Tesis final de grado

Tráfico determinístico en Internet de las Cosas (IoT)

Autor

Czarnitzki Estrin, Ana - Padrón 96812

czarniana@gmail.com

Tutor: J. Ignacio Alvarez-Hamelin
Cotutor: Georgios Z. Papadopoulos

Índice general

Capítulo 1

Introducción

En este capítulo se presentará una pequeña introducción a los temas en los que está inmersa esta tesis. Se empezará por explicar qué es Internet y describir de forma resumida su historia, estructura y características. En segundo lugar se hablará de los fenómenos conocidos como Industria 4.0 e Internet de las cosas. Por último, se enunciarán las necesidades de la industria y de las llamadas industrias inteligentes en este contexto.

1.1. Internet

Internet es una red de redes de computadoras que conecta millones de dispositivos, llamados *hosts* o anfitriones, permitiendo que los mismos intercambien paquetes de comunicación [1]. De esta manera, la misma permite la transmisión de información y la colaboración e interacción de individuos (humanos y máquinas) independientemente de su localización geográfica, logrando la divulgación a nivel mundial e inter-planetaria de todo tipo de contenidos [2].

Esta tecnología no fue diseñada para una aplicación en particular o una funcionalidad en específico, sino como una infraestructura general en la que se podrían desarrollar aplicaciones. De esta manera, su influencia no sólo se encuentra en los aspectos técnicos de la informática y electrónica, sino también en la sociedad como un conjunto a medida que más y más actividades, recreativas y comerciales, se ven modificadas por este mecanismo. Por esta razón, desde su establecimiento, la investigación en esta área siempre se enfocó

tanto en el desarrollo técnológico como en los posibles usos de este desarrollo.

Cada una de las redes que conforman a Internet pueden representar instituciones públicas o privadas, gobiernos, negocios, empresas o casas de particulares, entre otras. Las mismas son independientes entre si y su topología interna es independiente de Internet, tampoco existe un control global y/o centralizado a nivel operacional ni administrativo. A su vez, los *hosts* que pertenecen a estan redes se encuentran conectados entre sí y hacia el resto de las redes por un conjunto de enlaces de comunicación y dispositivos. A la secuencia de dispositivos y enlaces que recorre un paquete para ir de un *host* a otro, es decir, para lograr que un contenido se comparta entre estos, se la denomina ruta o camino.

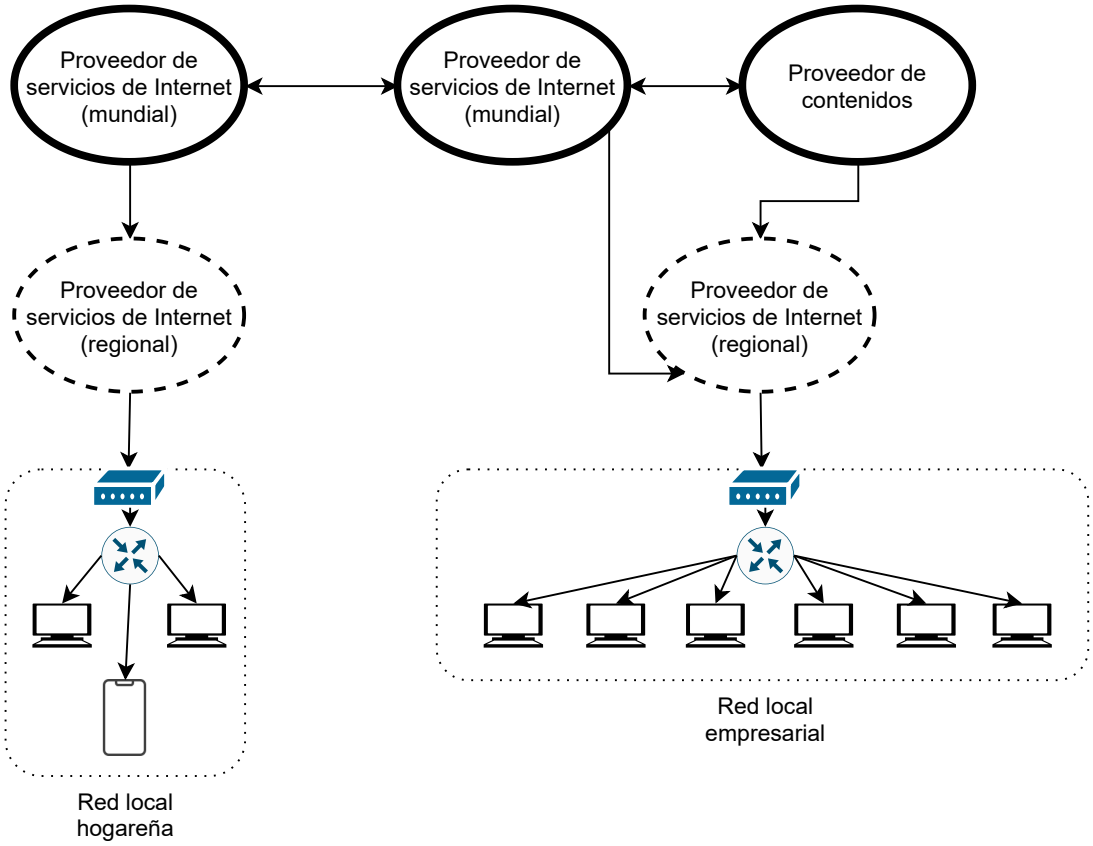

Figura 1.1: Vista simplificada de Internet

En la figura 1.1 se puede ver una vista simplificada de Internet. En la misma, las dos redes locales tienen acceso a diversos contenidos por medio de su conexión a algún proveedor de Internet. Todos los proveedores son, a su vez, redes.

Las diferentes piezas que son parte de Internet utilizan protocolos de comunicación que determinan como se envía y como se recibe la información. Dichos protocolos

definen el formato y el orden de los mensajes intercambiados entre dos o más entidades de comunicación además de las acciones que se toman al transmitir y/o al recibir un mensaje u otro evento. Para funcionar y realizar las tareas para las que fueron desarrollados, todos los protocolos de comunicación requieren que dos o más entidades los implementen y utilicen. Esto es, si alguien transmite es necesario que alguien reciba la información y viceversa.

Por otro lado, se debe destacar que Internet es un ejemplo exitoso de los beneficios de la financiación e inversión contínua, tanto en investigación como en maquinaria, para la infraestructura de la información, tanto por la industria, como los gobiernos y la academia, entre otros. Esta red no fue ni es desarrollada e implementada por una entidad u organización en particular, sino que es el resultado del trabajo cooperativo de múltiples entes de diferentes orígenes.

1.1.1. Arquitectura de capas

Los protocolos de Internet y el *hardware* y *software* asociados a estos se organizan en capas. Los mismos se conocen como la familia de protocolos de Internet.

5. Aplicación

4. Transporte

3. Red

2. Enlace

1. Física

Cada capa provee sus servicios ejecutando ciertas acciones dentro de la misma capa y utilizando los servicios de la capa inferior a esta. Por ejemplo, la capa de aplicación usará los servicios de la de transporte, la de transporte los de la de red y asi sucesivamente. Cada protocolo de cada capa se distribuye entre todos los componentes de Internet, usualmente hay una parte de al menos un protocolo de cada capa implementado en cada uno de estos componentes.

Capa 5: de Aplicación

La capa de aplicación es donde se encuentran las aplicaciones de Internet y sus protocolos. Ejemplos de estas aplicaciones pueden ser desde diferentes servicios de correo digital hasta redes sociales, como *Facebook* o *Twitter*.

Capa 4: de Transporte

La capa de transporte le provee servicios de comunicación a las aplicaciones que corren en los distintos *hosts*, logrando que, desde la perspectiva de la aplicación, los *hosts* que corren sus procesos estén conectados de forma dirécta. Las aplicaciones utilizan estos servicios para enviarse mensajes entre ellas sin preocuparse de cómo los mismos se envían. Los protocolos de transporte se implementan en los *hosts* y no en los dispositivos intermedios de Internet.

Se cuenta con dos protocolos principales para esta capa, *Transmission Control Protocol* (TCP, Protocolo de Control de Transmisión) y *User Datagram Protocol* (UDP, Protocolo de Datagramas de Usuario).

Capa 3: de Red

La capa de red provee comunicación lógica entre *hosts*, moviendo los paquetes de un *host* emisor a un *host* receptor, sin diferenciar a que aplicación en particular está dirigido cada paquete. Esta capa tiene la función de definir el camino que tomará un paquete de un *host* a otro además de determinar que enlace deberá utilizar cada paquete en cada "salto" en este camino.

La capa de red de Internet provee un servicio de tipo *best effort* o de mejor esfuerzo. Esto implica que no garantiza mantener el orden de entrega de los paquetes ni respetar un tiempo de entrega para los mismos o garantizar su correcta llegada.

Dentro de esta capa se encuentra el protocolo que le da nombre a Internet, *The Internet Protocol* (IP o el Protocolo de Internet), siendo IPv4 la versión más extendida y utilizada e IPv6 la nueva versión del protocolo que aún se encuentra en adopción. Esta segunda versión agrega mejoras y un mayor espacio de identificadores para los dispositivos que se encuentra conectados a Internet (direcciones IP).

Capa 2: de Enlace

El servicio básico que provee la capa de enlace es transportar un datagrama (siendo un datagrama el mínimo bloque de información transmitible en una red) de un nodo a otro nodo adyacente sobre un único enlace de comunicación o único salto. Sus servicios incluyen definir las reglas por las cuales un datagrama se transmite por un medio y la deteccion y corrección de errores en las transmisiones del mismo. La mayor parte de la implementación de la capa de enlace se encuentra en el *hardware*.

Capa 1: Física

La capa física se encarga de mover los bits individuales de cada mensaje de un nodo a otro. Los protocolos de esta capa dependen de los enlaces y del medio de transmisión que utilicen los mismos.

1.1.2. Redes inalámbricas

Las redes de Internet se pueden diferenciar entre aquellas que utilizan enlaces cableados y aquellas que utilizan enlaces inalámbricos. De esta manera, estos dos tipos de redes se diferencian especialmente por las características de sus enlaces. Un enlace inalámbrico, a diferencia de uno cableado, presenta caida en la intensidad de la señal, interferencia con otras fuentes y propagación en múltiples caminos. Debido a estas razones, los errores en los *bits* de los mensajes que circulan por los distintos enlaces son mucho más comunes en este tipo de redes.

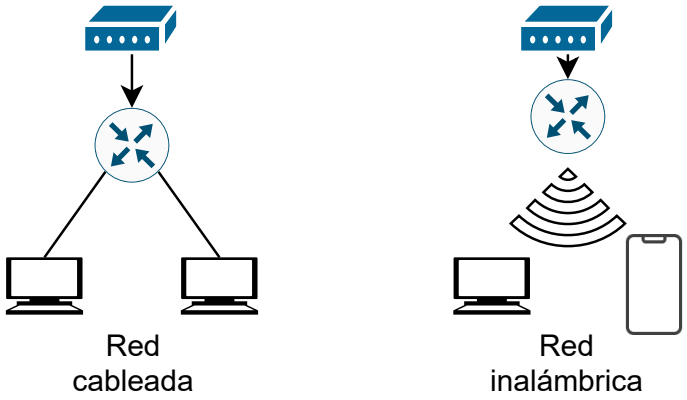

Figura 1.2: Ejemplo de una red cableada e inalámbrica.

En la figura 1.2 se tiene un ejemplo de una red cableada y uno de una inalámbrica. Se puede observar que en el primer ejemplo cada dispositivo tiene su propio cable mientras que en la segunda el medio es compartido. En el primer caso, frente a una transmisión, solo el dispositivo a quien se dirija el mensaje recibirá este, por su cable, sin embargo, en el segundo caso, cualquier dispositivo que esté en rango de escucha del receptor podrá escuchar el mensaje también.

Para evitar la degradación en las transmisiones, los protocolos de acceso al medio que utilizan redes con enlaces inalámbricos suelen implementar esquemas de evitación de errores que incluyen *acknowledges* (ACK, confirmaciones de recepción) y retransmisiones en la capa de red. En estos mecanismos, si luego de un determinado tiempo no se recibe un ACK, se enviará una retransmisión. No se implementan, en cambio, detección de errores dado que la habilidad de detectar colisiones es costosa y, además, no resulta posible detectar todas las pérdidas.

1.2. Industria 4.0

La industria es el área de la economía que produce bienes materiales que se encuentra altamente mecanizada y automatizada. El fenómeno de la Industria 4.0 fue mencionado por primera vez en el 2011 en Alemania como una propuesta de nuevas políticas económicas para este país basadas en el desarrollo de altas tecnologías. El término fue establecido bajo la idea de una "cuarta revolución industrial" y hace referencia a las formas del versionado de *software*.

Esta nueva revolución industrial se encontraría marcada por la automatización total y digitalización de los procesos productivos (utilización de sistemas informáticos para manejar y planear la producción) y el uso de maquinaria electrónica y tecnologías de la información en estos. En este tipo de producción los mismos productos controlan su proceso de manufactura de forma descentralizada. Se debe aclarar, además, que la industria 4.0 aún se encuentra en un estadío conceptual y se basa en tecnologías muy dinámicas que involucran a múltiples industrias, entre ellas: ingeniería eléctrica, ingeniería mecánica, ingeniería de los sistemas de comunicación, administración de negocios y ciencias de

la computación [3],[4].

La industria 4.0 se basa en la computación móvil (uso de computadoras que no se encuentran en una red cableada), *cloud computing* (computación en la nube, paradigma que proporcia servicios informáticos a través de una red, en general Internet) y *big data* (datos masivos, refiere a cantidades de información grandes y complejos que no pueden ser procesados por los medios tradicionales de la informática). Dentro de este fenómeno, hay una tendencia hacia el establecimiento de un canal de comunicación para el intercambio contínuo de information sobre las necesidades y entornos individuales y particulares de cada individuo hacia vendedores, médicos, proveedores de energía y todos los otros grupos que puedan responder a estas necesidades.

De esta manera, las máquinas envían datos mediante sensores *wireless* hacia servicios inteligentes donde estos grandes flujos de información son analizadas. El objetivo de esta automatización es la customización de los productos y servicios para cada cliente de manera que se incremente el valor agregado para las organizaciones y para los mismos clientes. Según esta idea, se forman *smart factories* (industrias inteligentes) donde el proceso productivo se encontrará íntegramente equipado por sensores, actores y sistemas autónomos.

Por otro lado, la Industria 4.0 no se limita sencillamente a la automatización de la producción, sino que incluye la digitalización de todo un negocio, involucra la procuración de materiales y los procesos de introducción del producto a su desarrollo y de como ese mismo producto terminado llegando a cada uno de sus clientes.

1.3. Internet de las cosas

Internet of Things (IoT, Internet de las Cosas) refiere a la conectividad de los objetos cotidianos y tangibles del mundo material entre ellos mismos. La reducción del tamaño, peso, consumo de energía y precio de los transmisores trajo consigo una nueva era donde casi cualquier objeto puede estar conectado a Internet. La IoT no es una tecnología única, sino que es un concepto en el cual se le permite a estos objetos o "cosas" que interactuen y se coordinen entre ellos, compartiendo su información y tomando decisiones en conjunto,

de manera de reducir la intervención humana y dar información del estado de cada objeto en las tareas de todos los días. Entonces, una definición posible para la IoT podría ser: "Una red abierta de objetos inteligentes que tienen la capacidad de auto-organizarse, compartir información, datos y recursos, reaccionando y actuando debido a diversas situaciones y cambios en su ambiente" [5], [6].

El posible primer ejemplo de uso de IoT es de principios de los 1980s, en la universidad de Carnegie Melon. En esta universidad, un grupo de programadores conectó una máquina que vendía gaseosas de manera de obtener información sobre cuándo había sido la última vez que la misma había sido abastecida. De esta manera, los progradores podían conectarse a la red, ver el estado de la máquina y en base al mismo decidir si bajar a comprar una gaseosa o esperar al próximo re-abastecimiento. Además de este ejemplo, hay otros múltiples usos de la IoT, siendo algunos de los destacados los que se enuncian a continuación:

- Infraestructura inteligente: se incorporan dispositivos inteligentes en edificios y construcciones. Los mismos pueden mejorar la flexibilidad, fiabilidad y eficiencia en las operaciones de infraestructura.
- Salud: diferentes sensores monitorean a los pacientes y envían esta información a sus médicos.
- Cadenas de producción y logística: la IoT puede mejorar a estos procesos proprorcionando información más detallada y actualizada, reduciendo errores de predicciones sobre el consumo y mejorando la trazabilidad.

Los usos de la IoT se basan en entornos donde la demanda de recursos es dinámica y donde se tienen aplicaciones de tipo *real time* (aplicaciones de tiempo real, aplicaciones en las que se tienen restricciones fijas y bien definidad respecto al tiempo que deben tardar ciertas tareas o respuestas). Debido a que la comunicación entre estos dispositivos y los servicios que los mismos emplean de Internet puede ocurrir en cualquier momento y lugar, sus conexiones suelen ser inalámbricas y de tipo *ad hoc* (conexiones temporales y automáticas que se generan para un fin específico, sin intervención de un coordinador

central para que la misma sea establecida). De esta manera, los servicios que estos emplean resultan más dinámicos, descentralizados y complejos.

Por otro lado, otra de las características principales de la IoT es la enorme cantidad de información que se transmite en sus redes (*big data*) hacia computadoras remotas para su análisis (*cloud computing*).

1.4. Necesidades de la industria e industrias inteligentes

La industria manufacturera se encuentra en un proceso de evolución hacia una organización distribuida de la producción con productos, equipamientos y logísticas inteligentes y conectadas entre si y con Internet. A pesar de que la IoT no es aún una parte considerable del proceso productivo, se espera una gran adopción de la misma en los próximos años, mayormente para mejorar la eficiencia operacional y la productividad. Por esta razón, se considera a esta rama el sector con mayor crecimiento potencial referido a estas tecnologías, incluso por sobre el de las llamadas ciudades inteligentes y los sistemas de transporte [7].

Tradicionalmente, las automatizaciones industriales se realizaban con comunicaciones cableadas, sin embargo, este tipo de sistemas son costosos debido a la necesidad de instalar estos cables y de mantenerlos. Con los recientes avances en redes de sensores inalámbricos, se volvió posible el utilizar este tipo de redes en entornos industriales. Estos sensores dan lugar a sistemas donde estos se instalan en equipos industriales y monitorean los parámetros críticos de los mismos, esta información se transmite para luego ser analizada de manera de tomar decisiones inteligentes y evitar fallas en la producción. [8]

Entre las ventajas que tienen estas redes inalámbricas sobre las cableadas tradicionales se encuentran la auto-organización, el rápido despliegue y su fléxibilidad. Las redes wireless en la industria permiten abaratar los costos al no necesitar cableado además de posibilitar casos de uso más amplias, por ejemplo el de dispositivos que rotan, son portables o tienen movimientos rápidos que un cable entorpecería [9].

Hoy en día, hay una enorme inversión en IoT para solucionar problemas industriales, se suele mencionar a esta rama de la IoT como IIoT (*Industrial IoT*, IoT Industrial).

IIoT refiere a los objetos industriales que contienen sensores y se comunican de forma automática sobre una red sin ninguna interacción entre humanos o entre humanos y computadoras para intercambiar información y tomar decisiones inteligentes con la ayuda de sistemas avanzados.

