RED DE IoT: ESTIMACIÓN DE CONSUMO DE ENERGÍA

Matías Siracusa*, Carlos Taffernaberry*, Gustavo Mercado*,Diego Dujovne§, Sebastian Tobar*, Ana Lattuca*

*GridTICs – Grupo en Tecnologías de la Información y las Comunicaciones

Departamento de Electrónica / Facultad Regional Mendoza / UTN- Argentina

{matias.siracusa, carlos.taffernaberry,sebastian.tobar, ana.lattuca}@gridtics.frm.utn.edu.ar

§Escuela de Informática y Telecomunicaciones, Universidad Diego Portales, Chile

diego.dujovne@mail.udp.cl

RESUMEN

Este trabajo tiene como objetivo proponer, desarrollar y evaluar alternativas de solución Multicast, para redes de IoT, basadas en algoritmos y procesos que permitan una mejora en el uso de los recursos y su escalabilidad. La aplicación concreta de estas técnicas se puede aplicar en nodos sensores que deban enviar información a múltiples nodos destino, o cuando un nodo intenta difundir una orden (una consulta, una actualización de firmware, un comando, etc.) a un grupo de nodos de sensores. Se debe tener presente que la transmisión de datos es el factor que mayor consumo de energía produce en un nodo.

Palabras Clave: IoT, Multicast, consumo energía, 6LoWPAN, CoAP

CONTEXTO

El presente trabajo se desarrolla en el ámbito de un proyecto de investigación acreditado por la Universidad Tecnológica Nacional con código CCUTIME0005150TC denominado "Multi-IoT: Mejora del desempeño en redes 6LoWPAN utilizando técnicas de Multicast". El proyecto es llevado adelante por investigadores y becarios de la Facultad Regional Mendoza, con el apoyo de Investigadores formados de la Universidad Diego Portales.

Este trabajo es la evolución de distintos proyectos anteriores llevados a cabo por el grupo Gridtics[1] [2], mejorando y cambiando el hardware utilizado, como así también protocolos de comunicaciones y software, siempre buscando mejoras en eficiencia y autonomía del sistema.

1 INTRODUCCIÓN

6LoWPAN [3] es una pila (stack) de protocolos definido por la publicación 4944 (Request For Comments 4944) del Grupo de Trabajo de Ingeniería de Internet (IETF) que posibilita a redes inalámbricas de área personal (Wireless Personal Area Networks - WPAN) el acceso a Internet. Un tipo particular de WPAN son las redes de nodos sensores inalámbricos (WSN), por lo que, usando este protocolo, cada nodo puede interactuar, con cualquier otro dispositivo conectado a Internet, constituyendo actualmente el standard más relevante en la Internet de las Cosas (IoT). Uno de los campos de mayor interés reciente en esta área es mejorar la eficiencia en el uso de recursos limitados que tienen los nodos de estas redes (ancho de banda, energía, almacenamiento, capacidad de procesamiento) y la escalabilidad para el encaminamiento (routing) de los datos. En forma genérica, el uso de técnicas de encaminamiento Multicast

permiten reducir considerablemente el tráfico de una red cuando los contenidos son compartidos por un número importante de nodos. Sin embargo, en la actualidad, los algoritmos asociados a esta alternativa sólo responden a adaptaciones incompletas de sus variantes en redes jerárquicas y redes móviles ad hoc (MANET), sin considerar una solución integral a partir de los requerimientos y restricciones de redes WPAN.

2 LÍNEAS DE INVESTIGACIÓN Y DESARROLLO

Uno de los objetivos de este proyecto es "Desarrollar una metodología de medición de consumo energético de una red 6LoWPAN para evaluar bajo la misma base las distintas propuestas de algoritmos de ruteo y sus mejoras."

Para la implementación de esta metodología se decidió trabajar con TSCH, Contiki-NG y hardware de OpenMote, a continuación se describe su utilización.

- 2.1 TIME-SLOTTED CHANNEL HOPPING (TSCH)

Uno de los desafíos más grandes de la comunicación IoT reside en obtener alta confiabilidad con mínimo consumo energético. Se han propuesto numerosos protocolos de comunicación que intentan resolver este dilema, entre ellos el protocolo TSCH Time-Slotted Channel Hopping [4] el cual fue propuesto como un mecanismo de capa MAC en el protocolo 802.15.4e. TSCH utiliza salto de frecuencias para mejorar la confiabilidad, reduciendo así el efecto de la interferencia externa y el desvanecimiento de múltiples trayectos. Para disminuir el consumo se utiliza una planificación que indica a un nodo exáctamente cuándo debe transmitir o recibir, evitando así desperdiciar energía en períodos de competencia por la captura del canal. Además pueden los nodos desactivar su hardware de comunicación cuando no se encuentre en uso.

