Universidad de la República

Facultad de Ingeniería - IIE

Redes de Sensores Inalámbricos

LLN Energy-Aware RPL Routing

Nicolás Alves
Mariángel Fernández
Melina Rabinovich

Tutores: Leonardo Steinfeld - Javier Schandy

Montevideo, Uruguay
15 de diciembre de 2014

Resumen

El presente proyecto apuntó a la implementación de un ruteo para las Low-power Lossy Networks basado tanto en los parámetros originales de ruteo RPL[1] como en una estimación de la energía remanente de los nodos que las componen. A tales efectos se hizo uso del módulo Energest existente en el Sistema Operativo Contiki (SO de los nodos) para realizar una estimación aproximada de la carga consumida en función del tiempo en que el nodo se encontró realizando cada una de sus funciones, y la estimación del consumo de corriente asociado a cada función. Luego, a partir de esto y de la carga inicial, se llegó a una noción de la energía remanente en cada nodo.

El desarrollo del proyecto tuvo dos etapas bien diferenciadas, y ordenadas cronológicamente como se detalla a continuación:

Estudio Previo

- Estudio de RFC relacionadas
- Profundización en la estructura RPL
- Familiarización con estructura de Contiki

Implementación y Pruebas

- Implementación de nueva Función Objetivo con foco en la energía del nodo.
- Simulaciones Cooja, Foren6 y estudio de autonomía en configuración de red básica.

[1] Routing Protocol for Low-Power and Lossy Networks

Tabla de contenido

Capítulo 1

Introducción

Las redes de sensores inalámbricos (WSN por sus siglas en inglés) están conformadas por un número de componentes autónomos denominados nodos, con capacidades sensitivas de parámetros físicos y de comunicación con otros nodos. En una WSN dada cada nodo tiene funciones específicas determinadas por software, pudiendo estas ser una combinación o alguna de las siguientes:

- Sensor
- Emisor
- Receptor
- Router de borde

Evidentemente, en una WSN un nodo tiene sentido estando en comunicación con el resto de la red, por lo cual alguna de las funciones de comunicación siempre estará presente.

Dada la amplitud de funciones que presentan sus componentes y, por ende la variedad de aplicaciones posibles que tienen, las WSN se han vuelto una herramienta imprescindible en varias áreas. Algunas de estas, en las que su aplicación es más común, son:

- Sistemas de eficiencia energética
- Sistemas de seguridad
- Sensores industriales
- Domótica
- Sensores ambientales y sistemas de control de plantaciones

Teniendo en cuenta la independencia que se pretende alcanzar en los sensores en este tipo de redes, resulta evidente que los recursos energéticos de los nodos que la componen sean un punto crítico a la hora de buscar una mayor autonomía. Si bien existen métodos de recolección de energía (paneles fotovoltaicos, conexión a redes eléctricas, etc.) el caso

más común es en el cual la energía se encuentra en una reserva (batería) y, por ende, se hace imperioso el buen aprovechamiento de los recursos disponibles. A raíz de esto es que ya existen algunas implementaciones en software para realizar una estimación del consumo de energía de los nodos al ejecutar sus funciones; y es en particular el módulo *energest* de Contiki, la implementación que sirvió como punto de partida de este proyecto.

1.1. Descripción del problema

Como una generalización del problema energético, y yendo más allá de la visión particular de la problemática de la carga remanente en cada nodo, se adoptó una visión global de la red en términos energéticos. Esto podría traducirse en *¿Cuáles son los puntos críticos de la red en términos de energía?* o *¿Podríamos prolongar la autonomía de una rama de la red con riesgo de perderse en el caso que un nodo queda fuera de servicio por falta de energía?*.

En efecto, y dado cierto tipo de topología de red, las limitaciones energéticas de un nodo pueden representar el riesgo de la pérdida de utilidad de un número determinado de otros nodos de la red con carga suficiente para seguir funcionando. Para ejemplificar esto, considérese la topología presentada en la figura 1.1. En este caso el nodo 1 es el root de la red[1] y sólo se accede él por los nodos 2 y 3, y el resto de la red sólo llega a estos nodos a través del nodo 4. Por lo cual esta topología presenta nodos críticos en distintos lugares y con distintas posibilidades a la hora de buscar la mejor autonomía:

***Primer nivel (nodos 2 y 3)*:** En este caso el enrutamiento hacia el root puede hacerse tanto a través del nodo 2, como del nodo 3, indistintamente. La elección del nodo la hará el protocolo de ruteo.

***Segundo nivel (nodo 4)*:** En este caso, a diferencia del anterior, no hay posibilidad de elección y el nodo 4 tendrá la tarea de enrutar toda la comunicación generada aguas abajo en la red.

***Tercer nivel (nodos 5 y 6)*:** Este caso, al igual que el del *primer nivel*, presenta la capacidad de opción entre los nodos 5 y 6 por parte del protocolo de ruteo.

En la configuración presentada en la figura 1.1 se consideran los casos frente a los cuales se puede estar a la hora de evaluar el ruteo a través de un nodo específico de la red. El caso del *segundo nivel* representa un caso crítico, ya que toda la comunicación de la parte de la red que se encuentre aguas abajo deberá ser ruteada hacia el root a través de este nodo. Este hecho generará que la autonomía del nodo sea sensiblemente menor, ya que además de las funciones de medición y envío de datos propios se encargará de recibir todos los datos generados en los nodos restantes, y reenviarlos hacia el root. Por ende, por cada dato recolectado y enviado por los nodos aguas abajo en la red, el nodo 4 deberá llevar a cabo una recepción y una retransmisión, teniendo estas funciones un consumo de energía asociado considerable. La solución al problema que se genera en este nivel escapa al alcance de este proyecto, ya que concierne a la eficiencia del uso energético por parte de los nodos.

[1] Gateway a internet o a equipo de control o adquisición, por ejemplo

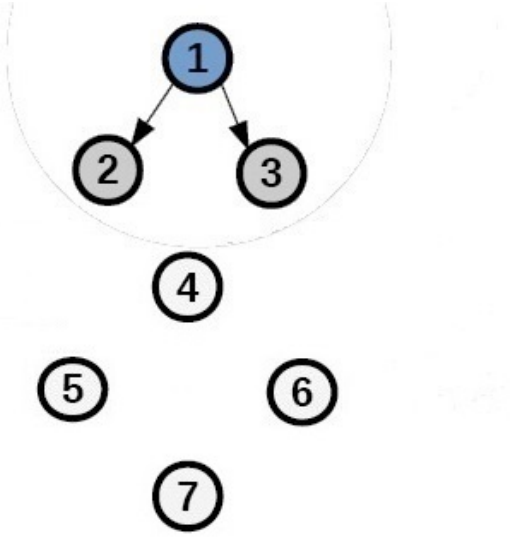

Figura 1.1: Topología de ejemplo

Centrando la atención en los otros dos niveles detallados en el ejemplo, se observa rápidamente que el protocolo de ruteo deberá, en algún momento y con algún criterio, optar por alguno de los nodos que lo puedan aproximar al root. El problema que se atacó en este proyecto surge en cómo se toma la decisión respecto a qué nodo realiza el ruteo en casos como el de los niveles que aquí se consideran; dado que la selección del nodo padre[2] en el ruteo ContikiRPL [3] se realiza con una métrica[4] de enlace basada en ETX[5], sin tener en cuenta el estado de carga de la batería del nodo. Esto puede generar que se sobrecargue un nodo con el ruteo del tráfico de gran parte de la red, llevando a una reducción drástica de su autonomía, incluso pudiendo optarse por otro padre que esté dentro del alcance en cualquier momento. En este escenario, y dependiendo de la topología de la red, es probable que la rama dependiente del nodo sobrecargado quede incomunicada antes de lo que se podría prever al hacer un estudio genérico de la autonomía de la red.

