAX_HEADER_LIST_SIZE.

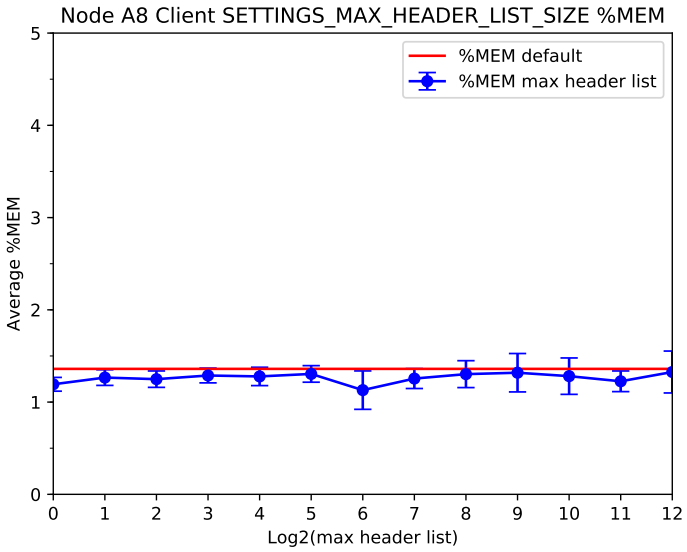


Figura 6.60: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_MAX_HEADER_LIST_SIZE.

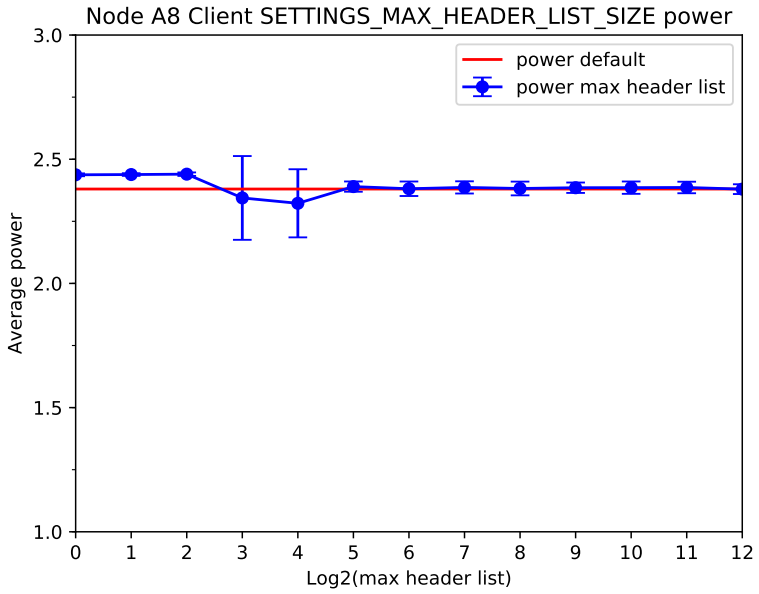


Figura 6.61: Potencia promedio para diferentes valores del parámetro SETTINGS_MAX_-HEADER_LIST_SIZE.

Medidas para distintos valores de tamaño de tabla de headers

En las Figuras 6.62 y 6.64 se puede ver una vez más el comportamiento más común entre los resultados de los experimentos realizados. En la Figura 6.63 se ve una tendencia a la disminución del uso de memoria conforme el tamaño de la tabla crece. Aún así, la diferencia es de 0,1 %, explicable por otros factores como la inestabilidad ya mencionada.

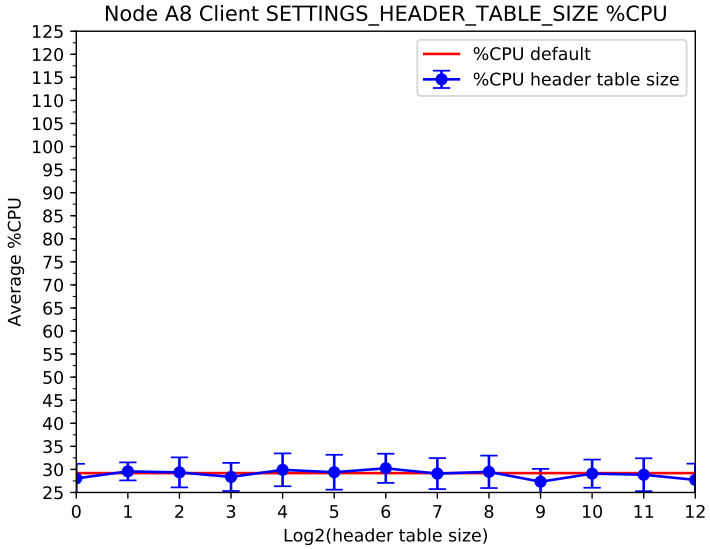


Figura 6.62: Porcentaje de uso de CPU para diferentes valores del parámetro SETTINGS_-HEADER_TABLE_SIZE.

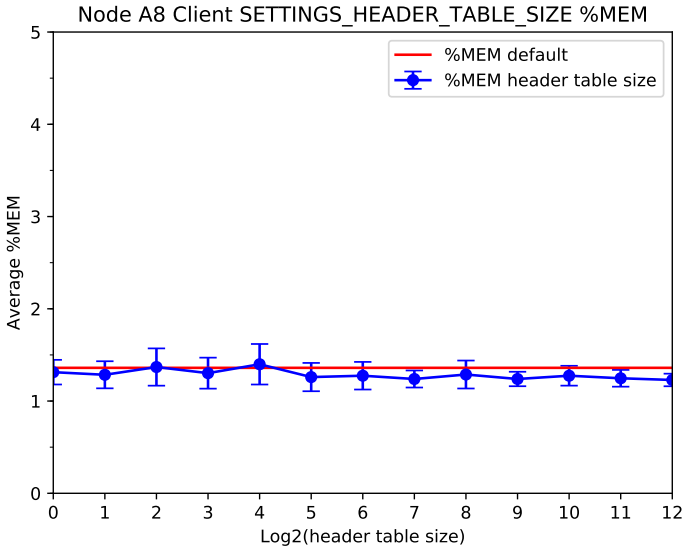


Figura 6.63: Porcentaje de uso de memoria RAM para diferentes valores del parámetro SETTINGS_HEADER_TABLE_SIZE.

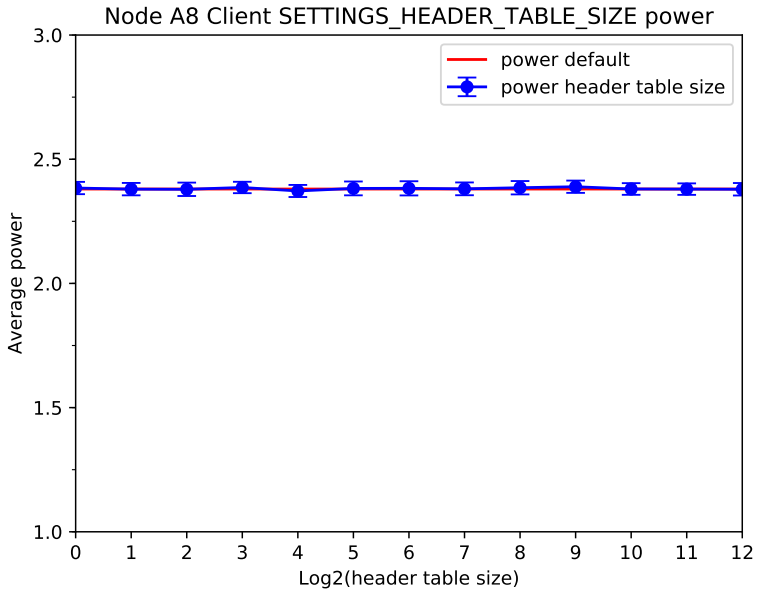


Figura 6.64: Potencia promedio para diferentes valores del parámetro SETTINGS_HEADER_TABLE_SIZE.