A pesar de estas ventajas, la implementación de redes inalámbricas para la industria tiene diversos desafíos, entre ellos se pueden destacar los siguientes:

- Limitaciones en recursos: los dispositivos IoT están limitados en su energía, memoria y capacidad de procesamiento.
- Topologías dinámicas y condiciones ambientales duras: la topología y conectividad de la red puede variar debido a errores y fallas en los enlaces de la misma o en los nodos de esta. Además, los dispositivos pueden estar expuestos a interferencia, agentes corrosivos, humedad, vibraciones, polvo y otras condiciones adversas, generando funcionamiento defectuoso.
- Requerimientos en la calidad de servicios: la mayoría de los sensores producen información que es sensible al tiempo, si la misma se recibe tarde puede conducir a malas decisiones.
- Despliegues a gran escala y arquitecturas *ad hoc*: la mayoría de estos sistemas se componen de un gran número de dispositivos que pueden estar dispersos de formas aleatorias en una planta productiva, al no haber una infraestructura de red predeterminada, se necesita que estos dispositivos pueden establecer conecciones y mantener la conectividad de la red de forma autónoma. Por otro lado, al tener despliegues de gran escala, aumentos o disminuciones pequeños en los costos de cada uno de los dispositivos de los mismos pueden dar lugar a aumentos o disminuciones grandes del costo total de la infraestructura.
- Integración con Internet: para poder brindar servicios via la información obtenida de los dispositivos y a su vez poder controlarlos, es necesarios que estas redes se encuentren conectadas con Internet.

Por otro lado, con estos desafíos en mente, se tienen a su vez los siguientes objetivos de diseño para las redes de dispositivos IoT en la industria:

- Arquitecturas escalables y protocolos eficientes: dado que se buscan soportar aplicaciones industriales heterogéneas con requerimientos diversos, es necesario desarrollar arquitecturas flexibles y escalables que puedan usarse para todas estas distintas aplicaciones. A su vez, es necesario que las mismas sean interoperables con soluciones de tipo *legacy* (tecnologías anticuadas que se encuentran aún en uso) existentes.
- Uso de recursos eficiente: esta característica es importante para maximizar el tiempo de vida útil de la red mientras, a su vez, se cumplen los requerimientos de calidad de servicio de la misma. Para esto se requieren protocolos que utilicen los recursos de forma eficiente y sepan de estas limitaciones. En estos dispositivos, suele ser crítico el diseño de sistemas operativos que compensen y equilibren los requerimientos de calidad de servicio con el uso de recursos.
- Auto-configuración y auto-organización: se busca que las redes pueden repararse solas ante fallos y que agregar nuevos nodos y nuevos dispositivos a la red no requiera modificar la misma de manera manual por un operario. Además, se quiere brindar rapidez a la instalación de nuevos dispositivos o al quitar dispositivos viejos.
- Tolerancia a fallas y confiabilidad: la información debe poder transmitirse de forma correcta a pesar de que ciertos nodos o enlaces de la red fallen. Se debe tener en cuenta que estos errores no solo ocurren por un alto *Bit Error Rate* (BER) (Tasa de error por Bit) en los mensajes recibidos sino también por resultar los nodos y/o enlaces inaccesibles durante tiempos prolongados. De esta manera, son necesarios mecanismos de corrección de la información y de auto-recuperación.

Por otro lado, uno de los principales obstáculos en la adopción de tecnologías inalámbricas en la industria es la baja confiabilidad que se asocia a las mismas comparándolas a tecnologías cableadas. Ademas, se tiene incerteza respecto al mantenimiento y soporte a largo plazo de estas nuevas tecnologías. Este sector es especialmente susceptible a la confiabilidad, teniendo líneas de producción altamente optimizadas y donde la necesidad de parar la misma conllevaría a grandes pérdidas.

De esta manera, no sólo resulta importante que la comunicación sea exitosa con una alta probabilidada sino que, ante fallas, el sistema pueda recuperarse de forma rápida. Sin embargo, los protocolos existentes que buscan determinismo lo hacen muchas veces a costa de una banda de ancha limitada debido a las limitaciones energéticas de los dispositivos IoT.

Capítulo 2

Estado del arte

En este capítulo se presentarán las diversas propuestas que al día de hoy están vigentes para tratar el desafío que busca solucionar esta misma tesis, esto es, ofrecer confiabilidad en entornos de redes IoT inalámbricas. En específico, se trataran soluciones basadas en caminos múltiples y/o la introducción de copias, en el uso de *schedules* inteligentes, entre otras alternativas.

2.1. Caminos múltiples

Las propuestas basadas en caminos múltiples buscan aumentar la confiabilidad de una red agregando oportunidades para que cada paquete que se transmite en la misma llegue de forma correcta a su destino. Para esto, en vez de enviar cada paquete una vez por un único camino, se generan copias de los mismos de manera de que cada copia atraviese un camino diferente. Estos caminos pueden o no tener tener nodos en común.

Los algoritmos mencionados en esta sección se diferencian unos de otros según los patrones de caminos múltiples que utilizan, cómo los mismos son elegidos, la cantidad de copias que se envian y como las mismas se generan, entre otros aspectos.

2.1.1. Patrones Disjunto, Triangular y Trenzado

Minet, Khoufi y Laouiti [10] presentan tres patrones de redundancia para generar caminos confiables de un origen a un destino. Estos tres patrones son llamados *Disjoint*

(Disjunto), *Triangular* (Triangular) y *Braided* (Trenzado). A su vez, los patrones presentados sirven para clasificar a todas las otras propuestas que se basan en caminos múltiples, incluyendo la de esta tesis.

El problema que se busca abordar es el siguiente: asumiendo que existe un camino principal entre un origen S y un destino D, se quiere aumentar la confiabilidad de las transmisiones entre S y D manteniendo un tiempo de entrega de paquetes corto y minimizando el consumo de recursos de la red. Para probar estos tres patrones, se basan en los protocolos ya existentes TSCH y RPL, estos mismos protocolos son utilizados en esta tesis y puede encontrarse una definición más extensa de los mismos en el capítulo 3.

Los autores expresan que luego de configurar a los protocolos de base de la mejor forma posible (esto es, utilizando las prácticas recomendadas), la única manera de mejorar la confiabilidad de la red es introduciendo redundancia ya sea por un método de enmascaramiento o un método de detección y recuperación frente a errores. En los métodos de enmascaramiento, los errores se "enmascaran" si la cantidad de los mismos es menor o igual a la máxima cantidad tolerada. Por otro lado, en los métodos de detección y recuperación frente a errores, se procesa y arregla un error si se detecta que alguno existe. Los métodos presentados a continuación son todos de tipo enmascaramiento.

Patrón Disjunto

Este primer patrón consiste en generar un camino alternativo al camino principal entre S y D de manera que estos dos caminos no compartan ningún nodo intermedio. De esta manera, se tolera una falla de cualquier nodo de la red, salvo de S y de D. Para esto, el origen envia una copia de su mensaje en el camino principal y otra en el camino secundario. Si no hay fallas, D recibe dos copias, quedándose con la primera que llegue y descartando la segunda.

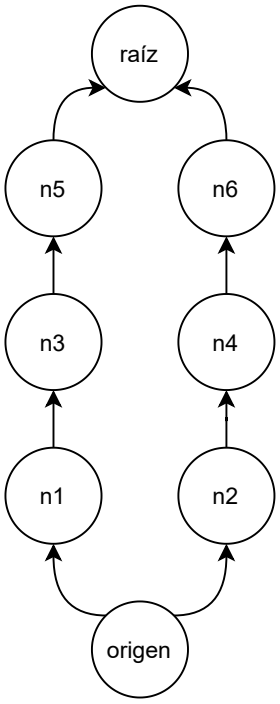

Figura 2.1: Ejemplo de patrón disjunto.

En la figura 2.1 se puede observar un ejemplo de este patrón para una topología con seis nodos intermedios, un origen (denominada antes S) y una raíz o destino (denominada antes D). En este caso se tienen dos caminos disjuntos, formados por $n1$, $n3$ y $n5$, por un lado, y $n2$, $n4$ y $n6$, por otro.

La principal ventaja de esta solución es el hecho de que solo el origen y el destino deben manejar la redundancia mientras que el resto de los nodos intermedios no ven modificado su comportamiento.

Patrón Triangular

El patrón básico se forma por un nodo N, el padre predeterminado de N o $dp(N)$ (*default parent*), el padre alternativo de N o $ap(N)$ (*alternative parent*) y el padre predeterminado del padre predeterminado de N (el "abuelo" de N) o $dgp(N)$ (*default grand parent*). Tanto $dp(N)$ como $ap(N)$ se encuentran conectados con $dgp(N)$. De esta manera, si ocurre una falla en el camino principal, es decir, entre N y $pp(N)$, se puede utilizar el camino alternativo entre N y $ap(N)$.

Todos los nodos en el camino principal, salvo el destino, tienen un padre pre-

determinado y un padre alternativo. Los padres alternativos de cada nodo tienen, a su vez, como padre al padre del padre predeterminado de su nodo hijo. En este patrón, la transmisión ocurre de la siguiente manera: cada nodo en el camino principal manda sus mensajes tanto a su padre predeterminado como a su padre alternativo mientras que los padres alternativos solo mandan sus paquetes a su único padre.

De esta manera, a diferencia del patrón anterior, todos los nodos intermedios del camino principal deberán seleccionar a sus padres predeterminados y a sus padres alternativos.

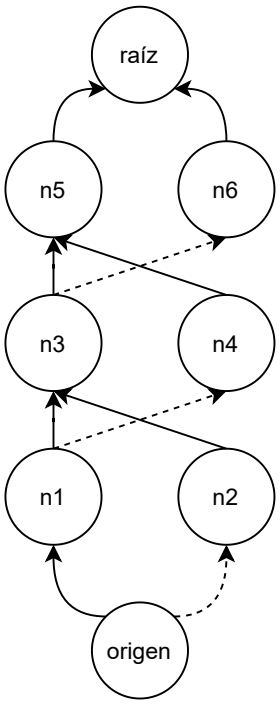

Figura 2.2: Ejemplo de patrón triangular donde las líneas enteras denotan a los padres predeterminados y las punteadas a los alternativos de cada nodo.

En la figura 2.2 se puede observar un ejemplo de este patrón para una topología con seis nodos intermedios, un origen y una raíz o destino. En este ejemplo, el camino principal es el formado por los nodos $n1$, $n3$ y $n5$, a su vez, $n2$, $n4$ y $n6$ son los padres alternativos de los nodos origen, $n1$ y $n3$, respectivamente. Siguiendo la nomenclatura inicial, si el origen es N, entonces $dp(N)$ será $n1$, $ap(N)$ será $n2$ y $dgp(N)$ será $n3$, notar que el padre de $n1$ y $n2$ es para ambos $dpg(N)$. Si hubiera una falla en $n1$, $n2$ reenviaría su paquete a $n3$ de todos modos, pudiendo continuar con la transmisión. Como se mencionó

anteriormente, los padres alternativos (en este caso, $n2$, $n4$ y $n6$) tienen un único padre.

Patrón Trenzado

El patrón Trenzado es una extensión del Triangular en donde se forma un nuevo camino alternativo con los padres alternativos como nodos intermedios. También puede verse como una extensión del Disjunto donde los caminos están conectados por múltiples enlaces, siendo posible ir de un camino al otro.

La transmisión se realiza de forma similar a la del patrón Triangular. Cada nodo n en el camino principal, salvo el destino, seleccionan como padre alternativo a un nodo entre sus padres posibles que tiene como padre posible al "abuelo" predeterminado de n (esto es, tiene como padre al padre predeterminado del padre predeterminado de n). Además de esto, se tiene una condición adicional, los padres alternativos de los nodos en el camino principal deben formar un camino alternativo. El origen y los nodos intermedios envian sus paquetes dos veces, una vez al padre predeterminado y otra al alternativo. A su vez, en cada nivel, cada nodo reenvía una vez los paquetes recibidos a cada uno de sus padres, independientemente de si recibió más copias. Las copias extras son descartadas.

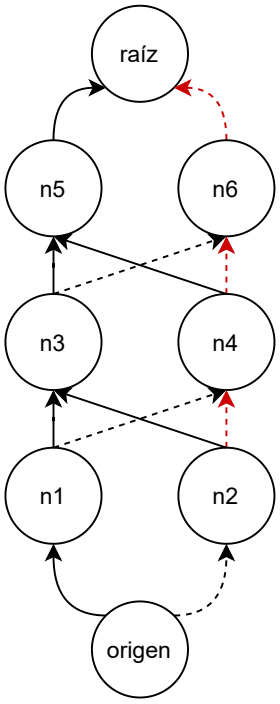

Figura 2.3: Ejemplo de patrón trenzado donde ls líneas enteras denotan a los padres predeterminados, las punteadas negras a los alternativos de los nodos del camino principal y las punteadas bordó a los alternativos de los nodos del camino alternativo.

En la figura 2.2 se puede observar un ejemplo de este patrón para una topología con seis nodos intermedios, un origen y una raíz o destino. En este ejemplo, el camino principal está formado por los nodos $n1$, $n3$ y $n5$ mientras que el alternativo se forma con $n2$, $n4$ y $n6$.

Comparación entre los patrones

Respecto a confiabilidad, los tres patrones solo pueden tolerar o una falla en un nodo o una falla en un enlace. Sin embargo, la confiabilidad que otorgan los tres patrones es distinta. Para calcular este valor se hicieron cálculos teóricos y simulaciones, los resultados obtenidos dan mejores valores para el patrón Trenzado, seguido por el Triangular y por último el Disjunto. Se debe tener en cuenta que los tres patrones superan a los protocolos base (TSCH y RPL, explicados en mejor detalle en el capítulo 3).

Respecto a la cantidad de nodos utilizados y a la cantidad de transmisiones realizadas en cada patrón, los valores son mayores para el Trenzado, seguido por el Triangular y por último el Disjunto. De esta manera, estos valores se ven reflejados en que

el Trenzado consume más energía que el Triangular y ambos consumen más energía que el Disjunto.

Por último, los tiempos de entrega para los tres patrones son similares. Esto se debe a que los tres se basan en estrategias de tipo enmascaramiento, donde no se espera a detectar un error para corregirlo. En estas propuestas, es indistinto si llega el paquete del camino original o el del alternativo, ambos llegarán a la vez.

Por último, en base a estos valores obtenidos, los autores concluyen que el aumento en la confiabilidad se logra en base a un mayor gasto de recursos, específicamente de energía. Dependiendo cuales sean las limitaciones e imposiciones que busquen cumplirse, se puede optar por usar uno u otro patrón. Por ejemplo, si la prioridad es tener una confiabilidad muy alta, se puede utilizar el patrón Trenzado, en cambio, si se busca mejorarla sin aumentar mucho el uso de energía, el patrón Disjunto será una mejor opción.

2.1.2. PAREO

Koutsiamanis, Papadopoulos, Jenschke, Thubert y Montavont [11] presentan las funciones de *Packet Automatic Repeat reQuest* (solicitud de repeteción automática de envío de paquetes o ARQ, por sus siglas en inglés), *Replication and Elimination* (replicación y eliminación o RE, por sus siglas en inglés) y Overhearing (sobre-escuchar o OH, por sus siglas en inglés) con la intención de mejorar la calidad de servicio y la predictivilidad en redes industriales al ser implementadas sobre redes de tipo de mejor esfuerzo. Estas tres funciones usadas en conjunto se denominan PAREO por la contracción de sus acrónimos en inglés.

Automatic Repeat reQuest (ARQ)

La función ARQ realiza la retransmisión de paquetes de datos cuando una transmisión previa falla. En este caso, la retransmisión se realiza en la capa de enlace, de manera que la decisión de re-transmitir recae en el nodo emisor. Esta función necesita el uso de paquetes de control de tipo *acknowledgment* (ACK, de reconocimiento) para cada transmisión.

La lógica de ARQ es la siguiente:

- Se envia un paquete y se asigna un tiempo corto para esperar a recibir un ACK del envio del mismo.
- Si se recibe el ACK en ese tiempo, la transmisión se marca como exitosa y termina, sino, la transmisión se marca como temporalmente fallida y es programada para realizarse de nuevo.
- Cada vez que una transmisión falla, se aumenta un contador asignado a ese paquete de datos, cuando ese contador llega a un umbral predefinido, el paquete se marca como permanentemente fallido y no es programado para enviar de nuevo.

El algoritmo de *scheduling* (es decir, aquel que asigna en qué momento y en qué canal cada nodo podrá transmitir o deberá estar recibiendo transmisiones) que se utilize modificará considerablemente los tiempos de las retransmisiones, pudiendo estas realizarse inmediatamente después de ocurrir un error en la transmisión anterior o mucho después del mismo.

Replication and Elimination (RE)

La función de replicación de paquetes modifica el envio de estos de manera de enviarlos no solo a un padre preferido sino a otros posibles padres de cada nodo. Es decir, esta función permite el uso de caminos múltiples de manera de mantener la conectividad de la red a pesar de que la misma tenga nodos inaccesibles. Para esto, los paquetes son clonados en cada nodo y se envian a un padre preferido y a uno alternativo. De esta manera, cada copia atraviesa la red de forma independiente.

Con el objetivo de evitar que se genere un fenómeno de *flooding* (inundación) debido a la replicación se implementa la función de eliminación de paquetes. En este fenómeno cada nodo de la red transmite paquetes a todos o a casi todos sus padres posibles, generando congestiones en la misma. Para que no ocurra lo mencionado anteriormente, esta función usa identificadores para todos los paquetes de manera de poder reconocer a las copias de cada uno. Estos identificadores se copian en memorias de tipo caché en cada nodo. Cuando un paquete llega a un nodo, se compara su identificador con aquellos ya presentes en memoria, si el mismo se encuentra, significa que ya fue enviado y el paquete

se elimina sin ser transmitido, en caso contrario, se guarda su identificador en la caché y se transmite.

Para poder enviar las copias a los padres alternativos de cada nodo, el *schedule* utilizado debe tener asignados tiempos y canales para estas transmisiones.

Overhearing (OH)

La función de sobre-escucha se puede usar para aprovechar la naturaleza compartida del medio de transmisión inalámbrico de manera de aumentar la confiabilidad. Esta función permite a un nodo simultáneamente enviar un paquete a múltiples padres que se asume se encuentran en rango de recepción. Si no se utilizara esta función, las transmisiones solo se realizarían una después de la otra según el *schedule* definido.

La sobre-escucha cobra sentido solo cuando se usa en conjunto con la función de replicación, permitiendo agregar posibilidades de transmitir de forma correcta el paquete sin agregar latencia. Esto es, la transmisión del paquete que se envia al padre preferido, debido al medio compartido, es escuchada también por el padre alternativo del nodo y viceversa. Se debe mencionar que solo el receptor del paquete realiza el ACK por el mismo y solo el padre preferido y el alternativo de cada nodo re-transmiten los paquetes a pesar de que otros nodos también pueden sobre-escuchar esta transmisión.

Elección del padre alternativo

Jenschke, Papadopoulos, Koutsiamanis y Montavont [12] proponen tres algoritmos distintos para elegir al padre alternativo que se utilizará con PAREO.

Estos tres algoritmos son *Strict CA* (CA Estricto), *Medium CA* (CA Mediano) y *Soft CA* (CA relajado o flexible), donde CA es un acrónimo de *Common Ancestor* (Ancestro Común). Los nombres de estos métodos refieren a cuán estricto es su criterio de selección del padre alternativo.

Considerando v, candidato a padre alternativo de un nodo u, las siguientes condiciones deben cumplirse para que v sea el padre alternativo de u:

- *Strict CA*: el padre preferido del padre preferido de u es el padre preferido de v.

- *Medium CA*: el padre preferido del padre preferido de *u* está entre los padres posibles de *v*, sin ser necesariamente su padre preferido.
- *Soft CA*: *u* y *v* tienen padres en común, pudiendo los mismos ser o no los preferidos o alternativos.