En redes TSCH, cada nodo utiliza una planificación común. Esta planificación se divide en ranuras de tiempo (*timeslots*), los cuales tienen una duración entre 10 a 15ms. El conjunto de todos las ranuras de tiempo en una ronda de transmisión se denomina *slotframe* y su longitud depende de la cantidad de nodos que formen parte de la red. TSCH contempla además un salto en frecuencia, con lo que a la coordenada de tiempo se le agrega una coordenada de frecuencia, resultando en una matriz bidimensional como la que se observa en la figura 1.

Una ranura de tiempo dentro de un timeslot puede ser de cuatro tipos: RX, TX, shared y off. RX y TX indican que un nodo deba recibir o transmitir respectivamente, mientras que shared permite el acceso de cualquier nodo mediante un mecanismo de competencia. Estos tipos de ranuras de tiempo son las que permiten la sincronización de la planificación entre nodos, así como la unión de nuevos nodos a la red. Conociendo la planificación, un nodo puede fácilmente determinar en qué intervalo de tiempo debe mantener su radio encendida y cuando puede apagarse para ahorrar energía.

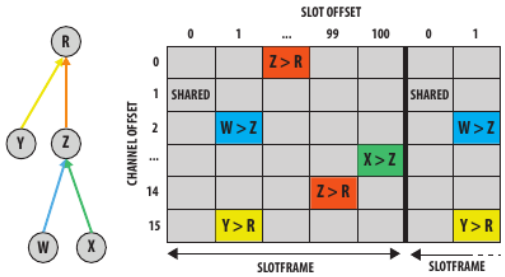

Figura 1: Ejemplo de planificación TSCH

Es importante destacar que el protocolo TSCH no define un mecanismo para iniciar y mantener la planificación, sino que los detalles de la implementación quedan a criterio del usuario.

Para habilitar TSCH en Contiki-NG, es necesario habilitarlo en el Makefile del proyecto

```
MAKE_MAC = MAKE_MAC_TSCH
```

Adicionalmente se debe realizar la configuración necesaria para que el hardware de los nodos soporte TSCH. Con lo anterior

queda implementada una función para el inicio y mantenimiento de la planificación.

- 2.2 CONTIKI-NG

El sistema operativo utilizado es Contiki-NG en la versión 4.4 [5]. Este sistema operativo se enfoca en comunicación de baja potencia en dispositivos de bajos recursos computacionales. Incluye soporte por defecto para protocoles estándar IETF, entre otros: IPv6/6LowPAN, 6TiSCH, RPL y CoAP.

Brevemente se exploró la posibilidad de utilizar OpenWSN[6], otro sistema operativo con énfasis en dispositivos de bajos recursos, sin embargo se encontró documentación confusa y a menudo obsoleta. En contraste, la documentación de Contiki-NG se mantiene actualizada y posee una comunidad dispuesta a evacuar dudas en la plataforma gitter.

Para la estimación del consumo, Contiki-NG utiliza un módulo de software llamado Energest. Provee una estimación basada en software aproximada para dispositivos motes de IoT. Al rastrear los diversos componentes de hardware, como la radio, cpu, etc. y conocer el consumo de energía cada componente, es posible estimar el consumo de energía.

El módulo Energest realiza un seguimiento de cuándo varios componentes se encienden y apagan. Al saber cuánto tiempo han estado los componentes en diferentes estados y el consumo de energía de cada uno de los componentes, es posible estimar el consumo de energía.

- 2.3 OPENMOTE

En el escenario presentado en este trabajo, el dispositivo utilizado es el *Openmote CC2538[7]*, un dispositivo modular que utiliza *open hardware*. Éste consta de CC2538 SoC (System on Chip), con un procesador ARM Cortex-M3 de 32 MHz y con 32 Kb de RAM y 256/512 Kb de memoria FLASH. Cuenta además con un trasceptor 802.15.4 de 2,4GHz y soporta múltiples periféricos como USB, NVCI, I2C, SPI, UUART entre otros. Presenta tres sensores digitales: temperatura/humedad, luminosidad y acelerómetro. El rango de transmisión es de 50 metros en interiores y de 200 metros en exteriores.

Se decidió trabajar con los mismos, debido a su existencia en el laboratorio GridTics.

3 RESULTADOS OBTENIDOS

La arquitectura ensayada al momento consta de 4 Openmote CC2538, los cuales forman un Destination Oriented Directed Acyclic Graph (DODAG) generado en base al protocolo de ruteo RPL-Lite.[8], como se puede observar en la Figura 2.