En la figura 1.2 se muestra una configuración básica elegida para evaluar el comportamiento del ruteo, donde cada nodo tiene alcance sólo a los nodos en vértices adyacentes, y donde el nodo elegido por el protocolo ContikiRPL como padre del nodo 4 es el nodo 3. El resultado final de este ruteo, en términos de la energía, se puede apreciar en el gráfico de la figura, donde se evidencia que el nodo 3 será el primero en quedarse sin carga, y resulta coherente pensar que en una red con mayor número de nodos y, por ende, tráfico la diferencia de autonomía crecerá. A raíz de esto es que surge la idea de una solución mediante la implementación de un protocolo de ruteo basado en la energía remanente en los nodos como métrica para la selección de padre.

[2]Vecino que elige cada nodo para enviar sus datos

[3]Ruteo implementado por defecto en Contiki-OS

[4]Valoración que se le da al nodo (métrica de nodo) o al enlace (métrica de enlace)

[5]Expected Number of Transmissions: Estimativo de cantidad de transmisiones necesarias para lograr envío exitoso

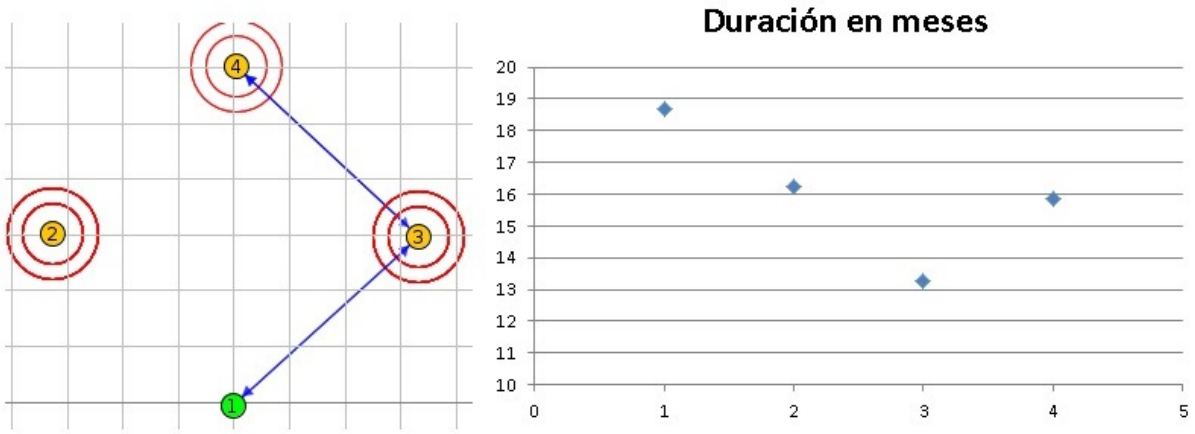

Figura 1.2: Comportamiento de simulación ETX

Con base en lo anteriormente desarrollado, se presenta a continuación una evaluación particular de cada posibilidad de ruteo, tomando en cuenta las dos métricas mencionadas:

Ruteo basado en ETX

Este ruteo presenta la ventaja de implementar un método recursivo para estimar, en base al historial de transmisiones anteriores, la cantidad de transmisiones esperadas para lograr una transmisión exitosa. Sin embargo, como ya se explicó, no tiene en consideración la carga remanente en las baterías de los posibles padres.

Ruteo basado en energía

Este ruteo apunta a paliar el punto débil del ruteo ETX ya que tiene en consideración la energía remanente en los nodos que integran el root-path[6] para optar por un camino. Sin embargo al no tener en consideración la cantidad de transmisiones que necesitará para un envío de datos exitoso, presenta una posible falencia, ya que si la conectividad con el nodo con más energía es mala, y por ende el envío de un paquete le lleva varios intentos, el uso de la energía disponible sería menos eficiente que en el caso ETX.

Ruteo combinado ETX-Energía

El uso de una métrica que combine la métrica de nodo (basada en la energía) y la métrica de enlace (basada en la ETX) antes mencionadas parece ser la solución más completa para el problema que aquí se presentó, ya que cada métrica afecta positivamente en el punto débil que presenta la métrica complementaria.

[6]Nodos que integran la posible ruta hasta el root de la red

1.2. Antecedentes

La implementación en Contiki-OS de un ruteo basado en métricas que consideren la energía remanente en los nodos no presentaba antecedentes visibles, o de acceso público. Sin embargo, sí se encontraron implementaciones de código mediante las cuales podían realizarse estimaciones de la energía consumida por el nodo. Esta implementación es el módulo **energest**[7] presente en el Kernel del sistema operativo Contiki, cuya ejecución fue realizada a través de tres funciones:

energest_init()
: Función que reinicia el conteo que realiza este módulo, y que se ejecuta en la inicialización del Contiki-OS bajo el supuesto de que las baterías están cargadas completamente en ese momento.

energest_type_time(type)
: Función que recibe como parámetro el estado del nodo del cual se quiere saber el tiempo de ejecución, y devuelve dicho tiempo expresado en ticks de reloj del sistema.

energest_flush()
: Función que actualiza el conteo al momento de ser ejecutado, sin importar en qué estado ni avance de estado se ejecute. La utilidad de esta función reside en que el conteo se actualiza por defecto cada vez que el nodo sale de un estado, pero el momento de solicitud no tiene porque ser necesariamente el tiempo de actualización. Con esta función se logra, en definitiva, actualizar el conteo hasta el momento en que ella sea ejecutada, reduciendo así el error cometido en el cálculo de tiempo de estados.

Este módulo realiza un conteo de ticks de reloj diferenciado por estados posibles en los que se puede encontrar el nodo. Este conteo se actualiza para todos los estados al cambiar de estado, pudiendo ser este uno de los siguientes: *ON*, *TX*, *RX*, *INT* o *LPM*. Sin embargo, el nodo puede estar en más de un estado a la vez, ya que al estar en los estados *TX* y *RX* (por ejemplo) deberá estar necesariamente también en estado *ON*. Luego, con estimados de consumo de corriente para cada estado, asumiendo una tensión media de la batería, y la última actualización de conteo mediante *energest_flush()* y *energest_type_time(type)* para cada estado, se estima la energía consumida por el nodo desde la última ejecución de *energest_init()*. Por ende, con la utilización de Energest se puede obtener un buen estimativo de la energía remanente en los nodos.

Cabe acotar, además, que en el mismo sentido se recurrió a la bibliografía de la implementación *Powertrace*[1] para alcanzar un entendimiento profundo del *Energest*, dada su similitud en el procedimiento para el cálculo de la energía consumida.

La bibliografía que se tomó como punto de partida y base del estudio previo para la resolución del proyecto es proveniente, en su gran mayoría, de la IETF (*Internet Engineering Task Force*), y en particular de sus RFC[8] (*Request For Comments*). También se

[7] Ruta Contiki: /contiki/core/sys/energest.c y energest.h

[8] Publicaciones (a modo de memorando) de ingenieros integrantes de la IETF que derivan en la Normalización de Internet

recurrió en primera instancia al documento *Energy-awareness metrics global applicability guidelines*[2] del ROLL[9] Working Group, pero dado el alcance final que tuvo el proyecto su utilidad fue mínima.

