UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

ANÁLISIS COMPARATIVO DE PROTOCOLOS FRAME SLOTTED-ALOHA EN REDES DE SENSORES

Jorge Cristóbal Ascaso

Tutor: Jorge Martínez Bauset

Cotutor: Miguel Ángel Rodríguez Hernández

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingeniería de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2020-2021

Valencia, 17 de septiembre de 2021

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS
VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS
Gestión ambiental verificada
REG.NO. ES-CV- 000030

Agradecimientos

A mis padres y a mi hermana Inés, porque sin vosotros no se podría haber recorrido este camino.

A mi tutor, Jorge Martinez-Bauset. Sin tu paciencia y ayuda este último paso nunca se hubiera podido dar.

A Miguel Ángel Rodríguez. Por su inestimable ayuda en la realización de este trabajo y en conseguir que el maldito filtro funcionase.

A mis profesores. Su guía, dedicación y enseñanzas han sido la causa necesaria para seguir en este ir y venir de ecuaciones, protocolos y peleas con mi propio conocimiento.

A mis hermanos de otra sangre, Guillem y Thomas. Porque me habéis demostrado que en cualquier recodo del camino puedes encontrar familia para siempre.

A Vera. Sin tu profesionalidad, ayuda y palabras no podría haber ganado las luchas que en mi mismo se tejían mientras recorría este sendero.

Y gracias por último a todos los que hayan andado conmigo estos años, sea un ratito o sigáis ahí. Vuestro granito en la construcción de lo que soy sigue ahí, aunque ya no transitemos los mismos parajes.

Resumen

El objetivo del TFG es realizar un análisis de prestaciones comparado de dos protocolos de acceso al medio (MAC) utilizados en redes de sensores: Frame Slotted-ALOHA with Reservation and Data Packets (FSA-RDP) y FSA-DQ (Distributed Queuing). El protocolo FSA-RDP requiere un mecanismo de access barring adaptativo con la carga que, en caso de congestión, excluya una parte de los nodos que contienden enviando de paquetes de reserva, consiguiendo así maximizar el número de éxitos en el acceso. El ajuste de este mecanismo debe ser realizado por el controlador de la red y puede ser complejo. En cambio, el protocolo FSA-DQ utiliza un mecanismo de cola distribuida que, por sí mismo, se adapta a las condiciones de carga. Los parámetros de mérito que se utilizarán para la evaluación son: la probabilidad de pérdida de paquetes, el caudal cursado y el retardo de acceso. El estudio se realizará mediante simulación por eventos discretos.

Palabras clave: Redes de sensores, tráfico, evaluación de prestaciones, Frame Slotted-ALOHA, Distributed Queuing

Resum

L'objectiu del TFG és realitzar una anàlisi de prestacions comparat de dos protocols d'accés al medi (MAC) utilitzats en xarxes de sensors: Frame Slotted-ALOHA with Reservation and Data Packets (FSA-RDP) i FSA-DQ (Distributed Queuing). El protocol FSA-RDP requereix un mecanisme d'access barring adaptatiu amb la càrrega que, en cas de congestió, excloga una part dels nodes que contenen enviant de paquets de reserva, aconseguint així maximitzar el nombre d'èxits en l'accés. L'ajust d'aquest mecanisme ha de ser realitzat pel controlador de la xarxa i pot ser complex. En canvi, el protocol FSA-DQ utilitza un mecanisme de cua distribuïda que, per si mateix, s'adapta a les condicions de càrrega. Els paràmetres de mèrit que s'utilitzaran per a l'avaluació són: la probabilitat de pèrdua de paquets, el cabal cursat i el retard d'accés. L'estudi es realitzarà mitjançant simulació per esdeveniments discrets.

Paraules clau: Xarxes de sensors, trànsit, avaluació de prestacions, Frame Slotted-ALOHA, Distributed Queuing

Abstract

The objective of the TFG is to perform a comparative performance analysis of two medium access protocols (MAC) used in sensor networks: Frame Slotted-ALOHA with Reservation and Data Packets (FSA-RDP) and FSA-DQ (Distributed Queuing). The FSA-RDP protocol requires an adaptive access barring mechanism with the load that, in case of congestion, excludes part of the nodes that contend sending reservation packets, thus managing to maximize the number of access successes. The adjustment of this mechanism is done by the network controller and can be complex. Instead, the FSA-DQ protocol uses a distributed queuing mechanism that adapts to load conditions by itself. The performance parameters of merit that will be used for the evaluation are: the packet loss probability, the packet throughput and the access delay. Discrete-event simulation will be used for the study.

Paraules clau: Sensor networks, traffic, performance evaluation, Frame Slotted-ALOHA, Distributed Queuing

Índice de contenidos

Capítulo 1. Índices de siglas, figuras y tablas.

1.1 Índice de siglas

1.2 Índice de figuras

1.3 Índice de tablas

Capítulo 2. Motivación, objetivos y metodología del trabajo.

2.1 Motivación.

Debido a la llegada de nuevas tecnologías y aplicaciones relacionadas con *Internet of Things* han aumentado bastante las poblaciones de dispositivos de sensorización, comunicación y gestión de los espacios y servicios. Esta tendencia en aumento del número de dispositivos genera una serie de desafíos a la hora de desarrollar las redes de acceso que presten servicio a dichos dispositivos. Los problemas en el acceso de los muchos dispositivos, la diferencia entre el tráfico humano y el generado por máquinas, la baja potencia de los sensores o la dispersión geográfica para realizar los servicios son algunos de los desafíos a superar.

Se requiere seguir investigando en mejores soluciones para el acceso y la gestión de los miles de dispositivos que se espera que converjan en este paradigma (*Internet of Things*). Los dos protocolos propuestos a comparativa buscan mejorar las capacidades de acceso de los distintos dispositivos y un uso más eficiente del canal inalámbrico de transmisión dentro de redes de sensores inalámbricos altamente pobladas.

El trabajo busca realizar una comparación entre dichos protocolos y proponer alguna mejora para ellos, ayudando así a los esfuerzos de la comunidad científica en la búsqueda de protocolos más eficientes y útiles en este nuevo paradigma tecnológico como es *Internet of Things*.

2.2 Objetivos

1. Revisar y proponer un marco tecnológico en el que trabajan los protocolos a estudio.
2. Explicar el funcionamiento de ambos protocolos.
3. Comparar cuantitativamente las capacidades de ambos mediante simulación orientada a eventos.
4. Proponer y evaluar cuantitativamente una mejora en el funcionamiento de uno de ellos.

2.3 Metodología.

La metodología usada en un primer lugar será un estudio deductivo del ecosistema tecnológico en el que se encuentran los dos protocolos que se van a comparar. En nuestro caso, bajo el paradigma de *Internet of Things* (Internet de las cosas) dentro del paraguas de las redes de acceso celulares de quinta generación (5G).

Tras ello se explica el funcionamiento de los dos protocolos que se llevarán a estudio en la comparación cuantitativa mediante simulación orientada a eventos.

Posteriormente se realiza un trabajo de diseño y propuesta de mejora a uno de los protocolos para automatizar uno de los parámetros configurables.

Finalmente se realiza una comparación cuantitativa de los dos protocolos. Esta comparación se realiza mediante datos extraídos por simulación orientada a eventos. También se realiza un estudio cuantitativo del funcionamiento de la mejora propuesta para observar si su funcionamiento es como el esperado.

2.4 Escenario de trabajo.

Una visión esquemática del escenario en el que se desarrolla este trabajo es el que se muestra en la siguiente figura donde una serie de sensores se conectan a sendos *gateways* para acceder a la red de celular.

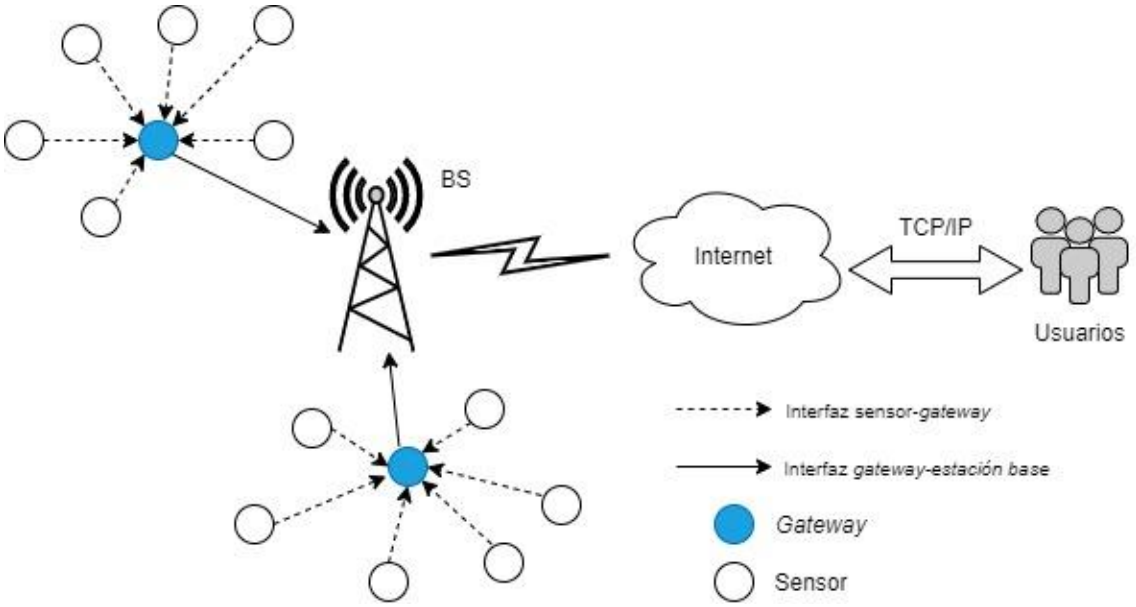

Figura 1. Modelo básico que se estudia en este trabajo.

El trabajo se centrará en el interfaz entre sensor y *gateway*. Los dos protocolos que se estudiarán están encargados de gestionar el acceso al interfaz aéreo entre estos dos elementos (los sensores y el *gateway*) buscando mejorar el uso de los recursos disponibles.

Estos protocolos vienen enmarcados en la llegada del paradigma tecnológico de IoT y la proliferación de las comunicaciones *machine-to-machine.* Por ello en la Figura 1 los *gateways* se conectan a una estación base de una red celular, para poder generar servicios y soluciones que sean accesibles mediante Internet.

El trabajo comparará cuantitativamente el funcionamiento de los dos protocolos con distinto número de sensores conectados al *gateway* para comparar sus capacidades en distintos entornos de trabajo. También se trabajará con distintas configuraciones de los protocolos para observar su mejora o detrimento.

Capítulo 3. Introducción.

3.1 Machine-to-machine communications.

Las comunicaciones M2M (*machine-to-machine*) son las comunicaciones que se generan durante la interacción autónoma entre dispositivos que desarrollan un proceso o prestan un servicio. [1]

El origen de esta tecnología se encuentra en los sistemas de control y adquisición de datos (SCADA) donde sensores toman información sobre el estado de un entorno, originariamente industrial, y una serie de controladores monitorizan y controlan los procesos (industriales). La mejora paulatina en las capacidades de los elementos, la reducción del tamaño y la mejora del rendimiento en general del hardware de comunicaciones ha permitido la expansión de las soluciones basadas en comunicaciones M2M. Además de las soluciones industriales podemos encontrarnos en la actualidad servicios para la mejora de la agricultura, desarrollo de las capacidades militares, servicios municipales, gestión sanitaria, etc. Debido a los entornos en los que trabaja esta tecnología podemos extraer una serie de características de estas comunicaciones:

a) *Arquitecturas y capacidades heterogéneas.* Debido a que muchos de los desarrollos han sido propietarios y adaptados a las necesidades específicas del proceso o servicio no se ha desarrollado una pila homogénea de protocolos para este tipo de comunicaciones. El tráfico que se genera y se transmite por la red no es igual entre las distintas agrupaciones de dispositivos.
b) *Baja movilidad o ninguna.* En la mayoría de los casos, una vez instalada la infraestructura no se espera movilidad de los dispositivos.
c) *Tráfico a ráfagas.* En muchos de los casos (ej. monitorizando una zona de exclusión aérea), cuando se activa un dispositivo lo suelen hacer otros al mismo tiempo haciendo que se sucedan largos tiempos de inactividad con otros momentos puntuales en los que se genera tráfico por parte de los dispositivos.
d) *Prioridad de envío.* Las soluciones pensadas para emergencias, usos militares, etc. Requieren que el tráfico generado tenga alta prioridad en la red de acceso y así conseguir mínima latencia en el envío de los paquetes.

Por tanto, las principales diferencias entre las comunicaciones M2M y las comunicaciones humanas (H2H *communications*) es que éstas últimas requieren un gran ancho de banda y el tráfico es producido en ambas direcciones (*uplink* y *downlink*). Por otra parte, en las comunicaciones M2M se requiere poco ancho de banda y el tráfico será prácticamente en su totalidad *uplink*. Además, en las comunicaciones M2M, debido a la ocurrencia de un evento, se puede generar la sincronización en el acceso de múltiples dispositivos a la red dando lugar a episodios de sobrecarga del sistema, lo cual es necesario gestionar. [2]

Es importante señalar la importancia del desarrollo de las redes celulares para la expansión de las comunicaciones M2M en la actualidad. Al desarrollarse la capacidad de acceso a internet inalámbricamente en espacios geográficos o infraestructuras de difícil acceso, se ha permitido la monitorización y el desarrollo de servicios que antes no hubieran sido posibles. También este desarrollo ha permitido generar infraestructuras más eficientes y con menores costos de mantenimiento, al no necesitar cables para la transferencia de la información, y que soportan un mayor número dispositivos (como sensores).

A partir del siglo XXI, con el desarrollo continuo de las redes móviles y la llegada del protocolo IPv6, que permite hasta $6{,}7 \cdot 10^{17}$ direcciones por milímetro cuadrado [3], se abrió el camino para un cambio de paradigma en las comunicaciones M2M. Éstas no solo se desarrollarán en entornos industriales, de seguridad o municipales; si no que se abrirán al mercado de consumo para generar productos, servicios y procesos pensados para el ciudadano y consumidor. Este paradigma se denomina IoT (*Internet of Things*).

3.2 ¿Qué es *Internet of Things*?

En [4] se nos presenta IoT (*Internet of Things*) como un paradigma de interconexión de redes facilitada por una pila de tecnologías que proveen la conexión entre los objetos virtuales y físicos facilitando el desarrollo de servicios inteligentes y aplicaciones con capacidades auto configurables.

Dicho paradigma de interconexión permite el desarrollo de nuevos servicios, en los que los usuarios tienen la capacidad de interactuar con "cosas". Las "cosas" que abarca este concepto van desde los dispositivos y recursos necesarios para la automatización de un proceso industrial, conectados a internet para su control telemático, hasta la posibilidad de que un usuario pueda activar su caldera de calefacción antes de llegar a casa, por ejemplo mediante un *smartphone* con conectividad a Internet. Estos procesos autónomos van a generar una gran cantidad de trafico de comunicaciones M2M. Por tanto, la IoT pretende universalizar esta tecnología y acercarla al ciudadano y consumidor.

Muchas de las soluciones provenientes del concepto de IoT se basan en la capacidad de monitorizar una actividad concreta, codificar los datos, retransmitirlos por la red y, una vez recibida en centros de gestión, transformar dichos datos en información útil que permita la toma de decisiones. Por ende, podemos dividir los desafíos que plantea este nuevo paradigma en tres niveles [5]:

- *Tier 1:* Obtener la muestra de datos del medio y codificarlo.
- *Tier 2:* Transmitir la información a través de la red.
- *Tier 3:* Una vez agregados los datos obtenidos se obtiene la información y se toma una decisión respecto a ella (automatizada o dirigida por una persona)

Los protocolos que soportan la IoT están pensados para superar varios de los desafíos más importantes que aparecen al conectar un gran número de sensores con una misma red inalámbrica, creando las conocidas WSN (*Wireless Sensor Network*) [6] La conexión inalámbrica masiva de dispositivos es habitual en escenarios de aplicación del paradigma de la IoT, por ejemplo, las *Smart Cities* (ciudades inteligentes), o espacios industriales de gran envergadura donde el cableado de los dispositivos es inviable. Los protocolos de acceso al medio deben gestionar de forma eficiente el problema del acceso múltiple al canal, y las situaciones de congestión. Los problemas más comunes que produce una gestión inadecuada son:

a) *Aumento del retardo:* Se retrasa mucho la obtención de la información y en aplicaciones críticas como las militares, la del control del tráfico u otras de emergencias puede ser fatal para el desempeño del servicio.
b) *Inoperatividad de la información obtenida:* Si se retrasa mucho la información puede que no sea fatal para el desempeño del servicio, pero puede haberse vuelto inútil (demasiado tarde para reaccionar o ya se ha reaccionado previamente a la llegada de la información) y, por tanto, se ha desaprovechado los recursos utilizados en la infraestructura IoT.
c) *Consumo de energía:* Si se retrasa mucho el acceso al canal, se obliga a los sensores a mantenerse conectados reintentando el acceso. Esto aumento del consumo de energía, aumentando así el costo de éstos, tanto en consumo de red eléctrica, o en baterías. Si a esto le sumamos que en muchas ocasiones se encuentren en puntos geográficos remotos, esto puede aumentar mucho los costes de mantenimiento, por ejemplo, sustitución de baterías y/o componentes de los sensores.

3.3 Las WSN y su implicación en el desarrollo de IoT.

Según [7], las WSN's son redes compuestas por microdispositivos distribuidos con capacidad de detección (sensores), utilizados para monitorizar un entorno y enviar información a los usuarios finales. Es común que los sensores adscritos a una zona geográfica envían la información hacia la Internet a través de una BS (Base Station).

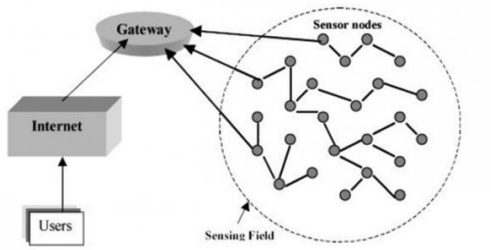

Figura 2. Esquema de una WSN conectada a una BS para enviar información a un usuario a través de internet. Fuente: [8]

Las WSN's nacieron hace más de 30 años, y como pasó con Internet, en un primer momento fue un proyecto desarrollado con fines militares [9], aunque rápidamente se vio utilidad para otros fines civiles (vigilancia de habitas o vigilancia meteorológica, por ejemplo). Desde entonces ha habido un constante desarrollo de sus posibilidades, dado el gran número de proyectos que han estado usando estas tecnologías.

El problema que subyace es la gran complejidad que tienen en su despliegue (sobre todo cuando los sensores no son dispositivos homogéneos), y en la creación de aplicaciones asociadas a dichas infraestructuras. Aun así, desde la aparición del paradigma de IoT, los investigadores se han visto en la necesidad de mejorar y acercar las WSN's a la sociedad, dejando de ser infraestructuras de "laboratorio". Así, unidas las redes de sensores con aplicaciones OTT (*Over The Top*) que proveen servicios útiles en la mejora del día a día de la población, son una de las claves del despliegue de la infraestructura IoT, y su migración del laboratorio al mercado definitivamente.

Siendo uno de los pasos previos al paradigma de IoT, este se diferencia sustancialmente en ciertos aspectos con las WSN's [7]:

1. *Aplicaciones*: A diferencia de las WSN que solo monitorizan, en IoT se prevé que haya soluciones que interactúan directamente con el medio.
2. *Dispositivos IoT*: En las WSN hay una red homogénea de sensores, en IoT se desarrolla una red de dispositivos distintos que se conectan a un mismo punto.
3. *Capacidad de comunicación.* Al no ser una infraestructura (IoT) de dispositivos homogéneos las capacidades de comunicación pueden variar entre ellos.
4. *Numero de dispositivos conectados.* En el paradigma IoT se espera un número creciente de dispositivos conectados.
5. *Objetivos de la tecnología.* Las WSN's nacieron para monitorizar un entorno y obtener información. El paradigma de IoT busca mejorar el día a día de consumidores y ciudadanos.

Aun con estas diferencias de paradigma, la existencia de las WSN's es totalmente necesaria a la hora de plantearse soluciones IoT como *Smart Cities*. El desafío nace porque el número de los sensores conectados a la red en dichas WSN va a ser muy superior en comparación con las infraestructuras de los últimos años al necesitar obtener información de amplios espacios altamente poblados. Cómo organizar la comunicación de la gran cantidad de los sensores con la BS, la heterogeneidad de estos y el uso del canal inalámbrico compartido son algunos de los retos que plantea el desarrollo del paradigma IoT dentro de lo que anteriormente denominábamos *tier 1*.