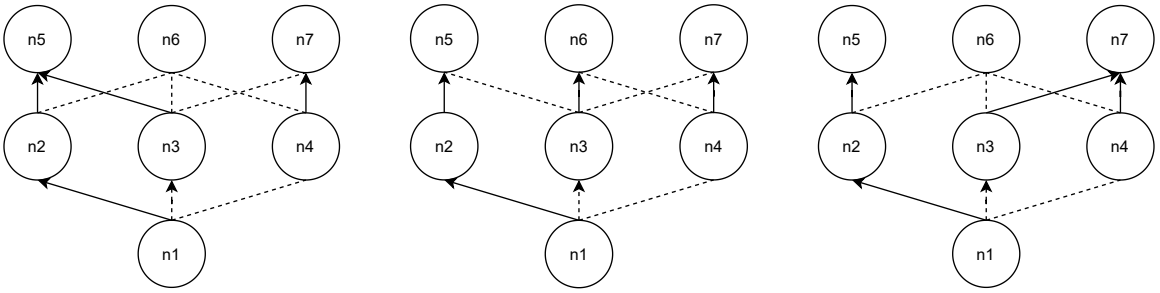

Figura 2.4: Ejemplo de los algoritmos *Strict* (en la izquierda), *Medium* (en el centro) y *Soft* (en la derecha) *CA*

En la figura 2.4 se pueden observar ejemplos para los tres algoritmos, las flechas enteras marcan a los padres preferidos de los nodos, las flechas punteadas a los alternativos y las líneas punteadas a los posibles. En los tres ejemplos, $n1$ es el nodo que se está analizando, teniendo el mismo a $n2$ como padre preferido y a $n3$ como padre alternativo. En el primer caso, en la imagen de la izquierda que refiere a *Strict CA*, tanto $n2$ como $n3$ tienen de padre preferido a $n5$. El segundo caso, en la imagen del medio que refiere a *Medium CA*, $n5$ es el padre preferido de $n2$ y este mismo se encuentra entre los padres posibles de $n3$, sin embargo, su padre preferido es otro, en específico $n6$. Por último, en la imagen de la derecha que refiere a *Soft CA*, el padre de $n2$ es $n5$ y el de $n3$ es $n7$, se puede observar que $n5$ y $n7$ comparten como padre posible a $n6$.

Evaluación de la propuesta PAREO

PAREO obtiene valores altos de entregas exitosas de paquetes con baja demora en los tiempos de entregas de los mismos, sin embargo, tiene un alto consumo de energía. En cuanto a los algoritmos para elección de padre alternativo, los más estrictos generan un menor consumo de energía y una menor cantidad de copias de los paquetes en la red, pero, a su vez, tienen menores valores de confiabilidad debido a que la probabilidad de

que se tenga un padre alternativo siguiendo sus criterios es menor, generando que menos nodos tengan un segundo camino a utilizar para enviar sus paquetes.

2.2. Scheduling

Ahrar, Nassiri y Theoleyre [13] proponen el uso de un algoritmo de *scheduling* para algoritmos de caminos múltiples de tipo trenzado como los introducidos en la sección 2.1.1. En este algoritmo, múltiples transmisores utilizan una misma *celula* (un par canal, rango de tiempo) asignada a un único receptor. Es decir, múltiples nodos pueden transmitir al mismo receptor en el mismo instante, en un mismo canal. Este *schedule* se construye de manera que solo un transmisor esté activo a la vez, evitando las posibles colisiones entre transmisiones. Al compartir celdas bajo este esquema, se reduce el consumo de energía y la capacidad de la red, generando además un *schedule* compacto. A su vez, la construcción de caminos se realiza de manera de maximizar la cantidad de nodos en común que comparten los mismos, pudiendo más nodos compartir celdas.

En su trabajo, el *schedule* no fuerza a los nodos a mantenerse despiertos para escuchar retransmisiones. Además, los paquetes no se encuentran replicados, se envía un único paquete alternativo de forma automática solo cuando la transmisión original falla. En este caso, se realiza una transmisión alternativa a un camino secundario en vez de retransmitir al original.

El objetivo de este enfoque es evitar agregar celdas consecutivas para cada padre alternativo. Estas celdas agregan redundancia (dado que solo es necesario que un paquete y no todas sus réplicas lleguen al destino de forma correcta, de manera que en la práctica solo una celda sería necesaria), lo que implica que los nodos deben estar más tiempo despiertos, consumiendo energía. Además, agregan demoras en los tiempos de entrega de paquetes dado que al ser el *schedule* más largo, cada nodo deberá esperar más para transmitir.

2.2.1. Algoritmo

El primer paso de este algoritmo es el de construir los caminos trenzados. De esta manera, para cada nodo se seleccionan dos padres hacia el destino, buscando maximizar la cantidad de ancestros comunes entre los mismos. El primer padre se selecciona utilizando alguna métrica de enrutamiento, luego, se selecciona como segundo padre al que entre los vecinos del primero se ajuste mejor a esta métrica de enrutamiento y que, a su vez, cumpla alguna de las siguientes condiciones (en orden de importancia):

- el vecino comparte todos los padres con el primer padre.
- el vecino comparte al menos un padre con el primer padre.
- el vecino no comparte padres con el primer padre.

Luego, para cada uno de estos caminos, se ordenan a los padres. De esta manera, luego se podrá iterar sobre este conjunto para asignar celdas a cada uno de estos enlaces (entre el nodo hijo y sus padres) garantizando que un paquete será recibido antes de ser re-transmitido (esto es, si un nodo A tiene como padre a B y B a C, la celda de la trama de comunicación correspondiente a la comunicación $A - B$ deberá ser de un intervalo de tiempo anterior a la de $B - C$).

Por último, se asignan las celdas a cada enlace con el objetivo de maximizar la cantidad de estas que pueden ser compartidas sin colisiones. Para esto, se buscan celdas que tengan el mismo receptor que fueron previamente asignadas en el mismo camino. Es decir, si A tiene como padre a C y B tiene como padre a C, las celdas para los enlaces $A - C$ y $B - C$ deberían ser las mismas. Debido a que B sólo recibirá un paquete para transmitir solo si la transmisión de A fue fallida (asumiendo que A es el primer padre y B el segundo de la capa anterior) C sólo recibirá un paquete y la celda será utilizada solo por uno de estos dos nodos a la vez.

2.2.2. Evaluación de la propuesta de *scheduling*

La propuesta enunciada (trenzada) con el uso del algoritmo de *scheduling* se compara con la versión de camino único por defecto del protocolo base utilizado (RPL,

explicado en mayor detalle en el capítulo 3) y con un algoritmo que construye dos caminos disjuntos siguiendo la propuesta enunciada en 2.1.1 y provee celdas independientes para cada camino. La topología de la red y los *schedules* que se utilizan son fijos.

Las tres versiones consiguen una alta confiabilidad gracias al hecho de que las tres aumentan la probabilidad de que un paquete llegue de forma correcta a su destino al agregar transmisiones alternativas de los paquetes. En cuanto a las demoras en la entrega, el algoritmo disjunto tiene el menor valor, mientras que el de camino único y el trenzado tiene la menor variabilidad en la demora. Esto último puede explicarse debido a que tanto la versión de camino único como la de trenzado utilizan los mismos intervalos de tiempo para hacer envíos de paquetes, presentando pocas diferencias en los tiempos de entrega; por otro lado, la de caminos disjuntos envía dos copias a la vez de forma independiente de la detección de fallas o pérdidas, generando que en el caso de que el paquete llegue de forma correcta, el mismo llegue más rápido que en las otras versiones.

En cuanto a la tolerancia a fallas, la propuesta es la que mejor se comporta frente a que uno o dos nodos de la topología colapsen y resulten inaccesibles, manteniendo mejores valores de entregas de paquetes exitosas y demoras en las entregas. La mejor tolerancia a fallas se debe a que esta versión es la que más caminos alternativos provee para la entrega de paquetes.

Por último, al comparar la cantidad de celdas asignadas en los tres casos, los valores obtenidos son equivalentes. Sin embargo, en el caso del algoritmo disjunto, los paquetes son replicados siempre, de manera que todas las celdas se utilizan, por esta razón, el consumo energético de este algoritmo es mucho más alto que los otros dos. Por otro lado, según cálculos teóricos, el crecimiento del *schedule* para la propuesta es mucho menor que el del algoritmo disjunto en el caso de agregar más padres alternativos o caminos, respectivamente.

2.3. Otras alternativas

Lohith, Narasimman, Anand y Hedge [14] desarrollaron *LinkPeek*, un módulo adicional que agrega funcionalidad al protocolo estandar RPL (descrito en el capítulo 3)

como medio de mejorar la proporción de paquetes entregados de forma correcta en una red de dispositivos IoT. Utilizando este módulo, al superarse un umbral de retransmisiones sin lograr el envio correcto de un paquete, el nodo que se encuentra transmitiendo retransmitite en su siguiente intento el paquete a un padre alternativo. La idea detras de esta lógica es la de poder continuar transmitiendo mensajes si los padres preferidos de los nodos se encuentran inalcanzables de forma temporal o definitiva.

Esta propuesta intenta evitar la creación de copias y el consumo extra de energía que traen consigo las alternativas de caminos múltiples. A su vez, busca cumplir los estándares ya establecidos de los protocolos utilizados actualmente agregando una librería sencilla y fácilmente configurable en vez de modificar a los protocolos en sí.

En específico, en el funcionamiento de *LinkPeek*, cuando un nodo detecta que una transmisión a su padre preferido ha sido fallida luego de una cierta cantidad de intentos, *LinkPeek* reenvía ese mismo paquete al siguiente mejor padre disponible (es decir, a un padre alternativo). Esta lógica podría reiterarse múltiples veces, escogiendo en cada instancia a un padre alternativo distinto. La cantidad de intentos o retransmisiones es configurable además de la cantidad de padres alternativos a utilizar. Para elegir a los padres alternativos, se utiliza a la lógica de elección de padres de RPL que ya provee a todos los padres posibles en una lista ordenada por preferencia.

El mecanismo propuesto es sólo de caracter reactivo, es decir, sólo se activa si ocurren fallas en las transmisiones y si las mismas superan al umbral establecido. En una red ideal, este mecanismo no se activaría y, en cambio, se usaría RPL por defecto.

En su analisis de desempeño, *LinkPeek* obtiene valores mejores respecto a la confiabilidad que la versión de RPL por defecto, sin embargo, no se proveen valores de la demora en la entrega de paquetes o el consumo energético.

Capítulo 3

Protocolos y software

En este capítulo se presentarán los protocolos y el *software* sobre los cuales se realizó esta tesis. En particular, los dos protocolos que se enuncian a continuación son RPL y TSCH, desarrollados para dispositivos IoT para las capas de routeo y MAC respectivamente. Las soluciones propuestas en este trabajo se basan y extienden a estos mismos.

Luego, se hará incapie sobre el sistema operativo Contiki y su simulador COOJA los cuales se utilizaron para implementar los algoritmos propuestos basados en los protocolos anteriormente mencionados y probarlos, pudiendo obtener métricas que reflejen su compartamiento, para luego analizarlos y compararlos.

3.1. RPL

IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [15] (Protocolo de routeo IPv6 para redes con baja energía y pérdidas) es un protocolo de tipo *distance-vector* diseñado específicamente para acoplar redes de dispositivos IoT a la red IPv6. Por esta razón, debe tener en cuenta en su diseño la baja velocidad de envio de datos de estas redes, junto con la alta cantidad de pérdidas y la necesidad de adaptabilidad debido a la inaccesibilidad a los enlaces por tiempos prolongados.

Este protocolo es utilizado por redes cuyo principal objetivo es que los nodos envien de forma periódica mediciones a un punto de recolección. Para esto, cada nodo elije

un padre de forma de generar un *Destination-Oriented Directed Acyclic Graph (DODAG)* (Gráfico acíclico dirigido con orientación a una destinación) que termina en una raíz. En este grafo, la raíz actúa como el *border router* de la red que conecta a los dispositivos, las hojas de la estructura, a Internet. Una red puede tener uno o varios DODAGs que forman una instancia de RPL, designada por un identificador único. Cada nodo puede unirse a varias instancias distintas pero solo se unirá a un DODAG en cada una de estas. El tener varios DODAGs en una misma instancia permite conectar redes no conexas (donde no todos sus miembros tienen caminos a todos los otros miembros).

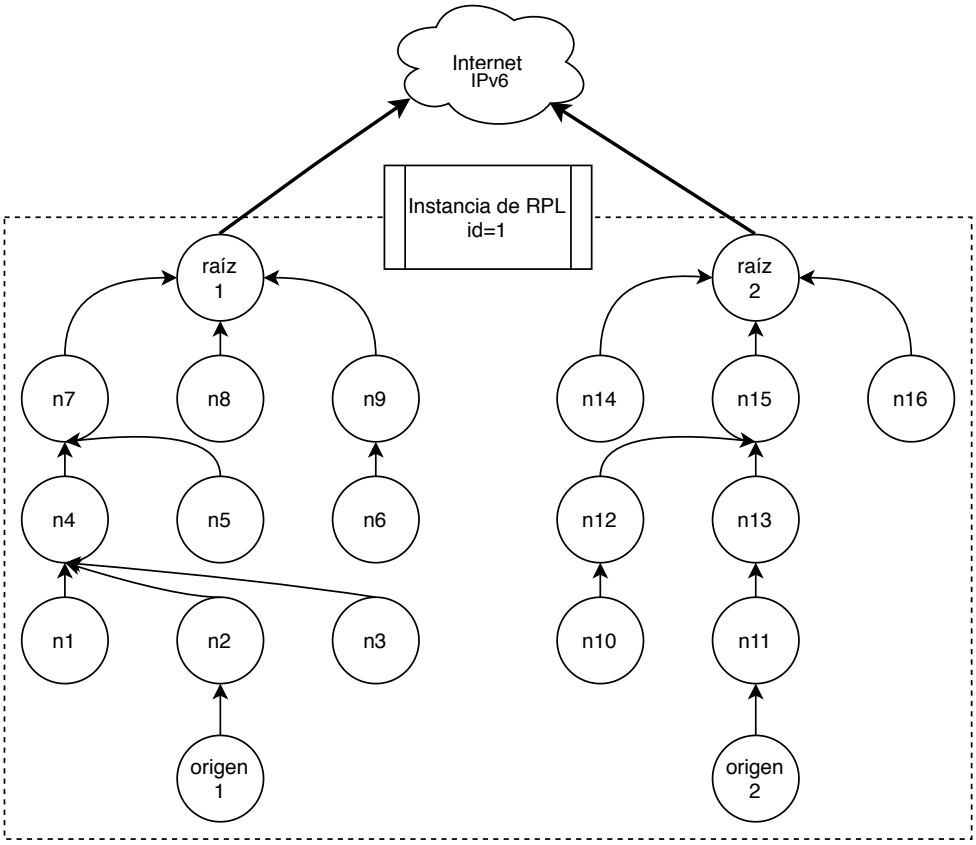

Figura 3.1: Un ejemplo de una instancia de RPL.

En la figura 3.1 se tiene una instancia de RPL con identificador 1, formada por dos DODAGs, uno que contiene a los nodos $n1$ a $n9$, el origen 1 y la raíz 1, el otro contiene a los nodos $n10$ a $n16$, el origen 2 y la raíz 2. Además, se puede observar, por ejemplo, que el padre preferido de $n1$ es $n4$ y de $n4$, $n7$, el cuál conecta diréctamente con la raíz 1 que conecta con la red de IPv6 de Internet.

La elección de los padres se realiza gracias a una *Objective Function (OF)* [16] (función objetivo) que toma varias métricas como parámetros y devuelve un único valor denominado *rank* (rango) para cada posible padre. Dicha salida busca reflejar la distancia o el costo de ir desde la raíz a un nodo en particular. De esta manera, todos los vecinos de un nodo que tengan un rango menor al mismo (es decir, que resulten "más cercanos a la raíz") serán considerados como posibles padres, agrupándolos en un *Parent Set (PS)* (set de padres) y se elegirá como *Prefered Parent (PP)* (padre preferido) al de menor rango entre todos estos. Además, el rango se utiliza para detectar y evitar los ciclos en el routeo y permite a los nodos diferenciar entre sus posibles padres (aquellos con rango menor) y sus hermanos (aquellos con igual rango).

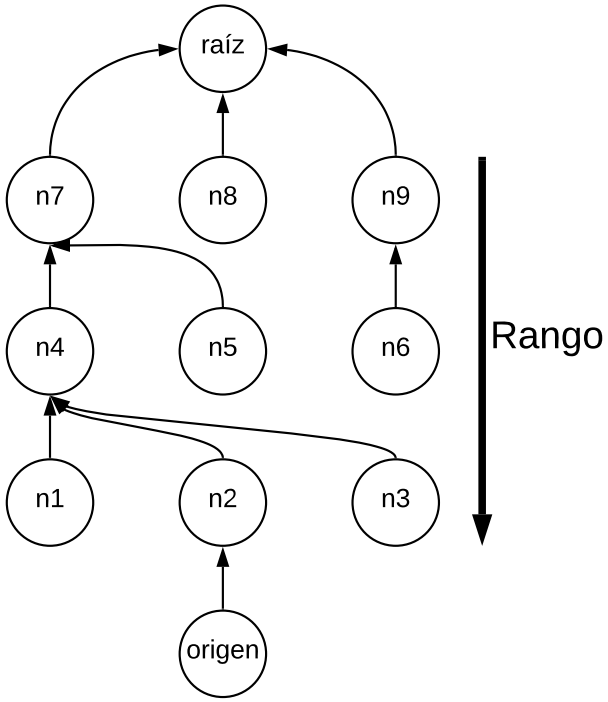

Figura 3.2: DODAG y valores creciente de rangos en el mismo, dónde el origen representa a un dispositivo IoT que busca conectarse con Internet.

En la figura 3.2 podemos observar un ejemplo de DODAG, dónde el origen representa a un dispositivo IoT (por ejemplo, un sensor de temperatura) que busca enviar información por medio de IPv6 a algún servidor remoto. La raíz tendrá el rango más bajo de todos y el mismo irá aumentando a medida que los nodos se alejen de la misma, teniendo la capa de $n1$, $n2$ y $n3$ los mayores valores.

Una de las métricas que se suele utilizar para la función objetivo de RPL es la llamada *Expected Transmission Count (ETX)* [17] (cantidad esperada de transmisiones) que mide la cantidad de transmisiones que se esperan de un paquete desde el nodo a su destino para que éste llegue sin error al mismo. Otro ejemplo podría ser el de la métrica *Hop Counting (HC)* (contador de saltos) que refleja la cantidad de "saltos" que un paquete necesita para llegar a un destino. La función objetivo permite tomar más de una métrica como entrada para calcular el rango de un nodo, logrando aprovechar los beneficios de varias de estas a la vez. Para las métricas mencionadas anteriormente, ETX hace incapié en la confianza de un nodo o un camino mientras que con HC se elijen los caminos de menor distancia.

RPL combina topologías de tipo jerárquicas y de tipo *mesh* (de malla). Resulta jerárquica por la forma de los DODAGs, dónde cada nodo tiene una relación de tipo padre-hijo. Sin embargo, se permite el generar rutas alternativas via los vecinos en caso de necesidad (por ejemplo, cuando el padre preferido resulte inaccesible), propiciando una topología de tipo de malla.

Este protocolo utiliza cuatro paquetes de control para crear y mantener a los DODAG, estos son:

- *DODAG Information Object (DIO)* (objeto de información del DODAG): es un paquete que se transmite de manera *multicast* (es decir, hacia todos los miembros de la red) generado por la raíz del DODAG o de forma *unicast* (es decir, hacia un nodo en particular) en caso de responder a un DIO. Se utiliza para actualizar y mantener el status de cada DODAG. Contiene información que permite a un nodo obtener el identificador del mismo, sus parámetros de configuración y además seleccionar un set de padres.

- *DODAG Informational Solicitation (DIS)* (pedido de información del DODAG): en este caso, este paquete se envia de manera *broadcast* (de la misma manera que el *multicast* se envia a múltiples nodos, pero no es un mensaje dirigido hacia un conjunto en particular) para recibir un paquete DIO en respuesta, con el objetivo de probar o detectar si resultan accesibles los nodos vecinos.

