Redes Digital Signage como sustrato para brindar conectividad a dispositivos IoT

Digital Signage Networks as Substrate for Connectivity to IoT Devices

Jorge D. de Hoz[1], Jose Saldana[1], Rebeca Guerrero-Rodríguez[2]

[1]I3A, Universidad de Zaragoza, Ada Byron Building, 50018, Zaragoza, Spain
[2]Universidad tecnológica de Durango, Carretera Durango – Mezquital, 34080 Dgo, México

e-mail: {dhoz, jsaldana}@unizar.es, rebeca.guerrero@utd.edu.mx

Resumen. El número de dispositivos conectados a Internet supera actualmente a la población mundial por más de tres veces y previsiblemente, esta cifra se duplicará en cinco años. El Internet de las Cosas es un concepto que describe esta tendencia y perfila ciertos aspectos de diseño y funcionalidad que los nuevos dispositivos deben incorporar para lograr una integración exitosa en Internet. En este sentido, las redes Digital Signage empleadas tradicionalmente para difundir comunicación audiovisual, cumplen muchas de las características recogidas en el paradigma del Internet de las Cosas. En este trabajo se plantea el poder emplear la red Digital Signage propuesta como sustrato para conectar otros tipos de dispositivos que puedan beneficiarse de las ventajas de estas redes, especialmente en escenarios de movilidad. Tras analizar los problemas de latencia inherentes a este esquema de comunicación y presentar soluciones factibles, se concluye que esta propuesta sería funcional en dispositivos móviles.

Palabras clave: Digital Signage, Internet de las Cosas, port forwarding, redes móviles.

Abstract. The number of Internet-connected devices exceeds the world's population by more than three times and this figure is expected to be doubled within the next five years. The Internet of Things is a concept that describes this trend and outlines certain aspects of design and functionality that new devices should incorporate for a successful integration into the Internet. In this respect, Digital Signage networks traditionally used for audiovisual media, accomplish many of the characteristics of the Internet of Things devices. This paper raises the power to employ a proposed Digital Signage network as a substrate to connect other types of devices that can benefit from the advantages of this kind of networks, particularly in mobility scenarios. After analyzing the latency issues arising from this communication scheme and presenting feasible solutions, it is concluded that this proposal would be functional on mobile devices.

Keyword: Digital Signage, Internet of Things, port forwarding, network mobility.

1 Introducción

El Internet de las Cosas (IdC) está cambiando el concepto tradicional de la red. En este nuevo contexto se plantea la conectividad universal como la interconexión de redes con diferentes dispositivos y servicios [7] [3]. Este concepto de ubicuidad cobra especial relevancia en la actualidad, ya que existen 6.6 dispositivos conectados en la red por cada individuo y se plantea que este ratio se duplique en cinco años [4].

La Digital Signage (DS), por su parte, es una tecnología de comunicación audiovisual para difusión selectiva, principalmente a través de pantallas. Esta tecnología, también se puede emplear para interconectar otros tipos de dispositivos de una manera transparente, proporcionando escalabilidad y movilidad tal y como se contempla en el paradigma del IdC. El elemento DS actuaría entonces como soporte para la conexión de otros dispositivos a la red, compartiendo su enlace backhaul con ellos. Para ello, este artículo se pretenden analizar las limitaciones del esquema de comunicación usado por DS en entornos de comunicaciones restrictivos y con movilidad, comunes en dispositivos IdC.

2 Trabajos relacionados

En [11], se propone una arquitectura descentralizada para una red DS que integra sistemas de identificación por radiofrecuencia (RFID). En el diseño se implementa seguridad en las comunicaciones y su arquitectura permite un despliegue de red flexible, la cual se basa en la descentralización de los servicios, aplicaciones y funciones de red. Este esquema permite la integración de diversos elementos relacionados con la tecnología RFID y también permite el envío de mensajes visuales a los usuarios a través de dispositivos DS. En [11], se remarca la importancia de la descentralización de los servicios, aplicaciones y control, ya que así se favorece la escalabilidad y la robustez general del sistema. Estas ideas se ponen de manifiesto en propuestas recientes de redes descentralizadas postulándose como plataformas adecuadas para diversos servicios y aplicaciones de IdC [2]. En este contexto, la contribución de este artículo se centra en dos puntos:

(1) Presentar una arquitectura de red DS basada en la tecnología abierta y protocolos maduros: Transmission Control Protocol (TCP) sobre Internet Protocol (IP), protocolo de transferencia de OpenSSH e HyperText Transfer Protocol (HTTP).