3.4 Clustering. La clave para organizar las WSN's y conectarlas a internet. [10]

Como hemos visto en la sección anterior, la existencia de las WSN's en las tecnologías IoT va a ser básica para muchas de sus soluciones. También en 3.2 vimos como el aumento de la población de nodos en la red de sensores puede provocar una serie de problemas que eviten el correcto funcionamiento del servicio o el desempeño general de la infraestructura:

a) *Aumento del retardo.*
b) *Inoperatividad de la información obtenida.*
c) *Consumo de energía.*

Una estrategia común para conectar un gran número de dispositivos a una WSN es la disposición de *gategays*. Un *gateway* es un dispositivo del conjunto de sensores, que se encarga de ejercer de interfaz entre el resto de los sensores y el punto de acceso a la red, normalmente una BS de una red celular, o un punto de acceso a una WLAN. Esto permite realizar particiones cuando se requieran grandes poblaciones de sensores por necesidades de la aplicación, consiguiendo así que el tamaño de la WSN pueda escalar y al mismo tiempo se haga un uso más eficiente de los recursos de la WSN.

Una de las soluciones más populares para realizar la partición de la población de sensores se basa en crear grupos (*clusters*) para jerarquizar el acceso al canal de comunicación. Agrupar los nodos se suele realizar en dos capas. En la primera se encuentran los sensores sin ninguna función adicional a la de tomar datos y enviarlos. De todos los nodos del *cluster* se eligen los mejores en su capacidad para comunicarse con la BS y con el resto de los nodos del *cluster*, para que sean los encargados de retrasmitir la información obtenida por el *cluster* a la BS; estos nodos funcionarán como CH (*Cluster Head*), que en la arquitectura que estudiamos, funcionarán de *gateway* entre el grupo de sensores y la estación base a la que se conecte. No se describe como se hace dicha selección, aunque sí que hay que indicar que en la actualidad se busca que sea auto-configurable debido a la difícil accesibilidad a alguna de las infraestructuras. [11]

Tras la selección del *gateway* (o CH), los nodos pertenecientes a la primera capa de la jerarquía no se comunicarán directamente con la BS. Cuando tengan información que transmitir se la enviarán al CH que se encargará de almacenarla, agregarla y eliminar los datos redundantes. Tras tener una cierta cantidad de datos el CH la transmite a la BS. Así se reduce el consumo de energía al no enviar información redundante a la red, y también se disminuye el uso del canal y, por tanto, se reduce el retardo de acceso (se limita la contienda en el acceso al canal de la BS).

La Figura 3 muestra un ejemplo. También se observa que no necesariamente cada uno de los CH tienen porqué conectarse directamente con la BS, en alguno de los algoritmos existentes puede haber una comunicación *multi-hop* entre CH's hasta la propia BS.

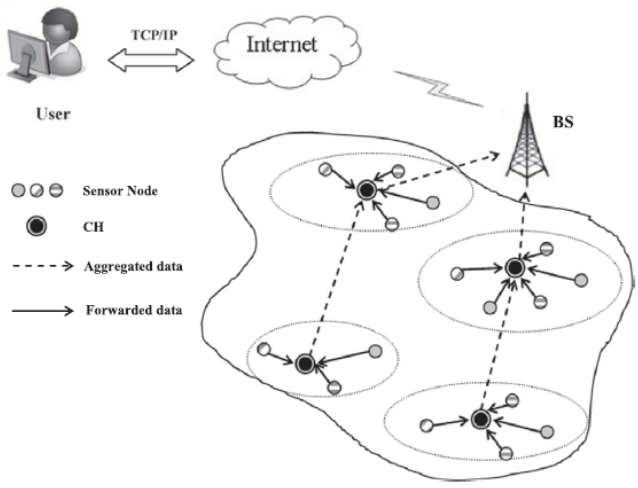

Figura 3 Una red agrupada cualesquiera. Fuente: [10]

Dado que se prevé que las infraestructuras IoT se desarrollen en grandes extensiones geográficas, como ciudades o zonas industriales, es lógico querer aprovechar las redes celulares y sus BS para interconectar los *clusters* de las WSN's a Internet. En la siguiente sección se describirán los esfuerzos de los organismos de estandarización por adecuar este tipo de comunicaciones con las infraestructuras celulares ya prexistentes, ya que éstas estaban pensadas para comunicaciones humanas cuyas características son distintas.

3.5 Las tecnologías celulares. 5G y su implicación en el desarrollo de IoT.

Como se comentó en el punto 3.2, el paradigma se divide en distintos *tiers*. Hasta ahora se ha hablado del marco de trabajo que rodea al *tier 1*, pero como la propuesta de este trabajo es la comparación de dos protocolos de acceso (que están en *tier 1*), es interesante que se trate de qué manera las tecnologías celulares se han ido adaptando a este nuevo paradigma. Es decir, analizar cómo funciona el *tier 2* en su tarea de conectar los elementos autónomos con Internet.

El paulatino desarrollo del paradigma de IoT y distintas soluciones dentro de este paradigma, ha obligado al 3GPP (*3rd Generation Partnership Project*) a seguir desarrollando y estandarizado una serie de tecnologías y protocolos para que las redes celulares sean capaces a dar servicio a los dispositivos basados en comunicaciones M2M. Las comunicaciones M2M son nombradas por el 3GPP como MTC (*machine type communications*).

Este punto se centrará en 5G dado que desde el inicio de su concepción ha sido planteada para dar servicio y superar los retos que plantea el paradigma de IoT. 5G se ofrece como una tecnología con menos coste por punto conectado, menor consumo de energía por parte de los dispositivos, y que permite el soporte a gran número de dispositivos.

Estas son los principales retos en el desarrollo de 5G que permita un despliegue eficiente de IoT [12]:

a) *Evolución de la tecnoloía de radiocomunicaciones.* Esto implica el uso de frecuencias más altas en las comunicaciones. El uso de ondas milimétricas [13] hace que la propagación y penetración sea mala en zonas exteriores, teniendo las BS que usan VHF (*Very High Frequency)* una menor cobertura. Por ello se han propuesto arquitecturas híbridas donde se encuentran BS que hacen uso de las ondas milimétricas para zonas donde se requiere alta tasa de transmisión, y tecnologías herederas de 4G para soluciones menos exigentes. En la Figura 4 se pude ver un ejemplo de ésto y también un ejemplo de una red con solo antenas que trabajan con ondas milimétricas.

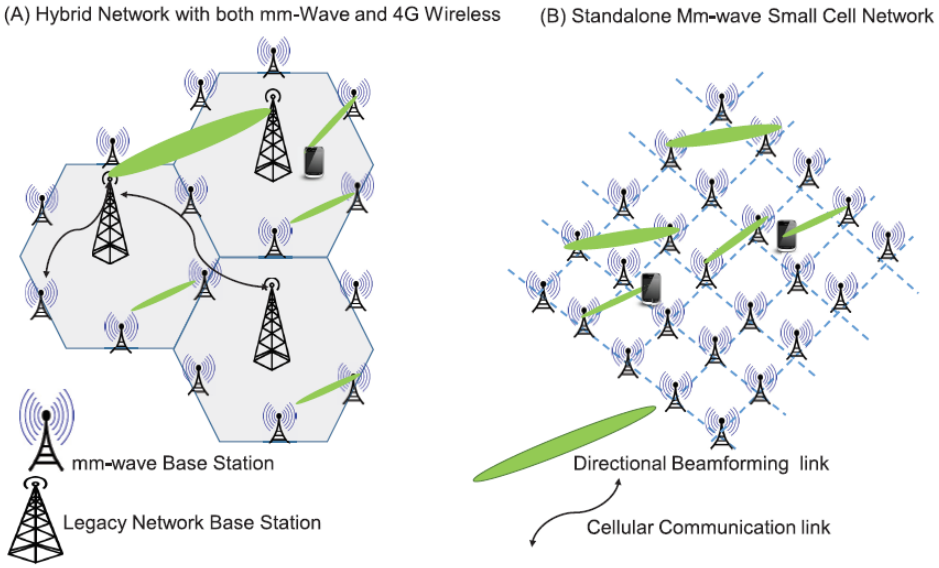

Figura 4. Ejemplo de una red híbrida (A) y una con solo antenas que trabajan en VHF. Fuente: [12]

b) *Interfaz aéreo avanzado.* Debido al uso de las ondas milimétricas se requieren antenas de un tamaño menor y por tanto se hace necesario el uso de *arrays* de antenas que puedan dar cobertura a la distribución geográfica. Debido al aumento de los arrays las antenas utilizadas dejan de tender hacia la omnidireccionalidad y tienden hacia antenas que son más direccionales para evitar las interferencias entre ellas.

c) *Nueva generación de antenas inteligentes.* Estas están diseñadas para poder dar cabida y control al uso de *arrays* de antenas y al gran número de dispositivos que a estas se tienen que conectar. Estas antenas inteligentes buscan reducir las interferencias entre ellas a la vez que mantienen el servicio y la cobertura tanto con los dispositivos como con otras estaciones base.

d) *Agilidad y resiliencia mediante SDN.* El SDN (*Software Design Network*) ofrece una solución basada en la división/separación del plano de control del de los datos. Esto se hace mediante componentes de software que se encargan del control, reduciendo el uso intensivo del hardware para ello. La interacción entre los dos planos se hace mediante interfaces abiertas en cada uno de los planos.

e) *Arquitectura centralizada mediante Cloud RAN (Radio Access Networks).* Es la solución para descentralizar el plano de los datos, a diferencia de SDN que es para remodelar el sistema de control y flexibilizarlo. Esta solución ofrece una reducción del coste de despliegue de las soluciones al centralizar y virtualizar las conexiones y los elementos en la nube. Esto también permite flexibilizar los servicios ofrecidos por las operadoras al combinarse con SDN a través de interfaces programables.

f) *Mejorar las capacidades para la conexión de redes heterogéneas.* Gracias a la plataforma de la nube para el plano de datos y la flexibilidad que ofrece SDN se abre la puerta a la integración del gran número de redes heterogéneas cuyos dispositivos generan tráfico de distinto tipo (mayor carga en subida o en bajada, trafico puntual o por ráfagas, etc.) y que tienen un tamaño geográfico y de dispositivos adscritos de distintos órdenes de magnitud.

Las anteriores mejoras vienen acompañadas por una arquitectura pensada para las MTC [14], permitiendo así una mejora en las capacidades de comunicación, interoperabilidad, escalabilidad de las redes, seguridad y control remoto. La plataforma *smartM2M*, que es la encargada de esta tarea, está basada en arboles de registro que guardan la información necesaria para gestionar y dar servicio a las distintas redes con MTC. Cada recurso (dispositivos, gateways, etc.) tiene una representación única en el árbol virtual de elementos. En la siguiente figura se puede ver una aproximación de cómo se presenta la arquitectura 5G para las comunicaciones M2M.

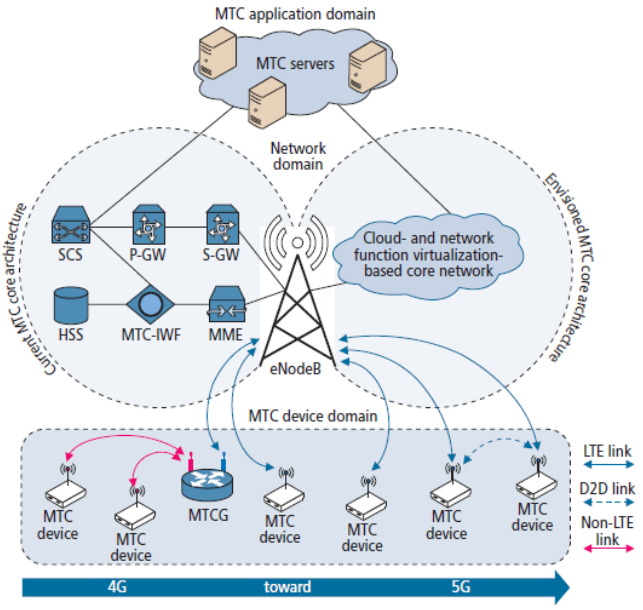

Figura 5. Arquitectura de las redes celulares para dar servicio a MTC. Fuente: [15]

Aunque se ha desarrollado una plataforma de software para la integración de las comunicaciones MTC, el aumento acelerado de las poblaciones de dispositivos en el paradigma de IoT y bajo el marco de 5G hace que se hable de mMTC *(massive Machine Type Comunnications*), que ha requerido nuevos esfuerzos en la estandarización y mejora en las capacidades de las tecnologías celulares de quinta generación.

3.6 Las mMTC (*massive Machine Type Comunnications*).

El aumento masivo de dispositivos en el paradigma de IoT bajo el paraguas de la tecnología celular de 5G ha creado una serie de retos a la hora de gestionar el masivo acceso de dispositivos a los canales de comunicación. Para superar dichos retos se han propuesto diversas soluciones, en las que el *gateway*, que se observa en la Figura 5 como MTCG (*Machine Type Communications Gateway)*, es un elemento fundamental. Aunque también es posible conectar los dispositivos directamente a la estación base de la red celular, el hacerlo impide por ejemplo la creación de *clusters* que como se ha visto en el punto 3.4 es una técnica muy interesante a la hora de crear infraestructuras con alta densidad de sensores.

Según [16] es posible categorizar las distintas soluciones propuestas para solucionar el problema del acceso masivo en MTC de esta manera:

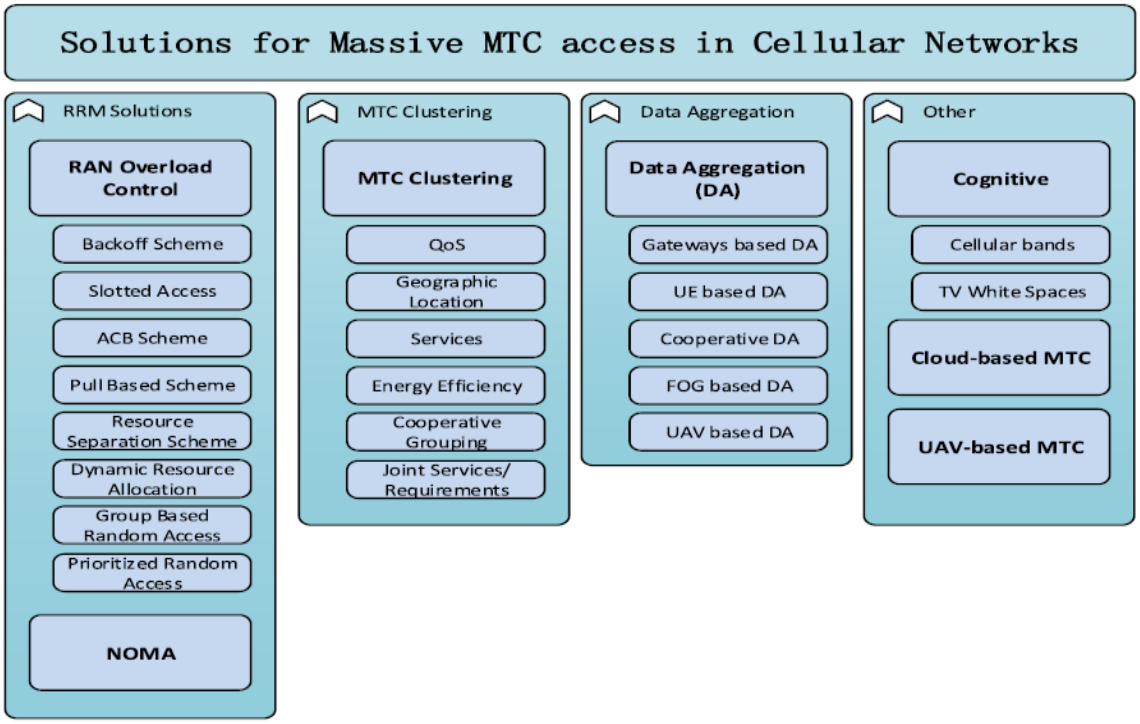

Figura 6. Categorización de las soluciones para los desafíos de mMTC. Fuente: [16]

a) *Control de sobrecarga de RAN.* Este tipo de mejoras se basa en aumentar las capacidades de control a la hora de gestionar el acceso de los dispositivos al canal aéreo. Dicho control de la sobrecarga se puede hacer a nivel estación base o a nivel MTCG. Los protocolos que se estudiarán tratan de mejorar el uso del canal aéreo entre los distintos dispositivos y el g*ateway*.
b) *Clustering de los dispositivos MTC.* Ya hablamos de ello en el punto 3.4. La introducción de técnicas de agrupación de dispositivos bajo el paraguas de un mismo *gateway* es básico para poder hacer las distintas infraestructuras escalables. Además, las técnicas de clustering permiten el uso de dispositivos de baja potencia que usen la capacidad de conectividad a las BS de otros dispositivos más potentes haciendo estos últimos de *gateaway* hacia la red de acceso.

c) *Agregación de los datos.* La función de MTCG no es solo la de funcionar como CH del cluster de dispositivos, también es el encargado de agregar la información de los dispositivos de los que es *gateway*. Al eliminar información redundante se reduce el consumo de energía, se utiliza más eficientemente el uso de los recursos de la red de acceso y se reduce el uso excesivo de la señalización.
d) *Otros como basados en la nube o en tecnología de radio cognoscitiva.*

3.7 Protocolos de acceso al medio entre los dispositivos y el *gateway.*

Antes de entrar a explicar los dos protocolos que se van a comparar en este trabajo, veamos cómo se clasifican algunos de los protocolos de acceso al medio (MAC) más comunes para las WSN por su relación con el trabajo realizado.

En [17] se ofrece una clasificación de distintos protocolos que se han ido desarrollando para el control de acceso al medio en redes de sensores:

a) *Asíncronos.* Son protocolos de acceso que funcionan con ciclos de estados entre activo e inactivo para ahorrar consumo de energía, por ello es necesario que los dispositivos comunicantes estén ambos activos a la hora de transferir la información. Estos protocolos centran los esfuerzos en gestionar la comunicación entre dispositivos con ciclos de trabajo de distinta velocidad.
b) *Síncronos*. Como los anteriores, son protocolos de acceso funcionan en ciclos de actividad e inactividad. A diferencia de los asíncronos, los dispositivos deben de sincronizar su ciclo de trabajo y activarse y desactivarse al mismo tiempo. Esto obliga que dispositivos que se conecten con dos o más ciclos de trabajos distintos se tengan que sincronizar a cada uno de estos.
c) Basados en trama-ranura. Son protocolos de acceso con el tiempo ranurado y basan su actividad en distribuir cada una de las ranuras a los distintos dispositivos que quieran acceder al canal para evitar colisiones.
d) Multicanal. Se basan en usar varios canales aéreos para la transmisión simultánea de la información de los diferentes dispositivos al *gateway* de la red.

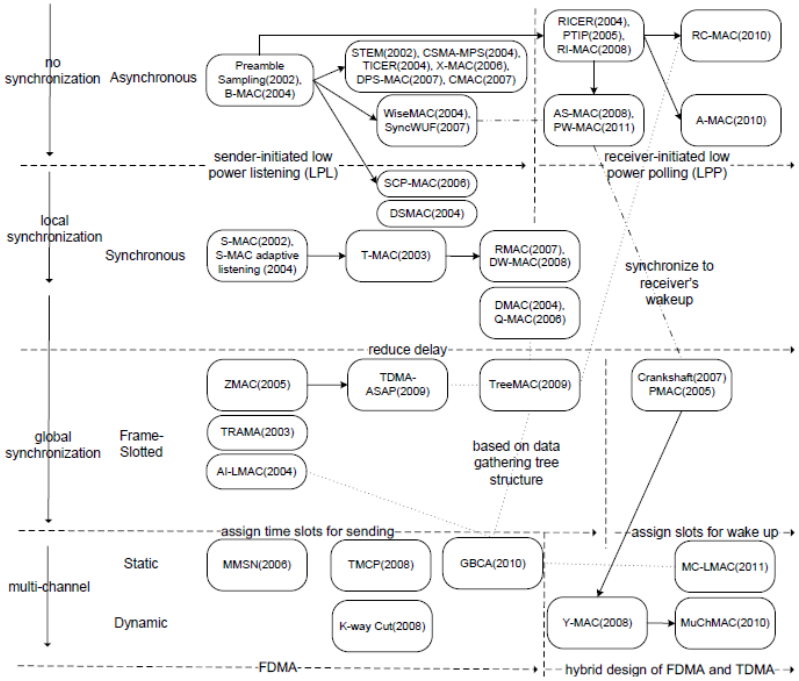

Figura 7. Clasificación de algunos protocolos MAC. Fuente: [17].

A continuación, se tratarán algunos de los protocolos de acceso al medio.

3.7.1 Variantes de ALOHA [18].

A. Protocolo ALOHA.

Protocolo de acceso al medio de nivel de enlace de datos desarrollado para la red ALOHAnet, desplegada en 1970. Esta versión del protocolo se componía de dos fases:

- Si un nodo de la red tiene datos a enviar, los envía por el medio compartido.
- En caso de haber colisión, se espera un tiempo aleatorio para volver a intentar reenviarlo.

En este caso de ALOHA no es necesaria la sincronización de los nodos dado que estos transmiten en el momento que tengan algo que transmitir. El uso máximo del canal en este protocolo está entorno al 18%. En la actualidad está en total desuso.

B. Protocolo S-ALOHA (ALOHA ranurado).

Una mejora posterior del protocolo ALOHA. En este caso se ranura el tiempo en *slots* y si un terminal tiene datos a enviar debe esperar al inicio de una nueva ranura para hacerlo. En el caso de existir colisión al transmitir la información se espera un número aleatorio de slots para volver a intentarlo. Debido a esta división temporal en slots, es necesario que los elementos de la red estén todos sincronizados.