- *Destination Advertisement Object (DAO)* (objeto de anuncio de destino): los DAOs se transmiten de forma *broadcast* cada vez que un nodo selecciona a un nuevo padre. Su objetivo es el de propagar la información a utilizar para generar los caminos desde las hojas a la raíz del DODAG (caminos *upward* o hacia arriba).
- *Destination Advertisement Object Acknowledge (DAO-ACK)* (confirmación de objeto de anuncio de destino): es un paquete *unicast* enviado por un receptor de un mensaje DAO en respuesta a esta recepción.

La construcción de cada DODAGs se basa en el proceso de *Neighbor Discovery* (ND) (descubrimiento de vecinos), para esto la raíz de cada DODAG envia de forma *broadcast* un mensaje de tipo DIO que contiene (entre otra información) el identificador del DODAG, el rango de la raíz y la función objetivo a utilizar. Cuando un nodo que no es aún parte de la red y que quiere sumarse a la misma recibe un DIO, agrega al emisor de este a su set de padres, luego calcula su propio rango y reenvia el DIO con el valor obtenido. Después de esto, elije a su padre preferido según se explicó anteriormente basándose en los rangos de los mismos. En el caso de que un nodo que ya es parte de la red reciba un DIO, puede descartarlo según algún criterio predefinido, procesarlo y mantener su rango o procesarlo y actualizar su rango si al calcularlo de nuevo con esta nueva información obtiene un valor menor. En este último caso, debe eliminar todos los padres de rango mayor al suyo de su set y reenviar el DIO con este nuevo valor.

En el caso de las rutas reversas, RPL cuenta con dos opciones para mantenerlas, modo *storing* (con guardado) y modo *non-storing* (sin guardado). En el primer modo, se envia un mensaje DAO de forma unicast desde cada nodo a su padre preferido con su dirección, a su vez, este padre envia un nuevo mensaje de tipo DAO con la información de los nodos que pueden llegar a él a su propio padre. Cada nodo guarda la información de sus hijos. Este envío de DAOs se realiza cada vez que un nodo elije a un nuevo padre.

En el segundo modo, la diferencia recae en que la información solo se guarda en la raíz del DODAG. De esta manera, si un nodo del DODAG quisiera comunicarse con otro, se le enviará un paquete del primer nodo hasta la raíz (utilizando el camino de los padres preferidos) y luego la raíz la enviará al segundo nodo (usando la información de las rutas reversas).

Para mantener los DODAGs y actualizarlos en caso de que parte de los enlaces resulten inaccesibles o se tenga una estructura incorrecta (por ejemplo, debido a un ciclo en la misma), todos los nodos envían de forma periódica mensajes de tipo DIO según un *trickle timer* (temporizador de goteo). La frecuencia de este temporizador aumenta al detectarse inconsistencias en la información recibida de estos DIOs y disminuye en caso contrario. Para realizar reparaciones en los DODAGs, RPL cuenta con mecanismos de reparación global (rearmar toda la estructura) y local (elegir un camino alternativo temporal para un nodo). El segundo tipo de reparación se intenta antes de realizar una reparación global debido a que es mucho menos costosa respecto a tiempo y a energía. Además, evita y detecta ciclos definiendo una máxima distancia o profundidad que puede tener el padre de un nodo, publicitando rangos infinitos para evitar ciclos en la actualización de los valores de rango de un sub-DODAG y guardando la información del camino que recorre un paquete para, en caso de descubrir que el mismo pasa dos veces por el mismo nodo, lanzar uno de los mecanismos de reparación anteriormente mencionados.

Por último, respecto a seguridad, el protocolo cuenta con tres modos de operación: inseguro (los mensajes de control de RPL viajan como texto plano), pre-instalado (los nodos se unen a una instancia de RPL utilizando *keys* (llaves) preinstaladas que les permiten procesar y generar mensajes de control de RPL seguros) y autenticado (similar al anterior, pero las llaves preinstaladas sólo permiten unirse a una instacia de RPL como hojas de los DODAGs).

3.2. TSCH

IEEE Std 802.15.4-2015 Time Slot Channel Hopping (TSCH) [18] es un protocolo de la capa MAC diseñado para ofrecer confiabilidad y determinismo con bajo uso de energía. Este protocolo combina las técnicas de *Time-Division Multiple Access* (TDMA) (acceso múltiple por división del tiempo) y *Frequency-Division Multiple Access* (FDMA) (Acceso múltiple por división de la frecuencia). TDMA permite el acceso a un medio de transmisión compartido otorgándole rangos de tiempo específicos a cada transmisor de modo de que dos o más no transmitan a la vez. Por otro lado, FDMA divide el espectro

disponible en canales asignándole cada uno a un transmisor, evitando también colisiones entre los mismos.

En el caso particular de TSCH, el protocolo divide el tiempo en intervalos que se agrupan en un *timeslot* (registro de tiempos) que se repetirá indefinidamente mientras la red exista. Cada intervalo es lo suficientemente largo para que se transmita un paquete del tamaño máximo permitido ($127B$) y que se le permita al receptor enviar un mensaje de *acknowledge* (confirmación).

Las transmisiones y recepciones de mensajes tendrán asignado un intervalo y un índice de canal. El índice se utiliza para calcular la frecuencia en la que se deberá transmitir con la siguiente ecuación, dónde v es el índice, n es el total de canales que se pueden utilizar y *ASN* refiere a *Absolute Sequence Number* (número absoluto de secuencia), un contador global de intervalos compartido por todos los nodos de la red.

$$frecuencia = (ASN + v) mod n \tag{3.1}$$

A estos pares de intervalos e índices que se les asignan a los dispositivos para transmitir y recibir se los llama *cells* (celdas). Las mismas se defininen de modo de que no ocurran dos transmisiones en el mismo momento en el mismo canal, evitando fenómenos de interferencia y *fading* (desvanecimiento). Las celdas pueden ser de dos tipos: dedicadas o compartidas. En las dedicadas se realizan transmisiones y recepciones de paquetes de datos y pueden ser utilizadas por un único nodo de la red mientras que en las compartidas se intercambian mensajes de control en dónde cualquier nodo puede transmitir o recibir paquetes. TSCH utiliza paquetes de control de tipo *Enhanced Beacons* (EB) (guías mejoradas) para mantener su registro actualizado y sincronizado entre todos los nodos.

Este mecanismo de celdas también permite que en los momentos que un nodo no tenga una celda asignada para transmitir o recibir paquetes, se mantenga en estado *sleep* (dormido). De esta manera, se optimiza la energía que gasta cada uno de los dispositivos,

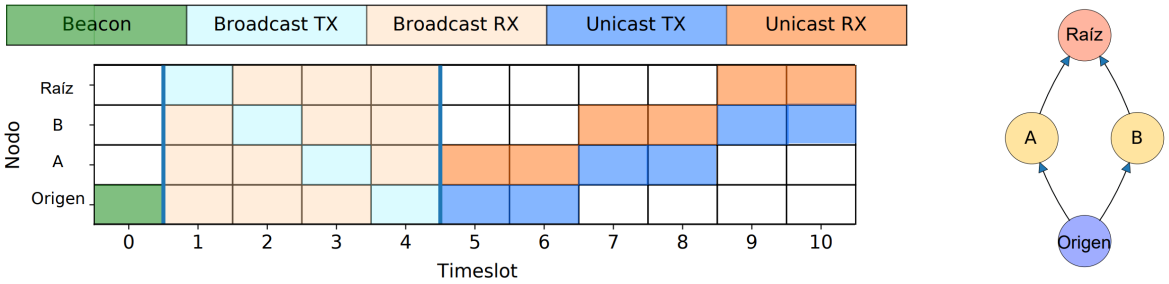

Figura 3.3: Un ejemplo de un registro de TSCH para la topología de la izquierda

En la figura 3.3 se puede observar un ejemplo de un registro para la topología que se ve a la derecha, consistente de tres nodos (*A*, *B* y origen) y una raíz, se tiene un único canal. El rectángulo verde representa un EB que envía el origen, los celestes representan transmisiones en modo *broadcast*, los beige recepciones *broadcast*, los azules, transmisiones *unicast* y los naranjas recepciones *unicast*. De esta manera, se puede apreciar que las celdas que se encuentran entre los intervalos 1 y 4 inclusive son compartidas (se tienen varias celdas que utilizan el mismo intervalo y canal), mientras que las que se encuentran del 5 hasta el 10 inclusive son dedicadas (no se tienen dos o más transmisiones o recepciones que utilicen el mismo canal en el mismo intervalo). Se debe tener en cuenta que la topología mostrada no es un DODAG de RPL, sino que muestra los enlaces entre los nodos de una red. Dado que RPL y TSCH son agnósticos el uno del otro, en este registro de TSCH se tienen asignadas celdas para que todos los enlaces existentes puedan ser usados independientemente de cuáles luego utilice RPL.

Para la formación del registro, el mismo puede crearse de forma centralizada o descentralizada. En el primer caso, uno de los dispositivos de la red se designa como coordinador y genera el registro en forma completa, para luego compartirlo con el resto. En el segundo caso, cada nodo toma decisiones locales de forma conjunta con sus vecinos, formando de forma comunitaria el registro completo. Los algoritmos de tipo centralizado resultan más sencillos de implementar y reducen la complejidad de la lógica que ejecutan los nodos, resultando también en un menor uso de energía por parte de los mismos, por otro lado, los descentralizados reducen el tráfico de punta a punta de la red y se adaptan más facilmente a las modificaciones en la misma en entornos cambiantes.

Como principal particularidad de este protocolo, TSCH no define cómo crear

el registro, sino que da un marco teórico de como interpretar al mismo una vez creado y otorga ciertos lineamientos que se deben cumplir (por ejemplo, el tamaño de cada intervalo). El registro a utilizar en cada caso depende exclusívamente de la implementación de TSCH. Algunos ejemplos de implementación son WirelessHART [19] y ISA100 [20] que utilizan registros centralizados, por otro lado se encuentran las especificaciones de 6TiSCH [21] que utiliza un registro distribuido.

3.3. Contiki

Contiki OS[1] [22] es un sistema operativo de código abierto diseñado para dispositivos IoT. El concepto de código abierto refiere a un modelo de desarrollo colaborativo donde tanto el código fuente como su documentación son accesibles por toda la comunidad.

Las principales características de diseño de este sistema operativo son su bajo consumo de recursos, en específico, de memoria y de energía, y sus implementaciones de protocolos *wireless* de forma nativa. Estas propiedades responden a las necesidades de los dispositivos que lo utilizan.

Por otro lado, el mismo se adapta a diversos aparatos y microcontroladores y entre los protocolos que implementa se encuentran tanto RPL como TSCH. Además de esto, es uno de los sistemas operativos modelo que se utilizan en el entorno académico para realizar experimentación de tecnologías IoT.

Contiki OS está implementado en el lenguaje C. Este lenguaje le otorga a los desarrolladores un mayor control sobre los recursos del objecto sobre el cual el sistema se ejecuta mientras se mantiene el uso de un lenguaje de alto nivel (es decir, un lenguaje de programación que resulta más fácilmente entendible por un humano).

3.3.1. COOJA

El simulador COOJA es la principal razón de que Contiki sea uno de los sistemas operativos más usados en el ambiente académico. COOJA es un simulador de redes para

[1] https://github.com/contiki-os/contiki

Contiki desarrollado en Java para Windows y Linux, siendo luego portado a MAC OS, el mismo se encuentra incluido en el código fuente de este primero.

COOJA permite compilar a la vez diferentes versiones de Contiki como librerías compartidas y cargarlas en Java, logrando simular distintos tipos de dispositivos en una misma simulación. El simulador controla y analiza luego los ejecutables generados.

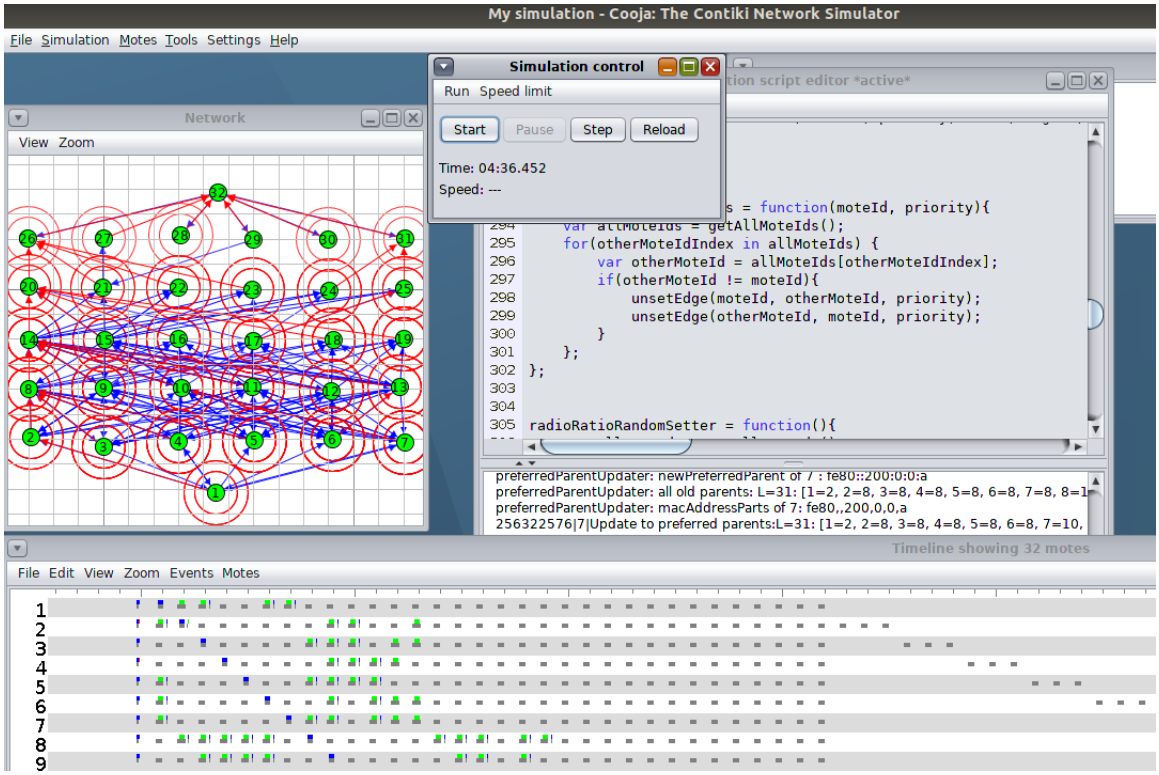

Figura 3.4: COOJA ejecutando una simulación.

En la figura 3.4 se puede observar una captura de pantalla de una computadora donde COOJA está ejecutando una simulación. Hacia la izquierda se puede ver la red simulada de treinta y dos dispositivos, donde las fechas representan envios de paquetes de un dispositivo a otro. A la derecha se observa de fondo el código en C que ejecuta cada uno de los nodos y abajo del mismo los mensajes de registro de estos.

Por último, abajo de todo se observa el registro TSCH de la red, que utiliza un único canal. Cada rectángulo gris es un intervalo de TSCH, los cuadrados verdes encima de los mismos representan paquetes enviados y los azules, recibido. A su vez, las líneas que aparecen al lado de las transmisiones y recepciones representan a los mensajes de *acknowledge* (ACK) (mensajes de confirmación) de estos. Se puede observar que el primer

bloque de intervalos representan a las celdas compartidas (en esta simulación se utiliza un único canal, todos los dispositivos pueden transmtir a la vez) y los siguientes a las dedicadas.

Capítulo 4

Nuevos algoritmos de ruteo determinístico

En este capítulo se introducen las dos propuestas que se desarrollaron durante esta tesis y los conceptos sobre los que estas se basan.

En primer lugar, se hace una breve explicación de las ideas de retransmisiones y réplicas, las cuales se utilizan en los dos algoritmos implementados. A continuación, se explica el primer algoritmo, *n-Disjunto: por defecto* y por último, el segundo *n-Disjuto: controlado*, que es una versión mejorada del primero.

Durante el resto de este capítulo se utilizará el siguiente DODAG de ejemplo para ilustrar los temas a tratar.

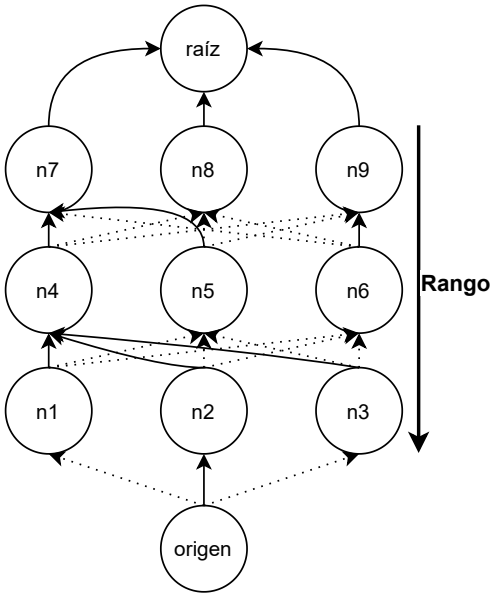

Figura 4.1: Ejemplo de un DODAG.

En el DODAG de la figura 4.1 se tienen once nodos (nueve intermedios más la raíz y el origen). En el mismo, cada nodo, salvo $n7$, $n8$, $n9$ y la raíz, cuenta con tres padres en su conjunto, donde el padre preferido se encuentra marcado con una flecha con línea entera y los alternativos con línea punteada. A su vez, el rango es creciente desde la raíz hacia el origen (por ejemplo, $n1$, $n2$ y $n3$ tendrán un rango mayor al de $n7$, $n8$ y $n9$).

4.1. Conceptos previos: retransmisiones y réplicas

Como ya fue mencionado con anterioridad en el Capítulo 3, las propuestas de este trabajo extienden a los protocolos TSCH y RPL. En específico, se aprovecha al mecanismo de retransmisiones de TSCH para utilizar a estas y al conjunto de padres de RPL para la generación de réplicas.

4.1.1. Retransmisiones

Las retransmisiones refieren a intentos de reenviar un paquete una vez que se detectó que el mismo no pudo entregarse de forma exitosa. Por ejemplo, si en el DODAG que se puede observar en la figura 4.1 se quiere enviar un paquete desde el origen a la raíz y el enlace entre $n4$ y $n7$ se encuentra inaccesible por un tiempo corto coincidente con

el momento en el que el mismo lo quiere atravesar, el paquete se perderá en dicho enlace. De esta manera, para lograr que el paquete llegue a destino, se podría hacer un segundo intento o retransmisión dado que el enlace volverá a estar funcional cuando ocurra este segundo intento. Entonces, las retransmisiones siempre ocurren luego de una primera transmisión fallida, que se detecta de forma activa al recibir un mensaje de non-ACK (*non Acknowledge* o no confirmación) o de forma pasiva al no recibir un mensaje de ACK (*Acknowledge* o confirmación) dentro del tiempo máximo estipulado. Es posible hacer uno o más re-intentos, esta cantidad depende de los protocolos de comunicación que se utilicen y de como los mismos se configuren. De aquí en adelante, si se hace referencia en este trabajo a t retransmisiones significa que se envia el paquete original y luego se pueden hacer hasta t reintentos con un total de $t+1$ transmisiones, es decir, si la transmisión i falla e $i < t$, entonces se hará una nueva transmisión $i+1$, sino, se desistirá.

La necesidad de realizar un nuevo intento agregará *delay* (retrasos) en la entrega de los paquetes. Esto se debe a que en vez de tener que esperar el tiempo que tarda un paquete desde el origen hasta el destino, se tendrá que esperar un tiempo proporcional a ese valor multiplicado por la cantidad de transmisiones que se necesiten para que el mismo llegue de forma correcta. Es decir, la cantidad de retransmisiones que se utilicen será proporcional al retraso total del paquete.