(2) Realizar un estudio del rendimiento de túneles bidireccionales seguros basados en port-forwarding de OpenSSH compartiendo el acceso a Internet. Ambas cuestiones se han aplicado y estudiado en escenarios reales.

3 Arquitectura propuesta para Digital Signage

La red DS ofrece servicios de intercomunicación básicos a los players, todos ellos incorporando encriptación y una gestión del establecimiento de comunicación y monitorización mediante tres canales bidireccionales de distinta prioridad para transmitir mensajes de señalización de la red, realizar gestión remota de cada

dispositivo en tiempo real y distribuir contenidos audiovisuales para reproducción en diferido respectivamente.

La red DS presentada puede proporcionar comunicación tunelizada con otros dispositivos. Las comunicaciones entre los players a través de la red DS también siguen un esquema de un túnel bidireccional. Este enfoque comparte algunas similitudes con algunos esquemas planteados en Movilidad IP [10] [17], lo que permite el acceso en tiempo real a estos dispositivos desde cualquier terminal conectada a Internet.

La seguridad en la red DS se aborda introduciendo encriptación en las comunicaciones tunelizadas vía Secure Socket Layer (SSL). Este tipo de seguridad ayuda a proteger la integridad de los datos y a certificar el origen del mismo, evitando posibles ataques de phishing en la actualización de contenidos [5]. Por otro lado, este esquema de seguridad permite el establecimiento de túneles independientes para tráfico entre un proceso local y un servicio remoto empleando la funcionalidad port-forwarding de Secure SHell (SSH). De este modo, se consiguen asegurar las comunicaciones sin requerir modificaciones importantes de software ni de servicios existentes.

4 Análisis y resultados

La arquitectura propuesta considera una serie de túneles dinámicos reversos entre las aplicaciones del dispositivo DS y el núcleo de red. En este escenario, SSH funciona como proxy extensible: una parte del proxy es local y la otra parte está en una máquina remota. Ambas partes se comunican entre sí a través de un canal port-forwarder TCP-IP [18] generado en la sesión SSH establecida. Sin embargo, la latencia en las comunicaciones tunelizadas puede degradarse ya que las implementaciones recientes de OpenSSH incluyen un buffer de salida con un tamaño fijo de 2 Mbytes [14]. Como resultado, cuando existe una aplicación que requiere un uso intensivo del ancho de banda, los valores globales de latencia aumentan en el resto de comunicaciones que comparten el canal o sesión ya que el tamaño fijo del búfer común de salida penaliza el rendimiento general.

Estos problemas son conocidos en la literatura [15], y se han propuesto implementaciones optimizadas de OpenSSH, pero destinadas principalmente a mejorar el rendimiento del throughput, no de la latencia. Para superar este problema, se propone no multiplexar las conexiones tunelizadas a través de la misma sesión de SSH. Esto permite que cada sesión de SSH tenga su propio buffer de salida, evitando así que los valores altos de latencia se propaguen de un flujo a otro. También hace posible a OpenSSH aplicar DiffServ mediante el campo Type of Service (TOS) de los paquetes de cada sesión, lo que permite emplear el sistema de colas por defecto en Linux (pfifo_fast) para coordinar el envío de datagramas en función de su prioridad [9]. Este enfoque no requiere modificaciones del kernel en la mayoría de los dispositivos y facilita su implementación en sistemas embebidos Android.