A continuación se presenta un resumen del contenido de los RFC consultados:

RFC 6550[3]: *IPv6 Routing Protocol for Low-Power and Lossy Networks*
Este RFC, como su título lo indica, establece los parámetros de funcionamiento del protocolo RPL para las WSN. En otras palabras, este documento especifica el protocolo RPL de ruteo IPv6 para las *Low-power and Lossy Networks*. Este protocolo establece los mecanismos para el tráfico multipunto-a-punto que se da desde varios elementos de la red hacia un nodo central (root, por ejemplo), así como para el tráfico punto-a-multipunto desde un elemento central hacia varios componentes de la red y para el tráfico punto-a-punto. Aquí se definen, además, la construcción de la topología de la red, los identificadores de RPL, los mensajes de control, los rangos y las métricas y restricciones utilizadas por RPL.

RFC 6551[4]: *Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks*
Este RFC, como su título lo indica, define las métricas de ruteo utilizadas para los cálculos de root-path en las LLN. Dadas las características particulares que presentan las WSN, respecto a las redes cableadas o inalámbricas tradicionales, se hace necesaria la especificación de nuevas métricas y restricciones. En este sentido, y contrariamente con los protocolos IGP[10] que utilizan conteo de saltos y métricas de enlace basadas en ancho de banda, en este documento se especifican un conjunto de métricas de nodo y enlace y restricciones para RPL.

El RFC 6550, en conjunto con el material de clase referente a RPL[5], fueron de gran utilidad para la profunda comprensión del protocolo RPL, su estructura, su implementación y el tráfico entre los elementos de la red con el fin de establecer los caminos para el intercambio de datos. A tales efectos, y para asentar los conceptos de esta bibliografía, también se profundizó en el código que implementa el protocolo ContikiRPL. Estudiando el RFC 6551 se adquirieron los conceptos respecto a las métricas de nodo y de enlace y de las restricciones del protocolo. Además se llegó a comprender cabalmente el concepto y cálculo de ETX, así como el uso de objetos y contenedores de RPL.

[9] *Routing Over Low-power and Lossy networks*
[10] Interior Gateway Protocol

Capítulo 2

Objetivos

Como objetivo principal del proyecto se planteó la **optimización del uso de la energía en las LLN mediante la implementación de un ruteo basado en la estimación de la energía disponible en los elementos de la red**.

Dado el estudio de las ventajas y debilidades de los protocolos basados en ETX y Energía se establecieron los siguientes objetivos particulares:

Objetivo primario: Implementación y pruebas de un ruteo con métrica en base a la energía remanente en los nodos del *root-path*.

Objetivo secundario: Implementación, pruebas y comparación de un ruteo que combine las métricas basadas en ETX y en Energía para alcanzar un ruteo que asegure una mayor autonomía de las redes. Este objetivo se planteó en dependencia al tiempo disponible al finalizar el objetivo primario.

Capítulo 3

Alcance

Al definir el plan de trabajo se decidió implementar en primera instancia el ruteo con la métrica de nodo basada en energía dejando de lado la calidad de enlace (ETX), para luego, en una segunda instancia y sujeto a la disponibilidad de tiempo, implementar un ruteo combinando las métricas de nodo (basada en energía) y de enlace (ETX). Sin embargo, y dados los problemas experimentados en las etapas de Diseño (**4.1.1**) e Implementación (**4.2.1**) no fue posible, en definitiva, llevar a cabo la implementación de un ruteo combinado con énfasis en la energía de los nodos y la ETX.

Por lo que en definitiva este proyecto presenta como alcance:

- Modificación del código de implementación del protocolo RPL en Contiki de forma de que utilice la nueva función objetivo.
- Implementación de la nueva función objetivo *rpl-energy.c* con métrica de nodo basada en la energía remanente, utilizando el módulo Energest de Contiki-OS.
- Simulación de una WSN y posterior evaluación de resultados con la nueva métrica implementada.

Quedando fuera del alcance del proyecto:

- Modificación del código de implementación del protocolo RPL en Contiki de forma de llevar a la práctica un ruteo combinando de la métrica de nodo basada en energía y métrica de enlace basada en ETX.
- Simulación y evaluación de una WSN con la nueva métrica combinada implementada.

Capítulo 4

Diseño e Implementación

4.1. Diseño

Como se explica en RFC 6551 la función objetivo (OF, por sus siglas en inglés) es la función que define cómo los nodos eligen y optimizan las rutas en RPL. En otras palabras, se especifican reglas para la selección de padre y cálculo de rank en base a la métrica y restricciones allí establecidas.

Para lograr el objetivo planteado se necesitaba, entonces, crear una nueva OF que cumpliera con los objetivos de un ruteo con enfoque energético; se nombró *rpl-energy.c* a esta función. Dicha función debía implementar, al menos, las funciones que se especifican como API básica en *rpl.h*. En este sentido, se trabajó sobre el código que se encuentra en /contiki/core/net/rpl/ de la implementación original de Contiki, identificando allí dos funciones objetivo definidas: la función por defecto llamada *rpl-mrhof.c* y la función *rpl-of0.c*.

Luego de haber comprendido cabalmente las funciones del código original se estuvo en condiciones de aplicar los conocimientos adquiridos al diseño de la nueva OF a implementar. Para esto, como ya se mencionó, se debió respetar el API definido en el código original, del cual se realiza un punteo continuación:

- ***reset(dag)***
- ***neighbor_link_callback(parent, known, etx)***
- ***best_parent(parent1,parent2)***
- ***best_dag(dag1,dag2)***
- ***calculate_rank(parent, base_rank)***
- ***update_metric_container(dag)***

Para aprovechar el funcionamiento del protocolo RPL planteado originalmente se respetaron la funciones descritas anteriormente en estructura, pero se modificó su contenido para implementar una métrica de nodo basada en la estimación de energía remanente en el nodo, sustituyendo las decisiones originales basadas en la métrica de enlace (ETX). Si bien esto debió ser realizado como se detalla, nada impedía que se definieran otras

funciones para realizar tareas necesarias, y obviamente es aquí donde se recurrió al uso del *energest* para la estimación de la energía remanente en el nodo. De hecho, el archivo *rpl-mrhof.c* (OF en la implementación original) posee algunas funciones no incluidas en la API para facilitar su funcionamiento.

4.1.1. Problemas

Los problemas experimentados en esta etapa del proyecto estuvieron fuertemente ligados al código original del Contiki-OS, del cual se partió. Si bien al momento de adentrarnos en el código ya se contaba, a través de los RFC, con un conocimiento teórico de la estructura, contenedores y mensajes intercambiados en RPL, cabe mencionar que al ahondar en el código original que implementa ContikiRPL nos encontramos con una estructura muy compleja, con llamadas a funciones de variados archivos, pero sin comentario explicativo alguno. A raíz de esto es que el tiempo dedicado a la comprensión del funcionamiento del código original resultó bastante más de lo planificado.

4.2. Implementación

Luego del diseño, donde se logró un entendimiento de la estructura de ContikiRPL, y al momento de implementar, se tomó como punto de partida el código ya implementado en Contiki. Con esto se hizo más simple la implementación del nuevo código, y por ende la implementación del ruteo basado en estimación de energía con una métrica de nodo. En otras palabras, se partió de la estructura original de Contiki, implementando las mismas funciones incluidas en el archivo de la OF original (/contiki/core/net/rpl/rpl-mrhof.c), pero con un enfoque en la estimación de energía remanente.