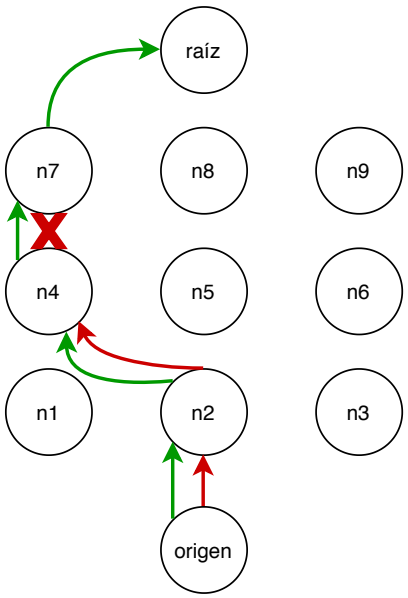

Figura 4.2: Los caminos que recorren una transmisión y su retransmisión en un ejemplo.

Siguiendo el ejemplo propuesto anteriormente, en el caso de detectar de forma pasiva que un paquete se perdió, si en el DODAG de la figura 4.1 se envía un paquete del origen a la raíz y el mismo en el primer intento se pierde entre los nodos $n4$ y $n7$ pero se logra entregar de forma correcta en el segundo intento, el tiempo total que llevará esa entrega será el tiempo que tarda el paquete en recorrer la distancia entre el origen y $n4$ más el tiempo máximo que puede tardar un mensaje de confirmación más el tiempo que tarda el paquete en recorrer la distancia entre el origen y la raíz. Los trajectos de la transmisión que no llega a destino y de su retransmisión que se entrega correctamente se pueden observar en la imagen 4.2 en rojo y en verde respectivamente, además, se marca con una cruz roja el enlace dónde ocurre la pérdida de la primera transmisión.

El otro inconveniente que presentan estos mecanismos es que no resultan útiles cuando un enlace en particular se encuentra inaccesible por un tiempo prolongado. Esto es debido a que los reintentos no utilizan un camino alternativo al original. En este caso, lo único que introducen es más retrasos a la red. Además, al utilizar retransmisiones se utilizará una mayor cantidad energía de la red al realizar más transmisiones por cada paquete. Dado que los dispositivos IoT tienen una cantidad de energía limitada y se encuentran aislados y no conectados a una fuente de energía fija, esto también resulta una desventaja para este tipo de redes.

Dentro del conjunto de protocolos que se utilizaron, estas retransmisiones se realizan en el protocolo TSCH, es decir, en la capa MAC. TSCH le permitirá a un nodo hacer una retransmisión dentro de una celda que tenga asignada para transmitir a ese destino.

4.1.2. Réplicas

Durante este trabajo, con réplicas se hace referencia a paquetes con el mismo contenido que otro paquete "original" que se transmiten de forma independiente al mismo con el objetivo de aumentar la probabilidad de que el contenido de este llegue a su destino. Es decir, al contrario de las retransmisiones que se envian al detectar una pérdida y utilizando el mismo camino que el primer paquete, las réplicas se transmiten siempre que se transmita el paquete original utilizando el mismo u otro camino. La cantidad de réplicas

que se utiliza depende, de la misma manera que las retransmisiones, de los protocolos de comunicación que se utilicen y de su configuración. De aquí en adelante, si se hace referencia en este trabajo a n réplicas significa que se envian $n+1$ paquetes, n réplicas más el paquete original.

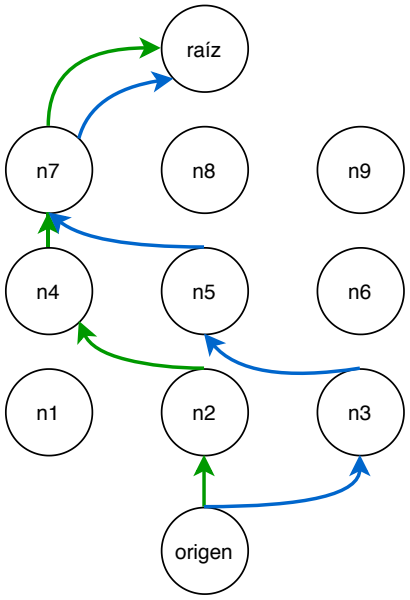

Figura 4.3: Los caminos que recorren una transmisión y su réplica en un ejemplo.

En la figura 4.3 se puede observar un ejemplo del camino que recorre un paquete, en verde, y su réplica, en azul. Considerando el caso que fue planteado en la sección 4.1.1, si hubiera una pérdida en el enlace entre los nodos $n4$ y $n7$, se perdería al paquete original pero la réplica llegaría a destino (considerando que el fallo está en el enlace entre $n4$ y $n7$ y no en el nodo $n4$).

La idea de réplicas fue planteada con el objetivo de mantener los beneficios del uso de retransmisiones (mejorar la confiabilidad de la red al aumentar la probabilidad de una entrega correcta) sin agregar retrasos en la entrega de los paquetes. Al enviarse los paquetes originales junto con sus réplicas, el tiempo de entrega es independiente de cuáles lleguen exitósamente y cuáles no (o, al menos, la diferencia entre los tiempos de entrega de todos los paquetes resulta despreciable frente al tiempo que toma detectar que un paquete se perdió y esperar de nuevo a que el mismo llegue a destino como sería necesario en el caso de las retransmisiones). Utilizar una mayor cantidad de réplicas no debería (al menos en la teoría) modificar los tiempos de entrega.

Sin embargo, las desventaja de utilizar réplicas recae en la consecuencia de utilizar más nodos y más enlaces de la red, lo que incluso podría generar un fenómeno de *flooding* (inundación). En este fenómeno, cada nodo reenvía los paquetes recibidos a todos sus vecinos, salvo a aquél que le envió el paquete.

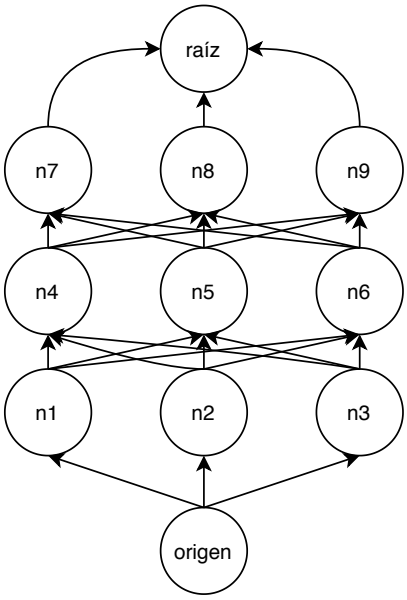

Figura 4.4: *Flooding* en la topología de ejemplo.

En la figura 4.4 se puede observar el fenómeno de inundación para la misma topología que fue muestrada al inicio de este capítulo en la figura 4.1. En este caso, cada línea representa una transmisión, de manera que cada nodo le reenviará los paquetes recibidos a todo su set de padres, es decir, a todos los enlaces de salida disponibles que tiene en vez de sólo a su padre preferido.

Este tipo de comportamientos son perjudicionales para las redes de dispositivos IoT debido a que estas tienen recursos limitados de energía, al utilizar una mayor cantidad de nodos y enlaces este consumo será mayor. Se debe aclarar que al mencionar a la utilización de una mayor cantidad de nodos y de enlaces se hace referencia a los componentes de la red que se utilizan para enviar un único paquete, una red que utilice todos sus nodos para enviar cada uno de sus paquetes será subóptima en su consumo energético, sin embargo, una que use un conjunto de los mismos para cada transmisión y que vaya rotando ese conjunto podrá funcionar por un tiempo mucho más prolongado. Muchos protocolos de routeo para dispositivos IoT (RPL, entre ellos) permiten ser con-

figurados para elegir los caminos a utilizar optimizando el consumo energético, esto es, seleccionarán los caminos cuyos nodos tengan la mayor cantidad de energía disponible. Dado que estos valores son dinámicos, los caminos a utilizar se irán actualizando según se vaya consumiendo la energía de los dispositivos.

Otra consecuencia negativa que trae el uso de más enlaces de la red, es la posibilidad de que se genere una saturación en los mismos y, en consecuencia, se congestione la red. En este fenómeno parte de los enlaces reciben más tráfico del que pueden procesar generando un aumento en la demora de entrega e incluso pérdidas de paquetes. En una red, existen *buffers*, *queues* o colas de entrada y de salida dónde se alojan los paquetes antes de ser procesados y enviados. Cuando un nodo recibe un paquete, lo guarda en su cola de entrada, luego, cuando el mismo tiene tiempo disponible de procesamiento, saca el primer paquete (el más antiguo que se encuentra en la cola), lo procesa y lo guarda en la cola de salida del enlace por el que será transmitido dicho paquete. De la misma manera, cuando un enlace se encuentra libre para realizar una transmisión, sacará el primer paquete de la cola de salida asignada al mismo y lo transmitirá. Si se reciben una gran cantidad de paquetes, se generarán demoras debido a que los mismos pasarán más tiempo en colas de entrada y salida esperando ser procesados o transmitidos y en casos extremos, se perderán paquetes debido a que no habrá espacio disponible para guardarlos en estas colas. Dado que los dispositivos IoT cuentan con recursos de memoria y procesamiento más limitados y enlaces más lentos que las redes estandar de internet, son más subceptibles a congestionamiento.

En el caso de las réplicas, ninguno de los protocolos que se utilizaron durante esta tesis las implementa de forma nativa.

4.2. Algoritmos n-Disjuntos

En este trabajo se proponen dos algoritmos que utilizan tanto retransmisiones como réplicas con el objetivo de aumentar la confiabilidad de la red sin aumentar el tiempo de entrega de los paquetes. Ambos algoritmos se basan en los protocolos TSCH y RPL, utilizando la opción nativa de TSCH para las retransmisiones e implementando

mecanismos nuevos de replicación sobre RPL, dónde el segundo algoritmo, “n-Disjunto: controlado“, es una mejora sobre el primero, “n-Disjunto: por defecto“.

4.2.1. n-Disjunto: por defecto

En el algoritmo “n-Disjunto: por defecto“ se busca que una cantidad $n + 1$ de paquetes, n réplicas más el paquete original, atraviesen la red por caminos disjuntos (caminos que no compartan ningún nodo) desde el origen, es decir, el dispositivo IoT que envia un paquete, hacia la raíz del DODAG de RPL, es decir, el punto de contacto que tiene ese dispositivo IoT con el Internet.

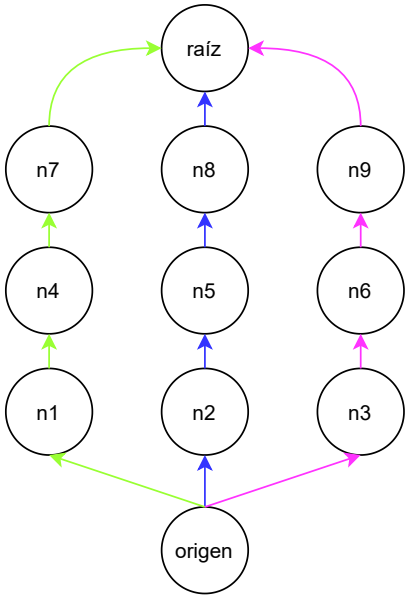

Figura 4.5: Tres caminos disjuntos en el DODAG de ejemplo

En el ejemplo de la figura 4.5 se pueden observar tres caminos disjuntos señalizados por las flechas en verde, azul y violeta. Por ejemplo, las flechas verdes podrían representar el trayecto que recorre el paquete original y las azules y violetas los de dos réplicas distintas. En este caso, n sería igual a dos.

Para obtener el comportamiento explicado en los párrafos anteriores, se modificó y amplió el código fuente de la implementación nativa de RPL del sistema operativo Contiki OS. El principal cambio al comportamiento de RPL fue permitir que el nodo origen transmita n réplicas, siendo n un valor entero positivo configurable menor al número de

padres que tiene el nodo origen. Por ejemplo, en el DODAG de la figura 4.1, el origen tiene tres padres, siendo entonces n menor o igual a dos. El nodo origen ordenará su set de padres, sin su padre preferido, según su rango, enviará el paquete original a su padre preferido y luego las n réplicas a n los padres del set con menor rango. Es decir, a diferencia de la implementación estandar de RPL, en esta propuesta el nodo origen genera n paquetes nuevos, copia del original, que luego transmite a los n padres alternativos seleccionados préviamente. Mientras tanto, los otros nodos mantienen la configuración por defecto del protocolo, esto es, reenvian los paquetes recibidos a sus padres preferidos. El resto del algoritmo utiliza la lógica y mecanismos ya existentes en RPL, incluida la construcción y mantenimiento del DODAG.

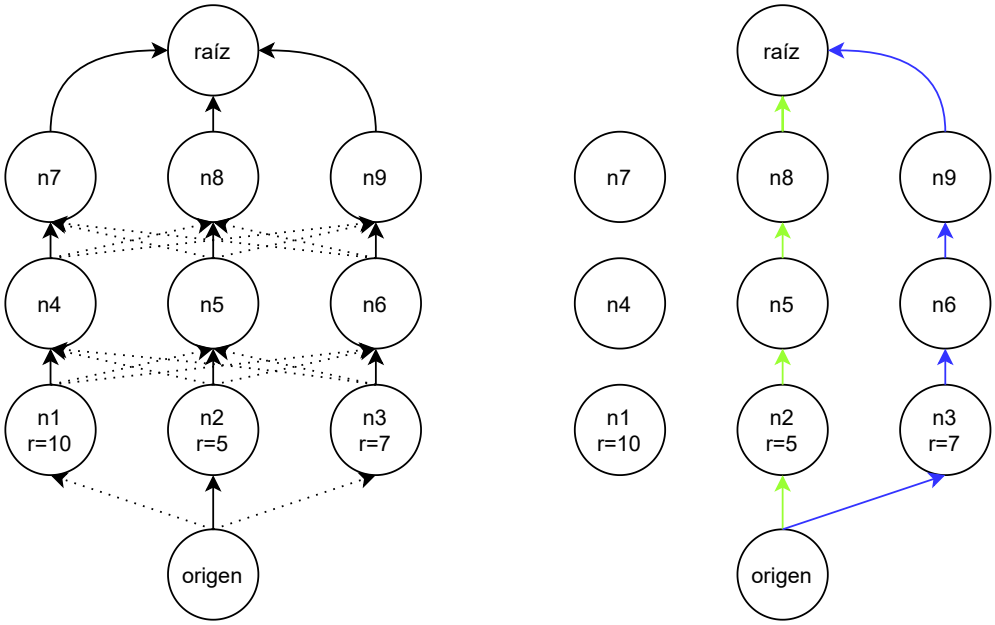

Figura 4.6: Un ejemplo del envío de paquetes del algoritmo n-Disjoint para $n = 1$ en el DODAG de la izquierda

En la figura 4.6 se puede observar un ejemplo de este algoritmo para $n = 1$. En el DODAG de la izquierda las líneas enteras representan a los padres preferidos de cada nodo y las punteadas los alternativos, además, se denotan con r los valores de los rangos de los tres padres ($n1$, $n2$ y $n3$) del nodo origen. Este nodo envia el paquete original a su padre preferido ($n2$) más una réplica al padre alternativo de menor rango ($n3$, con rango $r = 7$) a su vez, el resto de los nodos envía los paquetes que recibe a su padre preferido ($n2$ a $n5$, $n5$ a $n8$ y $n8$ a la raíz, lo mismo ocurre de $n3$ a $n6$, $n6$ a $n9$ y $n9$ a la

raíz). El comportamiento enunciado en este ejemplo es el ideal de este algoritmo donde la cantidad de réplicas que se envian se condice diréctamente con la cantidad de caminos disjuntos que se obtienen.

Por otro lado, con el mismo objetivo de maximizar la confiabilidad de la red, este algoritmo permite el uso de un número t de retransmisiones, siendo posible retransmitir una réplica o al paquete original. Las retransmisiones usadas son a nivel capa MAC, implementadas por TSCH de forma nativa.

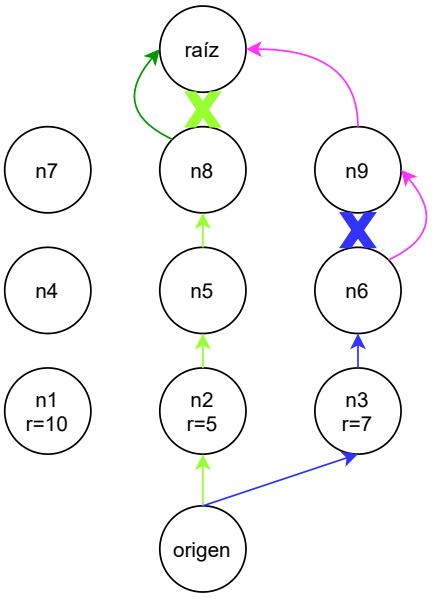

Figura 4.7: Ejemplo del envío de paquetes con réplicas y retransmisiones

En la figura 4.7 se puede observar un ejemplo de envío de paquetes del algoritmo n-Disjoint con $n = 1$ y con $t = 1$ para el DODAG mostrado en la figura 4.6. En este caso, las flechas en verde claro representan al paquete original y las azules a su réplica. Entre los nodos $n8$ y la raíz se pierde el paquete original y se retransmite, lo mismo ocurre entre $n6$ y $n9$ con su réplica.

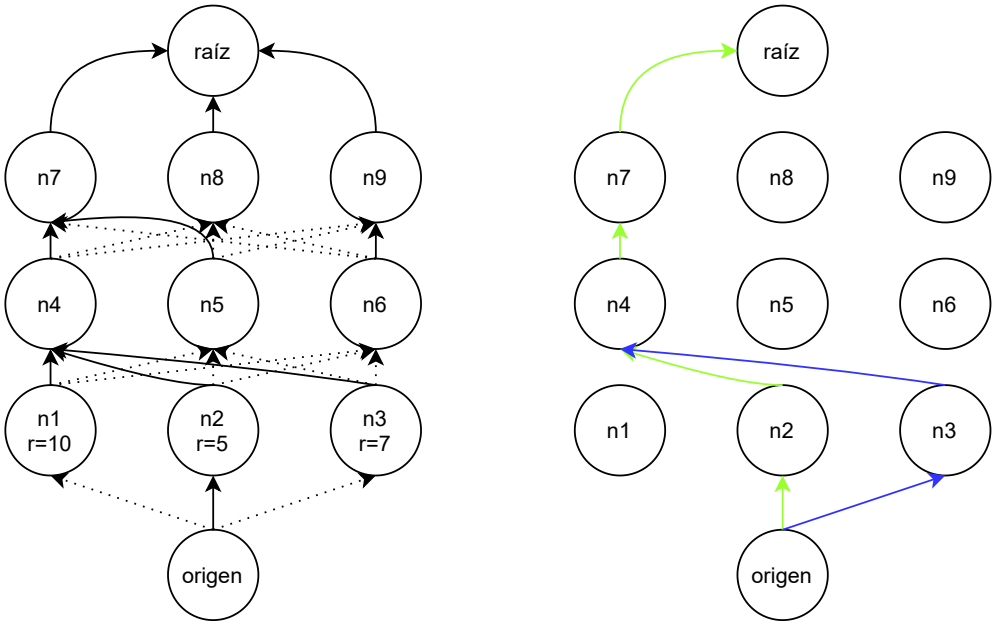

Figura 4.8: Un ejemplo del envío de paquetes del algoritmo n-Disjoint para $n = 1$ en el DODAG de la izquierda

Un caso no ideal en el que dos o más caminos convergen en uno solo se puede observar en la figura 4.8. En el DODAG de la izquierda se tiene de nuevo representados con líneas enteras a los padres preferidos de cada nodo y con punteadas a los alternativos. En el diagrama de la derecha se puede observar que debido a que los nodos $n2$ y $n3$ tienen ambos como padre preferido a $n4$, a partir de este nodo el camino que recorre el paquete original, en verde, y su réplica, en azul, es el mismo. Es decir, en un algoritmo "n-Disjunto: controlado", se tendrán a lo sumo $n + 1$ caminos disjuntos en el caso de que todos los nodos que utilicen el paquete original y sus n réplicas no compartan padres preferidos, en caso contrario, la cantidad de caminos disjuntos será menor a $n + 1$. En este algoritmo, frente al caso de que dos o más paquetes (sean todos réplicas o sean el paquete original y réplicas) lleguen a un mismo nodo (en el caso del ejemplo de la figura 4.8 el paquete original y la réplica llegan a $n4$), se transmitirá sólo uno al siguiente nivel y se descartará el resto (es decir, sólo un paquete recorrerá cada camino). Esto es así debido a que tanto el paquete original como su réplica comparten el mismo número de secuencia (un identificador único para cada paquete), de la misma manera que lo hacen un paquete y sus retransmisiones, por lo tanto, para un nodo es imposible diferenciar una réplica de una retransmisión que no es necesario transmitir dado que ya se ha recibido de forma

correcta (es decir, que el nodo ya recibió un paquete con ese número de secuencia). Se debe resaltar, además, que en estos últimos dos ejemplos no se tienen en cuenta las posibles pérdidas por desvanecimientos o congestión de la red, sino que se asume que todas las transmisiones se realizan corréctamente.