4.1 Comportamiento de los principales algoritmos de control de congestión TCP

Para estudiar el impacto del buffering en conexiones port-forwarding utilizando OpenSSH, se plantea la realización de una serie de medidas en las comunicaciones entre los players y los servidores DS. El escenario de medidas propuesto es el siguiente: cuatro players se encuentran conectados a Internet con un módem Huawei E173 3.5G+ cada uno. Este dispositivo permite High Speed Downlink Packet Access (HSDPA) de 7,2 Mbps para el canal de enlace descendente y para el ascendente un High Speed Uplink Packet Access (HSUPA) de 2.1 Mbps. Todos los players emplean OpenSSH 6.6 en un sistema operativo Linux basado en kernel 3.10.48. Cada player utiliza un algoritmo de control de congestión TCP diferente para probar el canal de subida. El nivel de la señal 3G es -75 dBm en todos los dispositivos y todas las mediciones se tomaron con el vehículo en reposo.

Cada serie de pruebas consiste en un conjunto de 15 transmisiones que se efectúan en diferentes horas del día. En cada prueba, se establecen dos conexiones port-forwarding por cada player a un servidor de pruebas que forma parte de la red DS, pero que durante los experimentos se reservó para este fin. En la primera conexión, la herramienta Iperf [6] se utiliza para generar tráfico durante 120s y para medir las estadísticas, y NetEm [12] se emplea para modelar las pérdidas de paquetes en ráfagas en la interfaz de red Protocolo Punto a Punto (PPP) de cada player. Los escenarios de pérdidas contemplados en la simulación son de 2% y 5% con el fin de estudiar los peores casos en High Speed Packet Access (HSPA), que según [8] pueden llegar experimentar los dispositivos mientras se encuentren en movimiento. En la segunda conexión tunelizada de cada player, se utiliza una secuencia de comandos Python para muestrear el Round Trip delay Time (RTT) utilizando el puerto echo del servidor remoto a través del segundo canal creado mediante port-forwarding para las conexiones interactivas. Las variantes TCP analizadas fueron Reno, Bic, Cubic y Westwood.

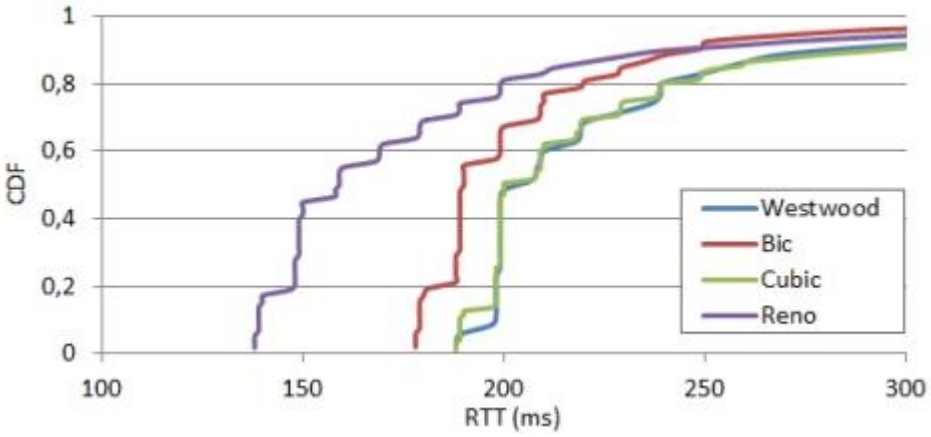

Fig. 1. Comparación de la latencia con un 2% de pérdida de paquetes.