Ahondando en la estructura de ContikiRPL se encontró que existen muchos otros archivos involucrados con el ruteo, además de las OF ya mencionadas (*rpl-of0.c* y *rpl-mrhof.c*), siendo estos:

***rpl.h*:** Archivo que define varias constantes de uso común en el código; como ser los valores asignados por la IANA para los tipos de métrica/restricción (7.2) con los cuales identificar los objetos en el contenedor de métrica, además de varias estructuras, por ejemplo la del contenedor de métrica.

***rpl.c*:** Archivo que define varias funciones básicas del protocolo RPL; siendo una de estas la función de inicialización.

***rpl-icmp6.c*:** Donde se definen funciones concernientes a los mensajes que maneja el protocolo RPL: DIO, DAO, DIS.

***rpl-config.h*:** Donde se definen distintas constantes de configuración; por ejemplo si hay RPL_CONF_DAG_MC seteado se lo asigna a RPL_DAG_MC, en caso contrario se le asigna RPL_DAG_MC_NONE.

La sigla mrhof de *rpl-mrhof.c* viene de Minimum Rank with Hysteresis Objective Function, o función objetivo de rango mínimo con histéresis en español. Por lo que resulta

obvio el hecho de que esta OF implementa histéresis para el cambio de padre, evitando así una fluctuación constante cuando dos padres presentan pequeñas diferencias entre sí, respecto a la métrica establecida (ETX en ese caso). Además, y exceptuando las restricciones que se puedan presentar, esta OF optará por el padre con menor rank para el root-path. El archivo *rpl-mrhof.c*, al igual que la mayoría de los archivos involucrados, toma decisiones según la variable RPL_DAG_MC. Como no está seteada por defecto el archivo *rpl-config.h* la setea en *RPL_DAG_MC_NONE* y bajo esa configuración no muestra el contenedor de métrica. Sin embargo, resultaba de interés poder ver el contenedor, puesto que es una buena forma de chequear que todo funcione acorde a lo esperado y así comprender mejor el funcionamiento, por lo que se seteó *RPL_DAG_MC_ETX*. Para ello se creó el archivo *project-conf.h*, de modo de no modificar los archivos originales y realizar una configuración específica para el proyecto en forma ordenada. Bajo dicha configuración entonces se intentó comprender el funcionamiento de ContikiRPL, que utiliza métrica de enlace ETX.

A continuación se realiza un breve desarrollo del comportamiento de cada una de las funciones incluidas en la OF *rpl-mrhof.c*:

***calculate_path_metric(parent, base_rank)*:** Esta función recibe como entrada un padre y devuelve el cálculo de camino dependiendo del seteo de la variable

RPL_DAG_MC. En el caso de setearla como *RPL_DAG_MC_ETX* devuelve la suma del ETX medido para el padre (*p->mc.obj.etx*) y la métrica de enlace (*link_metric*), si no tuviera padre da un valor fijo (*MAX_PATH_COST*) en base 256.

***reset(dag)*:** Esta función avisa que se reestablece el estado de la función objetivo para un DAG determinado imprimiendo en pantalla que se reseteó la función objetivo. Se utiliza cuando hay una reparación global del DAG.

***neighbor_link_callback(parent, known, etx)*:** Recibe información de los vecinos sobre la capa de enlace. El parámetro known se setea en 0 o 1 dado si el padre es conocido o no. El etx especifica el valor ETX (transmisiones estimadas necesarias) para el vecino previo al paso actual de la iteración.

***calculate_rank(parent, base_rank)*:** Calcula un valor de rank usando el rank del padre y un rank base. Si el padre es *NULL* (el nodo no es el padre), se realiza un incremento por defecto del rank, y se lo suma al rank base[1]. Sino, la función objetivo utiliza información conocida del padre para seleccionar un incremento para el rank base. El valor del rank para esta métrica se da en base 256. Además si el rank base es cero y el nodo no tiene un padre se le asigna rank infinito (*INFINITE_RANK*/256), si tuviera padre se define el incremento de rank igual al del padre (*p->rank*). En cambio, si el rank base no fuera cero y el nodo no tiene un padre el incremento de rank es fijo (*RPL_INIT_LINK_METRIC*), si tuviera padre el incremento de rank es la métrica de enlace del padre (*p->link_metric*). Por último, si el rank base sumado al incremento de rank no superan el rank infinito el nuevo rank será esa suma,

[1]Rank que trae el nodo vecino

sino será el rank infinito (*INFINITE_RANK*). Tiene entonces un rank mínimo de 1 (256 en base 256) y uno máximo de 256 (65535 en base 256).
La función entonces devuelve un rank comprendido entre *RPL_INIT_LINK_METRIC* y *INFINITE_RANK*/256.

***best_dag(dag1,dag2)*:** Esta función compara los dos DAG recibidos y retorna el mejor acorde a los parámetros establecidos por la OF. Para este proyecto no resultó de interés porque siempre se utilizaron topologías con un único DAG.

***best_parent(parent1,parent2)*:** Compara dos padres posibles y devuelve el mejor acorde a los parámetros establecidos por la OF. Es en esta función donde se implementa la histéresis para el cambio de padre.

***update_metric_container(dag)*:** Recibe una instancia RPL y actualiza el contenedor de métrica para los DIO salientes en un cierto DAG. Si en la función objetivo del DAG no se usan contenedores de métrica, la función debe setear el tipo de objeto a *RPL_DAG_MC_NONE*. Contempla el caso en que *RPL_DAG_MC* esté seteada en *RPL_DAG_MC_ENERGY* pero no utiliza el módulo energest para realizar una estimación de energía remanente en el nodo, solo setea las banderas dependiendo el tipo de alimentación del nodo.

Como se mencionó, el módulo *energest* es uno de los antecedentes que presentaba este proyecto respecto al cálculo de energía. Este módulo se encuentra implementado pero no utilizado en ContikiRPL, y es la herramienta que se utilizó en la estimación de energía de los nodos.

Finalizado el estudio y alcanzada la comprensión de las funciones antes mencionadas, se procedió a implementar una nueva OF, archivo nombrado como *rpl-energy.c* y ubicado en /contiki/core/net/rpl/. De forma de aprovechar la estructura existente de ContikiRPL, se definió $RPL\_CONF\_OF = rpl\_energy$ a efectos que Contiki utilice nuestra nueva OF en lugar de *rpl-mrhof.c*. Además, cada OF define, al inicio, una estructura compuesta de todas las funciones de la API definidas, además de un número *ocp* (objective code point) que la identifica. Este identificador es 0 para *rpl-of0.c* y 1 para *rpl-mrhof.c*), de aquí que se setea (en el archivo de configuración del proyecto, *project-conf.h RPL_CONF_OF* con el nombre de dicha estructura.

Como ya se mencionó, para poder visualizar correctamente el contenedor de métrica correspondiente se debe setear la variable *RPL_DAG_MC* en el archivo de configuración del proyecto (*project-conf.h*) con un valor distinto de *RPL_DAG_MC_NONE*, por lo que para ver el contenedor de interés se la estableció como *RPL_DAG_MC_ENERGY*.