Al usar retransmisiones junto con réplicas se pueden obtener los beneficios de ambas técnicas a la vez sin aumentar considerablemente las demoras en las entregas de los paquetes como ocurriría si se utilazaran solo retransmisiones. Esto es debido a que se espera que sea necesario un número bajo de retransmisiones para que el envío sea exitoso. Por ejemplo, cuando se utilizan una réplica y una retransmisión, se tienen en total cuatro oportunidades (en el caso ideal) de que el paquete llegue al siguiente salto (el paquete original tiene dos oportunidades con su transmisión y retransmisión y lo mismo ocurre para su réplica, dando un total de cuatro oportunidades). En cambio, si sólo se utilizaran retransmisiones, se necesitarían cuatro para llegar a ese número de oportunidades. Si se utilizan menos transmisiones, la demora en las entregas será menor.

4.2.2. n-Disjunto: controlado

El algoritmo “n-Disjunto: controlado“ es una versión mejorada del algoritmo anterior que busca evitar o mejorar los casos donde dos o más caminos converjen, igual que en la versión anterior, n debe ser un valor igual o menor a la cantidad de padres que tenga cada nodo. Lo que este algoritmo hace es que al detectar que dos o más réplicas (o una o más réplicas y el paquete original) arrivan al mismo nodo, generando que converjan dos caminos, las transmite cada una a un nodo distinto, de manera de volver a “separar“ los caminos de las mismas. Para esto, todos los nodos de la red ordenan a su set de padres según su rango de forma creciente y cuando se recibe una réplica de un paquete, se envía al padre de menor rango que no se haya utilizado anteriormente para enviar una réplica del mismo o al paquete original.

```
transmitir_paquete_recibido(paquete):
    if es_una_retransmision_ya_enviada(paquete):
        return

    indice = obtener_indice(numero_de_secuencia(paquete))

    if indice == 0:
        transmitir_paquete(paquete, padre_preferido)
    else:
        transmitir_paquete(paquete, set_padres_ordenados[indice-1])

    nuevo_indice = indice + 1

    actualizar_indice(numero_de_secuencia(paquete), nuevo_indice)
```

Pseudocódigo 4.1: Algoritmo para transmitir los paquetes que utilizan los nodos en "n-Disjunto: controlado"

En el pseudocódigo 4.1 se enuncia la lógica que sigue este algoritmo para decidir a qué padre enviar los paquetes que recibe un nodo. Todos los índices se inicializan préviamente en cero y el set de padres ordenados es un arreglo ordenado de todos los padres del nodo según rango creciente sin el padre preferido del mismo. Esta lógica la utilizan los nodos $n1$ a $n9$ de todas las figuras mostradas en este capítulo. Cada uno de estos nodos, al recibir un paquete llamará a esta función. Si el paquete es una retransmisión que ya fue enviada, significa que no es necesario transmitirla de nuevo y no se transmitirá nada, en caso contrario, se obtendrá el índice para ese número de secuencia. Si ese índice es cero se enviará al padre preferido, sino, al padre alternativo que coincida con ese índice menos uno. Por último, se actualiza el índice para ese número de secuencia sumándole uno. Es decir, cuando llegue un paquete con un número de secuencia que nunca fue transmitido por el nodo, su índice será cero (es la primera vez que se llamaría a esta función con ese número de secuencia, de manera que nunca se actualizó el contador préviamente inicializado en cero) y se enviará al mismo al padre preferido, el segundo paquete tendrá un índice igual a uno y se enviará al padre alternativo de menor rango,

el tercer paquete tendrá un índice igual a dos y se enviará al segundo padre alternativo de menor rango y así sucesivamente. Dado que la cantidad de réplicas será menor a la cantidad de padres de cada nodo por restricciones del algoritmo, siempre habrá un padre alternativo disponible para cada copia.

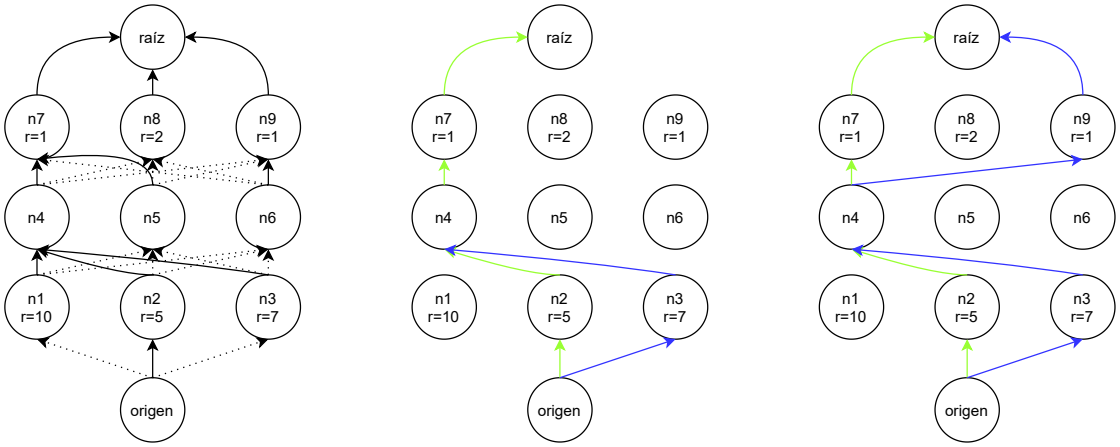

Figura 4.9: Un ejemplo del envío de paquetes de los algoritmos n-Disjoint: por defecto y controlado para $n = 1$ en el DODAG de la izquierda

En la figura 4.9 se puede observar el comportamiento de envió de paquetes para los algoritmos "n-Disjunto: por defecto" y "n-Disjunto: controlado" con $n = 1$ para el mismo DODAG que se encuentra a la izquierda. En este DODAG se denotan con flechas de línea entera a los padres preferidos y con línea punteada a los alternativos, por otro lado, se marca con r los valores de rango para los nodos $n1$, $n2$, $n3$ y $n7$, $n8$ y $n9$. En los otros dos diagramas, se marca en verde el camino que recorre el paquete original y en azul el que recorre la réplica. En ambos algoritmos, $n2$ y $n3$ envían el paquete que reciben a $n4$, su padre preferido, sin embargo, en el caso del primer algoritmo, al converger los caminos no se toma ninguna medida, en cambio, en el segundo algoritmo se envia el paquete original a $n7$, el padre preferido, y la réplica a $n9$, el padre de menor rango alternativo, con $r = 1$, luego, ambos nodos reenvian sus paquetes a la raíz.

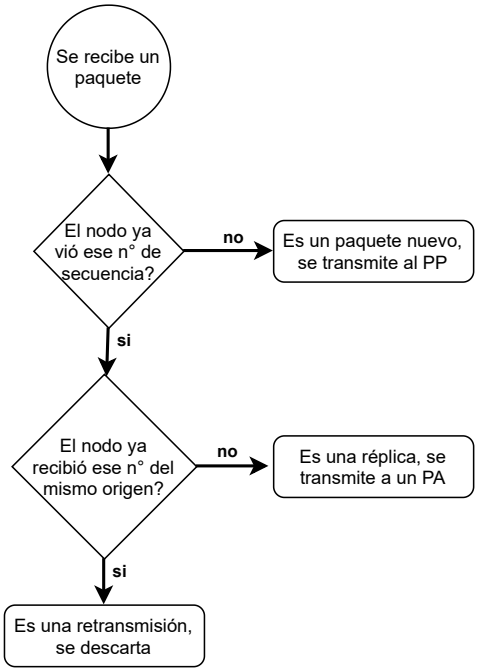

Figura 4.10: Flujo de decisión que toma cada nodo para determinar si recibió una réplica, una retransmisión o un nuevo paquete.

Para detectar estas convergencias de caminos, en el header de RPL se agrega un campo con un identificador del nodo que envía al paquete y, además, cada nodo guarda los números de secuencia de los paquetes que recibió y sus orígenes inmediatos. Entonces, cuando un nodo recibe un paquete, busca su número de secuencia entre aquellos que ya recibió, si no lo encuentra, es un paquete nuevo, si lo encuentra puede o ser una réplica o ser una retransmisión, para diferenciar esto compara el origen con los origenes previos de los que fue recibido el paquete, si el origen es igual, se trata de una retransmisión innecesaria y se descarta, sino, de una réplica cuyo camino convergió con otra recibida anteriormente. En la figura 4.10 se puede observar este flujo descripto.

El mecanismo que introduce esta mejora evita la pérdida de réplicas debido a la convergencia de caminos que ocurría en la versión “por defecto“ del algoritmo, sin embargo, no asegura que los caminos sean disjuntos ni “repara“ convergencias, sencillamente vuelve a dispersar a los paquetes.

El resto de este algoritmo, incluido el uso de retransmisiones, es equivalente al explicado para la versión “por defecto“.

Capítulo 5

Análisis de los protocolos n-Disjuntos en escenarios realistas

En este capítulo se describirá la experimentación que fue realizada en el marco de esta tesis. En específico, las simulaciones que fueron utilizadas para evaluar a los algoritmos propuestos, comparándolos entre si y comparándolos con los protocolos sobre los que los mismos se basan (RPL y TSCH, enunciados en el capítulo 3).

En primer lugar, se presentarán y explicarán las métricas a evaluar para cada uno de los algoritmos y las razones por las cuales los mismos fueron elegidos. En segundo lugar, se mencionará cuáles fueron las diferentes configuraciones y variantes utilizadas para ejecutar las simulaciones de esta tesis. Por último, se presentarán los resultados de los experimentos junto con conclusiones sobre estos resultados.

Durante este trabajo, se realizaron experimentos exclusívamente sobre *software*, utilizando el simulador COOJA del sistema operativo Contiki. Esto fue así debido a la simplicidad, flexibilidad y bajo costo que proveen este tipo de simulaciones en comparación a las que se realizan sobre un entorno real en *hardware*. Experimentar directamente sobre el código fuente permitió otorgarle más tiempo al desarrollo y diseño de los algoritmos en vez de a la configuración y al despliegue de estas experimentaciones, obteniendo, además, un entorno homogéneo y fácilmente reproducible para todos los experimentos.

5.1. Métricas a evaluar

En esta sección, como ya fue mencionado en la introducción de este capítulo, se presentan y explican aquellas métricas aceptadas por la comunidad científica para el análisis de este tipo de protocolos.

5.1.1. Packet Delivery Ratio (PDR)

El *Packet Delivery Ratio* (relación de entrega de paquetes o PDR por sus siglas en inglés) mide el porcentaje de paquetes distintos recibidos exitosamente frente a la cantidad total de paquetes distintos enviados.

$$PDR = \frac{cantidad\ de\ paquetes\ distintos\ recibidos}{cantidad\ de\ paquetes\ distintos\ enviados} * 100\,\% \tag{5.1}$$

Por ejemplo, siguiendo la ecuación 5.1 si se envían diez paquetes distintos y de estos sólo se recibe uno, el PDR será de 10 %. En una red determinística y de alta confiabilidad, es decir, en una red dónde enviar una cantidad n de paquetes siempre resulte en que estos mismos n paquetes se entreguen de forma correcta, se espera que el PDR tenga un valor de 100 %.

La definición de paquetes "distintos" se hace para poder aplicar esta métrica a entornos dónde se utilizan retransmisiones y/o réplicas. En estos casos, no es importante si llegan varias copias de un mismo paquete, sino que llegue al menos una de cada uno, esto es, que los paquetes no se pierdan. Siguiendo el ejemplo anterior, en el caso de que se envíen diez paquetes distintos y nueve réplicas por cada uno, si llegan las diez copias de un mismo paquete (el paquete original y sus nueve réplicas) el PDR volverá a ser de 10 %, en cambio, si llegara una copia de cada paquete, el PDR tendría un valor de 100 %. En ambos casos la cantidad de paquetes recibidos es de diez.

Se debe tener en cuenta, además, que cuando se hace referencia a "paquetes transmitidos" se refiere a aquellos que envía el dispositivo IoT que quiere comunicarse con Internet (marcado siempre como "origen" en los ejemplos enunciados durante este trabajo) y cuando se hace refencia a "paquetes recibidos" a aquellos que recibe el *border router* o raíz del DODAG. Es decir, se cuentan los paquetes que llegan a su destino final y

no las pérdidas que puedan ocurrir en los estadíos intermedios o los “saltos“ que se dan en la red debido a la topología de la misma.

5.1.2. Demora y su variabilidad

El *delay* (demora) refleja cuánto tarda cada paquete en ser entregado, esto es, la diferencia de tiempo entre el momento en que un paquete se transmite y el momento en el que el mismo paquete se recibe, siendo la transmisión aquella que hace el origen y la recepción el momento en que el paquete llega al *border router* o raíz del DODAG, es decir, su destino final.

$$Demora(paquete) = t_{transmisión}(paquete) - t_{emisión}(paquete) \tag{5.2}$$

Por ejemplo, si un paquete se transmite un día a las 4 de la tarde y se recibe ese mismo día a las 5 de la tarde, su demora será de una hora. Cuando se habla de demora de la red, entonces, se refiere al valor esperado de demora que tendrá un paquete en la misma. Para calcular este valor, se toma una muestra de un número n de paquetes que son transmitidos en la red y se hace el promedio de sus demoras.

Por otro lado, el *jitter* (variabilidad de la demora) refiere a la diferencia entre las demoras de los paquetes en una red, es decir, es la diferencia entre la demora máxima y la demora mínima que hay en esa red.

$$Variabilidad\ de\ la\ demora(red) = Max_{demora}(red) - min_{demora}(red) \tag{5.3}$$

Por ejemplo, si en una red $r1$ se toma una muestra de $n = 5$ paquetes con demoras $d1 = 5ms$, $d2 = 20ms$, $d3 = 5ms$, $d4 = 1ms$, $d5 = 4ms$ la demora de dicha red será de $7ms$ y su variabilidad de $9ms$. En cambio, en otra red $r2$ con la misma cantidad de muestras pero con demoras de $d6 = 7ms$, $d7 = 7ms$, $d8 = 7ms$, $d9 = 7ms$ y $d10 = 7ms$ se tendrá el mismo valor de demora para la red ($7ms$) pero una variabilidad de $0ms$.

En las redes de dispositivos IoT utilizadas en la industria, los valores de demora y su variabilidad suelen ser críticos. Para ejemplificar esto, se podría considerar una situa-

ción dónde una cinta transportadora transporta objetos que se mueven a una velocidad constante y se tienen sensores sobre la misma para detectar si estos objetos tienen alguna falla o desperfecto (es decir, no cumplen algún estandar de calidad), a su vez, se dispone de alguna pinza o mecanismo para sacarlos de la cinta en dicho caso. Si los sensores tardaran mucho en enviar un mensaje anunciando que un objeto está fallido, es decir, si hay mucha demora en la red de los sensores respecto a la velocidad de la cinta, este objeto podría quedar fuera del alcance de las pinzas. En este mismo caso, si la demora es baja pero la variabilidad muy grande, habrá casos donde las pinzas podrán reaccionar a tiempo y casos donde no, perdiendo la red predictibilidad al comportarse de manera distinta en situaciones equivalentes (en algunos casos las pinzas sacarán los objetos fallidos de la cinta y en otras no).

Debido a que varios mecanismos utilizados en redes para mejorar el valor del PDR de las mismas, por ejemplo el uso de retransmisiones, aumentan la demora y su variabilidad, se deben medir estos tres valores en conjunto al realizar experimentos y analizar como los mismos se relacionan.

5.1.3. Cantidad de nodos de la red utilizados

La cantidad de nodos de la red utilizados refiere a la cantidad de nodos distintos que un mismo paquete (contando al paquete original, sus réplicas y retransmisiones) atraviesa para llegar a su destino. En este trabajo, como ya se ha mencionado anteriormente, los paquetes siempre son enviados por el nodo origen y su destino es la raíz del DODAG.

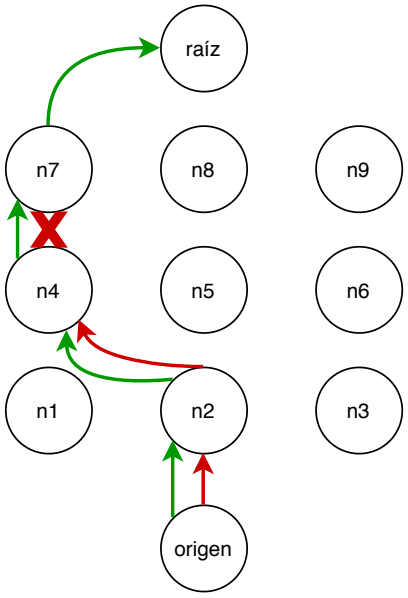

Figura 5.1: Los caminos que recorren una transmisión y su retransmisión en un ejemplo.

Por ejemplo, en la figura 5.1 donde se tiene el envío de un paquete con una retransmisión, se utilizan en total tres nodos. Notar que no se cuenta al nodo raíz, sino solamente a los nodos $n2$, $n4$ y $n7$ que conforman el camino entre el transmisor (el origen) y el receptor (la raíz) del paquete.

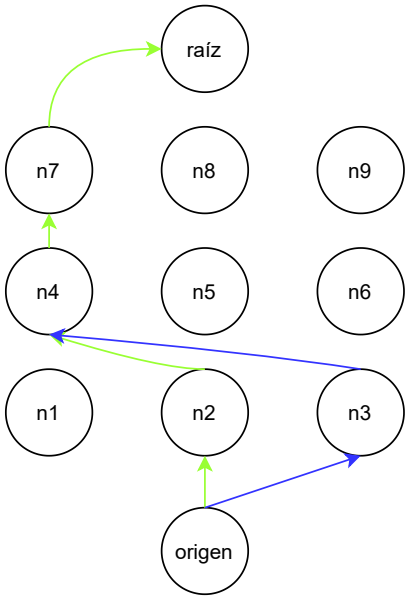

Figura 5.2: Ejemplo del envío de paquetes con réplicas dónde ocurre una convergencia

Se debe tener en cuenta que cada nodo se considera una única vez. Para ilustrar esto, en la figura 5.2 se utilizan en total cuatro nodos y no cinco, debido a que a pesar de

que dos réplicas llegan al nodo $n4$, el mismo se considera una única vez. Por otro lado, se puede observar en los ejemplos anteriores que a un mayor número de réplicas, se tiene una mayor cantidad de paquetes de la red utilizados mientras mientras que la cantidad de retransmisiones no influye en este valor considerablemente.