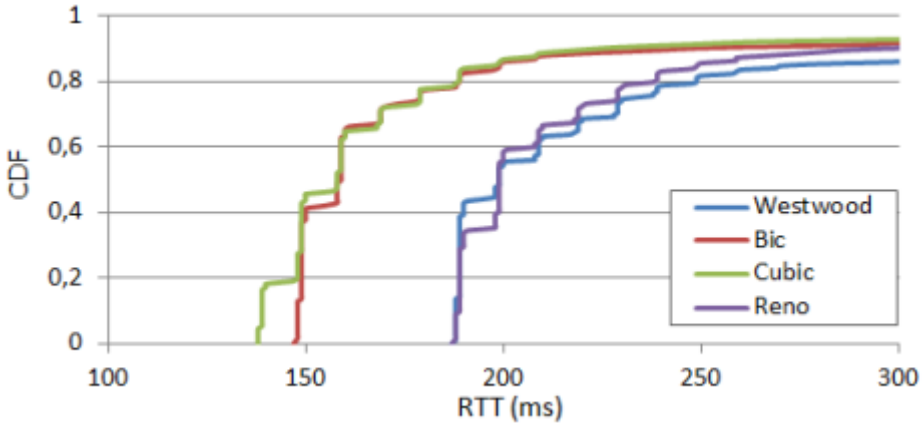

Fig. 2. Comparación de la latencia con un 5% de pérdida de paquetes

Tabla 1. Resultados de las comunicaciones IGMP y port-forwarded.

	Westwood	Bic	Cubic	Reno
2% de pérdidas	397 Kbps	340 Kbps	422 Kbps	355 Kbps
5% de pérdidas	193 Kbps	222 Kbps	240 Kbps	210 Kbps

La capacidad de ancho de banda se resume en la Tabla 1, donde Cubic obtiene la mejor marca. Este resultado es similar al obtenido en experimentos efectuados en conexiones no tunelizadas [13].

Los resultados presentados en la Fig. 1 y la Fig. 2 muestran la función de distribución acumulada (CDF) de los valores de RTT. En todas las pruebas realizadas, el 85% paquetes de echo se encuentran por debajo de 350 ms. de latencia mientras Iperf se encontraba transmitiendo. En estas circunstancias, si una aplicación interactiva con requisitos reducidos de ancho de banda necesita transmitir, su RTT no aumentará debido a los flujos de datos en paralelo, lo que permite que la aplicación interactiva funcione correctamente.

4.2 Degradación de la latencia a nivel de aplicación

Es necesario analizar el impacto de la tunelización de las comunicaciones a nivel de aplicación. Para ello, se plantea un escenario de pruebas que incluye movilidad: Se instala un player en un vehículo y se conecta a Internet utilizando el mismo tipo de módem que fue empleado anteriormente. El vehículo sigue un trayecto de 20 min típico de bus urbano por la ciudad de Durango, México. Se efectúa cada segundo un ping hacia el servidor DS sin tunelizar vía Internet Group Management Protocol (IGMP). También se mide el RTT a través de mensajes echo enviados por la

secuencia de comandos Python a través de la conexión tunelizada de alta prioridad. Todas estas medidas se realizan en presencia del tráfico de fondo generado a través de Iperf para medir la evolución del ancho de banda disponible en el tiempo.

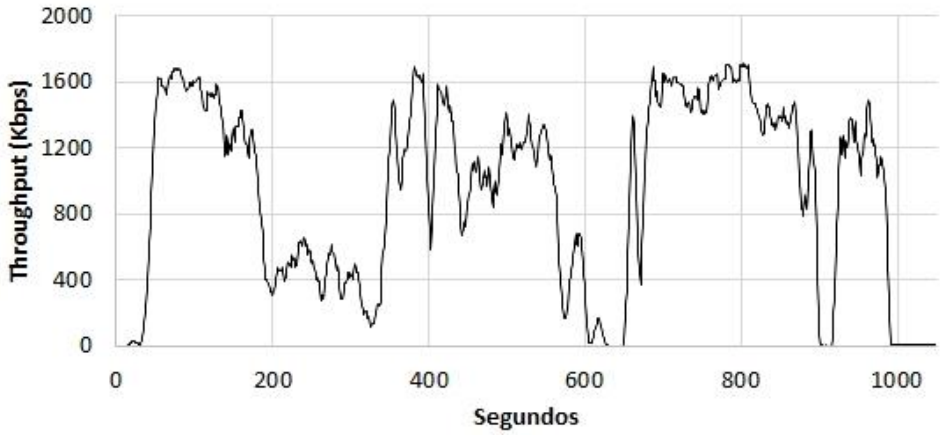

Fig. 3. Acho de banda de subida medido en la interfaz PPP del player en el vehículo en movimiento