Capítulo 7

Discusión

El problema de determinar el funcionamiento completo de un protocolo de comunicación en un escenario de Internet de las Cosas es complejo, no solo por el amplio espectro de aplicaciones existentes, sino que también por las misma naturaleza del concepto que implica el trabajo con dispositivos restringidos.

No solo las implementaciones de HTTP/2 son escasas, además no son compatibles con buena parte de las arquitecturas presentes en estos ambientes. La implementación utilizada para estos experimentos no correspondía a una implementación que pudiera utilizarse para realizar todas las pruebas directamente en los nodos sin pasar por algunas modificaciones, porque pese, a que se podía utilizar en sistemas operativos basados en Linux, no incluía todos los parámetros que se deseaban estudiar como opciones y, en particular, el tamaño máximo de frame y el tamaño máximo de lista de headers debieron ser parcialmente implementados.

Además, el acceso a los dispositivos puede ser limitado. En el caso de los nodos A8, todas las pruebas fueron realizadas a distancia y se debió simular múltiples clientes desde un nodo pese a sus propias limitaciones, lo que implicó bastante inestabilidad en el sistema. Se intentó mejorar esto haciendo uso de 6 nodos para simular 16 clientes cada uno, pero no hubo un cambio en el comportamiento que diera razones para volver a realizar los experimentos con esta nueva configuración.

Otro desafío lo imponen las mismas tecnologías de comunicación, ya que garantizar que estas han sido estables durante todo el proceso no es una tarea sencilla y su inestabilidad puede producir alteraciones en los resultados al tratarse de mediciones precisas relacionadas con recursos bastante limitados.

Por otro lado, HTTP/2 es un protocolo diseñado para entornos web aunque es bastante utilizado en entornos restringidos, pese a la existencia de protocolos diseñados específicamente para este escenario. Una de las particularidades de la Internet de las Cosas es que abarca una gran cantidad de aplicaciones, desde monitoreo de sensores que pueden estar a kilómetros de distancia hasta dispositivos inteligentes presentes en los hogares que no requieren conectividad mayor que una decena de metros. Es en estos últimos escenarios donde HTTP/2 con WiFi o Ethernet puede jugar un rol fundamental, dada las distancias existentes y la seguridad

requerida, y de ahí la utilidad de este trabajo, ya que es poco probable que en el futuro un solo protocolo sea utilizado para toda esta variedad de aplicaciones. Además, la evaluación del conjunto de parámetros de HTTP/2 puede potencialmente facilitar la reutilización de aplicaciones hechas para Internet de las Cosas ya implementadas en HTTP/1.x dado que la semántica no cambia.

Las diferencias encontradas en los resultados de los servidores de ambos dispositivos se deben posiblemente a tres factores: la forma en que fueron simulados los clientes, la tecnología utilizada para la comunicación y los recursos de los nodos. En el caso de la Raspberry Pi se tuvo la posibilidad de simular los clientes desde un computador tradicional, lo que no ocurrió para el nodo A8 que es además mucho más restringido.

Otro factor a considerar en las mediciones de potencia viene del hecho de que la forma de medir la potencia en ambos dispositivos se realizó de otra manera. FIT IoT-LAB ofrece poder medir la potencia en los nodos A8, sin embargo en la Rasperry Pi se utilizó un módulo Arduino cuyas mediciones tenían una desviación estándar que tendía a ser algo superior a las encontradas en las mediciones de los nodos A8, incluso sin realizar una modificación en la configuración. Esto se dio en particular para el caso de los servidores, ya que la potencia de los clientes presentaba muy poca variación.

Finalmente, en relación a las medidas para los clientes se observó una mayor inestabilidad en las métricas consideradas, probablemente porque el proceso de enviar una solicitud tiene momentos en los que se realiza un uso mayor de recursos que decae en el momento en que esta petición es finalmente enviada, pero no hubo un cambio significativo en relación a la configuración de los parámetros.

Capítulo 8

Conclusión

Las conclusiones de este trabajo relativas al tráfico y la cantidad de información utilizada en los experimentos son las siguientes:

- Una cantidad de tráfico equivalente al tráfico web de ninguna manera implica un comportamiento similar si esta información se envía en respuestas separadas a distintos clientes.
- Para lograr realmente forzar el realizar pruebas con streams y sus respectivas ventanas, debiera enviarse más de una petición a una única URI. Esto actualmente no es posible. Como alternativa, se podría poner más información en el archivo que se recibe como respuesta, pero esto sería equivalente a probar el caso web (una sola página con mucha información).

En relación a HPACK y los parámetros de HTTP/2 se concluye que:

- HPACK resulta poco efectivo en un escenario donde la cantidad de headers es muy baja. Existe una ganancia, en términos de compresión, demasiado pequeña como para ser notada.
- El ajuste de los parámetros de HTTP/2 puede resultar muy útil para el escenario para el que fueron diseñados. Sin embargo, dicho ajuste resulta prácticamente inútil para el caso de IoT donde la cantidad de headers es escasa y la cantidad de información por transacción también lo es.

Es más, en el trabajo "Performance Evaluation of HTTP/2 Window Size in the Internet of Things" [LGC+16], se evalúa el efecto de la ventana en el uso de los mismos recursos en Raspberry Pi y se encuentra que existe una diferencia bastante clara. Sin embargo, pese a que para esto se utilizó un dispositivo restringido, se hizo uso de tráfico de tipo web y no IoT, lo que explica la discrepancia de resultados, precisamente porque los parámetros de HTTP/2 fueron diseñados pensando en este tipo de tráfico.

Por otro lado, es importante mencionar que la herramienta de benchmarking de la librería utilizada, llamada h2load, tiene un comportamiento distinto al de los clientes simulados para las pruebas realizadas. La herramienta h2load, establece la conexión una única vez durante su

ejecución y envía las solicitudes a través de esta. Sin embargo, los clientes utilizados para las pruebas son clientes que abren y cierran las conexiones completamente tras cada operación, lo que tiene una exigencia de recursos superior. Esto puede explicar especialmente en el nodo A8, que este no haya sido capaz de manejar la tasa de solicitudes que sí pudo manejar durante el benchmarking.

Proponer una configuración para HTTP/2 que se adapte a la Internet de las Cosas como un conjunto, contribuyendo a optimizar su funcionamiento, es poco factible. Esto dada la amplia variedad de dispositivos que existe en estos escenarios. Además, proponer una forma realista de tráfico IoT es muy difícil, por la misma razón, y es probable que otro tipo de tráfico muestre resultados distintos.

En particular para las plataformas estudiadas, como conclusión final de este trabajo, se establece que las distintas configuraciones de cada parámetro de HTTP/2 no conllevan una diferencia significativa en el uso de CPU, memoria y potencia. Y, por lo tanto, la configuración por defecto es suficiente para lograr hacer un uso efectivo de HTTP/2 en un entorno IoT con un tráfico usual de paquetes del orden de 500 bytes. Esta conclusión implica que si se tiene una aplicación para Internet de las Cosas que funciona con HTTP/1.x y tráfico similar al estudiado, esta puede ser actualizada a HTTP/2 sin necesidad de ninguna configuración de parámetros adicional para obtener un mejor uso de las métricas analizadas.

8.1. Trabajo futuro

Como trabajo futuro, se plantea hacer experimentos donde se modifique más de un parámetro a la vez, aunque los resultados de este trabajo indiquen que no debiera haber una diferencia importante.

Además realizar las mismas pruebas en otras plataformas, por ejemplo Arduino, podría contribuir a confirmar o extender los resultados aquí obtenidos.