De forma de poder realizar la estimación de energía que nos brindara la posibilidad de llevar a cabo el ruteo deseado, se implementó la función *estimo_energia(void)*. Dado que el módulo energest devuelve valores de mediciones de tiempo, y que se contaba con la información del consumo de corriente en cada estado, se asumió un voltaje constante

de tres volts[2] para calcular así la estimación de energía consumida. Dicha estimación se calculó como la sumatoria de $V_{bat} * I_{Estado} * t_{Estado}$ para los distintos estados, siendo I_{Estado} la corriente del estado y t_{Estado} el tiempo total acumulado en cada estado, calculado a partir de la función *energest_type_time* de energest (expresado en segundos dividiendo entre *RTIMER_SECOND*). Como se explicó (1.2), previamente a calcular la estimación se invoca a la función *energest_flush* para que la estimación sea lo más actualizada posible. La función devuelve el porcentaje de energía estimada remanente en el nodo por lo cual se asume en los nodos una energía inicial, igual para todos, y fijada en 19.44mJ (lo que entregaría una batería de 1800mAh tomando en cuenta los mismos 3V). Luego, y en base a lo anterior, se calcula el porcentaje de energía remanente en el nodo. Por simplicidad al utilizar tipos de dato, la estimación se expresó en unidades de mJ.

Las corrientes consumidas en cada uno de los estados son las utilizadas en el laboratorio del curso y se definieron en el archivo de configuración del proyecto (*project-conf.h*); sus valores se expresan en la Tabla 4.1 .

Estado	I
ON	6.25 mA
TX	19.5 mA
RX	21.8 mA
INT	19.5 mA
LPM	54 uA

Tabla 4.1: Estimación de corrientes consumidas en los distintos estados.

Las funciones ya existentes en la OF original *rpl-mrhof.c* resultaron con los siguientes cambios:

- Se implementa nuevamente el contenido de la función ***calculate_path_metric*** de modo que devuelva la estimación de energía que está en el contenedor de métrica, en base 256 para que cierre con el código ya existente, no es más que el rank.
- La función ***reset(dag)*** se mantiene sin cambios respecto de *rpl-mrhof.c*, por lo que solamente imprime RPL: Reset ENERGY".
- La función ***neighbor_link_callback*** no fue editada, pero como es llamada de todos lados se dejó tal cual estaba.
- Se redefine la función ***calculate_rank*** en forma análoga a la de *rpl-mrhof.c*, manteniendo la base 256 para que todo cierre en el código existente de ContikiRPL. Tomando como referencia el cálculo de rank para ETX, se define como rank mínimo 1 (256 en base 256) y máximo 11 (2816 en base 256), siendo 1 cuando el nodo tiene la batería completa y 11 cuando la tiene agotada por completo. El nuevo rank se calcula como $[\frac{100-\%b}{10} + 1] * 256$.

[2]Dado que se estaba realizando una estimación, y aproximando a un voltaje promedio de la batería durante su ciclo de descarga, se estimó este valor como apropiado

- La función ***best_dag*** no se modificó.
- Se redefine la función ***best_parent*** de modo que devuelva el padre con mayor energía remanente. En cuanto a la histéresis se redefine el rango de modo que si un padre preferido no dista en un porcentaje mayor al 10 % de la carga total de la batería no se cambia de padre. Si no hubiera histéresis devuelve el padre con mayor energía remanente sin importar la diferencia.
- Se redefine la función ***update_metric_container*** de modo que se guarde la estimación que devuelve la función *estimo_energia* en el objeto adecuado, que es el node energy o NE dentro del campo *E_E*. También se setea una bandera (en NE) que indica que lo que hay guardado allí es una estimación de energía.

Luego de realizados los cambios anteriormente detallados, quedó implementada la nueva OF ***rpl-energy.c***.

4.2.1. Problemas

El mayor problema experimentado en la etapa de implementación fue al intentar simular el nuevo código, y al relacionarlo con el tipo de mote que se utilizó durante el curso. Al compilar el código se recibía constantemente un mensaje de overflow de la memoria de programación del mote y se truncaba el proceso. Luego de probar, sin éxito, cambiar el mote destino de la compilación por uno con mayor capacidad, se eliminaron del código las líneas originales que tenían el objetivo de implementar el ruteo por métrica con ETX (código que se había conservado con el objetivo de poder optar por una y otra métrica), pudiendo así compilar nuestro código satisfactoriamente. Este inconveniente llevó a pensar que en caso de implementar un ruteo combinado habrá que tener en consideración la memoria disponible para programación del mote, y que tal vez este ruteo no pueda ser utilizado en motes con la capacidad del Sky CC2420.

Capítulo 5

Pruebas

Se trabajó en Cooja simulando una modificación del ejemplo unicast-example.csc que se encuentra bajo /contiki/examples/ipv6/simple-udp-rpl/.
Se diseñó una topología de rombo como se observa en la figura 5.1, donde el nodo 1 tiene el código de unicast-receiver.c y los nodos 2, 3 y 4 tienen unicast-sender.c.

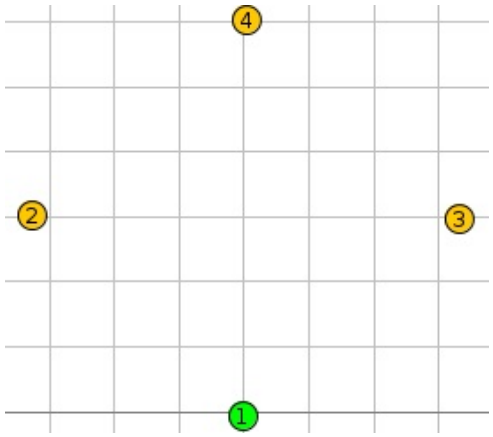

Figura 5.1: Topología de red elegida

Se utilizó foren6 para la interpretación de los mensajes DIO identificados en los archivos *.pcap generados con Cooja.

ETX

Como se mencionó en Impementación (4.2), se observó inicialmente el comportamiento original del Contiki para esta topología. Estudiando particularmente las funciones incluidas en el archivo /contiki/core/net/rpl/rpl-mrhof.c. Contiki por defecto utiliza ETX como métrica, por lo cual se realizaron las pruebas sobre dicha configuración, activando previamente la vista del contenedor de métrica (seteando *RPL_DAG_MC* en *RPL_DAG_MC_ETX*).

La figura 5.2 se obtuvo del foren6, en la misma se puede observar una captura de un mensaje DIO en el cual viaja un contenedor de métrica. El contenedor de métrica contiene 070400020000 (hexadecimal), el 07040002 se corresponde con el objeto genérico. Allí se indica en Routing MC Type (7.1.1) que el objeto que sigue es un ETX, pues tiene un 07. Y el 02 al final indica que a continuación llegarán 2 bytes, que corresponde a los 0000.

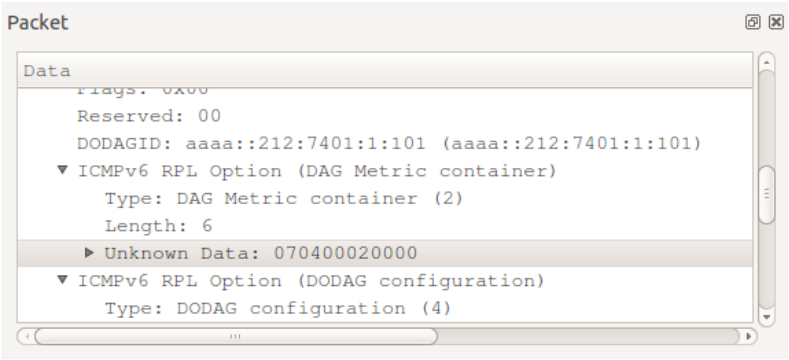

Figura 5.2: Captura Foren6 del MC para ETX

En la figura 5.3 se muestra el gráfico de duración (en meses) de las baterías de los distintos nodos. Se obtuvo como resultado de procesar los datos obtenidos a través de una simulación en Cooja con la herramienta PowerTracker. Allí se puede observar que la red, en su totalidad, tendrá una autonomía de alrededor de 13 meses, ya que es lo que dura la batería del nodo 3 y este es el primero en agotarla. Se entiende que la métrica ETX tendrá sus ventajas en cuanto a calidad de enlace, pero una vez que elige un padre con mejor calidad de enlace no habría razon para que lo cambie. Por lo tanto seguro hay un nodo que agota su energía antes haciendo que la autonomía de la red en su conjunto no sea la óptima, como sería deseable.