De la misma manera que ocurre para medir la demora en una red, para obtener la cantidad de nodos de la red utilizados de forma representativa, se toma una muestra de un número n de paquetes y se calcula el promedio entre las cantidades de nodos que cada paquete utilizó.

La cantidad de nodos de la red utilizados estará relacionada diréctamente con la cantidad de energía que esta misma red consuma, considerando a la energía que consume la red como la sumatoria de energía que consumen los nodos de la misma para realizar el envío de un paquete. Si más nodos son utilizados, más energía consumirá la red y viceversa. Siendo el consumo de energía un valor crítico para las redes en estudio, se busca que este valor sea el mínimo posible de modo de aumentar el tiempo en el que una red pueda funcionar sin necesitar re-cargarla, una operación que suele ser manual y costosa en estos contextos.

5.1.4. Cantidad de copias por paquete

La cantidad de copias por paquete es la sumatoria de la transmisión original más todas las réplicas y retransmisiones de este mismo paquete que son transmitidas, sin tener en cuenta si estas llegan o no a su destino.

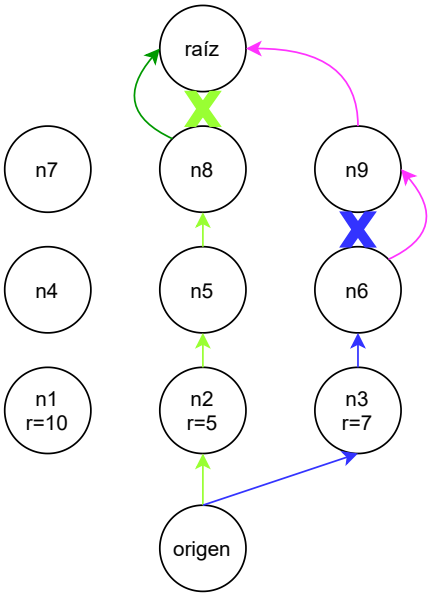

Figura 5.3: Ejemplo del envío de paquetes con réplicas y retransmisiones

Por ejemplo, en la figura 5.3, se tienen cuatro copias en total: la copia original, señalizada con las líneas verdes claras, que se pierde entre $n8$ y la raíz, su retransmisión, señalizada con las líneas verdes oscuras, enviada de $n8$ a la raíz, una réplica, señalizada con líneas azules, que se pierde entre $n6$ y $n9$, y su retransmisión, señalizada con líneas violetas, enviada de $n6$ a $n9$ y luego de $n9$ a la raíz. Se debe tener en cuenta que la cantidad de retransmisiones son las que se utilizan y no el máximo que permite la red.

Para obtener la cantidad de copias por paquete de una red, entonces, se toma una muestra de un número n de paquetes que se envían en la red y se calcula el promedio entre la cantidad de copias de cada uno.

De manera similar a como ocurre con el caso de la cantidad de nodos de la red utilizados, al utilizar una mayor cantidad de copias también aumentará el consumo de energía de la red. Esto es así debido a que cada transmisión de un paquete consume energía y se necesitarán transmitir más paquetes debido a la presencia de una mayor cantidad de copias. A su vez, la presencia de una mayor cantidad de copias puede generar fenómenos de *flooding* o saturación, generando también una mayor demora en la entrega de paquetes.

5.2. Configuración de las simulaciones

Durante este trabajo, para analizar y comparar las propuestas del mismo, se realizó experimentación via simulaciones de *software*. Se decidió utilizar este tipo de entorno debido a que resulta más fléxible y fácil de usar y de reproducir que un entorno real de *hardware*. En particular, permite en menor tiempo realizar una nueva muestra luego de introducir un cambio debido a no necesitar realizar instalaciones y, por otro lado, al ser más fácilmente reproducible, cualquier persona dentro de la comunidad podría comparar sus resultados con los de este trabajo. Las simulaciones se realizaron en COOJA, un simulador de redes desarrollado en el lenguaje de programación Java incluido en el código fuente de Contiki OS, el sistema operativo para dispositivos IoT que fue utilizado como base para la implementación de las propuestas a analizar.

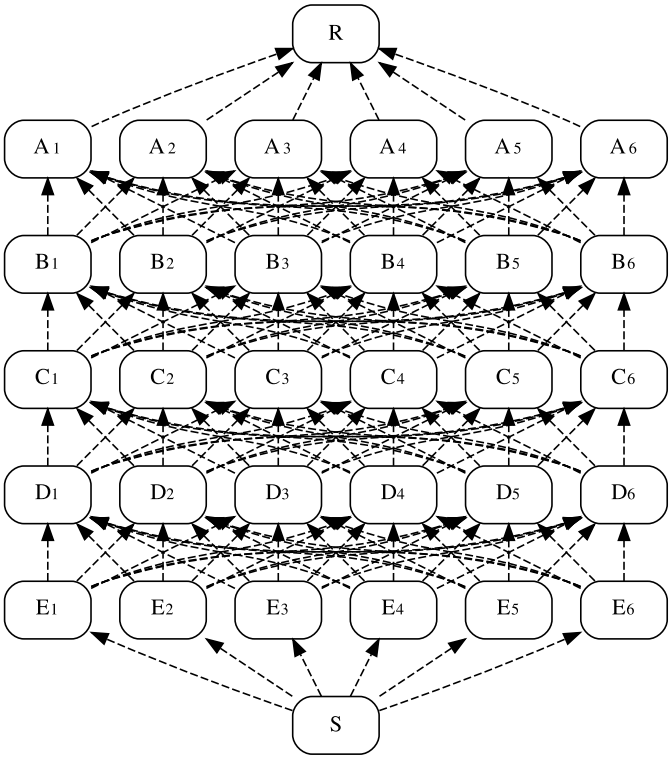

Figura 5.4: Topología utilizada para los experimentos.

En primer lugar, se utilizó una única topología de red para todos los experimentos, esta se puede observar en la figura 5.4. La misma cuenta con 32 nodos (S, R, $A1...E6$) dónde S actúa como el origen o emisor (en otras palabras, el dispositivo IoT que quiere

comunicarse con Internet, por ejemplo, un sensor) y *R* como el *border router* que se encuentra conectado a la red IPv6. Dentro de esta topología, cada uno de los nodos intermedios cuenta con cinco vecinos (por ejemplo, *A*1 cuenta con los nodos *A*2 a *A*6), formando cinco niveles (*Ai*, *Bi*, *Ci*, *Di* y *Ei*). A su vez, cada nodo se encuentra en rango de comunicación con todos los del nivel anterior y siguiente a si mismo (por ejemplo, los nodos del nivel *B* son visibles para aquellos del nivel *A* y el nivel *C*).

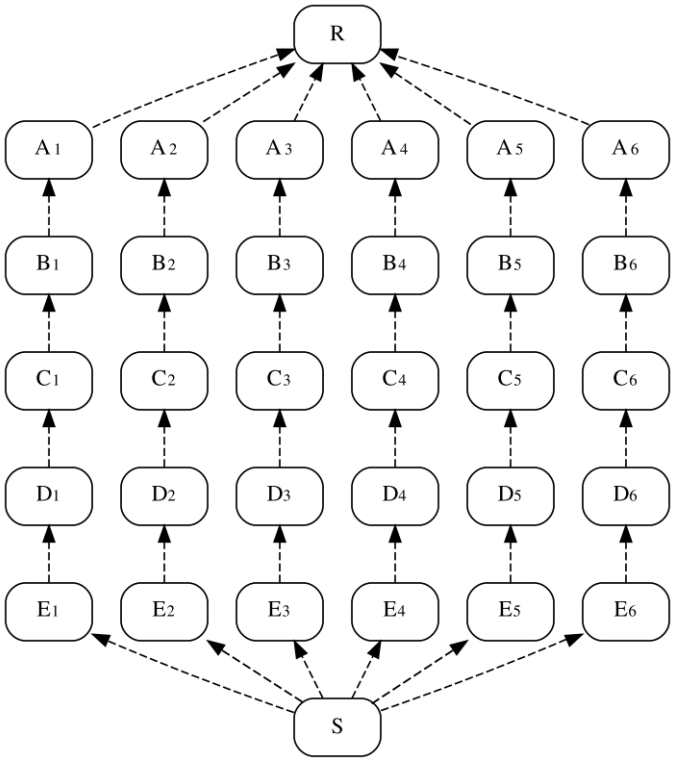

Figura 5.5: Caminos disjuntos en la topología utilizada para l0s experimentos.

Esta topología fue elegida de manera de poder maximizar la cantidad de caminos paralelos disjuntos disponibles que pueden utilizar los algoritmos propuestos en este trabajo. Como puede observarse en la figura 5.5, se tienen un máximo de seis caminos disjuntos si se utilizaran cinco réplicas.

Por otro lado, para TSCH se utilizó un registro centralizado y estático, esto significa que en vez de tener un nodo coordinador que genere el registro a utilizar y lo comunique al resto de los nodos de la red (como ocurriría en una versión centralizada normal) o que cada nodo genere su propio registro (como ocurriría en una versión descentralizada), todos los nodos obtienen como parámetro uno preestablecido. Esto se pudo

configurar de dicha manera debido a que la topología también es estática y conocida de antemano. TSCH, como fue préviamente explicado en el capítulo 3, utiliza un registro que se repite en el tiempo para coordinar la transmisión y recepción de paquetes. En este registro se le asignan celdas a los enlaces para transmitir y/o recibir paquetes, donde las celdas son pares dados por un intervalo de tiempo y un canal a usar para esa transmisión o recepción.

En específico, se usó un único canal y cada nodo cuenta con dos celdas para cada uno de sus vecinos (por ejemplo, $E1$ tendrá asignada dos celdas para cada uno de los nodos $D1$, $D2$, $D3$, $D4$, $D5$ Y $D6$). De esta manera, los paquetes originales y sus réplicas podrán ser transmitidos en la primer celda asignada a cada vecino mientras que si una de estas transmisiones fallara y la cantidad de retransmisiones permitidas fueran mayor a uno, su primera retransmisión se enviaría utilizando la segunda celda. En caso de haber una segunda falla y retransmisiones disponibles, este tercer intento se transmitiría en la primer celda al repetirse el registro y así sucesivamente. Se decidió utilizar esta configuración de manera de mantener las demoras bajas en el caso de necesitar utilizar una retransmisión sin estirar demasiado el registro, al agregar celdas que se utilizarían solo en pocos casos.

En el caso de la cantidad de retransmisiones utilizadas, se ejecutaron las simulaciones con valores de 0, 1, 3 y 7, buscando forzar estos "saltos" de celda y las esperas a las repeticiones del registro mencionados anteriormente. A su vez, para las réplicas, los valores fueron de 0, 1, 2, 3, 4 y 5, es decir, todos los valores posibles para esta topología. En la topología mostrada en la figura 5.4 el nodo origen S tiene seis posibles padres, $E1$ a $E6$, con la cantidad de réplicas igual a cinco el mismo enviará el paquete original a su padre preferido y una copia a cada uno de sus cinco padres alternativos.

Además, se utilizaron calidades de enlaces estáticas de 50 % y 75 % entre todos los nodos con el objetivo de comparar a los algoritmos en entornos de mayor y menor probabilidad de fallas. Estas calidades hacen referencia a la tasa de pérdidas de paquete que tiene un enlace. Una calidad de n % implica que en el n % de los casos los paquetes que utilizan ese enlace se transmitirán de forma correcta en el mismo. Se utilizaron estos valores en particular debido a la definición de una red saludable de dispositivos IoT dada

por Dust Networks[1] donde cada dispositivo debería tener al menos una calidad de enlace de 50 % con sus vecinos.

Por último, en cada simulación se enviarón 250 paquetes desde en nodo origen (*S* en la topología ya mostrada en la figura 5.4) y cada escenario se ejecutó veinte veces, dando un total de 5000 paquetes enviados en cada uno de los 96 escenarios.

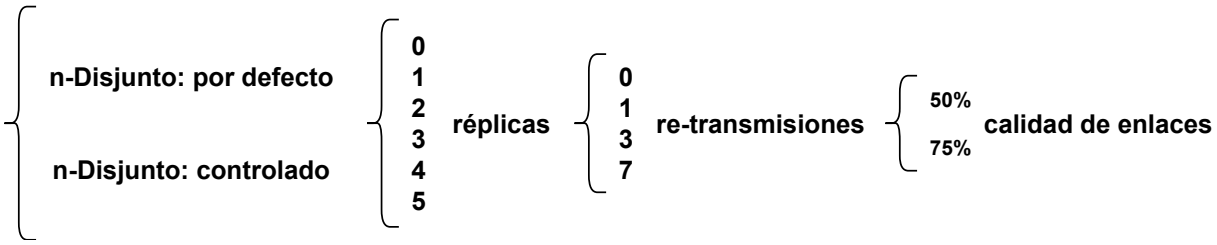

Figura 5.6: Escenarios utilizados para la experimentación.

En la figura 5.6 se pueden observar todas las combinaciones posibles de los valores utilizados en los experimentos que dan origen a todos los escenarios. Por ejemplo, uno de los escenarios es para el algoritmo "n-Disjunto: por defecto" con un valor de 4 para réplicas, 7 para retransmisiones y 50 % para la calidad de los enlaces, siendo otro de los escenario el mismo algoritmo y la misma calidad de los enlaces pero con valores 1 para réplicas y 0 para retransmisiones.

5.3. Resultados de los experimentos

En las siguientes figuras se muestran los resultados de los experimentos realizado. En todas ellas se encuentran a la izquierda los valores obtenidos para el algoritmo "n-disjunto: por defecto", en tonos naranjas, y hacia la izquierda los obtenidos para el algoritmo "n-disjunto: controlado", en tonos verdes. La cantidad de réplicas se marca en el eje horizontal de los gráficos y la cantidad de retransmisiones según los patrones señalizados hacia la derecha de los mismos. Por último, los escenarios que utilizan un 50 % de calidad de servicio se encuentran en la primer fila y los que utilizan un 75 %, en la segunda. De esta manera, cada "barra" refleja uno de los escenarios de la figura 5.6.

[1]SmartMesh IP Application Notes, Linear Technology Corp. 2012-2016.

Se debe tener en cuenta, además, que aquellos escenarios con cero retransmisiones y cero réplicas corresponden al protocolo RPL por defecto y que los mismos son equivalentes para ambos algoritmos. Graficarlos en los dos casos fue una elección para facilitar la comparación de los protocolos estandar con cada una de las propuestas.

5.3.1. Packet Delivery Ratio (PDR)

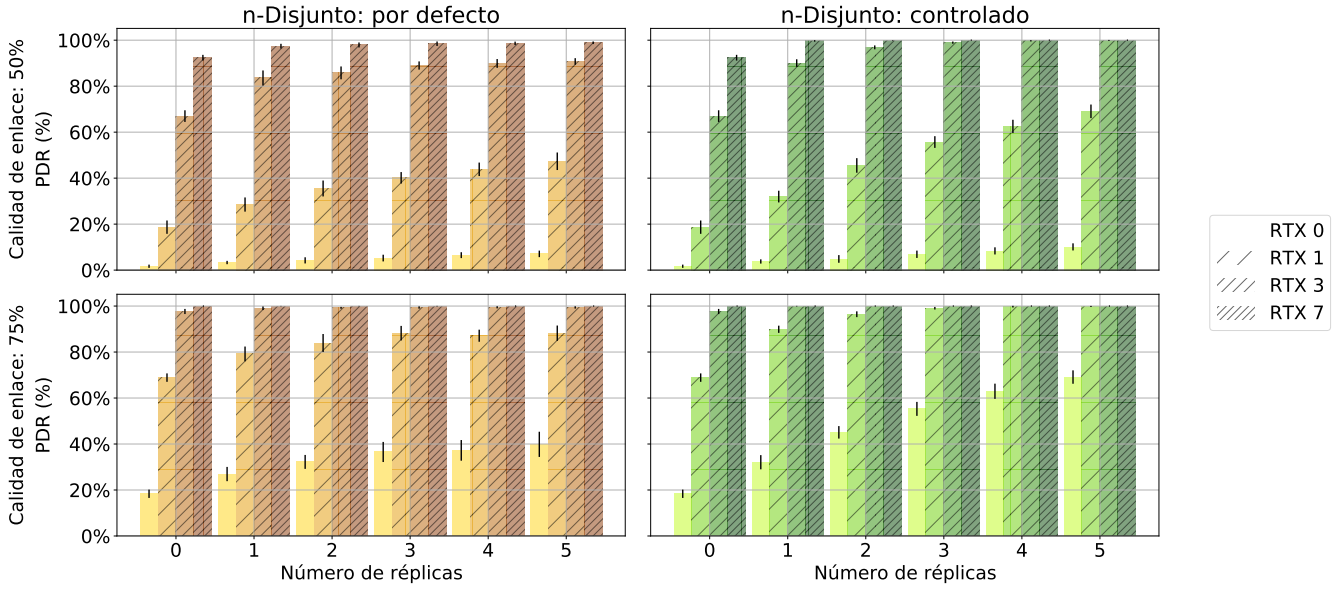

Figura 5.7: Resultados de PDR obtenidos durante el experimento.

En la figura 5.7 se encuentran los valores de PDR obtenidos durante la experimentación. En primer lugar, se puede observar que los mismos aumentan de forma líneal al aumentar la cantidad de réplicas. Este resultado era el esperable, debido a que al enviar una réplica más se está agregando una oportunidad más de que el paquete llegue de forma exitosa a su destino, aumentando el valor del PDR. De forma análoga, lo mismo ocurre al aumentar la cantidad de retransmisiones. En segundo lugar, es fácil ver, además, que la calidad de los enlaces de la red influye de forma directa en los valores de PDR obtenidos.

Por otro lado, los valores obtenidos al utilizar la versión “controlada“ del algoritmos son mayores a los obtenidos con la versión “por defecto“. Esto último deja en evidencia que en la topología utilizada RPL le asigna a varios nodos el mismo padre preferido, generando convergencia de caminos. Si no hubiera convergencia de caminos,

el algoritmo “controlado“ debería comportarse de igual manera que el “por defecto“.

Se puede destacar, además, lo bajo que es el resultado obtenido para el protocolo RPL por defecto, obteniendo menos de 10 % de PDR con una calidad de enlaces de 50 % cuando esta calidad es considerada normal para una red de dispositivos IoT saludable.

5.3.2. Demora y su variabilidad

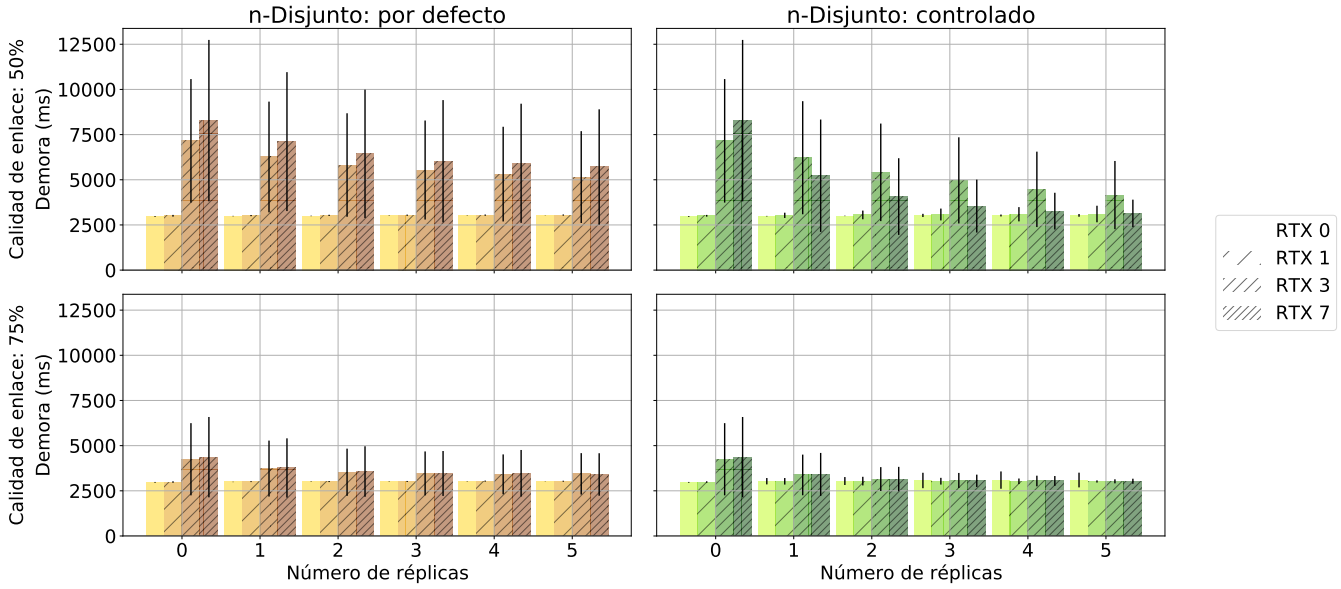

Figura 5.8: Resultados de demora obtenidos durante el experimento.