Bibliografía

[Ard] Arduino. Arduino Mega. http://store.arduino.cc/usa/arduino-mega-2560-rev3/.

[BPT15] M. Belshe, R. Peon, and M. Thomson. Hypertext Transfer Protocol Version 2 (HTTP/2). RFC 7540, RFC Editor, May 2015. http://www.rfc-editor.org/rfc/rfc7540.txt.

[Cab18] Benjamin Cabé. IoT Developer Survey 2018, 2018. https://es.slideshare.net/kartben/iot-developer-survey-2018.

[CGH+02] Ed Callaway, Paul Gorday, Lance Hester, Jose A Gutierrez, Marco Naeve, Bob Heile, and Venkat Bahl. Home networking with ieee 802.15. 4: a developing standard for low-rate wireless personal area networks. *IEEE Communications magazine*, 40(8):70–77, 2002.

[DR08] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, RFC Editor, August 2008. http://www.rfc-editor.org/rfc/rfc5246.txt.

[Ecu] EcuRed. Estándares IEEE 802.3 . http://www.ecured.cu/Est%C3%A1ndares_IEEE_802.3.

[Elm17] Asma Elmangoush. Evaluating the Features of HTTP/2 for the Internet of Things. 05 2017. https://www.researchgate.net/publication/320453832_Evaluating_the_Features_of_HTTP2_for_the_Internet_of_Things.

[Eva11] Dave Evans. The Internet of Things [INFOGRAPHIC], 2011. http://blogs.cisco.com/diversity/the-internet-of-things-infographic.

[FGM+99] Roy T. Fielding, James Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach, and Tim Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, RFC Editor, June 1999. http://www.rfc-editor.org/rfc/rfc2616.txt.

[For86] Internet Engineering Task Force. IETF, 1986. http://www.ietf.org/.

[Fou] Raspberry Pi Foundation. Raspberry Pi 3 Model B. http://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[Gri13] Ilya Grigorik. High Performance Browser Networking: What every web developer should know about networking and web performance, 2013. https://hpbn.co/.

[Gro] WiNet Research Group. WiNet Research Group. http://winetgroup.die.cl.

[h2o18] H2O - the optimized HTTP/1.x, HTTP/2 server, 2018. https://h2o.examp1e.net/.

[Htt] Httparchive. Page weight. https://httparchive.org/reports/page-weight.

[HV18] Egemen Hopali and Özalp Vayvay. Internet of Things (IoT) and its Challenges for Usability in Developing Countries. *International Journal of Innovation Engineering and Science Research*, 1:6–9, 01 2018. https://www.academia.edu/36373903/Internet_of_Things_IoT_and_its_Challenges_for_Usability_in_Developing_Countries.

[IET15] IETF. HTTP/2 Approved, 2015. http://www.ietf.org/blog/http2-approved/.

[IL] FIT IoT-LAB. A8 open node. http://www.iot-lab.info/hardware/a8/.

[INR67] INRIA. Inria, 1967. http://www.inria.fr/.

[LA17] Inc. LoRa Alliance. Lorawan 1.1 specification, 2017.

[Lab] NIC Chile Research Labs. NIC Chile Research Labs. http://niclabs.cl/.

[Lab18] NIC Chile Research Labs. TESTHTTP, 2018. https://github.com/niclabs/TESTHTTP.

[LC16] Diego Londoño and Sandra Céspedes. Performance Evaluation of CoAP and HTTP/2 in Web Applications. In *CEUR Workshop Proceedings*, volume 1727, pages 25–27, 2016. http://ceur-ws.org/Vol-1727/ssn16-final5.pdf.

[LG17] Stephen Ludin and Javier Garza. *Learning HTTP/2: a practical guide for beginners*. O'Reilly, 2017. https://www.imel.ba/edukacija/learninghttp2.pdf.

[LGC+16] Diego Londoño, Maite González, Sandra Céspedes, Javier Bustos, and Gabriel Montenegro. Performance Evaluation of HTTP/2 Window Size in the Internet of Things. October 2016. http://ceur-ws.org/Vol-1950/paper8.pdf.

[MCLS16] Gabriel Montenegro, Sandra Céspedes, Salvatore Loreto, and Robby Simpson. H2oT: HTTP/2 for the Internet of Things. Internet-Draft draft-montenegro-httpbis-h2ot-00, IETF Secretariat, July 2016. https://tools.ietf.org/id/draft-montenegro-httpbis-h2ot-00.txt.

[MCLS17] Gabriel Montenegro, Sandra Cespedes, Salvatore Loreto, and Robby Simpson. Http/2 configuration profile for the internet of things. Internet-Draft draft-montenegro-httpbis-h2ot-profile-00, IETF Secretariat, March 2017.

[Mil16] Robert Miller. Lora security. building a secure lora solution, 2016.

[Nai17] N. Naik. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–7, Oct 2017. https://ieeexplore.ieee.org/abstract/document/8088251.

[New18] NewGenApps. 13 IoT Statistics Defining the Future of Internet of Things, 2018. http://www.newgenapps.com/blog/iot-statistics-internet-of-things-future-research-data/, urldate = 2018-08-31.

[OAS15] OASIS. MQTT Version 3.1.1 Plus Errata 01, December 2015. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html.

[Ope] OpenSSL. OpenSSL. https://www.openssl.org/.

[otT13] FIT (Future Internet of the Things). FIT IoT-LAB, 2013. http://www.iot-lab.info/.

[Pos81] Jon Postel. Transmission Control Protocol. STD 7, RFC Editor, September 1981. http://www.rfc-editor.org/rfc/rfc793.txt.

[PR15] R. Peon and H. Ruellan. HPACK: Header Compression for HTTP/2. RFC 7541, RFC Editor, May 2015. http://www.rfc-editor.org/rfc/rfc7541.txt.

[SHB14] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). RFC 7252, RFC Editor, June 2014. http://www.rfc-editor.org/rfc/rfc7252.txt.

[Sim15] Robby Simpson. Deuterium, 2015.

[SSG+17] Arunan Sivanathan, Daniel Sherratt, Hassan Habibi Gharakheili, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Characterizing and classifying IoT traffic in smart cities and campuses. In *Computer Communications Workshops (INFOCOM WKSHPS), 2017 IEEE Conference on*, pages 559–564. IEEE, 2017. https://ieeexplore.ieee.org/document/8116438/.

[std16] IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, Dec 2016. https://standards.ieee.org/project/802_11bb.html.

[Ste] Daniel Stenberg. HTTP2 explained. https://daniel.haxx.se/http2/.

[TATM15] H. Tschofenig, J. Arkko, D. Thaler, and D. McPherson. Architectural Consi-

derations in Smart Object Networking. RFC 7452, RFC Editor, March 2015. http://www.rfc-editor.org/rfc/rfc7452.txt.

[TF16] Gabriel Tolosa and Marcelo Fernández. *HTTP/2. Un nuevo protocolo para la web.* 11 2016. http://www.researchgate.net/publication/321197226_HTTP2_Un_nuevo_protocolo_para_la_web.

[Tsu15] Tatsuhiro Tsujikawa. Nghttp2, 2015. http://nghttp2.org.

[Wor] RF Wireless World. LoRa vs Zigbee | Difference between LoRa and Zigbee. http://www.rfwireless-world.com/Terminology/LoRa-vs-Zigbee.html.

[YS16] T. Yokotani and Y. Sasaki. Comparison with HTTP and MQTT on required network resources for IoT. In *2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, pages 1–6, Sep. 2016. https://ieeexplore.ieee.org/abstract/document/7814989.