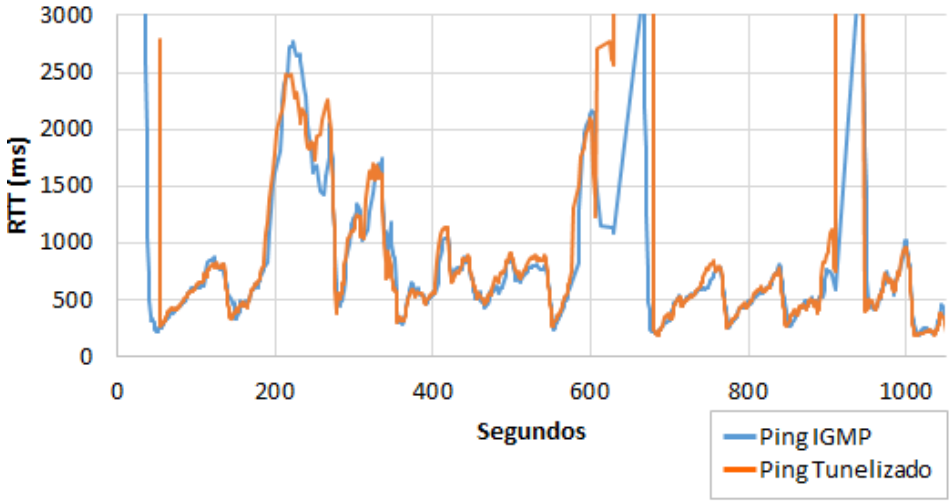

Fig. 4. Comparación entre la latencia existente en el túnel frente a la indicada por pings IGMP cuando Iperf está transmitiendo.

Según los resultados mostrados en la Fig. 3, Fig. 4 y su resumen en la Tabla 2, la degradación de las comunicaciones tunelizadas es menor de un 1,7 % en términos de RTT, por lo que la conectividad apenas se resiente.

Tabla 2. Resultados de las comunicaciones IGMP y port-forwarded

	IGMP	Port Forwarded echo
Conectividad	90%	89%
Promedio del RTT	622 ms	632 ms

4.3 Validación del esquema de comunicación DS

Las Comunicaciones no parecen verse afectadas cuando se utilizan conexiones tunelizadas a través del port-forwarding ofrecido por OpenSSH. La latencia media en presencia de tráfico de fondo compartiendo el enlace con conexión PPP es alta a pesar de emplear DiffServ debido a la incidencia de la movilidad y las fluctuaciones de cobertura 3G asociadas. Sin embargo, este enfoque puede todavía ser válido para las comunicaciones entre los dispositivos de la IdC. Por ejemplo, este esquema de tunelización puede emplearse por protocolos conformes a las condicionantes impuestas por la arquitectura de software REpresentational State Transfer (REST) [1], como el Constrained Aplication Protocol (CoAP) o HTTP, que no requiere bajos valores de RTT para funcionar. El uso de protocolos RESTful, particularmente HTTP, permite también simplificar la interoperabilidad con sistemas de información externos como ThingSpeak [16] a través de métodos HTTP GET, POST, PUT y DELETE.

5 Conclusiones

En este trabajo se ha presentado una red DS capaz de distribuir, obtener la información de los dispositivos y realizar control en tiempo real. Después de analizar las causas de las limitaciones del esquema de tunelización basado en el port-forwarding de OpenSSH se concluye que este planteamiento carece de buffers de entrada dinámicos, generando efectos negativos si comparte una sesión SSH con flujos de distintas aplicaciones con consumos elevados de ancho de banda. La solución a este problema es factible sin modificar el kernel del sistema operativo ni el código fuente de OpenSSH mediante el uso de diferentes sesiones SSH para cada flujo y la utilización de Diffserv.