Esta mejora permitió un aumento del uso del canal hasta un 37%, el doble que en el anterior caso. Este protocolo se sigue, por ejemplo, en las redes celulares actuales, cuando un terminal quiere iniciar el establecimiento de una conexión con las BS.

3.7.2 Variantes de CSMA.

CSMA (*Carrier-Sense Multiple Access)* [19]se trata de un protocolo de acceso al medio en capa de enlace de datos. Su funcionamiento se basa en verificar la ausencia de tráfico antes de transmitir en un medio compartido.

Para verificar la existencia, o no, de tráfico, un terminal "escucha" la portadora en el canal en el que quiere transmitir. Si una señal es detectada, el nodo espera a que la transmisión en progreso se finalice para iniciar la transmisión. Si se detecta colisión se deja de transmitir y se espera a que esté el medio libre. Hay dos tipos:

- 1-persistente: Transmite en el mismo momento que el canal se libera.
- p-persistente: Transmite con probabilidad p una vez que el canal está libre. Si no transmite espera un tiempo prefijado. Dicha probabilidad la calcula el propio nodo a la escucha. Si cuando pasa el tiempo de espera el canal está ocupado, vuelve a esperar a que esté libre y calcula de nuevo la probabilidad p.

A. Protocolo CSMA/CD [20].

CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*) es un protocolo de acceso al medio usado ampliamente en la tecnología *Ethernet* (IEEE 802.3) hasta la aparición de los *switches Ethernet*. Es una mejora del protocolo CSMA. En este protocolo en vez de dejar de transmitir cuando se detecta colisión, se mantiene esta durante el tiempo suficiente como para que todos sean conocedores de la colisión. Tras esto se espera un tiempo de *backoff* que se calcula en función de las colisiones previas al intentar enviar el paquete y se reenvía la trama por el medio con la expectativa de que no vuelva a haber colisión.

B. Protocolo CSMA/CA [21].

CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) es un protocolo de acceso al medio utilizado ampliamente en las tecnologías inalámbricas IEEE 802.11, entre las que se encuentra la tecnología *wifi* (802.11n). Se trata de una mejora del protocolo CSMA pensado para medios en los que los distintos elementos que se quieren conectar al g*ateway* de la red no tienen por qué detectar las colisiones en el receptor del *gateway* (sea este un punto de acceso o un router). Para ello el dispositivo que quiere enviar una trama pide permiso para enviar (*Request to Send)* al punto de acceso. Si el equipo destino está preparado para recibirla enviara un aviso al dispositivo para que inicie la transmisión. Una vez recibida toda la trama enviará un ACK(*Acknowledged*) para confirmar la correcta transmisión al dispositivo origen.

Capítulo 4. Protocolos que se estudian.

Los protocolos que vamos a estudiar y evaluar comparativamente en este trabajo están propuestos como mejoras de *Framed Slotted Aloha* (FSA) para entornos WSN. FSA es una generalización del estándar S-ALOHA que se trató anteriormente en el punto 3.7.1. FSA es una amplia familia de protocolos dentro de los protocolos de acceso aleatorio (RPA), usados en comunicaciones por cable e inalámbricas.

A diferencia de S-ALOHA donde los dispositivos eligen un *slot* temporal de forma no coordinada y, por tanto, pueden producirse colisiones. En FSA existe una división temporal en tramas compuestas por un determinado número de *slots* temporales. En FSA cada trama debe tener más de 1 *slot* temporal. Al inicio de la trama cada dispositivo elige con igual probabilidad uno de los *slots* para enviar el paquete y espera confirmación de envío del nodo receptor. Si hubiera colisión el dispositivo que envió el paquete no recibe confirmación (ACK). Si sucede esto (que haya colisión) el dispositivo no espera un número aleatorio de *slots* temporales, si no que al inicio de la siguiente trama vuelve a elegir aleatoriamente uno de los *slots* de la trama para intentar hacer el envío.

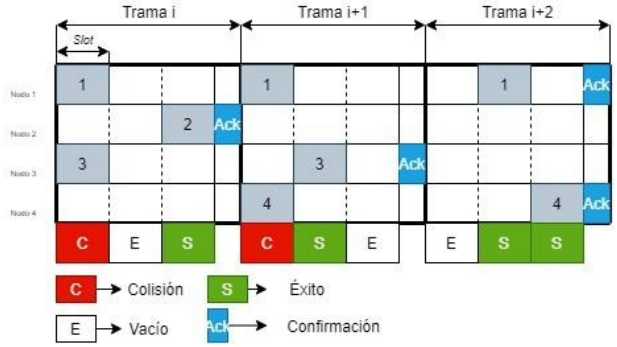

Figura 8. Ejemplo del funcionamiento genérico de FSA.

La generalización básica de FSA anteriormente comentada tiene dos inconvenientes importantes a la hora de adaptarlo a las necesidades de las redes WSN dentro del paradigma de IoT que se ha descrito en el capítulo de Introducción. El primero de ellos es el bajo uso de canal, dado que cada *slot* temporal tiene el tamaño de un paquete y con cada colisión se pierde bastante tiempo de oportunidad en el envío de los paquetes. También se pierde mucho uso de canal debido a los *slots* temporales no usados en cada trama. Si un nodo se activa en mitad de una trama temporal debe esperar el fin de ésta aunque todos los *slots* de datos estén sin utilizar. Una vez que acaba la trama puede intentar enviar el paquete en un slot de datos. El otro gran inconveniente dentro del paradigma de IoT es debido a la necesidad de que todos los nodos estén sincronizados, pudiendo esta aumentar el consumo de los sensores del *cluster*.

Los protocolos propuestos pretenden mejorar las capacidades de FSA y subsanar el primero de los problemas, el uso del canal y la pérdida de oportunidad debido a las largas tramas. Al estar pensados para su uso en sensores, buscan un uso más eficiente del canal y de los recursos de los dispositivos a la hora de contender por un *slot* de datos.

4.1 *Framed Slotted ALOHA with reservation data packets.*

FSA-RDP (*Framed Slotted ALOHA with reservation data packets)* es un protocolo propuesto en [22]. Este protocolo funciona bajo una topología de estrella donde el *gateway* funciona de controlador del acceso al medio de una población de sensores. Este control se hace mediante un modelo de permiso donde el *gateway* informa a los sensores conectados cual de ellos transmite en la trama donde han realizado la contienda.

En el protocolo se propone tiempo ranurado definiéndose dos tipos de *slots*. Los *minislots* (mslot) de contienda y los *slots* de datos. Así mismo la trama vendría definida por dos subtramas:

- *RSF (Subtrama de reserva)*: Subtrama de contienda con V *minislots* de contienda.
- *DSF (Subtrama de datos):* Subtrama de longitud variable de hasta V *slots* de datos.

La longitud de la DSF es variable en función del número de accesos exitosos en la subtrama de contienda. Por ende, la contienda se realiza durante la primera subtrama de reserva donde los nodos envían un pequeño paquete de control, varias veces más pequeño que el datagrama que se envía.

Cuando un nodo tiene un paquete que enviar al *gateway* al inicio de una trama, selecciona con la misma probabilidad uno de los *minislots* de contienda de la RSF y envía un el paquete de control para reservar un *slot* de datos de DSF. Suponemos que el *gateway* es capaz de detectar tres estados en la recepción de los *minislots* de contienda: vacío, colisión y éxito. Tras la subtrama de reserva el *gateway* informa del resultado de la contienda a la población de sensores conectados a él. La manera de informar de cuál es el resultado de la contienda es indicar el orden de envío de los paquetes y que nodos han de enviar dichos paquetes. Esta información se distribuye mediante el SAP (Mensaje de asignación de ranuras). El orden de enviar el paquete en DSF sigue el orden de llegada de los paquetes de control exitosos en RSF. Este envío está libre de colisiones y tras recibir los paquetes, el *gateway,* informa mediante un paquete ACK de la recepción exitosa de éstos.

DSF es variable, por tanto, en el caso de que no hubiera ningún nodo que pudiera enviar debido a que en la contienda solo hubiera colisiones, o que ningún nodo haya accedido, dicha subtrama no se daría y se iniciaría una nueva trama como se ve en la Figura 9 entre la trama i. En el caso de que todos los *minislots* de reserva recibieran exitosamente un paquete de control de los nodos, DSF tendría tantos *slots* para paquetes como numero de *minislots* de contienda tenga RSF.

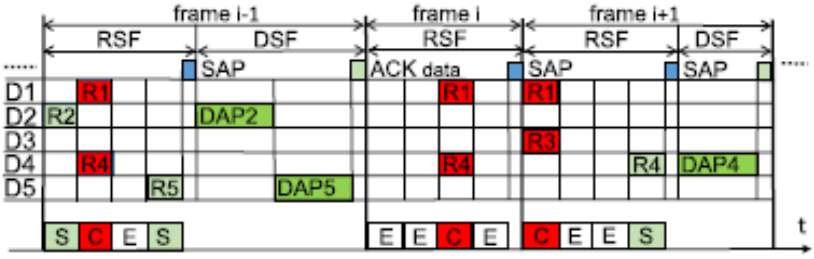

Figura 9 Ejemplo del funcionamiento del protocolo *FSA-RDP*. S: *success,* C: *collision,* E: *empty*. Fuente: [22].

El modelo asume que cada uno de los sensores tiene una cola de tamaño Q para que, en caso de que durante la contienda haya colisión, el nodo pueda guardar la información para intentar reenviarla en la siguiente trama. Se considera dos esquemas de gestión de dichas colas, *tail drop* (TD) y *push-out* (PO). TD es un algoritmo de gestión de colas que elimina los datos entrantes cuando la cola está llena y no permite añadir nuevos paquetes a la cola hasta que esta no tenga espacios libres. PO es un algoritmo de gestión de colas en el que, al llegar un nuevo paquete a una cola llena, se descarta el paquete más antiguo en la cola para que el nuevo pueda ser copiado.

Esta propuesta busca mejorar el uso del canal y los recursos de los distintos nodos gracias a que la contienda no se hace sobre los *slots* de paquetes directamente. Se realiza sobre RSF cuyos *minislots* son de un tamaño varias veces inferior al de los paquetes. Así las colisiones no desaprovechan tanto tiempo el canal. FSA-RDP también propone la aplicación de una DSF variable que permite tramas de muy corta duración (solo lo que dure RSF) en caso de que ningún nodo quiera o pueda enviar paquetes. Esto ayuda a un uso más eficiente del canal y un aprovechamiento de la oportunidad de envío por parte de los sensores que no tienen que esperar el final de una larga trama si quieren intentar acceder al canal. Esto en aplicaciones críticas como de emergencias o seguridad puede ser vital para el desarrollo del servicio.

4.1.1 La probabilidad de acceso

La probabilidad de acceso es un mecanismo de gestión del tráfico propuesto en [22] para mejorar las prestaciones del protocolo. Sobre todo, en ocasiones de gran cantidad de accesos para evitar la congestión de la RSF y que debido a ésta ningún nodo sea capaz de enviar paquetes.

Su funcionamiento se basa en que el sensor que tiene un paquete que enviar se activa y genera un número aleatorio $R \in [0,1]$ que debe ser menor que la probabilidad de acceso (r). Si el valor que genera el sensor es mayor que la probabilidad de acceso espera hasta la siguiente trama para repetir el proceso. Si el valor es inferior, el nodo envía el paquete de control a la RSF como se ha indicado anteriormente y se continua el proceso de contienda. Si la contienda es un éxito el nodo envía el paquete. Si se produce una colisión en la contienda el sensor deberá esperar a la siguiente trama y obtener un valor de R inferior a la probabilidad de acceso antes de reintentar la contienda.

Este valor de r es estático y configurable según el modelo propuesto. Se indica en el trabajo [22] que existe una probabilidad de acceso óptima (r_{opt}) calculable que maximiza la eficiencia del sistema. Mediante un proceso de búsqueda se puede obtener el valor óptimo de la probabilidad. Esto se consigue cambiando el valor de r en sucesivas simulaciones de tal manera que r_{opt} es el que permite que el caudal de tráfico servido sea mayor. En la Figura 10 se puede observar la diferencia de caudal entre un modelo con una r = 1 o r_{opt} para la carga total suministrada por el *cluster* al *gateway* (ρ_T). En este caso tenemos 10 sensores conectados al *gateway* y un tamaño de la RSF de 2 *minislots* y por tanto de 2 *slots* de datos en la DSF. El tamaño de los *slots* es 10 veces el tamaño de los *minislots*. Cada sensor tiene 10 posiciones de cola para guardar paquetes.

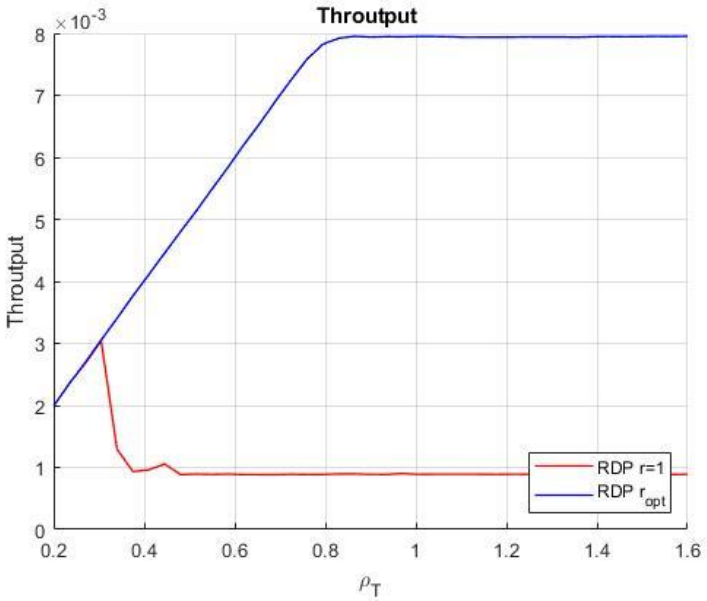

Figura 10. Caudal de FSA-RDP con *r*=1 y r_{opt} en función de la carga ofrecida por los sensores del *cluster*.

La evolución de *r* es inversamente proporcional a la carga total suministrada por los nodos, así que a mayor número de nodos que contiende en una misma trama, el valor de *r* debe ser menor para minimizar las colisiones. Así se impide proporcional y aleatoriamente (sin que unos tengan preferencias sobre otros) el acceso de los sensores. A continuación, se puede observar en la Figura 11 cuales han sido los valores óptimos de *r* para cada carga ofrecida en el caso de la Figura 10.

Hay más información de cómo se modela la carga o tráfico que se le suministra al *gateway* en la simulación en la sección 5.3.

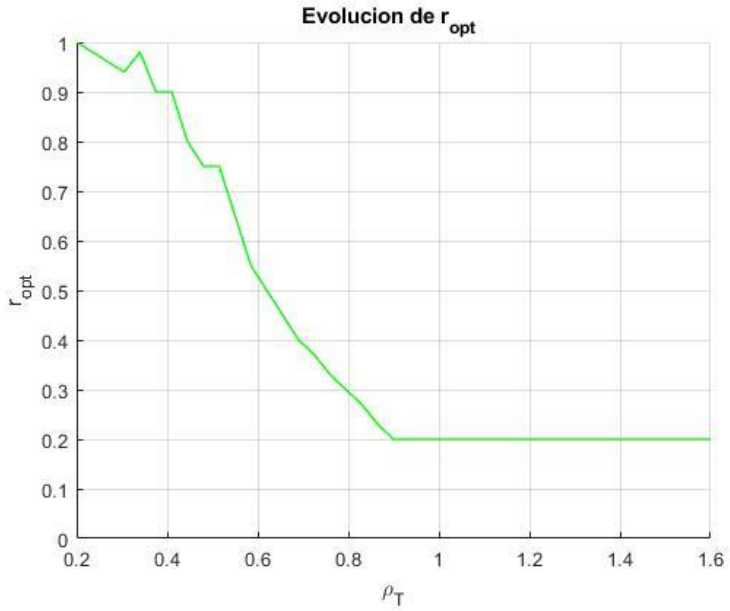

Figura 11. Evolución de r_{opt} en función de la carga ofrecida por los sensores del *cluster*.

En [22] se sugiere que, en vez de ser un valor estático y prefijado de antemano, se desarrolle un algoritmo de autoconfiguración del valor de *r*. En este trabajo se tomará esa sugerencia y la llevará a cabo buscando un algoritmo que permita ajustar *r* dinámicamente con la carga del sistema en cada ranura. El funcionamiento del algoritmo de ajuste está descrito en el Capítulo 6.

4.2 *Distributed Queueing Random Access Protocol.*

El protocolo que propone [23] se trata de una evolución de FSA mediante el uso de colas distribuidas, que denotaremos como FSA-DQ (*Frame Slotted Aloha with Distributed Queueing*). Trabaja bajo una topología de estrella con un *gateway* que funciona de coordinador de acceso al medio de una población de sensores. Esta propuesta, a diferencia de la anterior, no se basa en que el *gateway* de permiso a los nodos para enviar a los paquetes, sino que se gestionan los envíos de los paquetes mediante *tree-splitting* y el uso de colas distribuidas.

En esta propuesta el *gateway* envía a los sensores de la información mínima necesaria para que se desarrolle el protocolo autónomamente por parte de los dispositivos.

Como en el anterior protocolo el tiempo está ranurado en *slots*. Se definen dos tipos de *slots* temporales. Los *minislots* de contienda que son varias veces más pequeños que los *slots* de datos. Esto permite una mejora en el uso del canal aéreo pues la contienda no se da durante el propio slot de datos. Así mismo la trama temporal está dividida en tres partes:

- Subtrama de contienda: *Minislots* donde los nodos acceden aleatoriamente. Si acceden más de dos nodos en el mismo slot se produce una colisión y ambos se introducen en la cola de contienda.
- Subtrama de transmisión libre de colisiones: A diferencia de FSA-RDP, en la solución propuesta para FSA-DQ, la subtrama de datos está formada por un numero de *slots* fijo.

- *Minislot* de *feedback*: En este *minislot* el coordinador transmite la información necesaria para la consecución del protocolo por parte de los dispositivos contendientes.

Figura 12 Estructura de trama DQ, consiste en m *minislots* para la resolución de la contienda (*uplink*), un *slot* libre de colisiones para la transmisión (tanto *uplink* como *downlink*) y un *minislot* de difusión de información por parte del controlador (*downlink*). Fuente: [23]

A diferencia de FSA-RDP, FSA-DQ busca implementar un modelo de colas lógicas de dos tipos, la *Colisión Resolution Queue* (CRQ) y la *Data Transmision Queue* (DTQ). El *gateway* no coordina que sensores transmiten en la subtrama de datos, sino que transmite la información necesaria para la gestión de dichas colas. Estas colas son colas virtuales distribuidas entre los distintos sensores conectados al *gateway*.

Cuando un sensor tiene un nuevo paquete que enviar al inicio de la trama elige equiprobablemente uno de los *minislots* de contienda y envía un pequeño mensaje denominado *Access Request Sequence* (ARS). El *gateway* por tanto necesitará detectar tres estados posibles del *minislot* de contienda: éxito (solo un ARS ha sido enviado al *minislot*), colisión (se han enviado varios ARS a un *minislot* y ha sido imposible decodificarlos) y vacío. El *gateway* retransmite al final de la trama, mediante *broadcast*, la información sobre el resultado de la contienda mediante el paquete de *feedback* (FBP). En este paquete está la información necesaria para la ejecución del algoritmo DQ y, por tanto, para la gestión de las colas CRQ y DTQ. La entrada a una de las colas distribuidas viene dada por:

1) Los sensores cuyos paquetes han colisionado en la subtrama de contienda entran al CRQ. Para resolver las contiendas de la cola distribuida se hace uso de un algoritmo de *tree-splittig*. Posteriormente se tratará su funcionamiento.
2) Los sensores cuyos paquetes no han colisionado en la subtrama de contienda entran en la DTQ. Se trata de una cola *first-in first-out* (FIFO) que gestiona los nodos que han superado la contienda para el envío del paquete en las sucesivas subtramas de datos. Esta subtrama está libre de colisiones.

Las colas con las que trabaja FSA-DQ están representadas por dos enteros. El primer entero es el tamaño de la cola, este valor es actualizado por el *gateway* en el FBP al final de cada trama. El segundo entero es la posición de cada sensor en dicha cola, valor que gestiona cada uno de los dispositivos y por eso se denominan colas distribuidas. Se presupone un *buffer* de memoria en cada uno de los dispositivos para guardar los paquetes durante el proceso de resolución de la contienda, además de dos contadores para gestionar el valor de los punteros de las dos colas. En las siguientes secciones del capítulo se profundizará más en cómo se desarrollan las colas distribuidas, los valores de tamaño de cola y los punteros adscritos a dichas colas en los dispositivos.

La propuesta de [23], a diferencia de la que se hace en [22], mantiene una subtrama de datos de tamaño fijo. Facilitando así la sincronización de los sensores de la red dado que el tamaño de las tramas es invariable en el tiempo. Esto hace que esta propuesta sea más sencilla a la hora de desarrollarla en entornos reales al reducir la complejidad en el sincronismo de los distintos elementos del *cluster*.