Apéndice A

Tablas de resultados de Raspberry como servidor

A.1. Max concurrent streams

Número de streams	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
1	97.8238458742	7.6042427323	0.5993093421	0.0526003379	1.3880545455	0.1116869663
2	97.6869960044	7.5638578556	0.6017435525	0.0468158772	1.3521636364	0.0917284163
4	97.5026497278	7.6681463632	0.6048638838	0.0458709945	1.3752	0.1235588629
8	97.6651053014	7.7541481258	0.6099491649	0.0464793674	1.4073090909	0.1032316476
16	97.8098110465	7.4320519546	0.6083212209	0.0483814513	1.3398181818	0.1131443017
32	97.36712627	7.968214114	0.6081277213	0.0488634493	1.3800181818	0.0957660132
64	97.5118330309	7.9002443874	0.6050090744	0.0501662985	1.3705272727	0.1117314662
128	97.5092558984	7.9253336109	0.6031941924	0.0488498474	1.3722545455	0.089430755
256	97.6825281511	7.748816245	0.6062114057	0.0469404967	1.3633090909	0.1176296562
512	97.3558983666	8.8220318967	0.6058076225	0.0477035372	1.3944909091	0.090878607
1024	97.4663520871	8.0900238654	0.6046098004	0.0498051962	1.3990181818	0.1028750574
2048	97.6312999274	7.6628502678	0.6095860566	0.0543325519	1.3779636364	0.1015427229

Tabla A.1: Medidas para SETTINGS_MAX_CONCURRENT_STREAMS

A.2. Initial window size

Log2 initial window size	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
0	97.4526870007	8.4790051846	0.5929920116	0.0465080905	1.3546363636	0.089969617
1	97.8142857143	7.8616021631	0.6071246819	0.0397279669	1.3655454545	0.0882527553
2	97.3261248186	8.7221406947	0.5952104499	0.0407585869	1.3666545455	0.0716912666
3	97.7761627907	8.095530499	0.601380814	0.0437049728	1.3662363636	0.0940704944
4	97.5476034858	8.4102014832	0.6064633261	0.0438144587	1.3586727273	0.0747036254
5	97.6633127497	7.9658362682	0.5945877225	0.0403894426	1.3554363636	0.0828599362
6	97.7331515812	7.7775059469	0.6007633588	0.0557539952	1.3391818182	0.0918789644
7	98.2651746725	6.5709584108	0.6147016012	0.0502127085	1.3596	0.0820358097
8	97.9442909091	7.2342106228	0.6079636364	0.0447928678	1.3701272727	0.0736656748
9	98.0083303019	7.4126666651	0.6102946526	0.0456182714	1.3469272727	0.0938811018
10	97.8997819767	7.3012938076	0.602434593	0.0493273114	1.3864	0.0890725048
11	97.8207485465	7.4699121492	0.6107921512	0.0456364704	1.3202363636	0.0900746529
12	97.7680116321	7.54496786	0.6057797165	0.0508763296	1.3832363636	0.084255116
13	97.8552325581	7.0579932606	0.6052325581	0.0474146238	1.3689818182	0.0812910844
14	97.6049019608	7.379224521	0.605664488	0.0463394274	1.3530545455	0.0851763787
15	97.6315904139	7.5793799371	0.6011982571	0.0467856973	1.3439454545	0.0768812018
16	97.6066787659	7.3071809646	0.6143375681	0.04911588	1.3356	0.1135842973
17	97.9503455802	7.7080595491	0.6058566752	0.0463190455	1.3394181818	0.0738393378
18	97.521923775	7.6902742481	0.6128493648	0.0449527004	1.4014909091	0.0698036592
19	97.3379760609	8.0579989806	0.6027566195	0.0527854383	1.3501636364	0.0951758146
20	97.2117071403	8.5903472115	0.593439652	0.0397200944	1.3415272727	0.0788800906
21	97.4586569873	8.0265391548	0.5965517241	0.0385712583	1.3679636364	0.0794002901
22	97.714612868	8.2332936646	0.5933842239	0.0343224966	1.3605818182	0.0788793478
23	97.1506286267	8.7275420325	0.5982108317	0.0369585083	1.3449272727	0.0877563661
24	97.6792535143	8.4799138509	0.5889481338	0.0376827423	1.3632181818	0.0845762183
25	97.4297820823	8.696221736	0.5972881356	0.0353883053	1.3757636364	0.0814623211
26	97.6042130751	8.1716930035	0.6028087167	0.0351058025	1.3318909091	0.1024914943
27	97.735336888	8.0515372541	0.5881725642	0.0362621301	1.3569818182	0.0805532929
28	97.7876393602	8.2973893854	0.5881240911	0.0360450214	1.3881272727	0.0792705294
29	97.5436046512	8.5654610736	0.6016472868	0.0412739405	1.3534363636	0.0796912476
30	97.4229912875	8.0487460986	0.5946272991	0.0390589238	1.3596545455	0.0757804378

Tabla A.2: Medidas para SETTINGS_INITIAL_WINDOW_SIZE

A.3. Max header list size

Max header list size	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
1	98.2599927193	6.9252888802	0.6054969057	0.0498614129	1.3486909091	0.1015237379
2	97.8628	8.5217229202	0.6019636364	0.051352732	1.3512	0.0890219282
4	98.5056507474	6.7904055112	0.6087860007	0.0466931937	1.3673272727	0.0816791497
8	98.5227040816	6.8362060592	0.60196793	0.049751014	1.3707090909	0.0862508557
16	98.6252643092	6.1523230739	0.6083485235	0.0479287202	1.3760363636	0.0865051837
32	98.3628914785	6.9876406279	0.606372906	0.045322058	1.3818363636	0.0937476446
64	98.3611070648	6.6332854499	0.6069555717	0.047052072	1.3794545455	0.0712984905
128	98.4409836066	6.7074786202	0.6032058288	0.0464159665	1.3692727273	0.0869095857
256	98.6278790087	6.3341679238	0.5942419825	0.0475004972	1.3394363636	0.0813412482
512	98.5119854281	6.4708759867	0.6109289617	0.045784907	1.3556181818	0.0848681965
1024	98.1670913724	6.958779504	0.5997087732	0.0523301436	1.3447090909	0.0892222927
2048	98.1559883509	7.1771124293	0.6041863851	0.0518480628	1.3766181818	0.0940741415
4096	97.9218466012	7.0729255107	0.6119956379	0.0495218738	1.3345636364	0.1114119422

Tabla A.3: Medidas para SETTINGS_MAX_HEADER_LIST_SIZE

A.4. Header table size

Header table size	Uso de CPU promedio	Desviación de CPU	Uso de memoria	Desviación de memoria	Potencia promedio	Desviación de potencia
1	98.1449417758	7.1178032633	0.6055676856	0.0526847968	1.3692181818	0.099415084
2	98.1616084425	6.9819164121	0.6070596798	0.0520530176	1.3373090909	0.0864372761
4	97.9434181818	7.9571166906	0.6025454545	0.047804401	1.3536727273	0.0837514078
8	97.753217012	7.8566346955	0.6057070156	0.0481678877	1.3266	0.1170823705
16	98.021171335	7.5456646659	0.6053110222	0.0509104126	1.3687272727	0.0816207251
32	97.8199563795	7.7572088325	0.6066521265	0.0417275409	1.3451454545	0.0803069155
64	98.0696616952	7.0958474896	0.6053837759	0.0460219559	1.3513636364	0.0901137889
128	98.1723334547	7.4635584045	0.6044776119	0.0497441574	1.3268363636	0.0716397271
256	98.0666909091	6.5311562359	0.6011272727	0.0436865445	1.3743818182	0.0844289074
512	97.7354287791	7.7714161757	0.6087209302	0.0437712307	1.3437454545	0.0742483865
1024	97.9254181818	7.8020254916	0.6072	0.0477863561	1.3482	0.096966088
2048	98.0207348127	7.2029732185	0.5983266642	0.0382110066	1.3508	0.0806162377
4096	98.1455041864	7.6075108547	0.6025118311	0.0429801542	1.3236181818	0.1115208818