En la figura 5.8 se encuentran los resultados de demora obtenidos durante el experimento. Como fue enunciado anteriormente en este trabajo, al aumentar la cantidad de retransmisiones, aumentará la demora y su variabilidad mientras que la introducción de más réplicas no tendrá este efecto o el mismo será despreciable. Este efecto está explícito en la figura anterior.

Un resultado interesante que puede observarse es que al aumentar la cantidad de réplicas para un valor fijo de retransmisiones, la demora y su variabilidad bajan. Esto puede explicarse debido a que a un mayor número de réplicas se necesitarán menos retransmisiones para que los paquetes lleguen de forma exitosa a su destino, de manera que la demora que las mismas introduzcan será menor. Se puede corroborar este fenómeno en la figura 5.7, donde la diferencia del PDR al usar un mayor número de retransmisiones

disminuye al aumentar la de réplicas.

Por último, queda en evidencia que los valores obtenidos son mayores al disminuir la calidad de los enlaces. Esto es así ya que se necesitarán más intentos, por consiguiente, más retransmisiones, para poder entregar los paquetes, de manera que se introducirá más demora y variabilidad.

5.3.3. Cantidad de nodos de la red utilizados

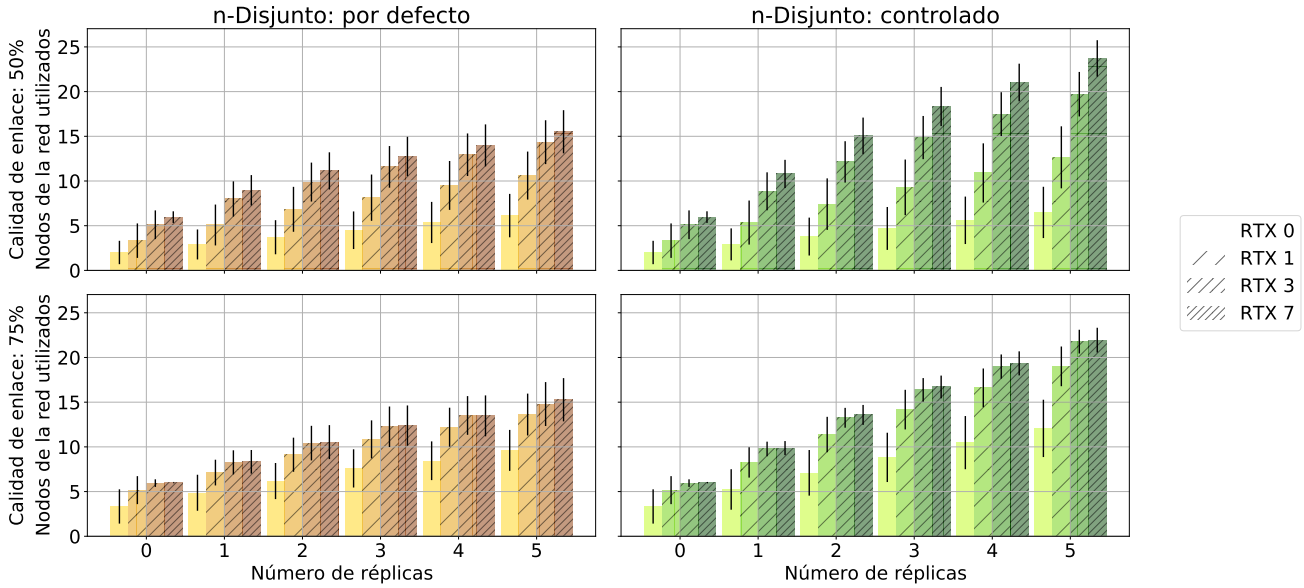

Figura 5.9: Resultados de cantidad de nodos utilizados obtenidos durante los experimentos.

En la figura 5.9 se pueden observar la cantidad de nodos de la red que utilizó cada escenario. Como era esperable, al aumentar la cantidad de réplicas aumenta la cantidad de nodos de la red utilizados. Además, estos valores son mayores en el caso del algoritmo "controlado" debido a que se tienen mecanismos de corrección ante la convergencia de caminos, lo que evita que de dos o más réplicas solo una atraviese la red.

En segundo lugar, aumentar la cantidad de retransmisiones también aumenta los valores obtenidos, especialmente cuando las mismas se usan junto con las réplicas. Esto se debe a que esta convinación de los dos mecanismos aumenta el PDR, como pudo observarse en la figura 5.7, generando que más paquetes recorran todo el camino hasta su destino, de manera de pasar por más nodos de la red. Esta relación con el PDR

también explica las diferencias obtenidas al disminuir la calidad de los enlaces entre las dos propuestas.

5.3.4. Cantidad de copias por paquete

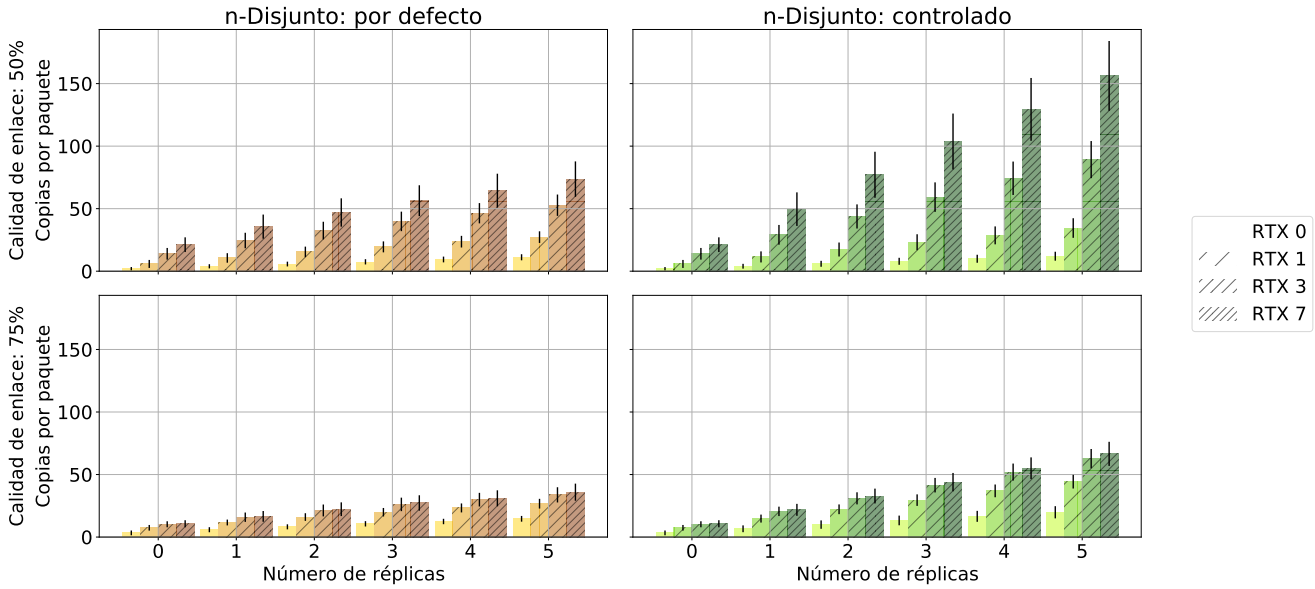

Figura 5.10: Resultados de cantidad de copias obtenidos durante la experimentación.

En la figura 5.10 se ven los resultados obtenidos para la cantidad de copias por paquete. En primer lugar, se puede observar que la cantidad de copias aumenta al aumentar la cantidad de réplicas y de retransmisiones. Esto se puede explicar debido a que cada réplica agrega una copia más en la red y que cada retransmisión agrega una copia por cada paquete que se envia (contando a las réplicas y al paquete original).

Además, de manera similar que en los gráficos anteriores, la versión "por defecto" presenta valores menores que la "controlada". Esto es debido a que, como ya fue mencionado anteriormente, al haber convergencia de caminos en esta primer versión, no todas las réplicas recorren la totalidad de la red, de esta manera, se tienen menos nodos en los que se realizan retransmisiones, agregando menos copias por paquete.

Por otro lado, se puede observar que la cantidad de copias es mayor al usar la calidad de enlace más baja. Este fenómeno es a causa de que en los escenarios con peor calidad de enlace, se necesitarán más retransmisiones por salto para que los paquetes

lleguen de forma correcta, de manera que se tendrán más copias por paquete.

5.3.5. Comparación entre las propuestas

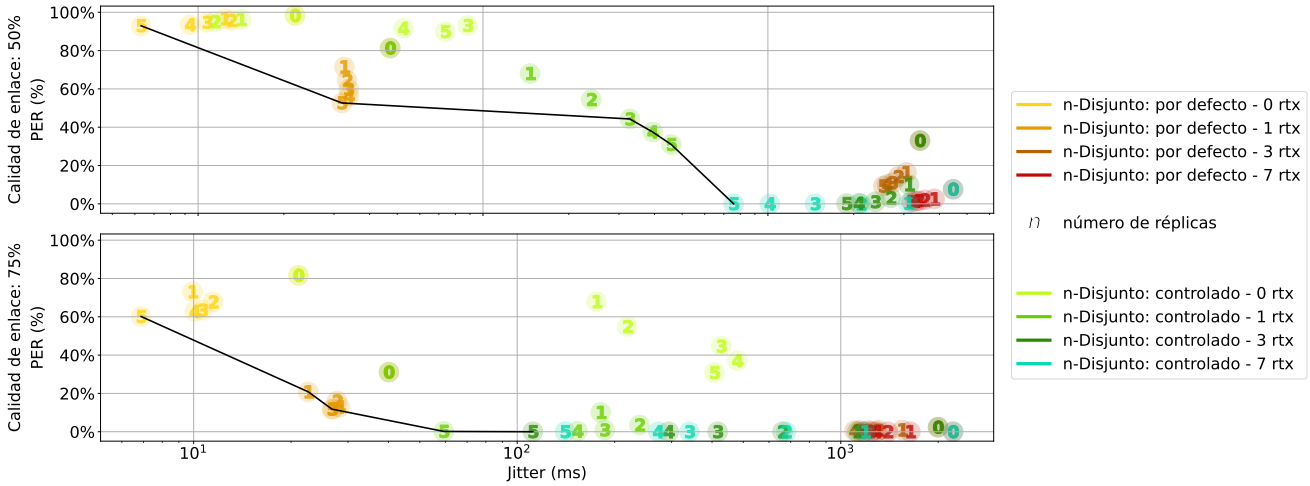

Figura 5.11: Valores obtenidos de PER contra *jitter* (demoras en la entrega) para todas las simulaciones. En este gráfico cada círculo representa una simulación distinta, donde su color refiere a si la misma es la versión controlada o por defecto del algoritmo y a la cantidad de retransmisiones utilizadas, por otro lado, su número denomina la cantidad de réplicas usadas.

Con el objetivo de poder comparar a las propuestas y a las diferentes configuraciones usadas para las mismas durante esta experimentación, se realizó la figura 5.11. Este gráfico muestra a la vez los valores obtenidos para el *Packet Error Ratio* (relación de error de paquetes, PER por sus siglas en inglés) y para el *jitter* o la variabilidad en la demora de entrega de paquetes para todos los escenario enunciados en la figura 5.6. El PER es el valor contrario al PDR y se puede calcular como:

$$PER = 100\,\% - PDR \tag{5.4}$$

Es decir, a mayor PDR, menor PER y viceversa. Entonces, valores más pequeños de PER serán equivalente a una mejor calidad de servicio (más paquetes entregados de forma correcta). La decisión de graficar esta métrica y no al PDR recae sencillamente en la claridad del gráfico generado.

En la figura 5.11 cada círculo representa un escenario distinto. Los colores amarillos, marrones y rojos refieren al algoritmo "por defecto" y los verdes y turquesas al "controlado", cada gama de color marca la cantidad de retransmisiones que utiliza este escenario. A su vez, cada círculo tiene en su centro un número que determina la cantidad de réplicas que se uso en cada caso. Por último, la línea negra representa a la frontera de Pareto, la misma marca el punto de equilibrio entre los dos parámetros que se buscan optimizar, esto es, para los pares PER-*jitter* que están sobre esa línea, no es posible obtener un mejor valor de PER sin obtener uno peor de *jitter* y viceversa.

Tomando un ejemplo, si se tuviera un escenario donde la calidad de los enlaces es del 75 % y la prioridad fuera el tener el valor de *jitter* más bajo posible con un PER menor al 20 %, se podría optar por el algoritmo "por defecto", utilizando una retransmisión y una o cinco réplicas. En cambio, si se tuvieran las mismas condiciones y una calidad de servicio de 50 %, se debería optar por el algoritmo "controlado", con siete retransmisiones y cinco réplicas. Este caso deja en evidencia como dependiendo de las condiciones del entorno y de los aspectos de una red que se busquen priorizar, que algoritmos serán mejores para dichos casos, si la comparación se hubiera hecho con valores de energía y demoras en la entrega, las elecciones seguramente hubieran sido distintas.

Capítulo 6

Conclusiones

Durante este trabajo se propusieron dos algoritmos basados en el uso de caminos disjuntos y en los mecanismos de retransmisión y de replicación para redes de dispositivos IoT. Estos algoritmos son: “n-Disjunto: por defecto“ y “n-Disjunto: controlado“. Durante los experimentos realizados se pudo comprobar que ambos cumplen con los requerimientos de calidad de servicio, específicamente los mismos tienen una alta confiabilidad y una baja demora y variabiliadad en la entrega de los paquetes; requerimientos bajo los cuales los mismos fueron pensados. A su vez, se corroboró que la versión mejorada o “controlada“ atenua los escenarios perjudiciales de la versión “por defecto“ donde la cantidad de caminos disjuntos aprovechados se ve reducida por la convergencia de caminos.

En contraposición, estos beneficios en la calidad de servicio traen como consecuencia negativa un mayor uso de recursos de la red. En específico, se observó un aumento en la cantidad de nodos de la red utilizados y en la cantidad de energía consumida. En base a estas observaciones y gracias a la comparación realizada entre ambas propuestas con distintas configuraciones y con los algoritmos estándares utilizados hoy en la industria, se puede inferir que es necesario a la hora de decidir una solución para una red en específico, definir cuáles son los parámetros a priorizar para dicha solución. Dependiendo si se quiere anteponer o favorecer la calidad de servicio o la vida util de la red, se deberá utilizar uno u otro algoritmo con una configuración específica.

Por último, el trabajo realizado en esta tesis junto con sus resultados pudo ser publicado en el 2020 IEEE Symposium on Computers and Communications(ISCC) [23]

Bibliografía

[1] Kurose F. James and Ross Keith. *Computer Networking: A Top-Down Approach*. Pearson, 2017.

[2] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff. A brief history of the internet. *SIGCOMM Comput. Commun. Rev.*, 39(5):22–31, October 2009.

[3] Vasja Roblek, Maja Meško, and Alojz Krapež. A complex view of industry 4.0. *SAGE Open*, 6(2):2158244016653987, 2016.

[4] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. Industry 4.0. *Business Information Systems Engineering*, 6:239–242, 08 2014.

[5] Shivangi Vashi, Jyotsnamayee Ram, Janit Modi, Saurav Verma, and Chetana Prakash. Internet of things (iot): A vision, architectural elements, and security issues. pages 492–496, 02 2017.

[6] Somayya Madakam, R Ramaswamy, and Siddharth Tripathi. Internet of things (iot): A literature review. *Journal of Computer and Communications*, 3:164–173, 04 2015.

[7] Borja Martinez, Cristina Cano, and Xavier Vilajosana. A square peg in a round hole: The complex path for wireless in the manufacturing industry. *IEEE Communications Magazine*, 57:109–115, 04 2019.

[8] Vehbi Gungor and Gerhard Hancke. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *Industrial Electronics, IEEE Transactions on*, 56:4258 – 4265, 11 2009.

[9] Ethan Grossman. Deterministic Networking Use Cases. RFC 8578, 2019.

[10] P. Minet, I. Khoufi, and A. Laouiti. Increasing Reliability of a TSCH Network for the Industry 4.0. *In Proceedings of the 16th IEEE International Symposium on Network Computing and Applications (NCA)*, 2017.

[11] Remous-Aris Koutsiamanis, Georgios Papadopoulos, Tomas Lagos Jenschke, Pascal Thubert, and Nicolas Montavont. Meet the PAREO Functions: Towards Reliable and Available Wireless Networks. In *IEEE International Conference on Communications (ICC)*, Dublin, Ireland, June 2020.

[12] T. L. Jenschke, G. Z. Papadopoulos, R. Koutsiamanis, and N. Montavont. Alternative Parent Selection for Multi-Path RPL Networks. In *Proceedings of the IEEE 5th World Forum on Internet of Things (WF-IoT)*, Limerick, Ireland, April 2019.

[13] Erfan Mozaffari Ahrar, Mohammad Nassiri, and Fabrice Theoleyre. Multipath aware scheduling for high reliability and fault tolerance in low power industrial networks. In *Journal of Network and Computer Applications*, pages 25–36, September 2019.

[14] Y. S. Lohith, T. S. Narasimman, S. V. R. Anand, and M. Hedge. Link peek: A link outage resilient ip packet forwarding mechanism for 6lowpan/rpl based low-power and lossy networks (llns). In *2015 IEEE International Conference on Mobile Services*, pages 65–72, 2015.

[15] Roger Alexander, Anders Brandt, JP Vasseur, Jonathan Hui, Kris Pister, Pascal Thubert, P Levis, Rene Struik, Richard Kelsey, and Tim Winter. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, March 2012.

[16] Pascal Thubert. Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL). RFC 6552, March 2012.

[17] Dominique Barthel, JP Vasseur, Kris Pister, Mijeom Kim, and Nicolas Dejean. Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks. RFC 6551, March 2012.

[18] IEEE Standard for Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011), April 2016.

[19] WirelessHART Specification 75: TDMA Data-Link Layer. *HART Communication Foundation Std., Rev. 1.1,*, HCF SPEC-75, 2008.

[20] ANSI/ISA-100.11a-2011: An ISA Standard, Wireless systems for industrial automation: Process control and related applications.

[21] X. Vilanosa, T. Watteyne, M. Vučinić, T. Chang, and K. Pister. 6TiSCH: Industrial Performance for IPv6 Internet-of-Things Networks. In *Proceedings of the IEEE*, pages 1153–1165, June 2019.

[22] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN)*, 2004.

[23] A. Czarnitzki Estrin, T. Lagos Jenschke, G. Z. Papadopoulos, J. Ignacio Alvarez-Hamelin, and N. Montavont. Thorough Investigation of multipath Techniques in RPL based Wireless Networks. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, 2020. doi: 10.1109/ISCC50000.2020.9219646.