Por contrapartida, si se requiere de un número mayor de *slots* de datos debido al número de sensores en el *cluster*, unido a un servicio que puntualmente tienda a sobrecargar el acceso al medio, se vuelve al problema descrito previamente. Se desaprovecha el canal al tener una trama con *slots* de datos posiblemente inutilizados durante periodos relativamente largos de tiempo.

4.2.1 Contention Reservation Queue (CRQ).

La primera de las colas que trataremos es la que se encarga de gestionar las colisiones que se dan en la subtrama de contienda. Esta cola trabaja bajo un algoritmo de *tree-splitting* para resolver las colisiones en el acceso. En el caso de que más de un nodo acceda a un *minislot* en la subtrama de contienda y, por tanto, las colisiones pasarán a formar parte de un subgrupo de contienda. Estos subgrupos de dispositivos se organizan en función del *minislot* al que han intentado acceder. Si por ejemplo acceden 2 al mismo *minislot*, dicho subgrupo estará formado por esos dos sensores. Se ve por ejemplo en la Figura 13 apartado a), en la primera trama, como el primero de los subgrupos está formado por los 4 dispositivos distintos que han intentado acceder al primer *minislot* de contienda.

Cada vez que se añade un subgrupo de contención la cola CRQ aumenta en 1 su tamaño, representando el tamaño de la cola el número de subgrupos que esperan volver a intentar la contienda en las siguientes tramas. Cada vez que se inicia una nueva trama el valor de CRQ se reduce en una unidad, siendo cero su valor único. El valor del tamaño de la cola es un entero que se denomina contador RQ y se transmite por el *gateway* al resto de dispositivos en el FBP.

Para que los sensores sepan su posición en la cola CRQ, el *gateway* les informa a través del FBP del tamaño de la cola y cuál ha sido el estado (colisión, éxito o vacío) de cada *minislot* de contienda en la trama donde envía el paquete. Las posiciones de la cola se llenan de manera ascendente con dependencia al *minislot* escogido. Esto quiere decir que, si ha habido colisión en todos los *minislots* de contienda, será el subgrupo del primer *minislot* el que tome la primera posición libre del CRQ, el segundo el que tome la segunda posición libre y así sucesivamente hasta el último subgrupo. La posición de cada dispositivo en la cola CRQ se denota mediante un entero denominado contador pRQ y es gestionado por el propio sensor. El valor de pRQ se reduce en una unidad al inicio de cada trama, así que cuando vale cero, el dispositivo que gestiona ese puntero reintenta la contienda sin saber si hay otros dispositivos que se encuentran en la misma situación que él o no.

En las tramas siguientes realizarán la contienda solo los subgrupos por orden en la posición de la cola CRQ. Así los sensores que deban esperar varias tramas para volver a intentar la contienda pueden desactivarse y esperar hasta ese momento para reactivar la emisión, ahorrando así energía.

Este proceso de *tree-spliting* se observa en la Figura 13 en el apartado a). El subgrupo que colisiona en el primer *minislot* de la trama 1 es el que entra a la contienda en la trama 2 (los dispositivos d1, d2, d3 y d4). El otro subgrupo (d7 y d6) pueden desactivarse hasta que le toque realizar la contienda, que será en la tercera trama. Dado que ninguno de los dispositivos del primer subgrupo tiene éxito en la contienda de la trama 2 se vuelven a dividir en dos subgrupos que contendrán en la trama 4 y 5.

4.2.2 Data Transmision Queue (DTQ).

Una vez que la contienda se resuelve y el *gateway* ha recibido y decodificado un ARS, el dispositivo que envió dicho mensaje se organiza virtualmente en la cola de transmisión DTQ. Que sea virtual, como en el anterior caso, significa que no está implementada en un elemento específico dentro de la red, sino que cada sensor almacena el paquete y el orden de acceso a la subtrama de datos se gestiona distribuidamente entre todos los elementos del *cluster*. Como se ha indicado anteriormente esta cola es una cola FIFO, las primeras posiciones libres las ocupan los primeros dispositivos que entren en ella. Por tanto, la abandonan primero los que primero accedieron a ella.

Tanto DTQ como CRQ funcionan en paralelo. Los dispositivos ingresan directamente en la DTQ si durante la contienda envían el ARS sin colisión. Si hubiera colisión, ingresan primero en la CRQ para resolver la contienda y en sucesivas tramas entrarán en la DTQ. Así como cada colisión aumenta en uno el tamaño de la CRQ, cada acceso exitoso en la subtrama de reserva aumenta en uno el tamaño de la DTQ.

Para la gestión de DTQ se hace uso de dos enteros. El primero de ellos es el entero que denota el tamaño de la cola, denominado *TQ counter*. El valor de *TQ counter* aumenta en una unidad por cada dispositivo conectado al *gateway* que es capaz de superar la contienda en la anterior trama. El valor *TQ counter* es gestionado y transmitido mediante broadcast por el *gateway* a través del FBP. El segundo entero, denominado *pTQ counter*, es un puntero gestionado por cada dispositivo que corresponde a la posición del dispositivo dentro de la DTQ. Cuando un dispositivo entra en la DTQ actualiza este valor para que apunte al final de la cola (*TQ counter*) y cada vez que se termina una trama resta tantas unidades como *slots* de datos tenga la trama. Cuando su valor sea igual o menor que cero el dispositivo enviará el paquete en uno de los *slots* de datos.

Como se ha indicado anteriormente los *slots* de envío de paquete son fijos en la propuesta original. Además, se estudia un escenario con un único *slot* de datos, aun así, su número puede ser superior. Si hay más de un *slot* de datos, los dispositivos utilizan dichos slots en el orden correspondiente en cola. Así el primero de los dispositivos en la cola transmitiría en el primer *slot* de datos, el segundo en la cola en el segundo *slot* de datos y así sucesivamente.

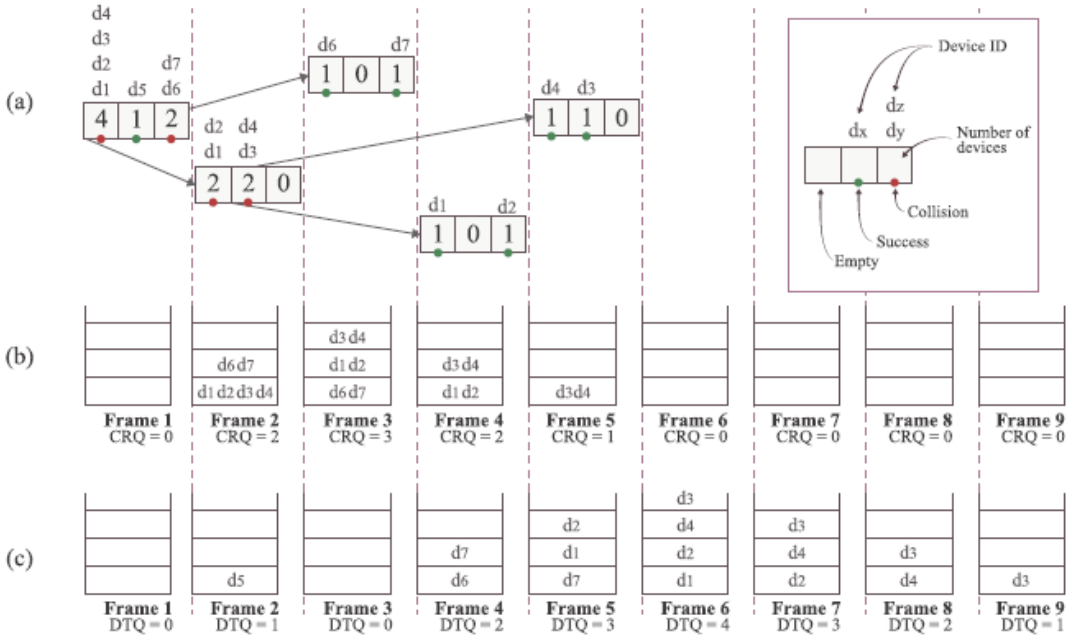

Figura 13 Ejemplo del funcionamiento de DQ con 7 nodos: (a) algoritmo tree-split. (b) Comportamiento de CRQ en cada trama. (c) Comportamiento de DTQ en cada trama. Tres minislots de contienda para cada trama. Fuente: [23].

Capítulo 5. El modelo de simulación.

Para diseñar el modelo de simulación se utiliza el entorno de simulación SMPL desarrollado en C++. Para el desarrollo del modelo de simulación que permite evaluar el protocolo FSA-RDP se ha utilizado parte del código facilitado por el tutor de este trabajo. Dicho código está validado mediante los resultados de un modelo analítico descrito en [22]. Para el caso FSA-DQ, en [23] no se propone un modelo analítico, por lo que no es posible utilizar esta estrategia para validar los resultados del modelo de simulación. Posteriormente, en este capítulo, propondremos una estrategia alternativa para validar el modelo de simulación que permite estudiar el protocolo FSA-DQ.

5.1 El entorno SMPL[24].

El entorno de simulación SMPL pertenece al entorno de simulación llamado SMPL. Dicho entorno de simulación proporciona una serie de herramientas de *debugging*, representación de datos, etc. En el *software* proporcionado por el tutor de este trabajo se hace uso únicamente del entorno SMPL para aprovechar su potencia a la hora de simular eventos discretos, además de sus funciones para generar números aleatorios.

La simulación por eventos discretos se trata de una forma de simular y programar que permite la obtención y organización de los resultados de procesos que avanzan o trabajan en momentos puntuales en el tiempo [25]. Como hemos indicado en el **¡Error! No se encuentra el origen de la referencia.**, el tiempo está ranurado en minislots y se considera la llegada de los paquetes en el último momento del slot, por ende, esta manera de trabajar la simulación coincide con las necesidades de nuestros casos.

5.1.1 Funcionamiento y características básicas del entorno.

El entorno, como se ha dicho anteriormente, se basa en eventos. Para crear, organizar y gestionar dichos eventos tiene tres entidades diferenciadas:

a) *Facilities (recursos/bloques funcionales).* Un sistema se forma por una serie de recursos o bloques funcionales que se interconectan entre ellos para llevar el proceso a cabo. La interconexión entre ellos no es explícita, se hace mediante el uso de tokens. Por tanto, los *facilities* son elementos estáticos dentro del funcionamiento del modelo generado.

b) *Tokens(fichas).* Los *tokens* representan las entidades activas del sistema. El comportamiento del sistema se modela mediante el movimiento de las fichas entre los bloques funcionales. Cada ficha puede representar una tarea en un modelo informático o un paquete en un modelo de comunicaciones. Para controlar el flujo de las fichas, smpl proporciona la capacidad de reservar recursos para una ficha, y si no fuera posible iniciar el proceso, el token se pone en cola hasta que se pueda iniciar la tarea que representa dicha ficha. Así se pueden obtener estadísticas y capacidades de funcionamiento del sistema que se simula

c) *Events (eventos).* Los eventos son los cambios que ocurren en el sistema simulado, sean estos cambios activos o pasivos.

5.1.2 Funciones del entorno usadas en el modelo de simulación.

- Función **SMPL(int m, char *s)**: Con dicha función inicializamos el proceso y limpia las distintas estructuras de datos. Además, reinicia a cero el intervalo de medida de las medidas. También es el encargado de ir cambiando las semillas que generan los eventos aleatorios en cada una de las distintas simulaciones (como por ejemplo los valores aleatorios de la siguiente función). Así se pude observar el desempeño del sistema ante la variabilidad.
- Función **random(int i, int n)**: Devuelve un número aleatorio en el rango [i, n]. Es utilizado tanto en la decisión que toman los nodos a la hora de enviar un mensaje de

reserva en el slot de contienda, como en la activación aleatoria de los mismos para el inicio del acceso al medio.

5.2 Parámetros configurables de la simulación.

Para el desarrollo de la simulación se configuran una serie de parámetros relacionados con las características de los sensores del *cluster*, la carga ofrecida por estos… Los parámetros se introducen mediante un *.txt* donde en cada línea representa una simulación del sistema con unas determinadas características. Por ejemplo, en la siguiente figura se puede observar un escenario de simulación donde se pretende obtener el comportamiento del sistema al ir cambiando la carga total ofrecida. Cada una de las líneas producirá una serie de resultados en función de lo configurado, cómo se calculan dichos resultados los trataremos posteriormente.