Tabla A.4: Medidas para SETTINGS_HEADER_TABLE_SIZE

A.5. Max frame size

Max frame size	Uso de CPU promedio	Desviación de CPU	Uso de memoria	Desviación de memoria	Potencia promedio	Desviación de potencia
16384	97.8651094652	4.5165528236	0.5699854121	0.0458404806	1.399463138	0.0546528934
32768	97.6925263432	3.9272585459	0.5695889414	0.0460113409	1.3994461248	0.057749605
65536	97.6233234362	5.1568070821	0.5711944647	0.0464847436	1.4021701323	0.0535662198
131072	98.3312572662	4.6803618913	0.5637824023	0.0480717077	1.4184763705	0.0551951474
262144	98.2269866918	4.3837050875	0.5617207437	0.0486156951	1.4149584121	0.0791940352
524288	98.7480926102	4.8545042792	0.5623040467	0.0495870564	1.4041361059	0.0464042369
1048576	97.9076501877	4.2107569819	0.5684310702	0.0519087089	1.4049855321	0.0620990709
2097152	98.3838968034	3.5107466147	0.5814779976	0.0478503877	1.4015086254	0.0699971721
4194304	98.5679550263	3.4829476968	0.5777475417	0.0451446178	1.4136800552	0.0414938775
8388608	97.6964578058	4.4011878874	0.5778834944	0.0463706624	1.4183029199	0.0648188123
16777215	97.8144427776	5.0574368921	0.5747780184	0.046933252	1.4019928626	0.0471849662

Tabla A.5: Medidas para SETTINGS_MAX_FRAME_SIZE

Apéndice B

Tablas de resultados de Raspberry como cliente

B.1. Max concurrent streams

Número de streams	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
1	57.5042075736	2.6282560108	0.3741020794	0.0144485779	1.3136635161	0.0163104927
2	56.8589060309	3.9379301749	0.3725897921	0.0187893956	1.3044650284	0.0167759179
4	54.4727910238	3.9897297212	0.3597353497	0.0228648228	1.2999489603	0.0141329621
8	55.0910238429	3.6716073608	0.3684310019	0.021106095	1.3090586011	0.0199713087
16	56.970687237	3.9385948332	0.3724007561	0.0192490348	1.3131833648	0.0176894452
32	55.7294530154	3.4001678504	0.3720226843	0.0167790493	1.314563327	0.0165152654
64	54.9720897616	3.4344007812	0.3771266541	0.0150737085	1.3110926276	0.0120547407
128	54.5279102384	2.509194629	0.3689981096	0.0181790161	1.3094593573	0.0169963949
256	54.2942496494	3.4867093749	0.3765595463	0.0182937371	1.3159451796	0.0139022266
512	54.4179523142	3.5896920166	0.3792060491	0.0174877033	1.3081701323	0.0236817376
1024	53.8451612903	3.89136259	0.3816635161	0.0161597818	1.3056748582	0.0156123608
2048	54.5931276297	2.8045787511	0.3803402647	0.0165232767	1.3064461248	0.0147986034

Tabla B.1: Medidas para SETTINGS_MAX_CONCURRENT_STREAMS

B.2. Initial window size

Log2 initial window size	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
0	53.4503506311	3.2090453394	0.3688090737	0.0148111892	1.302241966	0.0138249237
1	55.4913043478	4.0178867566	0.3741020794	0.0160274651	1.3026559546	0.020150324
2	55.2265077139	3.4379904952	0.3786389414	0.0187655209	1.3015689981	0.0188495865
3	55.5692847125	3.5015803319	0.3769376181	0.0150860955	1.3161285444	0.0175064296
4	54.7196353436	3.386485404	0.3784499055	0.0183752423	1.3050718336	0.0147068718
5	55.1964936886	4.6046611298	0.3714555766	0.0193033012	1.3050567108	0.0138013142
6	56.1900420757	4.3035490686	0.3703213611	0.0233274388	1.3131625709	0.0195751236
7	53.9267882188	3.3449600186	0.3780718336	0.020539144	1.3100396975	0.0205831128
8	54.6820476858	3.5825136722	0.37731569	0.0174384993	1.3079962193	0.0181205028
9	54.4576437588	3.1953737368	0.3725897921	0.0190618057	1.3142551985	0.0226613567
10	54.4454417952	2.6970809301	0.3843100189	0.0205500546	1.3079432892	0.0215814337
11	55.1022440393	3.3644832871	0.3716446125	0.0197887783	1.309610586	0.013480852
12	54.9130434783	3.4822978384	0.375047259	0.0148439434	1.3226143667	0.0294969722
13	53.8755960729	3.815381049	0.3722117202	0.0192101792	1.3060170132	0.0176525621
14	55.2624123422	3.8862962751	0.3792060491	0.0177800679	1.3122911153	0.0195230835
15	54.9535764376	3.0000889269	0.3824196597	0.0222268708	1.3088620038	0.0150142377
16	55.2342215989	3.2465666498	0.3771266541	0.0160134734	1.3040453686	0.0175274634
17	57.2959326788	3.2665542595	0.368241966	0.0155902807	1.2997844991	0.0186090026
18	55.2806451613	4.0201801661	0.3799621928	0.0221207271	1.3128090737	0.0192087453
19	55.2612903226	3.6391796708	0.3810964083	0.0216840684	1.3142079395	0.0151465656
20	55.7270687237	3.3410772762	0.3797731569	0.0165097052	1.3073931947	0.0175300517
21	55.0625525947	3.6477225595	0.3769376181	0.0163961727	1.3109224953	0.0186541453
22	55.6381486676	2.9593057458	0.3680529301	0.0163528236	1.3126408318	0.0149916412
23	54.94628331	2.9000703006	0.3758034026	0.0217717576	1.3076521739	0.0186625224
24	55.553997195	2.6451009927	0.372778828	0.0193651338	1.3028506616	0.0157128535
25	54.6381486676	3.4543636887	0.3792060491	0.020349098	1.3051795841	0.0187315553
26	55.7294530154	3.5964441428	0.3693761815	0.0179015208	1.3067334594	0.01461419
27	57.6485273492	2.7829458801	0.3746691871	0.0153244217	1.3111285444	0.0135682162
28	54.703085554	3.4884902467	0.368241966	0.0211149434	1.3184102079	0.0148224169
29	55.9913043478	3.8860763196	0.3741020794	0.0165029153	1.3152173913	0.016315406
30	56.0402524544	3.8451519354	0.3814744802	0.0182221211	1.3168298677	0.016645237

Tabla B.2: Medidas para SETTINGS_INITIAL_WINDOW_SIZE

B.3. Max header list size

Max header list size	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
1	54.6004207574	3.5444293833	0.3604914934	0.0129716761	1.3099432892	0.0192881499
2	54.6704067321	3.6270426606	0.3678638941	0.0199991034	1.3099017013	0.0220011374
4	55.5454417952	3.3527505357	0.3809073724	0.0204242312	1.3024045369	0.0191991912
8	54.3709677419	3.7694352295	0.3775047259	0.0189812805	1.2948601134	0.0172999873
16	54.8225806452	3.7655904724	0.3741020794	0.0142690532	1.3128487713	0.0207017913
32	55.346002805	3.2137605393	0.3676748582	0.0204278891	1.3104234405	0.0168560223
64	54.712342216	3.0585461572	0.3846880907	0.0188548952	1.2964102079	0.0142555412
128	54.846002805	3.5306810391	0.3837429112	0.0181275669	1.28868431	0.0151671062
256	55.4465638149	3.3222960671	0.3652173913	0.0197074322	1.3323270321	0.0383683383
512	54.6819074334	3.5743095617	0.3776937618	0.0153609461	1.3179848771	0.0198772293
1024	56.1849929874	3.067879785	0.3725897921	0.0197263798	1.3004763705	0.0170976215
2048	54.2291725105	3.9030221424	0.372778828	0.0153560812	1.304436673	0.0151225587
4096	54.7241234222	4.1533877638	0.3718336484	0.0189004122	1.307241966	0.0172447672