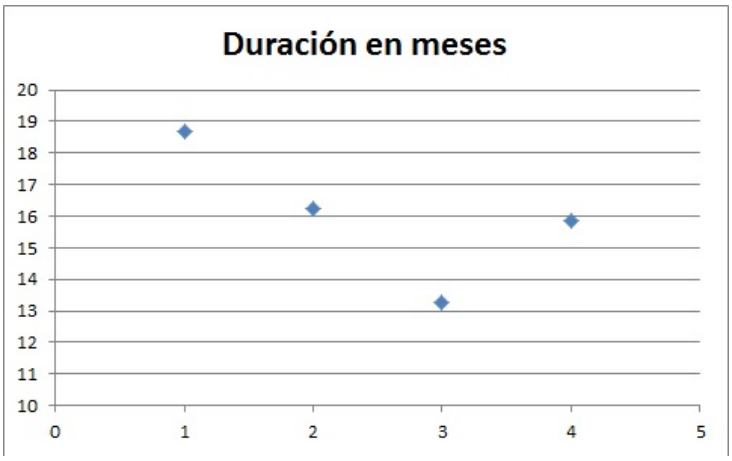

Figura 5.3: Powertracker para ETX

Luego de esto se realizaron las mismas pruebas sobre la nueva OF implementada (*rpl-energy.c*).

Energía

Análogamente a lo visto para ETX, se puede observar en la figura 5.4 un mensaje DIO en el cual viaja un contenedor de métrica. El mismo contiene en esta oportunidad el valor hexadecimal 020400020357. El 02040002 corresponde al objeto genérico, en él se indica que a continuación llegará un objeto NE (node energy) indicado con el 02 inicial, y que llegarán dos bytes indicando con el 02 final.

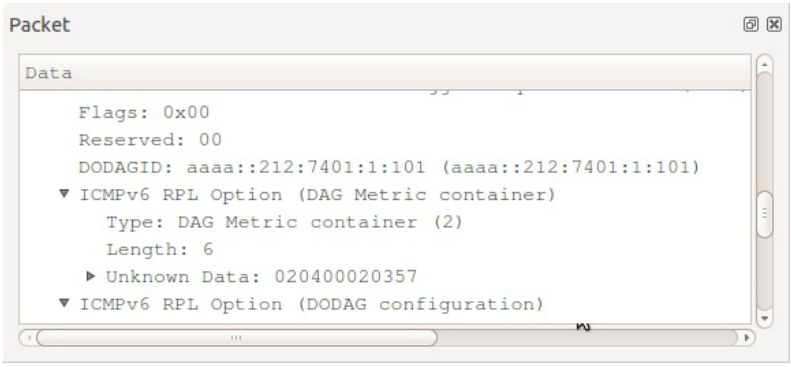

Figura 5.4: Captura Foren6 del MC para energía

El NE contiene el hexadecimal 0357, que indica con el 03 que es un nodo alimentado por baterías (tiene seteada la bandera T -type- en 01 binario) y que a continuación en el NE se tiene guardada una estimación de la energía remanente del nodo (tiene seteada la bandera E).

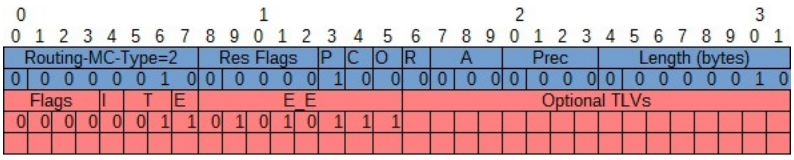

Figura 5.5: NE bit a bit

Luego el 57 en hexadecimal no es más que la estimación de energía remanente del nodo (*E_E*), que como puede visualizarse en la captura de la simulación de Cooja, figura 5.6, coincide con el valor estimado que es 87 %.

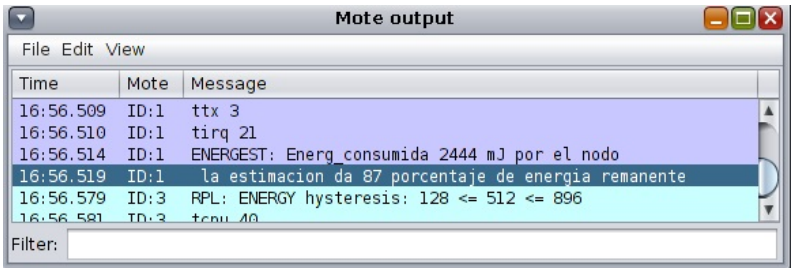

Figura 5.6: Captura de *Mote Output* de la simulación en Cooja

En la figura 5.7 se puede ver que el rank de los nodos 2 y 3 es similar, como era de esperar dado que el tráfico se reparte parejo entre los nodos y por lo tanto su energía remanente baja en forma similar.

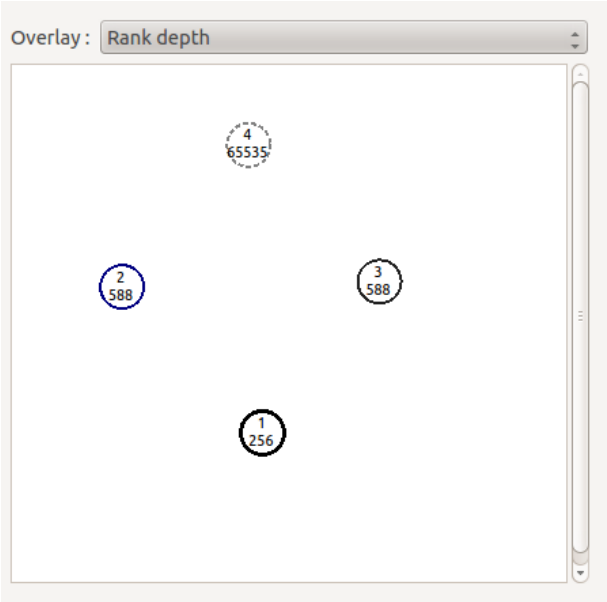

Figura 5.7: rankEnergy - Nodos

En la figura 5.8 se muestra el gráfico de duración (en meses) de las baterías de los distintos nodos. Se obtuvo como resultado de procesar los datos obtenidos a través de una simulación en cooja con la herramienta PowerTracker. Se puede observar que la red tendrá una autonomía de alrededor de entre 15 y 16 meses, que es lo que dura la batería del nodo 2, que es el primero en agotarla. Se observa que los nodos agotan la batería en forma más pareja que lo hacían para ETX. Se entiende que si bien mejora el ruteo de modo que alterna entre un padre y otro, sin que uno sólo soporte todo el tráfico como pasaba para ETX, a veces intenta enviar a través de un nodo con un enlace pobre y manda varias veces por un nodo con mejor energía agotándolo, en vez de mandar algunas veces menos por un nodo con menor energía pero que hace el tráfico más eficiente.