PARAMETERS.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
1         10        20        30        40        50
Parametros:
//PhO FSA N   Qmax W  V  saccx R     rho_t TSim  #Sim
//-----------------------------------------------------
  1   2   10  10   10 2  2     1.0   0.200 1e7    1
  1   2   10  10   10 2  2     1.0   0.235 1e7    1
  1   2   10  10   10 2  2     1.0   0.270 1e7    1
  1   2   10  10   10 2  2     1.0   0.305 1e7    1
  1   2   10  10   10 2  2     1.0   0.340 1e7    1
  1   2   10  10   10 2  2     1.0   0.375 1e7    1
  1   2   10  10   10 2  2     1.0   0.410 1e7    1
  1   2   10  10   10 2  2     1.0   0.445 1e7    1
  1   2   10  10   10 2  2     1.0   0.480 1e7    1
  1   2   10  10   10 2  2     1.0   0.515 1e7    1
  1   2   10  10   10 2  2     1.0   0.550 1e7    1
  1   2   10  10   10 2  2     1.0   0.585 1e7    1
  1   2   10  10   10 2  2     1.0   0.620 1e7    1
  1   2   10  10   10 2  2     1.0   0.655 1e7    1
  1   2   10  10   10 2  2     1.0   0.690 1e7    1
  1   2   10  10   10 2  2     1.0   0.725 1e7    1
  1   2   10  10   10 2  2     1.0   0.760 1e7    1
  1   2   10  10   10 2  2     1.0   0.795 1e7    1
  1   2   10  10   10 2  2     1.0   0.830 1e7    1
  1   2   10  10   10 2  2     1.0   0.865 1e7    1
  1   2   10  10   10 2  2     1.0   0.900 1e7    1
  1   2   10  10   10 2  2     1.0   0.935 1e7    1
  1   2   10  10   10 2  2     1.0   0.970 1e7    1
  1   2   10  10   10 2  2     1.0   1.005 1e7    1
  1   2   10  10   10 2  2     1.0   1.040 1e7    1
  1   2   10  10   10 2  2     1.0   1.075 1e7    1
  1   2   10  10   10 2  2     1.0   1.110 1e7    1
  1   2   10  10   10 2  2     1.0   1.145 1e7    1
  1   2   10  10   10 2  2     1.0   1.180 1e7    1
  1   2   10  10   10 2  2     1.0   1.215 1e7    1
  1   2   10  10   10 2  2     1.0   1.250 1e7    1
  1   2   10  10   10 2  2     1.0   1.285 1e7    1
```

Figura 14. Ejemplo de parámetros configurables

Se expondrán a continuación los elementos usados para la comparación de los protocolos:

- **PhO:** Indica el tipo de gestión que se realizará en las colas del caso de FSA-RDP. Que puede ser *push-out* o *trail-drop*. En todas las simulaciones que hagamos se hará gestión *push-out* de las colas
- **FSA:** Este parámetro elige entre los dos protocolos, FSA-RDP y FSA-DQ
- **N:** Este parámetro indica el número de sensores que están conectados al *gateway*.
- **Qmax:** Con este parámetro se configura el tamaño de cola para cada uno de los sensores.

- **W:** Este parámetro indica el tamaño en *minislots* de los *slots* de datos de la DSF en el caso de FSA-RDP, y de los *slots* de datos de FSA-DQ.
- **V:** Este parámetro indica el número de *minislots* de contienda en ambos protocolos.
- **Número de *slots* de datos (saccx):** Se trata del parámetro que indica cuantos paquetes se pueden enviar por trama. En el caso de FSA-RDP este valor tiene que ser igual al de V. En el caso de FSA-DQ puede ser inferior.
- **R:** Se trata del valor de probabilidad de acceso con el que trabajará el *cluster* durante la simulación. Es estático en todo el tiempo de simulación, aunque posteriormente se propondrá un algoritmo para que sea autoadaptativo.
- **ρ_T (rho t):** Se trata de la carga total ofrecida por todos los sensores conectados al *gateway*. Es adimensional y expresa el factor de utilización, su expresión es:

$$\rho_T = N \cdot \rho_i \quad \textbf{(1)}$$

  Donde ρ_i es la carga ofrecida por un sensor del *cluster* cuya expresión es:

$$\rho_i = \lambda \cdot W \quad \textbf{(2)}$$

  Así mismo, λ es la tasa de tráfico generado por dicho sensor.
- **Tiempo simulación (TSim):** Número de *minislots* que durará la simulación de dicha línea.
- **Nº de simulaciones por cada línea de parámetros (#Sim):** Como se ha indicado previamente, la generación de números aleatorios se realiza mediante una semilla. Se puede realizar la misma simulación con los mismos parámetros anteriormente comentados, pero con distintas semillas para obtener intervalos de confianza.

5.3 La generación del tráfico en el modelo de simulación.

El tráfico en ambos modelos está caracterizado por una distribución de Poisson [26] cuya tasa es λ (paquete/*minislot*). Como se ha visto en el anterior punto, mediante el parámetro configurable ρ_T se puede extraer la tasa de tráfico generado por parte de los sensores conectados al *gateway* durante el tiempo de simulación.

Se denota la probabilidad de que lleguen *k* paquetes a la cola de un sensor en la trama *i* como:

$$P[X = k \ paquetes \ en \ la \ trama \ i] = \frac{(\lambda \cdot t_i)^k}{k!} e^{(\lambda \cdot t_i)} \quad \textbf{(3)}$$

Siendo t_i la duración de la trama *i*. Que en total será:

$$t_i = \mathrm{V} + \text{accesos exitosos} \cdot \mathrm{W} \quad \textbf{(4)}$$

Para reducir el tiempo de ejecución del modelo de simulación, y dada una determinada carga definida por λ, la distribución acumulada de probabilidad (CDF) para la variable aleatoria número de paquetes que llegan por ranura se precomputa y se almacena en una tabla. Los índices de la tabla son el número de paquetes y el tamaño de la trama, ya que ésta es variable [26].

Para cada paquete que llega al sistema se genera un *token*. Un *token* es un identificador único de paquete. Su uso simplifica el diseño del modelo de simulación, es decir, simplifica la gestión de los paquetes en la cola, la determinación de su tiempo en el sistema, su retrasmisión por el canal, … El tiempo de espera de los paquetes en las colas de los sensores, junto con otros estadísticos acumulados durante la ejecución del modelo de simulación, permite determinar las prestaciones de los protocolos.

5.4 Prestaciones obtenidas de la simulación.

En la siguiente tabla se indican los parámetros de mérito que se han definido para determinar las prestaciones de los protocolos, su símbolo y las unidades con la que se mide.

Nombre de la prestación	Símbolo	Unidad de medición
Caudal del sistema (*Throutput*)	Th	paquete/*minislot*
Utilización del canal	S	[0, 1]
Probabilidad de perdida de paquete	P_L	[0, 1]
Retraso medio de transmisión de paquete	D	*minislots*
95º percentil del retraso de paquete	D_{95}	*minislots*

Tabla 1. Prestaciones estudiadas en el trabajo

- **Caudal**: Es el número de paquetes por unidad de tiempo que es capaz de servir el sistema. Se calcula obteniendo la relación de paquetes servidos entre el tiempo total de simulación.
- **Utilización del canal**: Se trata del tiempo en el que el canal envía paquetes respecto al tiempo total de simulación. Se obtiene mediante la inversa del tiempo promedio en que durante una trama no ha servido ningún paquete. Es la fracción de tiempo que el canal está ocupado en la transmisión de paquetes de datos que se transmiten con éxito. Nótese que la subtrama de contienda es un *overhead* e impide que la utilización del canal puede llegar a ningún caso a 1.
- **Probabilidad de pérdidas**: Es la tasa de paquetes perdidos respecto a los ofrecidos al sistema. Las pérdidas se generan por desbordamiento de las colas de los nodos, pues una vez generado un paquete se intentará retransmitir indefinidamente. Así se perderán paquetes siempre que se generen paquetes en nodos cuyas colas estén llenas.
- **Retraso medio de transmisión**: Es el tiempo medio en *minislots* que tarda un paquete desde que llega a la cola de un sensor hasta que es transmitido al *gateway*.
- **95º percentil de retraso**: Es el valor del retardo que determina que el 95% de los paquetes han tenido dicho retardo o inferior.

5.5 Validación del código para el caso DQ.

El modelo de simulación del FSA-RDP está validado con el modelo analítico que se presenta en [22], al coincidir los resultados analíticos y los de simulación (como se indicó anteriormente). No ocurre lo mismo con modelo de simulación desarrollado para analizar las capacidades de FSA-DQ dado que los autores de [23] no desarrollan un modelo analítico.

Por ello se propone un escenario en el que FSA-RDP y FSA-DQ puedan conseguir las mismas prestaciones. Para que también se aproxime todo lo posible, en vez de poder enviar un paquete en cada trama como se propone en [23], FSA-DQ podrá enviar dos paquetes por cada trama y la trama será variable.

Ambos (FSA-RDP y FSA-DQ) tendrán 2 *minislots* de contienda, 2 sensores intentando acceder al canal y se define la trama con dos *slots* de datos. La cola en cada nodo es hasta de 10 paquetes y se configura *r*=1 para que los nodos activos intenten acceder siempre al canal de contienda.

En el escenario para la verificación del código de simulación para el protocolo FSA-DQ suponemos que el sistema está en estado de saturación, en el que al inicio de la trama ambos nodos quieren transmitir, por lo que su acceso será exitoso para ambos o colisionarán en la subtrama de reserva. Los posibles casos de llegadas son los representados en la Figura 15. Dado que las fuentes de tráfico son independientes y la selección de la ranura es aleatoria consideramos equiprobables cada uno de los casos.

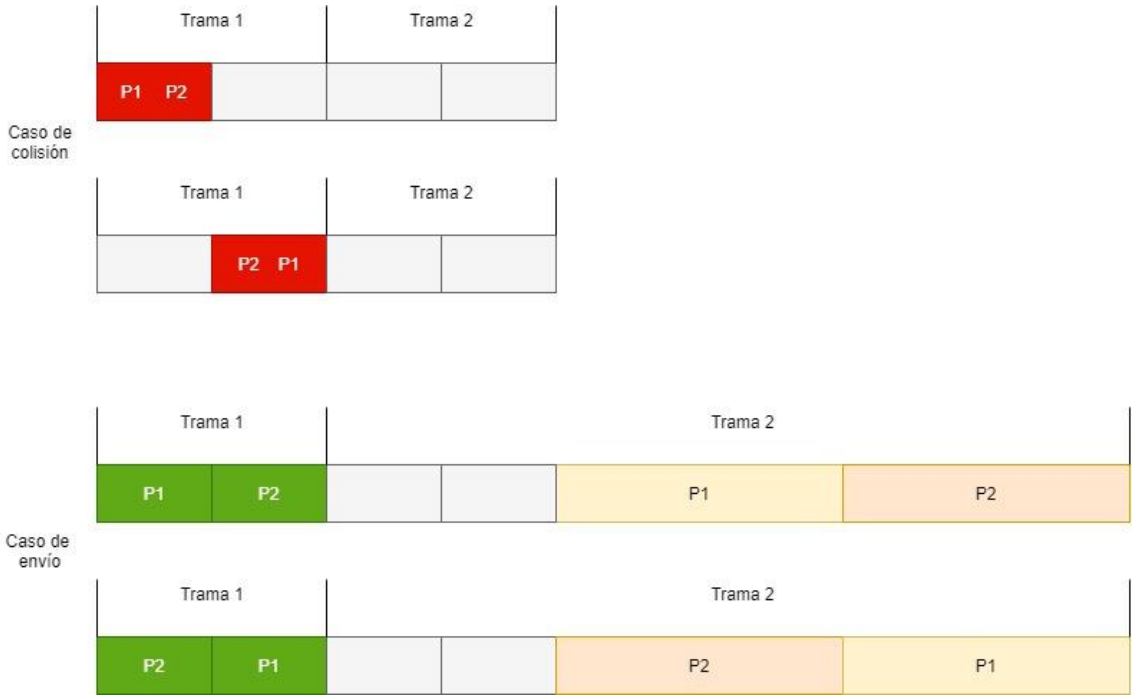

Figura 15 Posibles llegadas en el escenario para la validación para FSA-DQ.

Denotamos $P_{tx}(i)$ a la probabilidad de transmitir *i* paquetes y a $\bar{N}_{tx}$ el número medio de paquetes transmitidos por trama en el caso de que en todas las tramas cada nodo quiera enviar un paquete.

$$P_{tx}\,(0) = 0.5 \quad \textbf{(5)}$$

$$P_{tx}\,(1) = 0 \quad \textbf{(6)}$$

$$P_{tx}\,(2) = 0.5 \quad \textbf{(7)}$$

$$\bar{N}_{tx} = 0 \cdot P_{tx}\,(0) + 1 \cdot P_{tx}\,(1) + 2 \cdot P_{tx}\,(2) = 1 \quad \textbf{(8)}$$

El uso medio de canal se define como:

$$\bar{S} = P_{tx}\,(0) \cdot \frac{0}{V+2W} + P_{tx}\,(1) \cdot \frac{W}{V+W} + P_{tx}\,(2) \cdot \frac{2W}{V+2W} = \frac{W}{V+2W} \approx 0.5 \quad \textbf{(9)}$$

Este resultado coincide con el uso de canal en la simulación cuando saturamos de tráfico el escenario ($\rho_T > 0{,}8$) como se ve en la Figura 16.

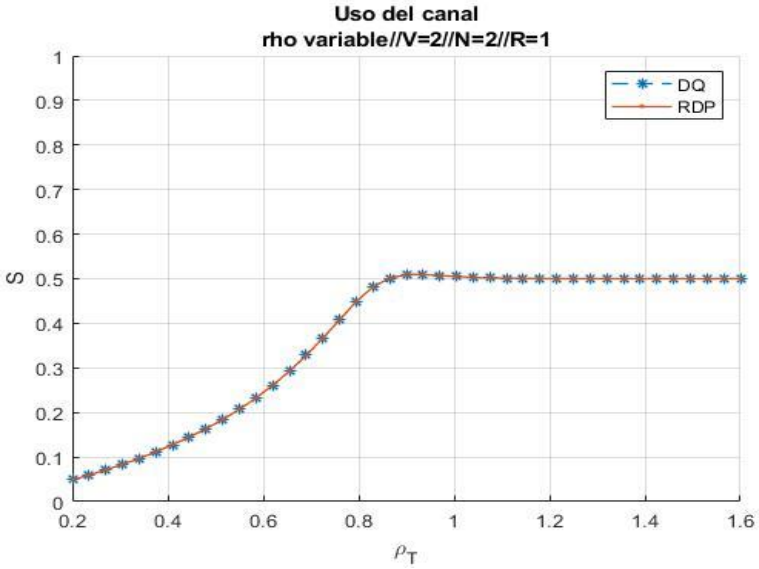

Figura 16. Uso del canal en el escenario para la validación.

La Figura 16 ilustra lo planteado inicialmente. Si el software de simulación de FSA-DQ estaba correctamente diseñado, los resultados de ambos protocolos debían de ser los mismos. Esto se repite también en el caso del retardo medio en el sistema como se puede observar en la Figura 17.

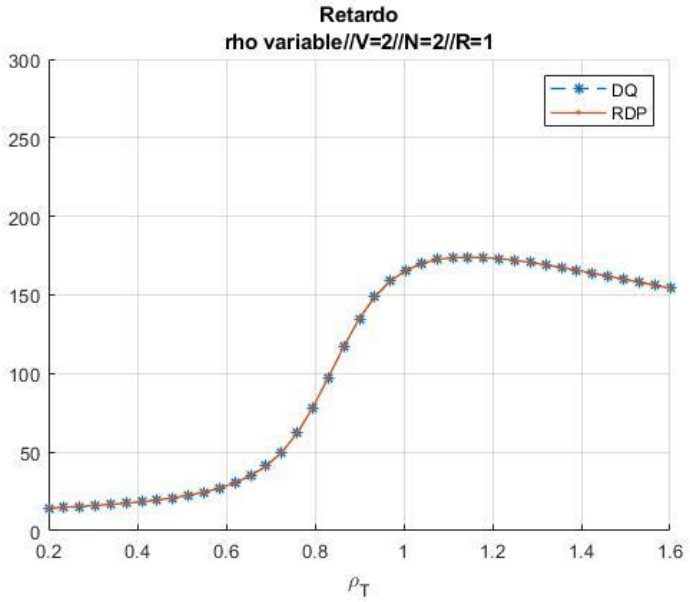

Figura 17. Retardo medio en el acceso al canal de datos en el escenario de validación.

Capítulo 6. Propuesta de mejora del protocolo FSA-RDP.

La propuesta de mejora se basa en integrar un filtro digital adaptativo en el *gateway* del *cluster* para que el valor de probabilidad de permiso (*r*) no sea estático, sino que se vaya autoconfigurando en función del tráfico que ofrezcan los distintos nodos del *cluster*. La idea se toma del trabajo [26] donde se propone un método mediante filtro digital adaptativo para el control de acceso de comunicaciones MTC dentro del paraguas tecnológico de 5G.

6.1 Filtros digitales

Los filtros digitales [27] son filtros diseñados para el tratamiento de señales discretas en el tiempo, a diferencia de los filtros analógicos que funcionan bajo señales continuas en el tiempo. Los filtros digitales pueden realizar tareas de filtrado en el terreno analógico de la señal, pero requiere del muestreado de las señales analógicas. Dicho muestreado necesita de unos cuidados tanto en separación de las muestras, la frecuencia de muestreado o la cuantificación de las muestras; pero no nos detendremos en ello pues no es relevante para el trabajo. La función de filtrado se realiza mediante el procesado de la señal muestreada mediante *software*.

Hay dos maneras de diseñar los filtros digitales:

- Diseño indirecto: que se basa en transformar un filtro analógico que existía previamente en uno digital. [28]
- Diseño directo: que se basa en determinar los coeficientes del filtro en base a unas condiciones de la respuesta esperada de éste. [28]

Los filtros digitales pueden o no tener realimentación con los valores de salida de este, por ello se dividen en dos grupos principales:

- Los filtros de respuesta finita al impulso donde la salida solo depende de los valores muestrales de la entrada.