Tabla B.3: Medidas para SETTINGS_MAX_HEADER_LIST_SIZE

B.4. Header table size

Header table size	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
1	55.4516129032	3.8228591032	0.3669187146	0.016254291	1.2991550095	0.0206056844
2	55.5639551192	2.7142488946	0.3650283554	0.0146258992	1.3068166352	0.0219507485
4	55.0748948107	2.5845709887	0.3725897921	0.0182323692	1.3034083176	0.0240824961
8	54.7106591865	4.0070929297	0.3769376181	0.0175111867	1.3131304348	0.0121175786
16	55.1570827489	3.0580483008	0.3737240076	0.017266263	1.3021228733	0.0237506439
32	53.029312763	3.7297419997	0.3790170132	0.0191166036	1.3069432892	0.0181132488
64	55.0726507714	3.1624970779	0.3776937618	0.0180842364	1.3155860113	0.0198189948
128	54.6192145863	3.2770333254	0.3826086957	0.0197509846	1.3127882798	0.0167614414
256	54.6434782609	3.6717419192	0.3663516068	0.0174020417	1.3117769376	0.014838266
512	53.9504908836	3.2861619463	0.3712665406	0.0156620045	1.3109754253	0.0192006474
1024	54.824684432	3.1406691373	0.3608695652	0.017782169	1.3021304348	0.0194117167
2048	55.2896551724	3.1522789474	0.3733459357	0.0207814488	1.3101096408	0.0177613249
4096	54.4260869565	3.9141358034	0.3744801512	0.0174063348	1.2998223062	0.0208388275

Tabla B.4: Medidas para SETTINGS_HEADER_TABLE_SIZE

B.5. Max frame size

Max frame size	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
16384	55.2663206124	2.9545218147	0.3769376181	0.0458404806	1.3037698883	0.0546528934
32768	56.0522149464	2.8314746317	0.3748582231	0.0460113409	1.3159840945	0.057749605
65536	54.9679630798	3.8741821327	0.3718336484	0.0464847436	1.3192982941	0.0535662198
131072	55.8965066245	3.1793865754	0.3822306238	0.0480717077	1.2918490102	0.0551951474
262144	55.7217574678	3.2320545799	0.3786389414	0.0486156951	1.3053832589	0.0791940352
524288	54.6613358883	3.5799891477	0.3640831758	0.0495870564	1.3037748374	0.0464042369
1048576	54.7840013743	2.9739469166	0.3752729223	0.0519087089	1.3032630439	0.0620990709
2097152	55.704802153	2.8542242887	0.3766565728	0.0478503877	1.2939749680	0.0699971721
4194304	55.7644311817	4.1658404932	0.3664800315	0.0451446178	1.2971196719	0.0414938775
8388608	54.5949007698	2.8922908985	0.3679385435	0.0463706624	1.2880683087	0.0648188123
16777215	54.4254962698	3.24097736	0.3777268095	0.046933252	1.2947500618	0.0471849662

Tabla B.5: Medidas para SETTINGS_MAX_FRAME_SIZE

Apéndice C

Tablas de resultados de nodo A8 como servidor

C.1. Max concurrent streams

Número de streams	Uso de CPU promedio	Desviación de CPU	Uso de memoria	Desviación de memoria	Potencia promedio	Desviación de potencia
1	79.2695726496	4.9354406064	3.6059288538	0.3875892759	2.4316854978	0.0437038472
2	78.4738461538	3.4292362170	3.4625823452	0.4107443727	2.4407323550	0.026232028
4	75.5029059829	4.2465211503	3.7812911726	0.2129477515	2.4555756794	0.0632522441
8	75.4924786325	5.8038193426	3.7714097497	0.2223989286	2.4481561544	0.0521699739
16	73.8664957265	3.1532794950	3.5201581028	0.405399547	2.4513670133	0.0537979099
32	78.444957265	5.9276850737	3.3779973650	0.4056968779	2.4663059783	0.0717368819
64	74.268034188	3.5427162324	3.7695652174	0.2184728531	2.4546785233	0.0556946543
128	76.1776068376	6.4121679892	3.3487483531	0.4243570105	2.4652101661	0.0695591078
256	75.6343533783	4.2495109377	3.5407114625	0.3976342169	2.4543607544	0.0519848000
512	75.3206837607	6.9282662743	3.7996047431	0.228386	2.4543614117	0.0543661664
1024	76.9478632479	4.6990804297	3.7613965744	0.2195349677	2.4668740550	0.0726954064
2048	74.532991453	6.2003629820	3.4262187088	0.4295108178	2.4541589567	0.0521750958

Tabla C.1: Medidas para SETTINGS_MAX_CONCURRENT_STREAMS

C.2. Initial window size

Log 2 initial window size	Uso de CPU promedio	Desviación de CPU	Uso de memoria	Desviación de memoria	Potencia promedio	Desviación de potencia
0	79.3485470085	5.7443760900	3.4707509881	0.4161770051	2.4538119750	0.0540849766
1	75.7232478632	3.7423354447	3.7806324111	0.2151901907	2.4657093817	0.0736259292
2	75.2976068376	6.0424427828	3.6508563900	0.3515350567	2.4536912294	0.0525388137
3	73.3394871795	3.8403721907	3.7885375494	0.2307571017	2.4538221511	0.0512072462
4	77.0090598291	5.6927041938	3.7762845850	0.2106455862	2.4634944083	0.0680084794
5	72.7694017094	6.2702515458	3.5728590250	0.3906002577	2.4543176806	0.0505841475
6	77.8615384615	3.4267242375	3.7864295125	0.2178439786	2.4658827317	0.0684250349
7	75.1364102564	5.3046187297	3.7387351779	0.2930438888	2.4543278728	0.0499710222
8	78.1560683761	7.0404067266	3.7934123847	0.2190079055	2.4650267217	0.0695045269
9	72.3054700855	4.6427565261	3.7819499341	0.2112002148	2.4553741850	0.0504996176
10	76.4304273504	4.5938081771	3.6827404480	0.3587623206	2.4559500339	0.0518371585
11	75.9311111111	5.9637459573	3.7206851120	0.3187860775	2.4676402572	0.0724734252
12	77.5685470085	3.2261508421	3.7894598155	0.2173152864	2.4546053406	0.0501078677
13	73.6712820513	6.5562046422	3.8179183136	0.2089721831	2.4610912528	0.0635061393
14	74.8588034188	4.0508673679	3.7754940711	0.2223638317	2.4559694028	0.0535564364
15	77.7102564103	5.9031643560	3.7686429513	0.2469371873	2.4557194456	0.0538572635
16	79.7811965812	5.5681404581	3.6357048748	0.3682116478	2.4681079433	0.0742353668
17	72.9644444444	3.3102976878	3.7699604743	0.2231211293	2.4556841756	0.0527374135
18	78.9646153846	5.4745393073	3.8217391304	0.2121821542	2.4674886572	0.0685588415
19	78.5982905983	3.1251823524	3.6996047431	0.3285565247	2.4553931983	0.0498838512
20	72.7509401709	5.6465385049	3.1616600791	0.2463160866	2.4560948911	0.0518670952
21	76.4288888889	3.9373855461	3.7803689065	0.2223991701	2.4676524939	0.0714968375
22	75.9165811966	7.0755907206	3.5669301713	0.3859567253	2.4556021278	0.0534734436
23	76.9924786325	3.7830652038	3.3357048748	0.4091893375	2.4678125622	0.0743576133
24	74.1124786325	4.9631932952	3.7603425560	0.2187154174	2.4554369378	0.0519691507
25	75.0018803419	3.6168269119	3.8075098814	0.2133213381	2.4559043011	0.0534682759
26	78.5061538462	5.3212969237	3.6154150198	0.3613163338	2.4675002022	0.0717162604
27	75.6948717949	4.3453942681	3.2587615283	0.3658220299	2.4561186489	0.0519938120
28	75.1418803419	6.2479054496	3.7646903821	0.2180743551	2.4678858261	0.0696958669
29	74.5776068376	3.7339820271	3.4762845850	0.4337843035	2.4551490233	0.0493506841
30	77.9488888889	7.2290981514	3.4271409750	0.4175229501	2.4592476483	0.0557004324