Los resultados de las pruebas realizadas concluyen que este esquema funciona en ambientes con pérdidas de paquetes y con movilidad sin degradar la latencia. Como trabajo futuro se propone conseguir y verificar que todas las características de seguridad proporcionadas por OpenSSH como protocolo proceso-a-proceso tengan una degradación del rendimiento apenas perceptible a nivel de aplicación. Por otro lado, constatar que aunque este enfoque no se pueda implementar en un sentido amplio, este esquema de comunicación tiene una aplicación inmediata en muchos dispositivos existentes en el mercado, lo que permite el rápido desarrollo de nuevos dispositivos y servicios del IdC.

Agradecimientos

Este trabajo ha sido parcialmente financiado por CONACYT (PEI 682/2014); Servicios de TI de Durango S.A. de C.V.; Ateire S.A.C., y ER H2020 Wi 5 project (Grant Agreement no: 644262).

Referencias

1. Aijaz, A., Hamid Aghvami, A. H.: "Cognitive Machine-to-Machine Communications," IEEE Internet of Things Journal, vol. 2, nº 2, pp. 103-112 (2015).
2. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog Computing and Its Role in the Internet of Things. Cisco Systems Inc. San Jose, California (2012).
3. CISCO: "The Internet of Things Reference Model," (2014) http://cdn.iotwf.com/resources/71/ IoT_ Reference _Model _White _Paper_June_4_2014.pdf, 28/10/2015.
4. CISCO: The Zettabyte Era: Trends and Analysis. Visual Networking Index. (2015) http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.pdf, 28/09/2015.
5. Cooper, E. A. et al: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, IETF (2008).
6. Dugan, J. et al: Iperf 2.0.5 (2010) https://iperf.fr/.
7. Edson, B.: Creating the Internet of Your Things. Microsoft Corporation (2014). http://download.microsoft.com/download/E/1/F/E1FFDADF-C0FF-4E72-A834-B173A079F393/Microsoft_Internet_of_Things_White_Paper.pdf, 28/09/2015.
8. Esquerra-Soto, J.A., Pérez-Díaz, J.A., Amezcua-Valdovinos, I., and García-Hernández, C.F.: Performance Analysis of 3G+ Cellular Technologies with Mobile Clients. Instituto Tecnológico de Monterrey (2012).
9. Graf, T., et al,: Simple, classless Queueing Disciplines. Linux advanced routing and traffic control. http://lartc.org/howto/lartc.qdisc.classless.html.
10. Jaehoon, J., Jungsoo, P. Hyoungjun, K.: Dynamic Tunnel Management Protocol. IEEE Xplore, vol. 7, pp. 4754 - 4757, (2004).
11. Kotak, D. B., Gruver W. A.: Distributed Intelligent RFID Systems. IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, Texas (2009).
12. Linux Foundation: NetEm: Network Emulation (2009) http://www.linuxfoundation.org/collaborate/workgroups/networking/netem 28/11/2015.
13. Mascolo, S., De Cicco, L.: TCP Congestion Control over HSDPA: an Experimental Evaluation. 22nd Mediterranean Conference of Control and Automation (MED) (2014).
14. OpenSSH 6.9 Source Code (channels.h), 2015.
15. Rapier, C., Stevens, M., Bennett, B., Tasota, M.: High Performance SSH/SCP -HPN-SSH. Pittsburgh Supercomputing Center, (2012) https://www.psc.edu/index.php/hpn-ssh 28/10/2015.
16. ThingSpeak Community "ThingHTTP",: http://community.thingspeak.com/documentation/apps/thinghttp/ 28/11/2015.
17. Xiaoming, W.: A framework of enhanced local mobility routing. IEEE Xplore, vol. 3, pp. 2030 - 2034, (2003).
18. Ylonen, T., Lonvick, C.: The Secure Shell (SSH) Connection Protocol. RFC 4254, IETF (2006).