- Los filtros de respuesta infinita al impulso cuya salida depende de los valores muestrales de la entrada y de los resultados de la salida.

A continuación, se profundizará un poco más en la forma de los filtros digitales según su realimentación, pues nuestra propuesta de mejora es un filtro adaptativo de respuesta finita al impulso.

6.1.1 Filtros de respuesta finita al impulso.

Los filtros FIR (*Finite Impulse Response*) [29] son sistemas cuya respuesta al impulso es de duración finita, esto quiere decir que estando en reposo el filtro, al ser estimulados con una entrada su salida vuelve al reposo tras este estimulo.

La salida de un filtro FIR de orden N será la suma ponderada de los N valores de entrada más recientes al filtro. Se define por tanto la salida por ecuación en diferencias como:

$$y(i) = w_0 x(i) + w_1 x(i-1) + \cdots + w_N x(i-N) = \sum_{k=0}^{N} w_k x(i-k) \quad \textbf{(10)}$$

Donde:

- **x(i):** La entrada al filtro digital.
- **y(i):** La salida o respuestas del filtro digital
- **N:** Es el orden del filtro digital. En el caso de los filtros directos también es el orden del filtro.
- $\boldsymbol{w_i}$**:** Son las respuestas al impulso del filtro digital. En el caso de ser un filtro digital directo también corresponden a los coeficientes del filtro. [28]

En la Figura 18 se pude ver el diagrama de un filtro FIR directo donde cada uno de los coeficientes del filtro son también las respuestas al impulso de este. El filtro en el que trabajaremos tendrá esta forma, pero sus coeficientes serán variables en el tiempo.

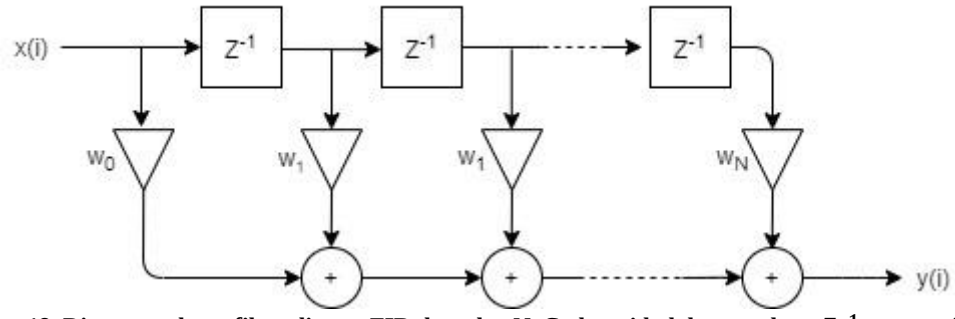

Figura 18. Diagrama de un filtro directo FIR de orden N. Cada unidad de retardo es Z^{-1} en notación de transformada Z.

6.1.2 Filtros filtros de respuesta infinita al impulso.

Los filtros IIR (*Infinite Impulse Response*) son sistemas cuya respuesta al impulso puede ser de duración infinita dado que tras ser estimulados su salida no vuelve necesariamente a estado de reposo debido a la recursividad que tiene la salida consigo misma.

La salida de un filtro IIR se define por ecuación en diferencias como:

$$y(i) = w_0 x(i) + w_1 x(i-1) + \cdots + w_M y(i-M) - v_1 y(i) - v_2 y(i-1) - \cdots - v_N y(i-N) = \sum_{k=0}^{M} w_k x(i-k) - \sum_{k=1}^{N} v_k y(i-k) \quad \textbf{(11)}$$

Donde:

- **x(i):** La entrada al filtro digital.
- **y(i):** La salida o respuestas del filtro digital
- **N:** Es el número de coeficientes que corresponden a la salida del filtro.
- **M:** Es el número de coeficientes que corresponden a la entrada del filtro.
- $\boldsymbol{w_i}$**:** Son las respuestas al impulso de entrada del filtro digital. En el caso de ser un filtro digital directo también corresponden a los coeficientes del filtro. [28]
- $\boldsymbol{v_i}$**:** Son las respuestas al impulso de la salida del filtro digital. En el caso de ser un filtro digital directo también corresponden a los coeficientes del filtro. [28]

El orden de un filtro IIR es el máximo de M y N.

En la Figura 19 se puede ver el diagrama de la ecuación por diferencias de un filtro IIR.

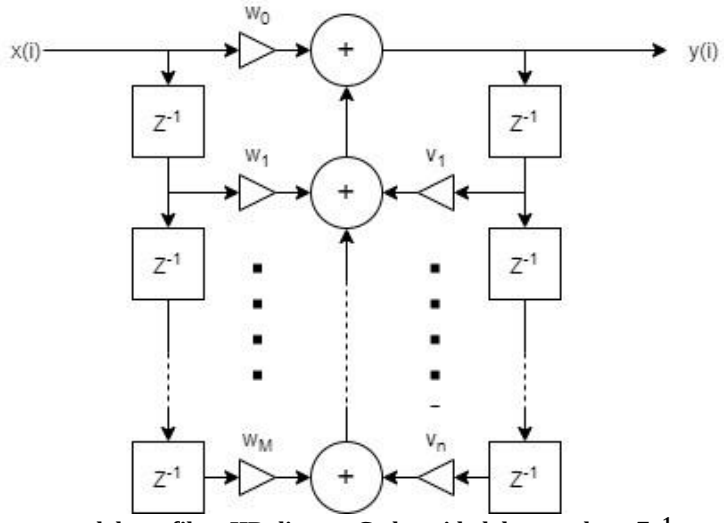

Figura 19. Diagrama general de un filtro IIR directo. Cada unidad de retardo es Z^{-1} en notación de transformada Z.

6.2 Los filtros adaptativos.

Un filtro adaptativo es un filtro lineal digital que tiene una función de transferencia cuyos coeficientes son variables en el tiempo y controlados acorde a un algoritmo de optimización (se autoajustan). Son usados para aplicaciones que requieren de un filtro que se adapte a las circunstancias de una señal en tiempo real como puede ser la cancelación de eco o ruido, la predicción lineal o la identificación de sucesos en un sistema. Por ello son filtros que pueden ser ajustados para mantenerse fijos o que tengan una constante adaptación.

El diagrama de bloques que representa un filtro adaptativo es este:

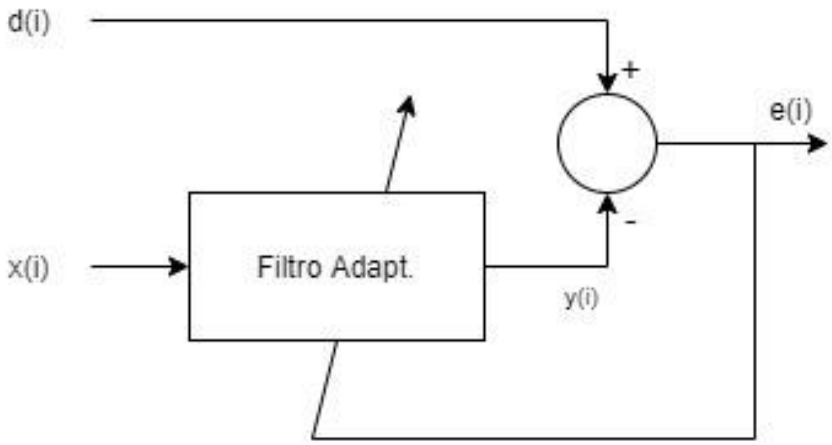

Figura 20. Diagrama de un filtro adaptativo.

- **x(i):** Señal de entrada en la ranura temporal i.
- **y(i):** Señal de salida
- **d(i):** Referencia o señal deseada.
- **e(i):** Señal de error.

El filtro adaptativo se presenta como un filtro de respuesta infinita al impulso (FIR) directo donde los coeficientes pueden variar en cada ranura temporal *i*:

$$y(i) = \sum_{k=0}^{L} w_k(i)x(i-k) \quad \textbf{(12)}$$

Al ser un filtro FIR directo L es la longitud del filtro y su orden.

Denominamos tanto el vector de valores de entrada como el vector de coeficientes del filtro como:

$$\underline{X}(i) = [x(i), x(i-1), \dots, x(i-L)] \quad \textbf{(13)}$$

$$\underline{W}(i) = [w_0(i), w_1(i), \dots, w_L(i)] \quad \textbf{(14)}$$

Por ende, la expresión (12) nos queda como:

$$y(i) = \underline{W}^T(i)\underline{X}(i) \quad \textbf{(15)}$$

Para la adaptación de los valores del filtro se busca minimizar la potencia de la señal de error. Eso quiere decir reducir al máximo la diferencia entre la salida del filtro $y(i)$ y la señal esperada representada por $d(i)$. Esta señal esperada es la señal con la que se busca adaptar los coeficientes del filtro para que su salida $y(i)$ sea lo más parecida a esta.

Hay distintos algoritmos para actualizar los valores del filtro y reducir al máximo el error anteriormente comentado. Para la mejora del protocolo FSA-RDP he decidido hacer uso del algoritmo LMS (*least-mean-square*) que permite adaptar el filtro sin conocer a priori el desarrollo de la señal de entrada y de la deseada. Además, dicho algoritmo también se usa en [26] de donde se ha extraído la idea de mejora.

6.3 Funcionamiento del algoritmo LMS

El algoritmo LMS funciona de la siguiente manera:

1) Se recibe $x(i)$ y $d(i)$.
2) Se actualiza el vector de entrada $\underline{X}(i)$
3) Se calcula la salida como se indica en la ecuación 15.
4) Se calcula la señal de error:

$$e(i) = d(i) - y(i) \quad \textbf{(16)}$$

5) Se actualiza los coeficientes del filtro según la siguiente expresión [30]:

$$\underline{W}(i+1) = \underline{W}(i) + 2\mu(i)e(i)\underline{X}(i) \quad \textbf{(17)}$$

$\mu(i)$ se trata de una variable que controla la velocidad de convergencia de los valores del filtro. Si el valor de $\mu(i)$ es muy bajo el algoritmo tenderá a no llegar nunca a un valor de convergencia; y si es más grande que el límite superior tampoco convergerá por la desestabilidad de los valores del filtro. En el caso de LMS se demuestra [30] que para que el algoritmo converja:

$$0 < \mu(i) \leq \frac{1}{(L+1) \cdot Pot_x} \quad \textbf{(18)}$$

Donde:

$$Pot_x = E[x(i)^2] \quad \textbf{(19)}$$

Esta es la forma general del algoritmo LMS, pero para nuestro caso hace falta adaptar la entrada y la salida para que funcione como deseamos, que autoconfigure una probabilidad de permiso en función de la carga ofrecida por el *cluster*.

6.4 Adaptación del algoritmo LMS al caso de FSA-RDP

Nuestro objetivo es reducir el valor de *r* (probabilidad de acceso) conforme vaya cambiando el tráfico que se le ofrece al *gateway*. En primer lugar, debemos presuponer que el *gateway* es capaz de detectar colisiones, no necesariamente tiene que conocer cuántos nodos han intentado acceder a la RSF (subtrama de reserva), solo detectar si en un *minislot* ha ocurrido una colisión. En muchos casos esto se puede detectar observando el nivel de potencia recibida en el *minislot* correspondiente. Obviamente si solo un nodo ha accedido a un *minislot*, el paquete será recibido con éxito por el receptor/controlador.

Cabe destacar que este algoritmo está pensado para que el *gateway* no conozca explícitamente cuantos elementos activos hay en el *cluster*, sino simplemente los resultados de la contienda y el número de nodos que tiene el *cluster*. Así no es necesario que los sensores conectados al *gateway* tengan que avisar de su activación, solo funcionar como se ha explicado en la sección 4.1.

El algoritmo que se propone será ejecutado por el *gateway* del *cluster* e informará del cambio del valor de *r* para la siguiente ranura dentro del mensaje SAP donde asigna los distintos DSF.

Denotaremos como:

- $sacc(i)$: Accesos exitosos a DSF (subtrama de datos) por parte de los nodos en la trama i, serán aquellos que han superado la contienda.

- $Arsf(i)$: Número de *minislots* de contienda (RSF) al que ha intentado acceder algún nodo en la trama i (haya habido colisión o acceso exitoso).

- $g(sacc(i), Arsf(i))$. Es la función previa a nuestro filtro adaptativo. Con ella se escalan los valores de entrada al filtro. Tiene en cuenta el número de nodos que pueden acceder a la RSF, así se generaliza para cualquier tamaño de *cluster*.

  Su forma es la siguiente:

$$g\big(sacc(i), Arsf(i)\big) = \begin{cases} Si\ Arsf(i) - sacc(i) = 0, & g(sacc(i), Arsf(i)) = 1 \\ Si\ Arsf(i) - sacc(i) > 0\ , & g(sacc(i), Arsf(i)) = \dfrac{V}{sacc(i) + \frac{N}{V}(Arsf(i) - sacc(i))} \end{cases} \quad \textbf{(20)}$$

  Desarrollando la formula queda resultante como:

$$g\big(sacc(i), Arsf(i)\big) = \begin{cases} Si\ Arsf(i) - sacc(i) = 0, & g(sacc(i), Arsf(i)) = 1 \\ Si\ Arsf(i) - sacc(i) > 0\ , & g(sacc(i), Arsf(i)) = \dfrac{V^2}{N(Arsf(i) - sacc(i)) + V \cdot sacc(i)} \end{cases} \quad \textbf{(21)}$$

  Que es la utilizada en el desarrollo de la simulación.

  La expresión (20) nace de reducir el valor de *r* en función de las colisiones, que es la información que puede tener el *gateway* de la carga que se está generando dentro del *cluster*. Si el tráfico aumenta significa que más sensores quieren acceder y por tanto más colisiones se suceden. La expresión $(Arsf(i) - sacc(i))$ es el número de *minislots* de la RSF donde se ha dado una colisión. Esta expresión está multiplicada por la esperanza del número de accesos en cada uno de los *minislots* ($E[\text{nº accesos por } minislot] = \frac{N}{V}$). Tiene esta forma porque la selección de los *minislots* se realiza mediante una distribución

normal, y por tanto es igual al número de nodos que hay entre los posibles *minislots* de reserva. Así lo que se hace es suponer que por cada una de las colisiones han intentado acceder tantos sensores como corresponden de media. Si el *cluster* es grande se supone que han contendido más que si el *cluster* es pequeño.

- L: número de coeficientes del filtro.

- $x(i) = g(i)$. Es el valor de entrada a nuestro filtro.

- $\underline{W}(i)$: coeficientes del filtro en la trama i. Es necesario darle valores al filtro antes del inicio del proceso, ósea $\underline{W}(0)$. En nuestro caso inicializamos el filtro con los valores:

$$\underline{W}(0) = [1, \frac{1}{2}, \frac{1}{4} \dots, \frac{1}{2^L}] \quad \textbf{(22)}$$

Así el primer valor de $\underline{X}(i)$, que es $x(i)$, tiene más peso que el resto y propicia una convergencia más rápida en caso de cambios en la carga ofrecida por los sensores del *cluster*.

- $\underline{X}(i)$: Conjunto de valores que entra en el filtro adaptativo. El *buffer* se inicializa con todas las posiciones igual a 1. Así nos aseguramos de que durante unas tramas *r* mantenga un valor cercano a 1 y posteriormente vaya adaptándose a las características del *cluster* en el que trabaja el *gateway.*

$$\underline{X}(0) = [1, 1, \dots, 1] \quad \textbf{(23)}$$

- $d(i) = x(i) = g(i)$: El valor esperado del filtro es el valor de $x(i)$. Esto es debido a que se busca que se reduzca la salida del filtro proporcionalmente a la infrautilización de las DSF debido a colisiones en la RSF. Así el filtro buscará reducir o aumentar *r* en función de la cantidad de colisiones que se den en la subtrama de reserva.

- $\mu(i)$: Es el parámetro para controlar la convergencia y se calcula en cada iteración en función del valor $x(i)$. Tomando como valor de potencia de la señal de entrada $x(i)^2$ debido a que la elección de ranura está generada aleatoriamente por una normal, la salida de la función $g(i)$ mantiene la misma distribución. Por lo tanto, la potencia de la señal que entra al filtro es la autocorrelación de dicho valor [31].

$$\mu(i) = \frac{1}{(L+1) \cdot x(i)^2} \quad \textbf{(24)}$$

Se trunca el valor de de $\mu(i)$ según la ecuación (18) dado que buscamos un cambio rápido en los valores del filtro. Así cuando sucede un gran número de accesos al *gateway* el valor de *r* puede reducirse rápidamente para evitar colisiones innecesarias.