Tabla C.2: Medidas para SETTINGS_INITIAL_WINDOW_SIZE

C.3. Max header list size

Max header list size	Uso de CPU promedio	Desviación de CPU	Uso de memoria	Desviación de memoria	Potencia promedio	Desviación de potencia
1	76.1632478632	5.8000463043	3.7753623188	0.2123704616	2.4566609556	0.0518352174
2	73.0837606838	2.6142358584	3.5230566535	0.420460417	2.4561193206	0.0529175310
4	78.4712820513	5.9143708960	3.3669301713	0.4080663219	2.4680899978	0.0706114950
8	75.6420512821	3.6082238739	3.7665349144	0.2061939184	2.4555828006	0.0498182113
16	75.2063247863	5.7589033483	3.3632411067	0.402263637	2.4682531189	0.0693021806
32	73.9034188034	2.9273487795	3.5266139657	0.4048701593	2.4563092517	0.0515809071
64	78.5314529915	6.0225001197	3.8047430830	0.2114943781	2.4552965511	0.0501914509
128	74.5774358974	3.6671351941	3.7635946113	0.2104066507	2.4685227639	0.0710925603
256	76.0967521368	5.4962053962	3.4233267457	0.40736097	2.4552120950	0.0501652794
512	72.3083760684	3.2740141800	3.4615283267	0.4124425279	2.4677508461	0.0705102732
1024	76.1302564103	4.0704246488	3.7801054018	0.2128493745	2.4558707344	0.0505632015
2048	74.8017094017	5.6842845338	3.2463768116	0.3479562709	2.4556407856	0.0493988311
4096	73.7473504274	2.8671140025	3.6450592885	0.3735096696	2.4630464894	0.0661196419

Tabla C.3: Medidas para SETTINGS_MAX_HEADER_LIST_SIZE

C.4. Header table size

Header table size	Uso de CPU promedio	Desviación de CPU	Uso de memoria	Desviación de memoria	Potencia promedio	Desviación de potencia
1	76.1316239316	6.2756795021	3.7861660079	0.2203131218	2.4562689094	0.0545830798
2	72.5970940171	5.3982127346	3.7747035573	0.2126117368	2.4676840322	0.0733465149
4	77.6302564103	3.7781233648	3.3163372359	0.3877572174	2.4557679672	0.0522210161
8	74.6899145299	4.9184252235	3.5636363636	0.3886486241	2.4683779283	0.0735963188
16	79.2548717949	5.1020731996	3.7851119895	0.2269684694	2.4550394817	0.0492225522
32	75.6333333333	4.2495109377	3.7330698287	0.2778039289	2.4562098739	0.0541067442
64	75.3206837607	6.9282662743	3.7951251647	0.228340456	2.4791527383	0.0817043560
128	76.9478632479	4.6990804297	3.7810276680	0.2271500356	2.4476735344	0.0543655012
256	72.7717948718	3.6666056543	3.6693017128	0.3490906689	2.4501767967	0.0535847687
512	71.6292307692	4.3058470005	3.7293807642	0.2907969915	2.4632708378	0.0699288124
1024	74.5329914530	6.2003629820	3.7924901186	0.2246985818	2.4518461923	0.0529809390
2048	73.9548717949	3.2394115434	3.8137022398	0.2102667347	2.4636547683	0.0698483675
4096	77.7037606838	6.9280546373	3.7764163373	0.2327154121	2.4525913361	0.0525476475

Tabla C.4: Medidas para SETTINGS_HEADER_TABLE_SIZE

C.5. Frame size

Max frame size	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
16384	79.2948023982	3.9288474589	4.0302528135	0.2739454568	2.4529182817	0.0532135631
32768	78.3408859656	3.5314141789	4.2813318889	0.3490408219	2.4653393794	0.0706387451
65536	81.1205179031	4.0358323904	4.1095519097	0.2197095594	2.4539540733	0.0519592988
131072	79.2181217813	3.8067992519	4.3678465742	0.2081422622	2.4658935356	0.0721040081
262144	79.2128488426	5.0478132507	4.5583182947	0.318029965	2.4542718750	0.0510253270
524288	78.9834493783	6.9282662743	4.5232160405	0.21483302	2.4543169817	0.0555352602
1048576	81.302823614	7.706421994	4.064763836	0.2163609284	2.442949143	0.0622950078
2097152	81.4380149974	6.5374748555	4.2731139257	0.2665294116	2.4589017739	0.0513638248
4194304	68.9974339938	7.2197138335	4.4316137992	0.2712940823	2.4576561828	0.0671166793
8388608	68.7989493573	6.6970796912	4.5696102848	0.2542440359	2.4676247869	0.0562523902
16777215	71.2751239438	7.0224608018	4.073570037	0.2345266946	2.4537316288	0.0583991633

Tabla C.5: Medidas para SETTINGS_MAX_FRAME_SIZE

Apéndice D

Tablas de resultados de nodo A8 como cliente

D.1. Max concurrent streams

Número de streams	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
1	30.4088757396	3.2403883758	1.3668639053	0.2510605633	2.4387480113	0.0169677609
2	27.6644970414	2.3816866599	1.2710059172	0.1105015394	2.4362357353	0.0168893053
4	30.2656804734	3.7766653724	1.2461538462	0.1412469127	2.4370181626	0.0169933832
8	30.4414201183	4.1827608759	1.2887573964	0.0848258097	2.4389026371	0.0132292287
16	28.9721893491	3.515048711	1.2402366864	0.1206554194	2.4393873516	0.0120277274
32	30.001183432	1.7731474099	1.2485207101	0.1120186223	2.4397485406	0.0149287708
64	27.8721893491	3.0792195572	1.2514792899	0.1592135695	2.4384075841	0.0141746869
128	30.0982248521	3.2361465534	1.2846153846	0.1376758334	2.4399539433	0.0093250594
256	29.8881656805	3.194283626	1.2834319527	0.1698080525	2.4389834556	0.0129115682
512	29.9650887574	3.6529156918	1.2627218935	0.1073183137	2.4383337429	0.0122274048
1024	29.9928994083	3.7634970276	1.2887573964	0.1867561612	2.438435276	0.011457316
2048	29.8278106509	3.0517956987	1.2615384615	0.1926663322	2.4382552665	0.0100164046