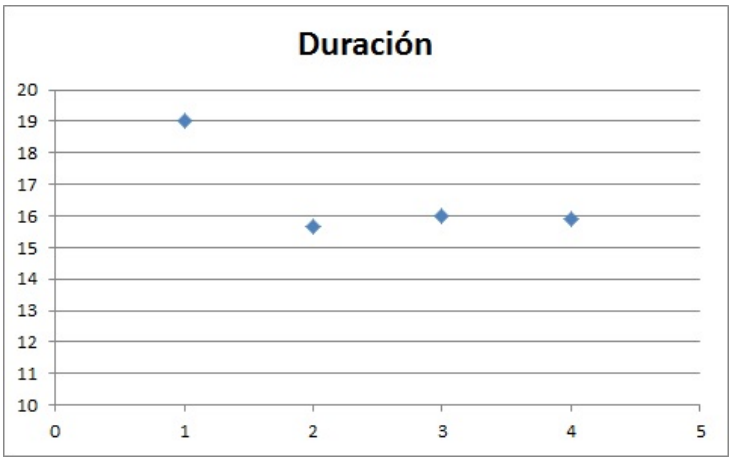

Figura 5.8: PowerTracker Energía

En suma, si bien el ruteo por energía balancea más el gasto de energía en los nodos, la autonomía de la red resulta ser muy similar a la del caso ETX, por lo que sería bueno probar con una combinación de ambas, de modo de mejorar este punto.

5.1. Problemas

No se logró probar que funcionara el cambio de padre contemplando histéresis. En principio se creyó que quizás la topología no fue la adecuada, pues para cambiar de padre debía haber una diferencia de más de un 10 % entre la energía remanente en los nodos y dicha situación no se dio nunca bajo las simulaciones con dicha topología. Sin embargo, en esa situación el protocolo debía haber utilizado el mismo nodo todo el tiempo, por lo que podemos concluir que quedaron detalles por pulir en la implementación de la histéresis.

Una vez sorteados los problemas que se presentaron o, en otras palabras, una vez finalizadas las pruebas en una topología adecuada, se podría explorar una combinación de ambas métricas de ruteo, para lo cual sería útil el MC indicando si envía ETX o NE.

Capítulo 6

Conclusiones

Mediante la ejecución del proyecto *LNN Energy Aware Routing* se logró un amplio entendimiento del ruteo RPL-ETX, en base a lo que se logró una implementación de energy-awareness routing mediante una nueva OF. Para esto debimos profundizar en las características del protocolo RPL, y en su implementación en Contiki, por lo que se considera que esta ejecución fue de gran utilidad para lograr la comprensión del protocolo implementado en las Redes de Sensores Inalámbricos.

Además, se puede concluir que el Contiki-OS tiene un gran potencial para el desarrollo de las WSN, ya que cuenta con una herramienta de gran utilidad como el *Energest* en su implementación actual, pero no se hace uso de ella. Se supone, entonces, la existencia de otras herramientas para la implementación de funcionalidades que mejoren el eficiencia y el funcionamiento de las WSN, tanto en la versión actual como en versiones futuras del Contiki.

Por último, parece ser claro que tanto el ruteo con métrica ETX, como el de métrica de energía, por sí solos presentan falencias que redundan en una reducción de las capacidades de las WSN. De igual forma, se puede prever una mayor eficiencia en las WSN en caso de contar con un ruteo que combine ambas métricas, y si bien no fue posible esta implementación en el marco de este proyecto, se considera que se avanzó en este sentido.

Capítulo 7

Anexo

7.1. RFC 6551

Este RFC apunta a establecer los lineamientos para determinar el mejor *root-path* desde un nodo de la red. El mejor camino es el que satisface todas las restricciones establecidas (si existe alguna) y tiene el menor costo con respecto a la o las métricas específicas.
El conjunto de métricas y restricciones de ruteo usadas por el desarrollo de RPL es mostrado a lo largo de DAG que es construído de acuerdo a la función objetivo, las métricas y restricciones de ruteo son publicadas en el DIO (DODAG Information Object).

7.1.1. Formato del contenedor de métricas DAG

Las métricas y restricciones de ruteo son transportadas dentro del objeto contenedor de métricas DAG definido en la RFC6550. Varias métricas o restricciones pueden estar presentes en el contenedor de métricas DAG y su uso para determinar el mejor camino puede estar definido por una función objetivo. El objeto representa una métrica o restricción de un tipo particular, pueden aparecer en cualquier orden en el contenedor de métricas DAG. El formato es el que se muestra en la figura 7.1.

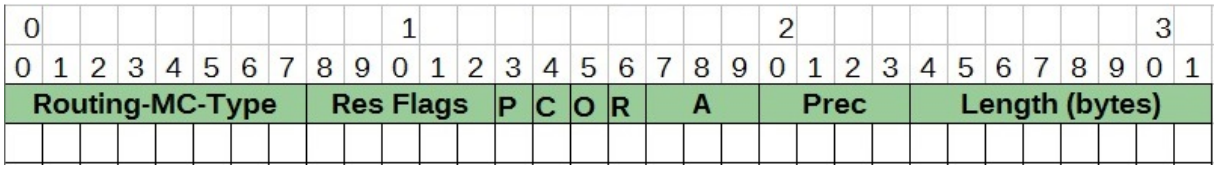

Figura 7.1: Formato genérico del DAG

El cuerpo del objeto transporta uno o varios subobjetos. El formato genérico de objeto que se muestra en la figura 7.1 está compuesto de:

- Routing-MC-Type, 8 bits: identifica la restricción o la métrica de ruteo y es definido por IANA.

- Length (largo), 8bits: define el largo del cuerpo del objeto. Puede variar de 0 a 255.
- Res Flags, 16 bits: el campo flags es manejado por IANA. Los bits no asignados son considerados reservados, tienen que ser transmitidos en cero e ignorados en la recepción.
- Flag P, 1 bit: es usado solamente para métricas grabadas. Cuando está en cero todos los nodos a lo largo del camino graban sucesivamente la métrica correspondiente. Cuando esta seteada indica que uno o varios nodos no pueden grabar la métrica de interés.
- Flag C, 1 bit: cuando está seteada indica que es una restricción de ruteo, cuando está en cero indica que es una métrica de ruteo.
- Flag R, 1 bit: sólo es relevante para métricas de ruteo, es decir para C=0. Cuando C=1, R=0. Cuando C=0 y R=1, indica que la métrica de ruteo es grabada a lo largo del camino. Cuando C=0 y R=0, la métrica de ruteo es agregada.
- Flag O, 1 bit: sólo se utiliza para restricciones de ruta, es decir cuando el flag C está seteado. Cuando el flag O está seteado indica que la restricción es opcional cuando está en cero indica que es mandatorio, si el flag C está en cero el flag O tiene que ser seteado en cero en la transmisión e ignorado en la recepción.
- Campo A, 3 bits: el campo A es relevante unicamente para métricas y es usado para indicar cuando una métrica de ruteo agregada es aditiva, multiplicativa, reporta un máximo, o reporta un mínimo.
  Si A=0, la métrica es aditiva.
  Si A=1, la métrica reporta un máximo.
  Si A=2, la métrica reporta un mínimo.
  Si A=3, la métrica es multiplicativa.
  El campo A no tiene significado cuando C=1 y sólo es válido cuando R=0. En cualquier otro caso el campo A tiene que ser configurado en cero en la transmisión y ser ignorado en la recepción.
- Prec, 4 bits: el campo Prec indica la precedencia de este objeto relativa a los otros objetos en el contenedor. Esto es útil cuando un contenedor de métricas DAG contiene varios objetos de métrica de rutas. Su valor varia de 0 a 15. El valor 0 indica la máxima precedencia.