- $y(i)$: Salida del filtro de impulso lineal representado por $\underline{W}(i)$.

$$y(i) = \underline{W}^T(i)\underline{X}(i) \quad \textbf{(25)}$$

- $f\big(y(i)\big)$: Función encargada de dar la salida del filtro que será el valor de $r(i+1)$. Es decir, el valor de probabilidad de permiso en la trama siguiente. Este añadido es posterior a la idea inicial debido al desarrollo del propio filtro. Dado que en ocasiones de gran congestión durante la simulación $y(i)$ tendía a valores inferiores a 0 durante sucesivas tramas, ha sido necesario realizar una función de activación que impidiera estos casos y

aunque *r* se mantiene en un valor pequeño, permite que parte de los nodos puedan ir accediendo a la RSF. Esto mejora el rendimiento e impide fluctuaciones aleatorias de comportamiento en el sistema al ser simulado.

El límite superior es obvio, dado que la probabilidad no puede ser superior a 1. El límite inferior debe ser la mínima probabilidad de acceso que permita el máximo de accesos. En una trama *i* la probabilidad óptima sería *V/Na*, siendo *Na* el número de nodos activos en dicha trama. Así conforme haya más nodos activos que *minislots* de contienda se reducirá *r* proporcionalmente. Dado que el subconjunto mayor de nodos activos es la población total de los nodos del *cluster* activados, el límite inferior será V/N.

$$f\big(y(i)\big) = \begin{cases} r(i+1) = V/N & si \quad y(i) < V/N \\ r(i+1) = y(i) & si \quad V/N \le y(i) \le 1 \\ r(i+1) = 1 & si \quad y(i) > 1 \end{cases} \tag{26}$$

En la Figura 21 se puede observar la representación mediante un diagrama del sistema que se encarga de adaptar *r*.

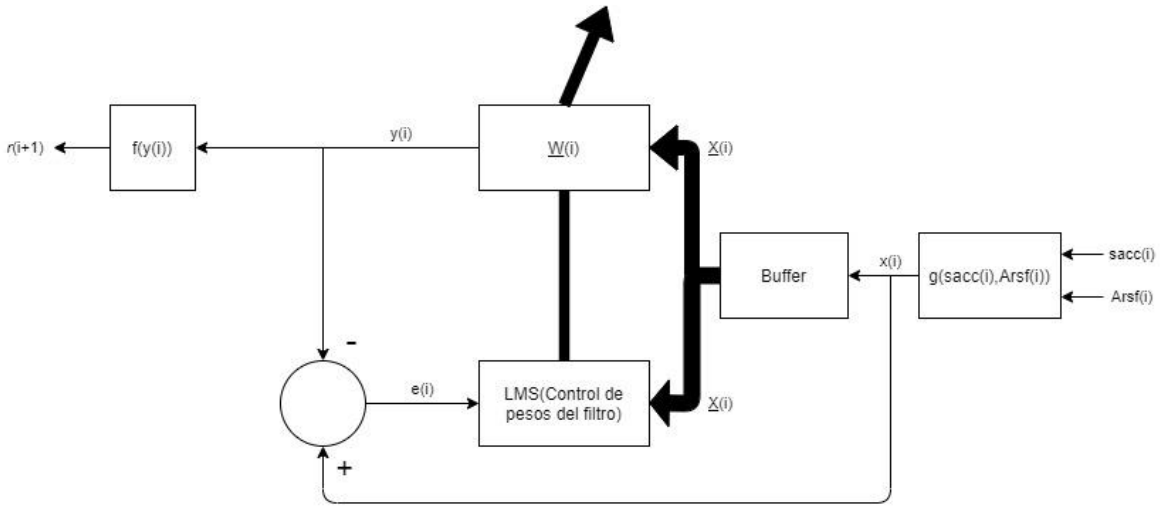

Figura 21. Diagrama del sistema de adaptación de r

Así el algoritmo de LMS adaptado a nuestro caso queda como:

Inicialización del algoritmo

Se inicializan los valores del filtro y del *buffer* como se indica en la expresión 22 y 23 respectivamente

Desarrollo del algoritmo

1) Se realizan las contiendas en las subtramas RSF y se obtiene el número de accesos exitosos ($sacc(i)$) y el número de subtramas que han tenido acceso, aunque haya habido colisión ($Arsf(i)$).
2) Se calcula $x(i) = g(sacc(i), Arsf(i))$, cuyo desarrollo se indican en las ecuaciones 20 y 21.
3) Se actualiza el buffer que se representa como $\underline{X}(i)$.
4) Se toma como valor deseado $x(i)$. O sea, $d(i) = x(i)$.
5) Se calcula $\mu(i)$ como se indica en la ecuación 24.
6) Se calcula y(i) como se indica en la ecuación 25.
7) Se obtiene el valor de error $e(i)$ como se indica en la ecuación 16.

8) Se actualizan los valores del filtro como:

$$\underline{W}(i+1) = \underline{W}(i) + \mu(i)\, e(i)\underline{X}(i) \quad \textbf{(27)}$$

9) Se obtiene el valor de $r(i+1)$ como se indica en la ecuación 26.
10) Se envía el valor de $r(i+1)$ con la información de éxitos de contienda y se espera a la siguiente trama para empezar otra vez el algoritmo desde el punto 1)

Capítulo 7. Comparación cuantitativa de FSA-RDP y FSA-DQ

Para probar el rendimiento de ambos protocolos se plantean dos escenarios distintos. En el primero de ellos se comparan las prestaciones de FSA-DQ y de FSA-RDP, buscando también comparar las prestaciones de FSA-DQ con la subtrama de datos fija o variable. En el segundo escenario y tras tomar el mejor caso de FSA-DQ (con subtrama fija o variable), se comparan las prestaciones de ambos protocolos (FSA-DQ y FSA-RDP) con poblaciones de distintos tamaños conectados a un mismo *gateway*.

Todas las figuras a las que se refiera el texto se encontraran en el **¡Error! No se encuentra el origen de la referencia.**. Además, cuando se hable de la carga se utilizará la siguiente convención:

$$\begin{cases} \rho_T \approx 0.3 \ \rightarrow \text{Carga baja} \\ \rho_T \approx 0.5 \ \rightarrow \text{Carga media} \\ \rho_T \approx 0.8 \ \rightarrow \text{Carga alta} \end{cases} \quad \textbf{(28)}$$

En todos los casos de simulación para el protocolo FSA-RDP se ha calculado la probabilidad de permiso optima (r_{opt}) mediante método empírico. Para cada uno de los puntos de carga total del sistema se buscaba mediante sucesivas simulaciones cual es el valor de la probabilidad para maximizar el caudal del sistema.

Las prestaciones que son representadas en las distintas gráficas son:

- El 95º percentil del retardo de paquete (D_{95}).
- La probabilidad de pérdidas del sistema (P_p).
- El uso medio del canal (S).
- Y el caudal cursado por el sistema (*Th).*

7.1 Primer escenario.

La configuración de este escenario se describe en las siguientes tablas. Todos los casos tienen el mismo número de dispositivos conectados al *gateway* para así poder observar el comportamiento de estos ante la misma carga del sistema.

Caso FSA-RDP			
Parámetros	Valor		Símbolo
Número de dispositivos	10	dispositivos	N
Tamaño de la cola del dispositivo	10	paquetes	Q
Duración de los paquetes de datos	10	minislots	W
Número de subtramas de reserva	4	subtramas	V
Número de subtramas de datos	4	subtramas	saccx

Tabla 2. Configuración del caso de simulación de FSA-RDP en el primer escenario.

Caso FSA-DQ con DSF fija			
Parámetros	Valor		Símbolo
Número de dispositivos	10	dispositivos	N
Tamaño de la cola del dispositivo	10	paquetes	Q
Duración de los paquetes de datos	10	minislots	W
Número de subtramas de reserva	4	subtramas	V
Número de subtramas de datos	2	subtramas	saccx

Tabla 3. Configuración del caso de simulación de FSA-DQ con DSF fija en el primer escenario.

Caso FSA-DQ con DSF variable			
Parámetros	Valor		Símbolo
Número de dispositivos	10	dispositivos	N
Tamaño de la cola del dispositivo	10	paquetes	Q
Duración de los paquetes de datos	10	minislots	W
Número de subtramas de reserva	4	subtramas	V
Número de subtramas de datos	4	subtramas	saccx

Tabla 4. Configuración del caso de simulación de FSA-DQ con DSF variable en el primer escenario.

Tanto el caso de FSA-RDP como el de FSA-DQ con DSF variable tienen el mismo número de subtramas de reserva como de datos. El caso de FSA-DQ con DSF fijo se elige con menos subtramas de datos para que se asemeje el caso lo máximo al modelo propuesto por [23]. La duración de los paquetes y el tamaño de las colas se mantiene entre los casos para, como se ha indicado antes, equiparar cargas generadas por los dispositivos y así ver las características de funcionamiento de los distintitos casos de simulación.

7.1.1 Comentario sobre las gráficas obtenidas por simulación

- **95º percentil del retardo de paquete**

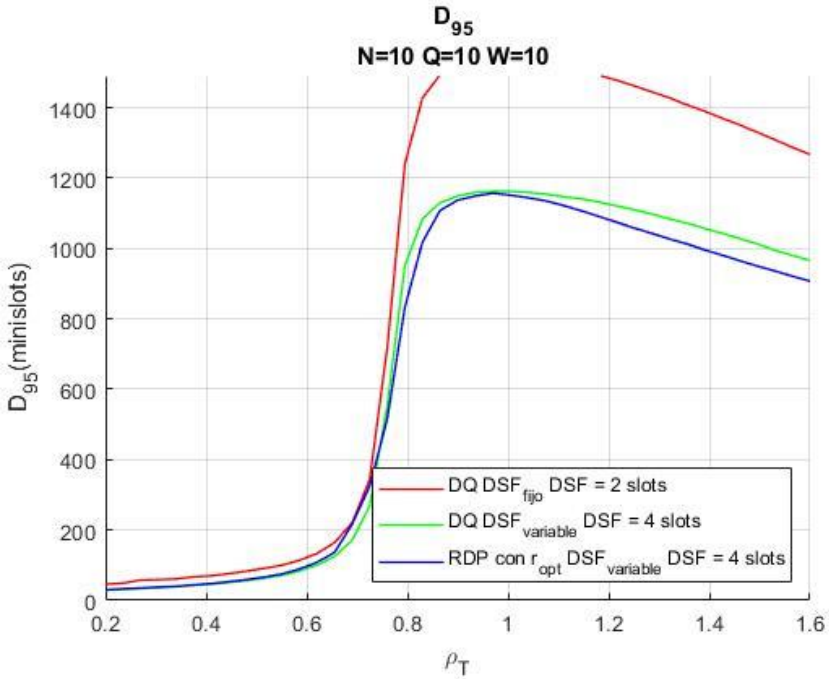

Figura 22. Percentil del 95% del retraso en el escenario 1 de simulación.

En la Figura 22 se puede observar el 95º percentil del retardo de paquete para la configuración de los distintos protocolos en este escenario. FSA-RDP y FSA-DQ con la subtrama de datos variable tienen un retraso parecido, siendo el caso de FSA-RDP menor a partir de cargas altas y con el sistema en saturación. FSA-DQ con trama variable tiene un retraso un poco menor en cargas medias respecto a FSA-RDP. En cargas bajas trabajan igual los dos protocolos con dicha configuración.

En contra partida FSA-DQ con la subtrama de datos fija tiene un retraso parecido a FSA-RDP y FSA-DQ con subtrama fija hasta cargas medias. A partir de ese punto se dispara el retardo de FSA-DQ con subtrama de datos fija, siendo bastante superior a los dos casos anteriores.

- **Caudal cursado**

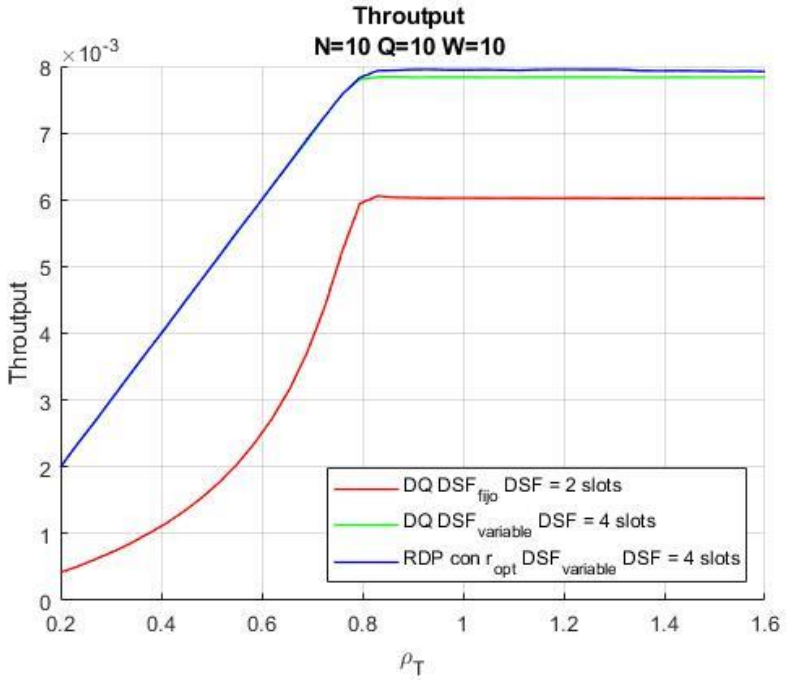

Figura 23. Caudal medio servido en el escenario 1 de simulación.

El caudal cursado por parte del sistema se muestra en la Figura 23. Por parte de los protocolos toman su valor máximo a partir de cargas altas. Tanto FSA-RDP como FSA-DQ con subtrama de datos variable consiguen un caudal 25% superior respecto a FSA-DQ con subtrama de datos fijo.

Los caudales de FSA-RDP y de FSA-DQ con subtrama de datos variable son iguales hasta cargas altas, aunque a partir de este punto de utilización del sistema, el caudal servido por FSA-RDP es un poco superior respecto a FSA-DQ con subtrama de datos variable.

- **Probabilidad de pérdidas**

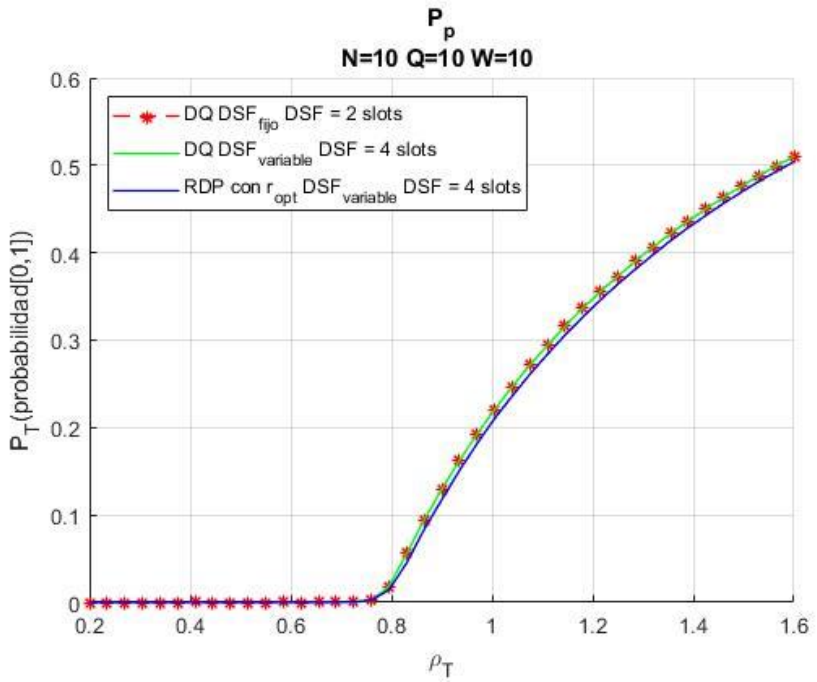

Figura 24. Probabilidad de perdida en el escenario 1 de simulación.

La probabilidad de pérdidas se muestra en la Figura 24. Tanto FSA-RDP como FSA-DQ con subtrama de datos fija y variable mantienen las mismas pérdidas en toda la carga ofrecida. Estas pérdidas se mantienen en un 0% hasta cargas altas. A partir de este punto ascienden hasta un 50% de perdidas cuando $\rho_T = 1.6$.

- **Uso de canal**

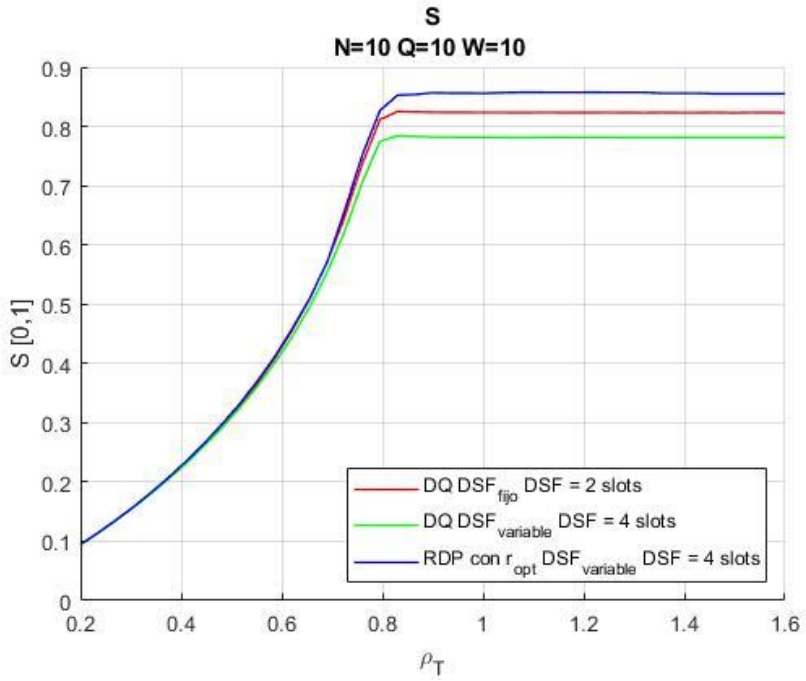

Figura 25. Uso medio del canal en el escenario 1 de simulación.

La tasa de uso efectivo del canal (que se utiliza para enviar paquetes) se muestra en la Figura 25. El uso del canal en este caso es prácticamente el mismo en FSA-RDP, FSA-DQ con subtrama de datos fija y FSA-DQ con subtrama de datos variable.

A partir de cargas altas, cuando el caudal de todos ellos llega al máximo, el uso del canal es diferente para éstos. FSA-RDP es el que mayor uso de canal tiene, seguido por FSA-DQ con subtrama de datos fija y el que menor uso de canal hace es FSA-DQ con subtrama de datos variable.

7.1.2 Conclusiones sobre los resultados del escenario.

A la vista de los resultados obtenidos para este escenario se puede afirmar que el funcionamiento de FSA-RDP (con r_{opt}) es mejor en todas las prestaciones estudiadas. La diferencia entre FSA-RDP y FSA-DQ con DSF variable es mínimo, sobre todo en el caudal servido por el sistema (Figura 23). En cuanto a las pérdidas es despreciable la diferencia que tiene FSA-RDP con ambas versiones de FSA-DQ (DSF fijo y DSF variable).

Este escenario de estudio también permite comparar el funcionamiento de FSA-DQ con DSF variable y fijo. Salvo en el uso medio del canal (Figura 25) que FSA-DQ con DSF fijo es superior, en el resto de los parámetros FSA-DQ con DSF variable es superior. Sobre todo, se observa dicha diferencia en el caudal (Figura 23) y en el 95º percentil de retardo de trama (Figura 22). Este segundo parámetro es de gran interés, dado que en momentos de carga alta en redes IoT se requerirá la respuesta más rápida posible del protocolo. Este uso superior del canal por parte de FSA-DQ hipotéticamente es debido a que tiene el doble de *minislots* de contención que *slots* de paquetes.

Por ende, podemos afirmar que con bajo número de nodos el protocolo FSA-RDP tiene un desempeño mejor tanto en el uso del canal inalámbrico como en reducir al máximo el retardo de

las tramas que llegan al *gateway*, que posteriormente está encargado de reenviarlas hacia la BS de la red de acceso a la que esté conectada. Además, el desempeño de FSA-DQ es mejor con DSF variable que fijo dentro de los parámetros de mérito para este trabajo, aunque el uso del canal sea inferior al del caso con DSF fijo.

7.2 Segundo escenario.

Dado que en el anterior escenario se observó que FSA-DQ tiene mejor rendimiento en los parámetros de mérito propuestos para este trabajo cuando DSF es variable, en los siguientes casos de simulación utilizaremos esta nueva forma de operación del protocolo, más eficiente que el propuesto en [23].

7.2.1 Caso de simulación con 10 dispositivos.

FSA-RDP y FSA-DQ			
Parámetros	Valor		Símbolo
Número de dispositivos	10	dispositivos	N
Tamaño de la cola del dispositivo	10	paquetes	Q
Duración de los paquetes de datos	10	minislots	W
Número de subtramas de reserva	2	subtramas	V
Número de subtramas de datos	2	subtramas	saccx

Tabla 5. Configuración del caso de simulación para FSA-RDP y FSA-DQ.

Este escenario de estudio está pensado para ver el comportamiento de un *cluster* de pocos dispositivos. Para que se ajuste al máximo los recursos de comunicación que tendrían los sensores tanto la subtrama de colisión como de datos es la mínima posible (2 subtramas para colisión y 2 para enviar los datos).

- **95º percentil del retardo de paquete**

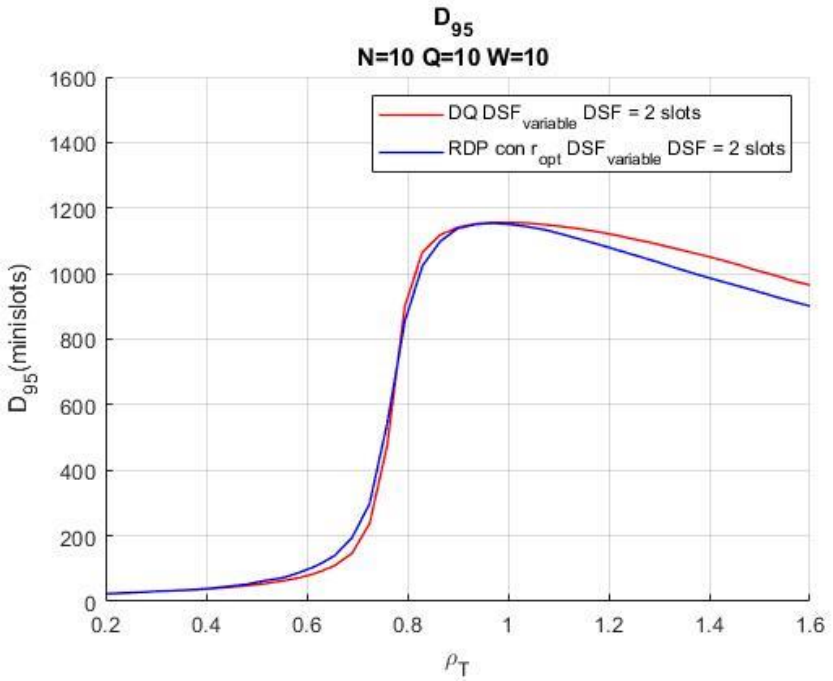

Figura 26. Percentil del 95% del retraso en el escenario 2 de simulación en el caso con 10 nodos.

En la Figura 26 se muestra el 95º percentil del retardo de paquete para el primer caso de este escenario. El comportamiento de ambos protocolos es semejante en toda la carga ofrecida.

Aun así, es un poco menor (el retardo) en el caso de FSA-DQ en cargas medias, mientras que cuando el sistema entra en saturaciones es FSA-RDP el que mantiene un retardo inferior.