Tabla D.1: Medidas para SETTINGS_MAX_CONCURRENT_STREAMS

D.2. Initial window size

Log initial window size	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
0	28.7136094675	3.2223105904	1.2710059172	0.1447010741	2.4406198053	0.0135254333
1	29.2662721893	3.3045774518	1.2556213018	0.1437094457	2.4374566749	0.0111296502
2	28.826035503	4.1808447596	1.2443786982	0.116285112	2.4392753951	0.0103157236
3	28.8278106509	1.5812901957	1.2781065089	0.0929913865	2.4385437599	0.0114442934
4	29.7301775148	2.8342308502	1.2230769231	0.0842064711	2.4387284008	0.013494016
5	29.6088757396	2.7208883102	1.2366863905	0.1211448448	2.4386383592	0.0117632093
6	30.0195266272	3.8846758251	1.2396449704	0.1153780406	2.4387295558	0.0100925797
7	28.6875739645	2.8608913979	1.2538461538	0.1019533095	2.4400381966	0.012805199
8	28.3289940828	2.7370065168	1.2568047337	0.1202618142	2.4371508696	0.0124983066
9	28.8396449704	3.2291080072	1.249704142	0.0788611858	2.4384225728	0.0100066274
10	29.1236686391	2.5946072536	1.2639053254	0.078029518	2.4399147127	0.0107593001
11	29.6656804734	2.360374338	1.2881656805	0.1026097039	2.4372501021	0.011641715
12	29.675739645	3.8165693546	1.2680473373	0.0791252816	2.4389534499	0.0129673767
13	29.0964497041	2.957037231	1.2828402367	0.0918876043	2.4385368072	0.0093648908
14	27.9952662722	1.4838165064	1.2692307692	0.1007369296	2.4394392401	0.0113388455
15	28.7544378698	4.071623868	1.2514792899	0.1247239403	2.4381317769	0.0109284172
16	30.3319526627	4.0591653009	1.2597633136	0.1043980513	2.4387030038	0.0108414671
17	31.449704142	3.7187986854	1.2704142012	0.1329696527	2.4389915142	0.0115978821
18	28.1834319527	3.2482461451	1.2668639053	0.1384396213	2.4385345142	0.0131635653
19	29.4189349112	3.1546475145	1.2520710059	0.1254244803	2.4402920775	0.0135756539
20	29.1591715976	3.1069232747	1.2390532544	0.1230954128	2.4388368885	0.0108242198
21	29.7165680473	3.7301517413	1.3023668639	0.1017298418	2.4393700265	0.0143389894
22	30.2201183432	4.0893009227	1.2834319527	0.0912787824	2.4389280227	0.0121721885
23	29.6538461538	1.9814823212	1.2437869822	0.1135054072	2.4387249754	0.0091222819
24	29.3668639053	2.6424716939	1.2852071006	0.0928117402	2.4384883875	0.0096157444
25	27.5372781065	2.8202771105	1.2786982249	0.0610682231	2.437625189	0.0102375856
26	27.8715976331	3.8006855821	1.249112426	0.0910999225	2.4390907637	0.0078188857
27	29.6846153846	3.186250045	1.275147929	0.13471256	2.4386672628	0.0061482455
28	28.6372781065	3.8801597272	1.2142011834	0.0859364332	2.4380290718	0.0073222735
29	28.7550295858	2.4865294805	1.2426035503	0.0891343082	2.4388103195	0.0076402807
30	29.4538461538	3.4711064348	1.2360946746	0.0848705137	2.4390642193	0.0060469061

Tabla D.2: Medidas para SETTINGS_INITIAL_WINDOW_SIZE

D.3. Max header list size

Max header list size	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
1	29.8556213018	3.5515687478	1.1928994083	0.0746101993	2.437295104	0.0057223286
2	29.0715976331	3.7430980331	1.2656804734	0.0848839203	2.4383302533	0.0053352907
4	28.6686390533	3.1611476793	1.2485207101	0.0889554004	2.4398547259	0.007023373
8	29.7183431953	2.7303512473	1.2881656805	0.0799170789	2.3441734045	0.168640811
16	28.5633136095	4.2700116563	1.2781065089	0.0994484868	2.322516087	0.1371367624
32	29.4159763314	2.5809888736	1.3053254438	0.0897830288	2.3898341493	0.0206209851
64	29.7408284024	2.9030766252	1.1295857988	0.2081629557	2.3811121191	0.0291951459
128	27.8437869822	3.2856771607	1.2550295858	0.1079139731	2.3865567561	0.0245631389
256	27.9343195266	4.2160438556	1.3029585799	0.1458004656	2.3822130718	0.0276618371
512	29.5532544379	3.7051412406	1.3183431953	0.2081556669	2.385271155	0.0209220592
1024	30.3532544379	2.2706481048	1.2810650888	0.1973346906	2.3855366181	0.0248499632
2048	31.0443786982	3.1078543907	1.225443787	0.1120186223	2.3862693932	0.0232194479
4096	29.0568047337	2.4844040921	1.326035503	0.2271411048	2.3795761418	0.0196778924

Tabla D.3: Medidas para SETTINGS_MAX_HEADER_LIST_SIZE

D.4. Header table size

Header table size	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
1	28.050887574	3.1675710405	1.3130177515	0.1335815471	2.3840629319	0.0247445252
2	29.5710059172	1.9530766551	1.2846153846	0.1467934722	2.3795888507	0.0249674735
4	29.3568047337	3.2529042379	1.3686390533	0.2019740238	2.3789021928	0.0269627564
8	28.3704142012	3.0492309086	1.3023668639	0.1679844731	2.3862647788	0.0229453596
16	29.9106508876	3.5641521927	1.398816568	0.2196205746	2.3722089338	0.0243129317
32	29.3917159763	3.7809935003	1.2597633136	0.1534338657	2.3825108015	0.0278557473
64	30.2420118343	3.1606440425	1.274556213	0.1496341566	2.3828247089	0.0285582585
128	29.0899408284	3.35942459	1.2390532544	0.0924390912	2.3807867221	0.0257458824
256	29.4786982249	3.5173256071	1.2875739645	0.1515058686	2.3852700284	0.0269153134
512	27.3307692308	2.7873817538	1.2390532544	0.0783496902	2.3892882949	0.0247377906
1024	29.0792899408	3.0558725187	1.274556213	0.1074807728	2.3802027694	0.0234622066
2048	28.8443786982	3.5678920371	1.2467455621	0.0909332253	2.3796696295	0.0231050175
4096	27.7301775148	3.5200320813	1.2284023669	0.0682541237	2.3790926125	0.0251066903

Tabla D.4: Medidas para SETTINGS_HEADER_TABLE_SIZE

D.5. Max frame size

Max frame size	Uso de CPU promedio	Desviación de CPU	Uso de memoria promedio	Desviación de memoria	Potencia promedio	Desviación de potencia
16384	28.6910949848	2.7699326613	1.2300858754	0.1546827307	2.4185533198	0.0191342614
32768	28.823914871	3.1447090237	1.2667239411	0.1283741727	2.3804104345	0.0219284939
65536	29.997813863	2.696783076	1.2684933262	0.1474434033	2.3870341364	0.0200870874
131072	29.2264905587	3.2777631427	1.2549574022	0.1592782882	2.3973122006	0.0186655529
262144	28.847128729	3.2963098797	1.2582015052	0.144050298	2.41197515	0.018802618
524288	30.0567617709	3.1812132048	1.2220447102	0.1334211731	2.3898714557	0.0218334389
1048576	30.1419088459	3.2987888753	1.2975557508	0.1572388603	2.4025955002	0.0220259136
2097152	28.7152872873	3.1496630832	1.2471260386	0.0798079657	2.3869357249	0.021743066
4194304	30.0622171473	2.6166938157	1.2530960219	0.1351293583	2.3879801527	0.0207819507
8388608	30.1639975157	2.5351325927	1.2445311582	0.1459824634	2.3945577213	0.0231199843
16777215	29.4367609944	2.9496527218	1.2891235012	0.0978340169	2.4037616879	0.0197286955

Tabla D.5: Medidas para SETTINGS_MAX_FRAME_SIZE