7.1.2. Objetos: métricas/restricciones de rutas

Objeto energía de nodo (NE)

A veces es deseable evitar seleccionar nodos con baja energía residual. En esos casos el motor de protocolo de ruteo debe calcular un camino más largo para algún tráfico de forma de aumentar el tiempo de vida de la red. Potencia y energía son claramente recursos críticos en la mayoría de los LLN. La solución mas simple está basada en un campo de 2 bits que codifica 3 tipos de fuentes de energía:

- Powered
- Battery
- Scavengers

Este acercamiento simple puede ser suficiente para varias aplicaciones.

El objeto energía de nodo (NE) es usado para proveer información relacionada con la energía del nodo y puede ser usada como métrica o restricción.

El objeto NE puede estar presente en el contenedor de métricas DAG. No debe haber más de un objeto NE como restricción por contenedor de métrica DAG y no debe haber más de un objeto NE como métrica por contenedor de métrica DAG.
El tipo de objeto NE fue asignado por la IANA con valor igual 2. El formato del cuerpo del objeto NE se muestra en la figura 7.2.

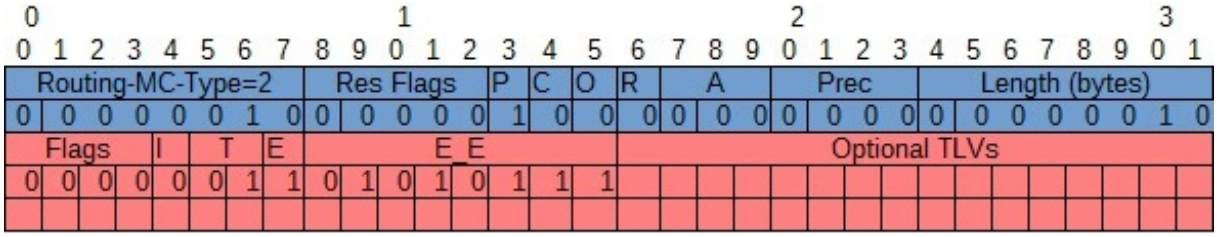

Figura 7.2: Formato NE (Node Energy) DAG

- Campo Flags, 8 bits: las banderas están definidas:
  - I (included), 1 bit: sólo es relevante cuando el tipo de nodo es usado como una restricción. El bit I es usado para estipular inclusión o exclusión. Cuando el Bit I=1 se indica que los nodos del tipo especificado en el campo "Node Type" deben ser incluidos. De forma opuesta cuando I=0 indica que el tipo de nodo especificado en el campo "Node Type" debe ser excluído.
  - T (node type), 2 bits: indica el tipo de nodo.
    T=0 energizados de la red,
    T=1 energizados por baterias y
    T=2 energizados por recolectores de energía.
  - E (Estimación), 1 bit: cuando está seteado para una métrica, la estimación del porcentaje de la energía restante del nodo es indicada en el campo E_E de 8 bits. Cuando E=0 no se provee el porcentaje de energía restante, cuando E está seteado para una restricción el campo E_E define el nivel para la inclusión o exclusión del nodo.
- E_E (Energía estimada), 8 bits: indica el porcentaje estimado de la energía remanente. El campo E_E es solamente relevante cuando la bandera E=1 y tiene que ser enviado en cero cuando E=0.

Si el objeto NE comprende varios sub-objetos cuando es utilizado como restricción, cada sub-objeto adiciona o quita un conjunto de nodos dado que los sub-objetos se procesan en orden.
No hay TLVs definidos actualmente, por lo que los sub-objetos NE pueden contener también un conjunto de TLVs usados para convenir varias características del nodo.

Confiabilidad de enlace

En LLN, la confiabilidad del enlace puede degradarse por varias razones:
- Atenuación de señal
- Varias formas de interferencia, etc.

Un cambio en la calidad del enlace puede afectar la conectividad de la red, es por ello que la calidad del enlace puede ser tomada en cuenta como una métrica de ruteo crítica. Varias métricas de confiabilidad de enlace pueden ser definidas reflejando varios aspectos, en este caso se estudia la confiabilidad de enlace con la métrica ETX.

Objeto de confiabilidad ETX
La métrica ETX es el número de transmisiones que un nodo espera realizar hacia un destino para entregar exitosamente un paquete. La métrica ETX provee un valor discreto (que puede no ser un entero) calculado de acuerdo a una fórmula específica.

El objeto ETX puede estar presente en el contenedor de métricas DAG. No debe haber más de un objeto ETX como restricción/métrica en el contenedor de métricas DAG.
El objeto ETX está formado de Sub-objetos ETX y debe al menos contener uno.
Cada sub-objeto ETX tiene un largo fijo de 16 bits.
El objeto ETX no contiene TLVs adicionales.
El tipo de objeto ETX fue asignado por la IANA con el valor 7.

El formato del objeto se muestra en la figura 7.3.

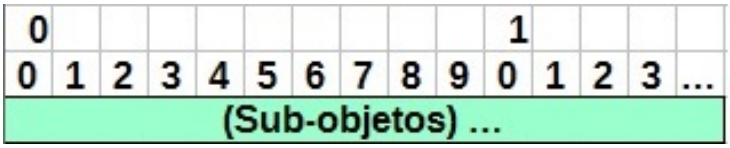

Figura 7.3: Formato ETX del DAG

7.2. IANA

Tipos de Métrica/Restricción de ruteo

La IANA[1] ha creado un subregistro, llamado *Routing Metric/Constraint Type*, para los tipos de objeto de Métrica/Restricción de ruteo, y cuyos posibles valores varían entre 0 y 255; estando el valor 0 sin asignar.

Value	Meaning
1	Node State and Attribute
2	**Node Energy**
3	Hop Count
4	Link Throughput
5	Link Latency
6	Link Quality Level
7	**Link ETX**
8	Link Color

Tabla 7.1: Valores asignados por la IANA para los tipos de métrica/restricción.

7.3. Archivos fuente

Conjuntamente con este documento se entregan todos los archivos y funciones modificadas para lograr implementar el ruteo RPL con métrica de energía. Estos archivos son:

***rpl-energy.c*:** Esta es la nueva OF implementada, cuya ruta en el sistema operativo es /contiki/core/net/rpl/.

***project-conf.h*:** Este es el archivo de configuración del proyecto donde se indica a qué OF apuntar para el ruteo.

***Makefile*:** Makefile del proyecto.

***unicast-example-rombo.csc*:** Esta es la simulación realizada para evaluar el funcionamiento del ruteo implementado.

[1]Internet Assigned Numbers Authority

Capítulo 8

Bibliografía

[1] N. Finne, N. Tsiftes, A. Dunkels, and J. Eriksson, "Powertrace: Network-level power profiling for low-power wireless networks."

[2] A. Junior and R. Sofia, "Energy-awareness metrics global applicability guidelines."

[3] P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "Rpl: Ipv6 routing protocol for low-power and lossy networks," 2012.

[4] K. Pister, N. Dejean, D. Barthel, E. JP. Vasseur, and E. M. Kim, "Routing metrics used for path calculation in low-power and lossy networks."

[5] L. Steinfeld, "Rpl: Routing protocol for low-power and lossy networks."