En la raíz del DODAG se ubica el router de borde, el cual corre el programa rpl-border-router, utilizando parte de los ejemplos provistos por Contiki-NG. La función de este nodo es actuar como punto de acceso para los nodos y enrutar las comunicaciones provenientes desde estos hacia Internet.

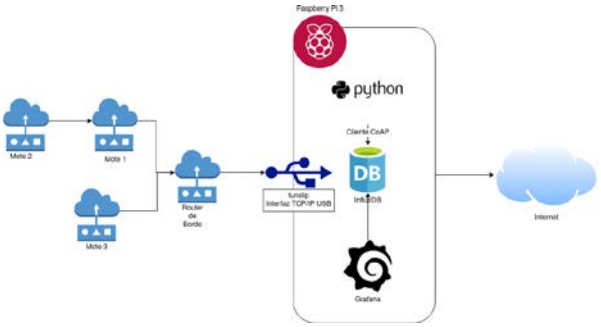

Figura 2: Arquitectura para ensallos

En el extremo de la red se ubica una Raspberry Pi, la cual sirve como plataforma para consultar a los nodos y almacenar información. Para poder conectar la *Raspberry Pi* al router de borde, es necesario utilizar el protocolo SLIPv6 (Serial Line Interface Protocol). Este protocolo crea un puente entre la red RPL y la Raspberry Pi, permitiendo que los nodos tengan acceso a Internet.

El resto de los nodos que forman parte del DODAG, ejecutan una aplicación servidor CoAP [9], el cual responde consultas con información de temperatura, humedad y luminosidad. La aplicación crea recursos propios de *Openmote CC2438* y un recurso adicional CoAP para consultar el consumo energético del nodo,entregado por el módulo

Energest.

CoAP (*Contrained Application Protocol*) es un protocolo de comunicación especializado para dispositivos de bajos recursos computacionales, definido en RFC 7252. Corre sobre UDP y es similar a HTTP.

Cada lectura tomada por los sensores del Openmote se representa como un recurso CoAP, el cual puede ser accedido en una URL propia. Si, por ejemplo, la dirección IPv6 de un nodo es 2001:0db8:85a3::8a2e:0370:7334, el recurso temperatura se puede leer consultando la URL: coap://[2001:0db8:85a3::8a2e:0370:7334]/sensors/temperature y, de manera similar, el recurso luminosidad se encontrará disponible en coap://[2001:0db8:85a3::8a2e:0370:7334]/sensors/light.

La mayor dificultad en la implementación la presentó el cliente que realiza consultas a los servidores CoAP en los nodos. Se decidió dejecutar en la placa *Raspberry Pi el* programa que haga las consultas, puesto que dispone de mayores recursos computacionales y energéticos que el router de borde, para poder mantener comunicaciones con todos los nodos. También en la placa *Raspberry Pi* se encuentra alojada la base de datos, por lo que el programa que realiza las consultas CoAP también se encarga de almacenar los resultados obtenidos en ésta última.

Adcionalmente, se eligió a *InfluxDB[10]* como tecnología para almacenar los datos provenientes de los servidores CoAP. *InfluxDB* es una base de datos orientada a series de tiempo (*Time series database*) optimizada para almacenar datos colectados a intervalos regulares y en grandes volúmenes, tales como lectura de sensores o telemetría.

La lectura de los sensores se realiza mediante consultas CoAP de manera muy similar a un HTTP GET Request. El código del cliente se desarrolló en Python 3 y se utilizó la librería aiocoap. Ésta tiene la particularidad de utilizar funciones no bloqueantes de Python de manera tal que se puedan consultar todos los nodos de la red de manera asincrónica. Es importante destacar que, de utilizar métodos síncronos, la consulta a un nodo en particular, bloquearía el programa hasta que éste no regrese una respuesta. En el caso que su batería se haya agotado, produciría un retardo importante, por los reintentos que haría el cliente.

El funcionamiento del programa cliente se detalla en el diagrama de secuencia en la Figura 3. La comunicación la inicia el cliente para descubrir los nodos que conforman el DODAG. Para esto se hace una consulta HTTP al router de borde. El firmware de éste último incluye un servidor HTTP que contiene la lista actualizada de los nodos de la red. El cliente los obtiene a partir de esta respuesta y crea una estructura de datos con los nodos y sus respectivas direcciones IPv6.

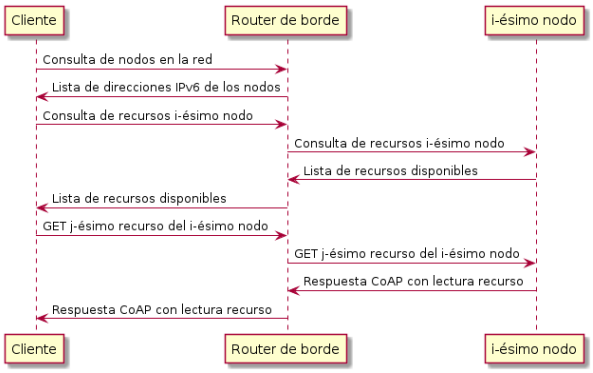

Figura 3: Secuencia de comunicación

El cliente luego itera por esta estructura y consulta el recurso =.well-known/core= de cada nodo. Éste es un endpoint en el cual el servidor CoAP registra los recursos disponibles. La lista de los recursos CoAP se agrega a una tabla hash de tal manera que la llave de dicha tabla es la dirección IPv6 de los nodos y el valor es una lista con los recursos disponibles.

Finalmente el cliente formatea una serie de peticiones que son enviadas de manera asíncrona a los nodos a través del router de borde. Las respuestas devueltas por los nodos son almacenadas en la base de datos, junto con etiquetas que identifican el recurso CoAP al cual se asocia el valor y el nodo del cual proviene. Opcionalmente se puede configurar a nivel de código etiquetas como

geolocalización o fecha y hora. Pero estos valores no son leídos desde los nodos, sino que son generados por software.
Justamente este punto nos encontamos en la actualidad. A futuro inmediato se comenzarán a medir los consumos de energía, para distintos algoritmos de ruteo multicast, para finalmente proponer las mejoras de los mismos.

4 FORMACIÓN DE RECURSOS HUMANOS

Desde la creación del GridTICs, uno de los objetivos a realizar fue la capacitación de los recursos humanos tanto del grupo como externos. Esta actividad de formación se viene cumpliendo desde el comienzo, junto a la divulgación de las tecnologías de redes de sensores. La meta como investigadores es fortalecer la capacidad para realizar investigación científica, generar conocimientos y facilitar la transferencia de tecnología que permita el desarrollo humano.
Este proyecto de investigación posibilita la colaboración inter-institucional y la ejecución de proyectos conjunto entre grupos I+D .
Los cantidad de integrantes que componen el presente proyecto son:
- Dos Investigadores formados
- Tres Investigadores de apoyo
- Dos Becarios graduado (Beca BINID UTN)
- Dos Becarios alumnos (Beca alumno UTN)
- Un Tesista de carrera de grado

Para lograr estos objetivos se desarrollan:
- Dictado de cursos, seminarios y conferencia para público especializado de la región.
- Promoción, coordinación y asistencia técnica de tesinas de grado para alumnos de ingeniería de sistemas de información e ingeniería electrónica de la frmza
- Promoción, coordinación, dirección y asistencia técnica a tesis doctorales, postgrado y/o maestría.
- Presentación de trabajos en congresos y reuniones técnicas/científicas.
- Publicación de trabajos en revistas con/sin referato.

5 BIBLIOGRAFÍA

[1] A. Diedrichs, C. Taffernaberry, G. Mercado, G. Grunwaldt, M. Pecchia, G. Tabacchi, M. González, N. Altamiranda, “RED SIPIA-LP Estudio de mecanismos de bajo consumo energético aplicados a Red de Sensores Inalámbricos en el ámbito de Agricultura de Precisión”, WICC 2016, ISBN: 978-950-698-377-2, Abril 2016.
[2] C. Taffernaberry, G. Mercado, “GW-CIAA-IoT: Gateway con CIAA para red inalámbrica de IoT”, WICC 2016, ISBN: 978-950-698-377-2, Abril 2016
[3] S. Chakrabarti, “IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN)”, RFC 8066, ISSN: 2070-1721, IETF, February 2017
[4] S. Duquennoy, A. Elsts, B. A. Nahas and G. Oikonomo, "TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation," *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Ottawa, ON, 2017, pp. 11-18.
[5] Contiki-NG: disponible en https://github.com/contiki-ng consultado el 29 de Marzo de 2020.
[6] “OpenWSN” disponible en http://www.openwsn.org/ consultado el 29 de Marzo de 2020.
[7] OpenMote: Open-Source Prototyping Platform for the Industrial IoT Conference Paper · September 2015 ISSN 1867-8211
[8] T. Winter, Ed. “RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks”, RFC 6550, ISSN: 2070-1721, IETF, March 2012
[9] Z. Shelby, K. Hartke, C. Bormann “The Constrained Application Protocol (CoAP)” RFC: 7252 ISSN: 2070-1721 June 2014
[10] “InfluxDB open-source time series database”. Disponible en: https://www.influxdata.com consultado el 29 de Marzo 2020.