- **Caudal cursado**

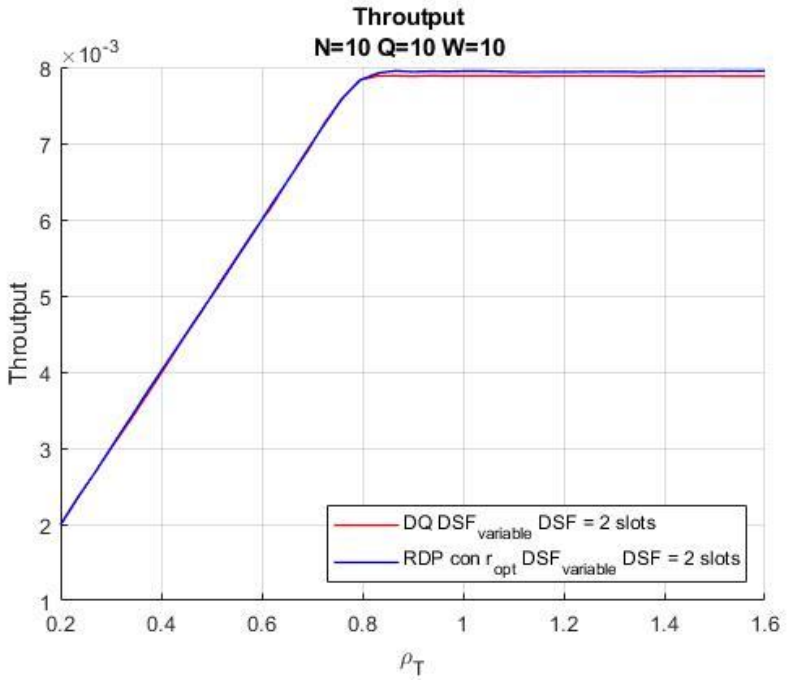

Figura 27. Caudal medio servido en el escenario 2 de simulación en el caso con 10 nodos.

En la Figura 27 se muestra el caudal cursado por el sistema en el primer caso del segundo escenario. En esta gráfica se observa como el caudal cursado es idéntico en ambos protocolos hasta cargas altas que, entrando el sistema en saturación, es FSA-RDP el que sirve un caudal un poco superior.

- **Probabilidad de pérdidas**

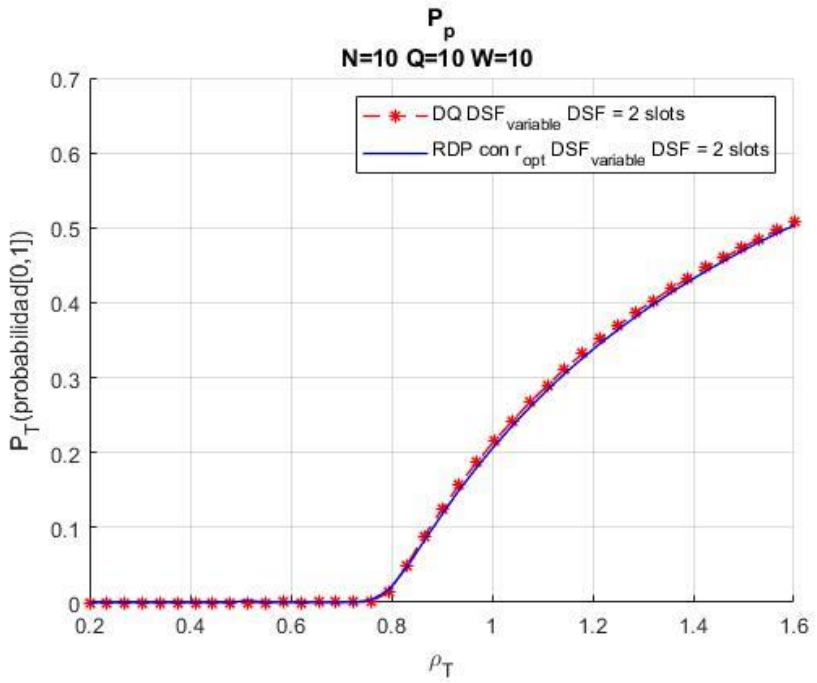

Figura 28. Probabilidad de perdida en el escenario 2 de simulación en el caso con 10 nodos.

En la Figura 28 se observan las pérdidas por desbordamiento de las colas en el primer caso del segundo escenario de estudio. Se observa como para ambos protocolos las pérdidas son idénticas en toda la carga suministrada. Se mantienen al 0% hasta cargas altas y ascienden hasta el 50% en la mayor carga ofrecida.

- **Uso de canal**

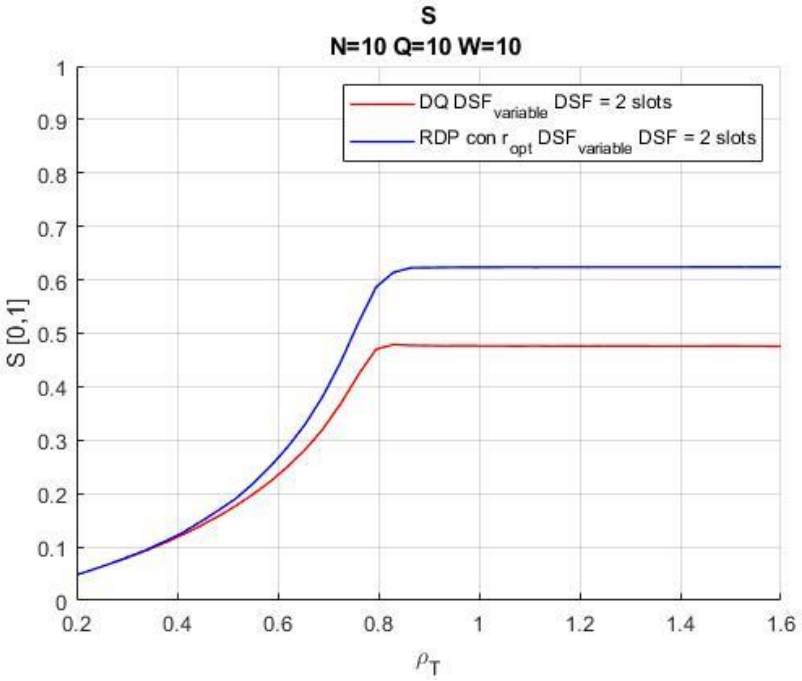

Figura 29. Uso medio del canal en el escenario 2 de simulación en el caso con 10 nodos.

En la Figura 29 se observa el uso medio del canal en el envío de paquetes en este caso. El uso del canal es superior en FSA-RDP a partir de cargas medias, aumentando su diferencia respecto a FSA-DQ hasta que el sistema entra en saturación donde la diferencia de uso llegas hasta el 15% aproximadamente.

7.2.2 Caso de simulación con 20 dispositivos.

FSA-RDP y FSA-DQ			
Parámetros	Valor		Símbolo
Número de dispositivos	20	dispositivos	N
Tamaño de la cola del dispositivo	10	paquetes	Q
Duración de los paquetes de datos	10	minislots	W
Número de subtramas de reserva	4	subtramas	V
Número de subtramas de datos	4	subtramas	saccx

Tabla 6. Configuración del caso de simulación para FSA-RDP.

Este escenario de estudio está pensado para ver el comportamiento de un *cluster* tamaño medio. Para que se ajuste al máximo los recursos de comunicación y dado que se han duplicado el número de dispositivos, el número de subtramas de colisión y de datos también se ha duplicado.

- **95º percentil del retardo de paquete**

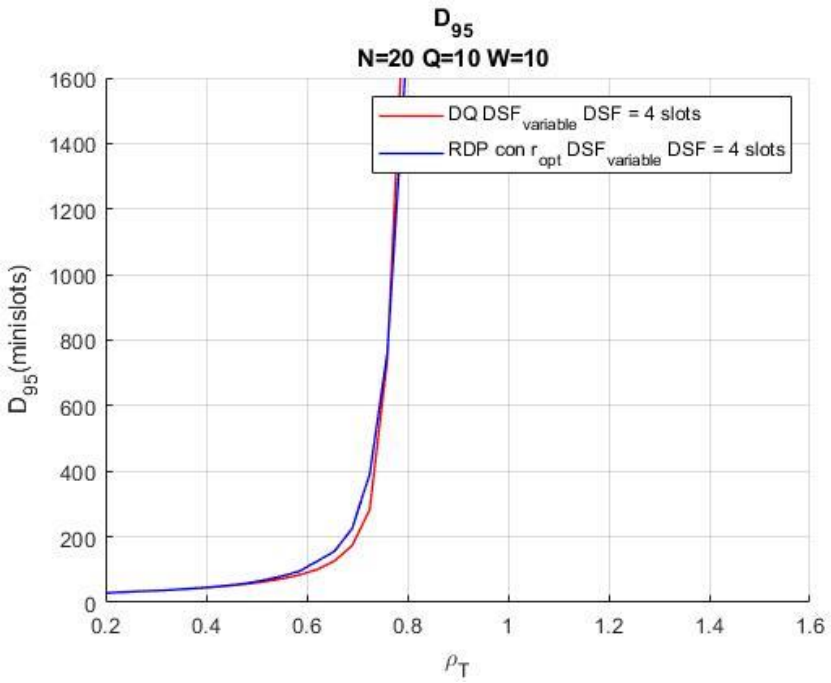

Figura 30. Percentil del 95% del retraso en el escenario 2 de simulación en el caso con 20 nodos.

En la Figura 30 se muestra el 95º percentil del retardo de paquete para el segundo caso de este escenario de estudio. Se observa como el retardo se mantiene parejo hasta cargas medias, donde FSA-DQ tiene un retardo un poco inferior a FSA-RDP. Tras esto y acercándose a cargas altas ambos protocolos tienden a disparar el retardo por encima de los 1600 *minislot*s.

- **Caudal cursado**

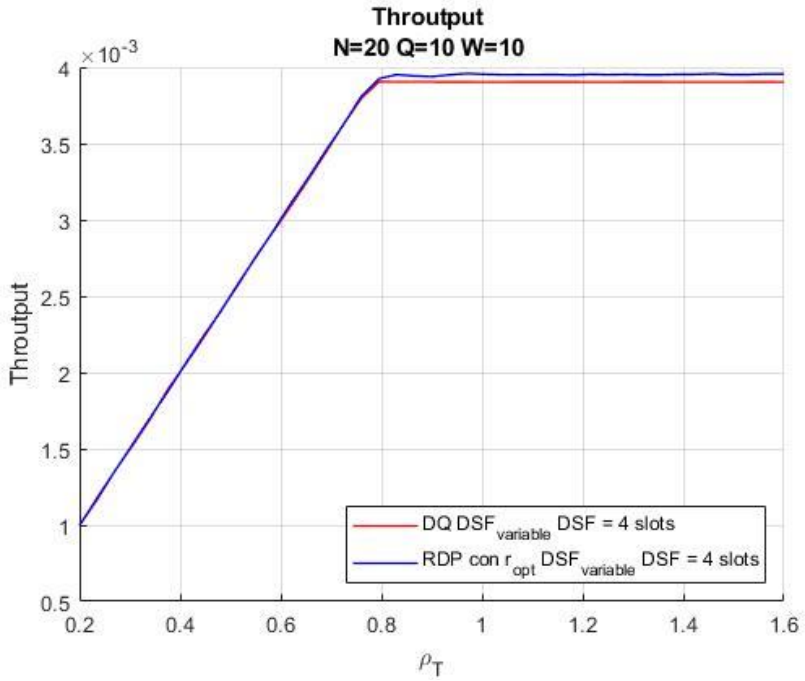

Figura 31. Caudal medio servido en el escenario 2 de simulación en el caso con 20 nodos.

En la Figura 31 se observa el caudal cursado por parte de los protocolos en este segundo caso del escenario de estudio. Como en el anterior caso, el caudal cursado se mantiene igual para ambos hasta cargas altas, donde a partir de ese punto de carga FSA-RDP es capaz de suministrar un poco más de caudal que FSA-DQ.

- **Probabilidad de pérdidas**

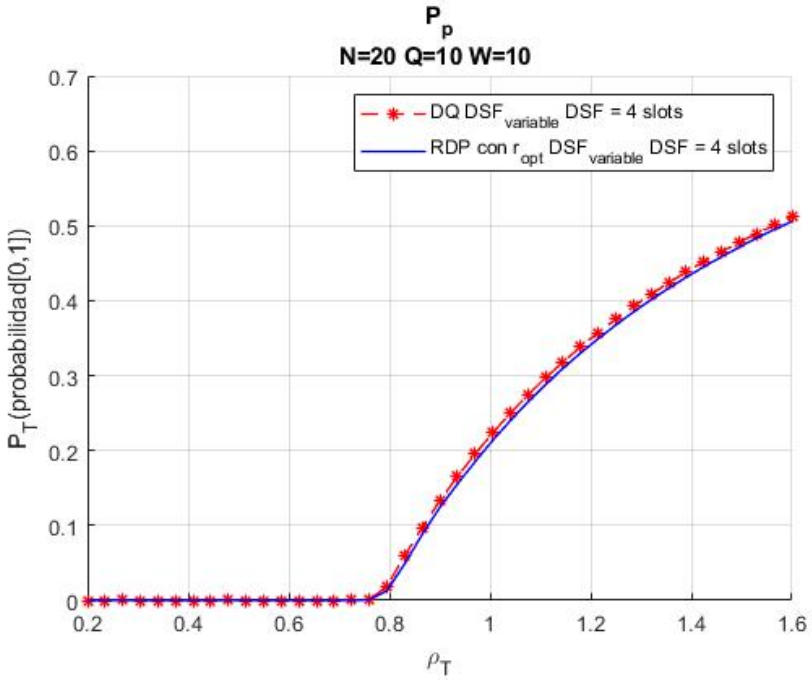

Figura 32. Probabilidad de perdida en el escenario 2 de simulación en el caso con 20 nodos.

En la Figura 32 se observa la probabilidad de pérdidas en este caso del segundo escenario de estudio. Como en el anterior hasta cargas altas ambos protocolos mantienen un 0% de probabilidad de pérdidas debido al desbordamiento de las colas. Tras ese punto de carga aumenta la probabilidad de pérdidas hasta el 50% en el protocolo FSA-RDP, siendo un poco superior la probabilidad de pérdidas para el protocolo FSA-DQ.

- **Uso de canal**

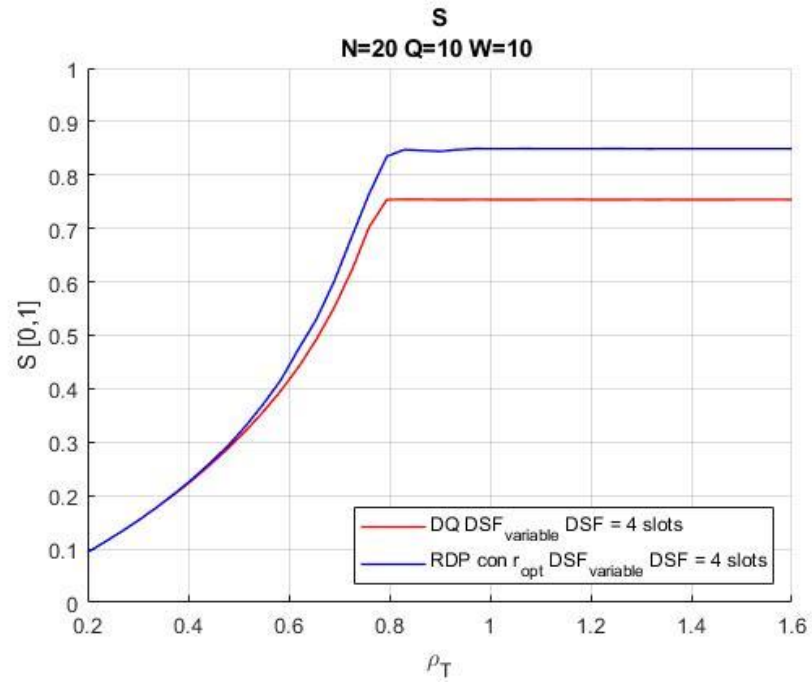

Figura 33. Uso medio del canal en el escenario 2 de simulación en el caso con 20 nodos.

En la Figura 33 se observa el uso medio del canal para el envío de paquetes. Como en el anterior caso de este escenario, se mantiene parejo hasta cargas medias donde FSA-RDP aumenta la capacidad de uso de canal hasta un 85% cuando el sistema supera cargas altas, mientras que FSA-DQ se mantiene en un 75% de uso.

7.2.3 Caso de simulación con 40 dispositivos.

FSA-RDP y FSA-DQ			
Parámetros	Valor		Símbolo
Número de dispositivos	40	dispositivos	N
Tamaño de la cola del dispositivo	10	paquetes	Q
Duración de los paquetes de datos	10	minislots	W
Número de subtramas de reserva	8	subtramas	V
Número de subtramas de datos	8	subtramas	saccx

Tabla 7. Configuración del caso de simulación para FSA-RDP.

Este escenario de estudio está pensado para ver el comportamiento de un *cluster* de tamaño grande. Para evitar que el sistema se sature con cargas medias de tráfico se vuelve a duplicar el número de subtramas de reserva y colisión.

- **95º percentil del retardo de paquete**

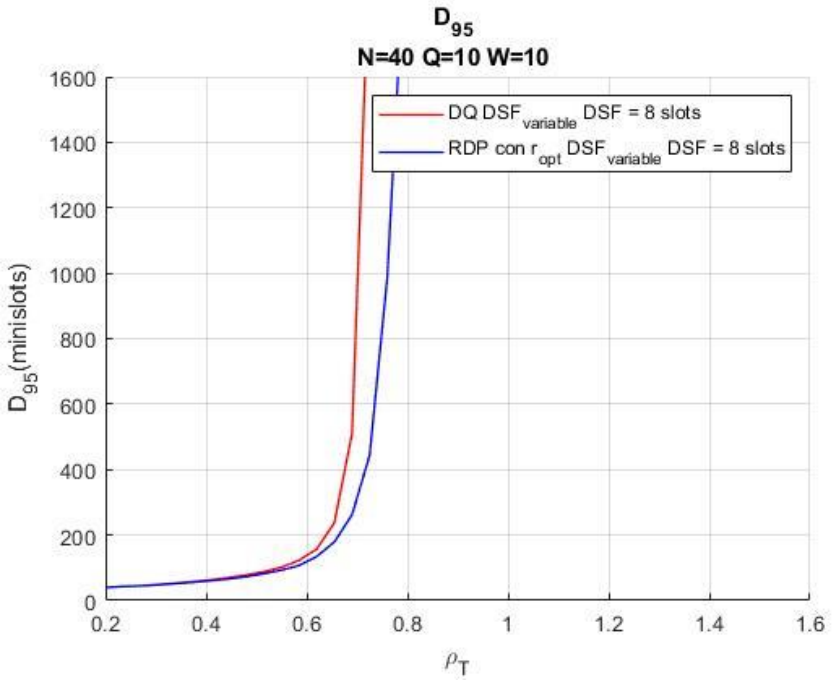

Figura 34. Percentil del 95% del retraso en el escenario 2 de simulación en el caso con 40 nodos.

En la Figura 34 se muestra el 95º percentil del retardo de paquete para el tercer caso de este segundo escenario de estudio. Se observa como hasta cargas medias el retardo de ambos protocolos es parejo. A partir de ese punto de carga el retardo en FSA-DQ es superior al de FSA-RDP. También, como en el anterior caso, a partir de dicho punto de carga media se dispara el retardo por encima de los 1500 *minislots*.

- **Caudal cursado**

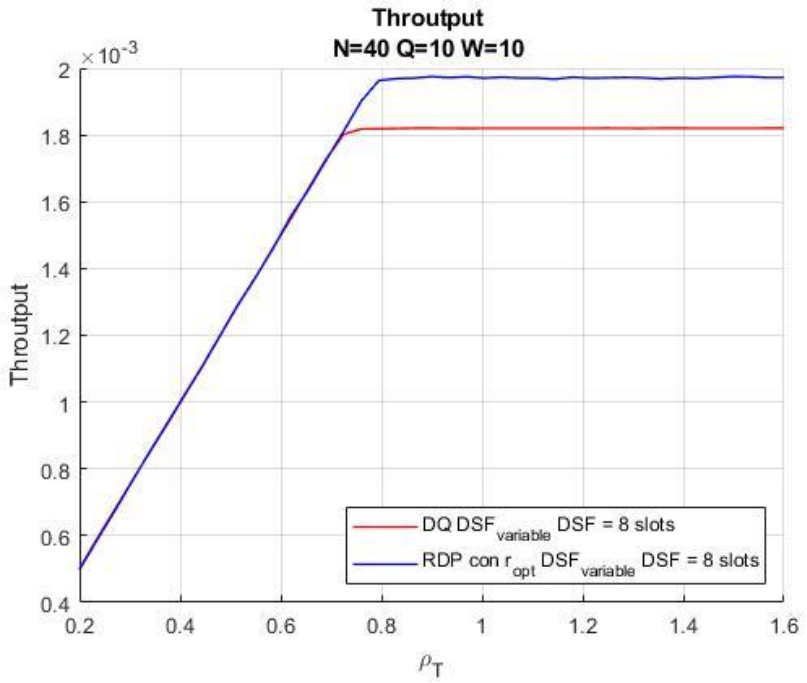

Figura 35. Caudal medio servido en el escenario 2 de simulación en el caso con 40 nodos.

En la Figura 35 se observa el caudal cursado por los protocolos en este tercer caso del segundo escenario de simulación. En este caso el caudal cursado es el mismo hasta un poco antes del punto de carga alta ($\rho_T \approx 0.75$), donde FSA-DQ llega a su máximo mientras que FSA-RDP sigue aumentando el caudal hasta el punto de carga alta. Así en este escenario, FSA-RDP sirve un caudal proporcionalmente superior al de FSA-DQ.

- **Probabilidad de pérdidas**

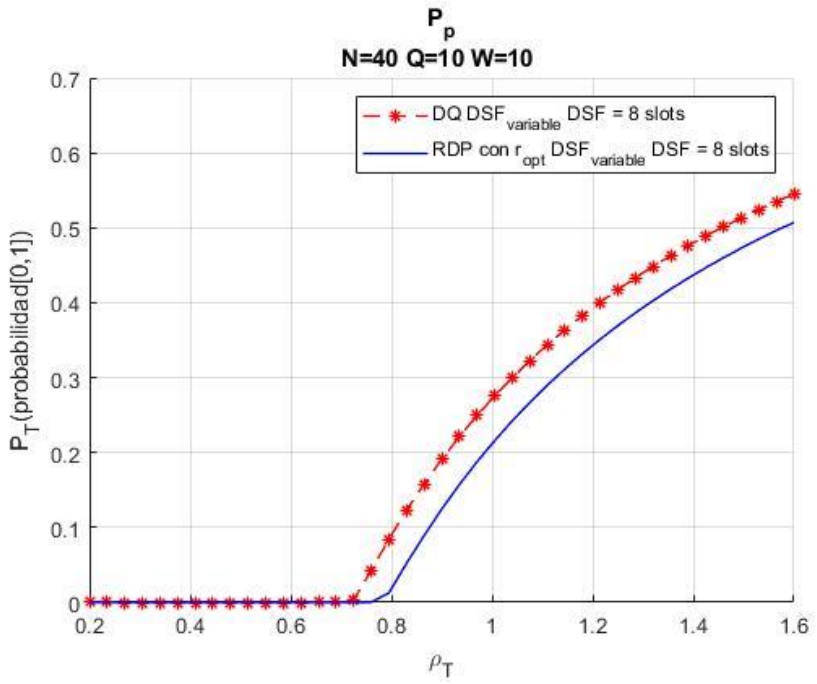

Figura 36. Probabilidad de perdida en el escenario 2 de simulación en el caso con 40 nodos.

En la Figura 36 se muestra la probabilidad de perdidas del tercer caso de este segundo escenario de simulación. FSA-DQ mantiene las pérdidas al 0% hasta $\rho_T \approx 0.75$ donde al entrar en saturación aumenta dichas pérdidas hasta el 55%. Mientras, FSA-RDP mantiene a 0% las

perdidas hasta cargas altas y como en los anteriores casos la probabilidad de pérdidas sube hasta el 50%.

- **Uso de canal**

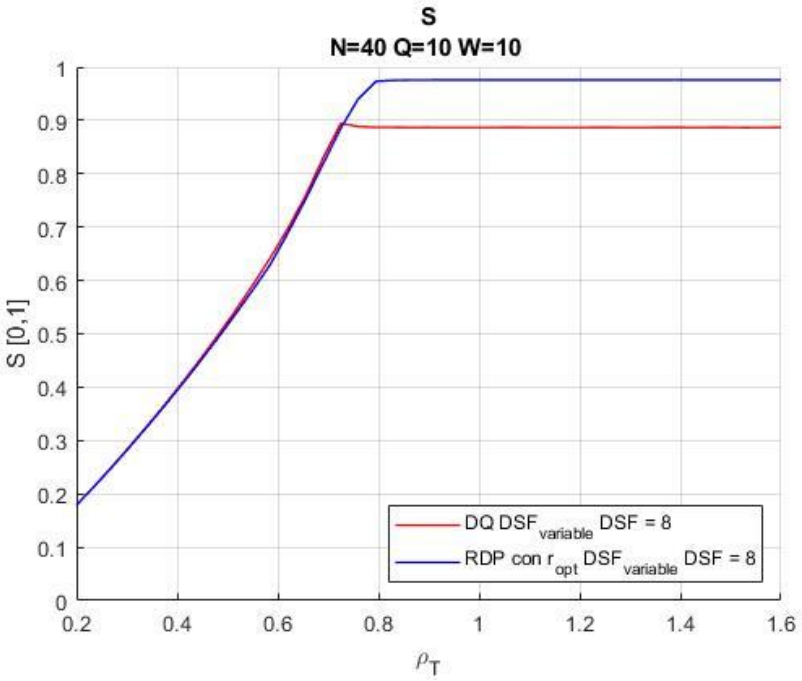

Figura 37. Uso medio del canal en el escenario 2 de simulación en el caso con 40 nodos.

En la Figura 37 se muestra el uso del canal medio en el envío de paquetes en este tercer caso del segundo escenario de simulación. El uso del canal es parejo en ambos protocolos hasta $\rho_T \approx 0.75$ donde FSA-DQ se mantiene aproximadamente en el 90%. FSA-RDP aumenta el uso del canal hasta aproximadamente el 98% a partir de cargas altas.

7.2.4 Conclusiones sobre los resultados del escenario.

En todos los casos el comportamiento del 95º percentil del retardo es semejante para los dos protocolos. Tanto en el caso de simulación de 20 nodos como de 40 nodos (Figura 30 y Figura 34 respectivamente), cuando la carga total del sistema se acerca a cargas altas se dispara el retardo de los paquetes. En ambos protocolos a cargas altas puntuales se prevé un retraso considerable en el envío de los paquetes, aunque FSA-RDP tiene un retardo inferior en el estado de saturación.

En cuanto al caudal ofrecido tanto en el caso de 10 dispositivos como de 20 dispositivos en el *cluster* (Figura 27 y Figura 31 respectivamente), la diferencia entre ambos protocolos es despreciable. Se puede observar una diferencia sustancial en el caso de 40 nodos (Figura 35) donde es superior el protocolo FSA-RDP respecto FSA-DQ.

Respecto a la probabilidad de pérdidas, ambos protocolos mantienen un 0% de pérdidas hasta llegar a cargas altas. A partir de dicha cota en los casos de *cluster* pequeño (Figura 28) y mediano (Figura 32) son semejantes con diferencias despreciables. Es en el caso del *cluster* grande (Figura 36) donde la diferencia de pérdidas aumenta. FSA-RDP mantiene las mismas pérdidas máximas en un 50%, mientras que FSA-DQ llega hasta un 55%, un 5% superior nada despreciable.

El uso del canal es superior en el protocolo FSA-RDP en todos los casos de simulación. La mayor diferencia es en el caso del *cluster* pequeño (Figura 29), siendo esta diferencia superior al 10%. Tanto en el caso del *cluster* medio (Figura 33) como en el grande (Figura 37), la diferencia se mantiene entorno al 10%. Es de señalar que en el caso del *cluster* grande FSA-RDP es capaz de llegar casi al 100% de uso del canal.

Capítulo 8. Estudio cuantitativo de la propuesta de mejora a FSA-RDP

Para el estudio cuantitativo del funcionamiento de la mejora propuesta, es decir, obtener el rendimiento del filtro probaremos los mismos casos del escenario 2 del anterior capítulo. En vez de comparar FSA-RDP y FSA-DQ, compararemos FSA-RDP con el filtro y sin el filtro. Sin el filtro tendremos dos posibles configuraciones: configurado con r_{opt}y configurado $r = 1$ (que funciona como si no hubiera probabilidad de acceso). Así podremos ver la mejora del funcionamiento con el filtro adaptativo instalado respecto a FSA-RDP sin probabilidad de acceso y cuán lejos se queda del funcionamiento óptimo del protocolo con una probabilidad fija.

El filtro usado en los distintos casos de simulación tiene el mismo tamaño, 20 índices en el filtro adaptativo, y por tanto 20 valores de *buffer* para guardar un histórico del valor de entrada al filtro.

En las figuras resultantes de las simulaciones se ha representado mediante una línea gruesa verde el comportamiento del protocolo con el filtro. Con una línea azul discontinua con estrellas el comportamiento del protocolo configurado con r_{opt}. El comportamiento del protocolo con permisividad total al tráfico se ha dibujado con una línea roja continua.

La convención para hablar de la carga será la misma que en el anterior capítulo, representada en la expresión (28).

8.1 Comentario de los resultados de las distintas simulaciones.

8.1.1 Percentil 95 de retardo.

El funcionamiento del protocolo añadiendo el filtro adaptativo respecto al retardo de transmisión del paquete no es homogéneo en los tres casos.

En los casos con 10 nodos (Figura 38) y 20 (Figura 39) nodos el percentil 95% de retardo es inferior a FSA-RDP sin probabilidad de acceso en toda la carga ofrecida. En comparación con FSA-RDP configurado con r_{opt} es igual en cargas bajas y altas. En cargas medias el retardo con el filtro añadido es superior a la configuración con r_{opt} (unos 150 *minislots* de diferencia) pero bastante inferior al retardo que sufren los paquetes sin el mecanismo de probabilidad de acceso

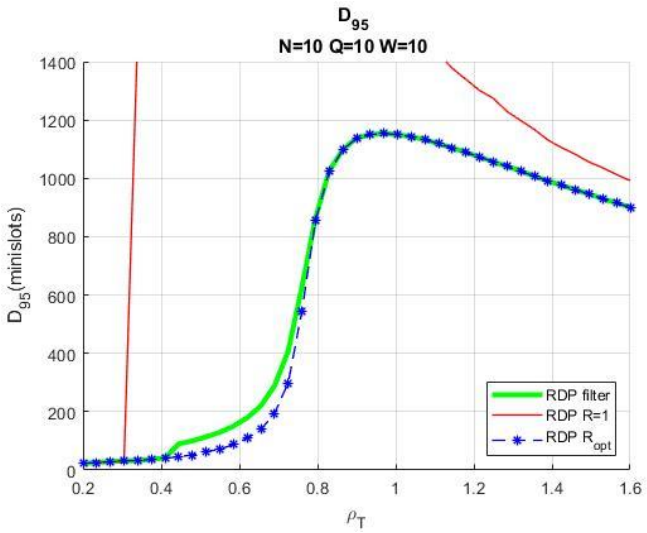

Figura 38. Percentil del 95% del retraso con 10 nodos.

.

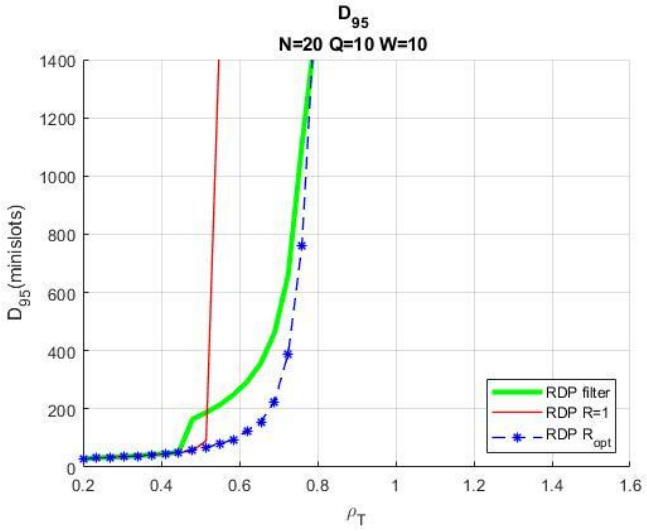

Figura 39. Percentil del 95% del retraso con 20 nodos.

En el caso con 40 nodos (Figura 40) el percentil 95% de retardo con el filtro es superior a la configuración con r_{opt} y sin la probabilidad de acceso en las cargas bajas y medias. Se aproxima al comportamiento con r_{opt} a partir de las cargas altas, mejorando considerablemente el comportamiento respecto a FSA-RDP sin probabilidad de acceso. Este comportamiento se debe posiblemente a una reducción drástica de *r* cuando trabaja el filtro, para evitar un exceso de colisiones en la RSF y así haya un aprovechamiento superior del canal.

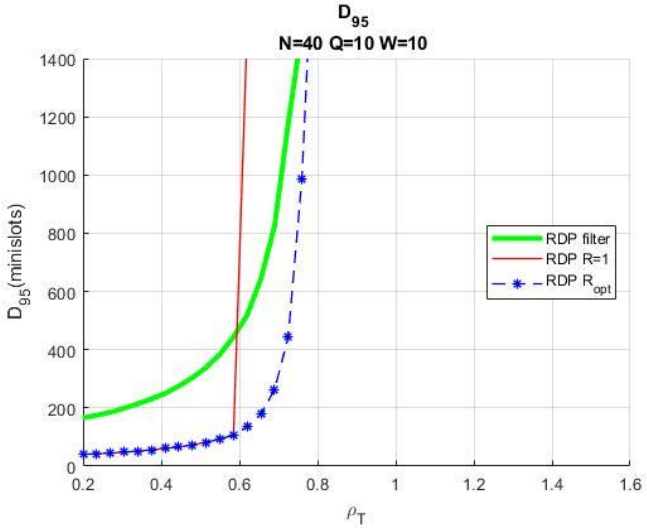

Figura 40. Percentil del 95% del retraso con 40 nodos.

8.1.2 Probabilidad de pérdidas.

El funcionamiento del protocolo añadiendo el filtro adaptativo es homogéneo en todos los casos respecto a la probabilidad de pérdidas.

Tanto en el caso de 10 nodos (Figura 41), 20 nodos (Figura 42) y 40 nodos (Figura 43) la probabilidad de pérdidas con el filtro es la misma que cuando se configura FSA-RDP con r_{opt}. Por tanto, reduce a la mitad las pérdidas generadas en comparación con FSA-RDP sin el mecanismo de probabilidad de acceso.

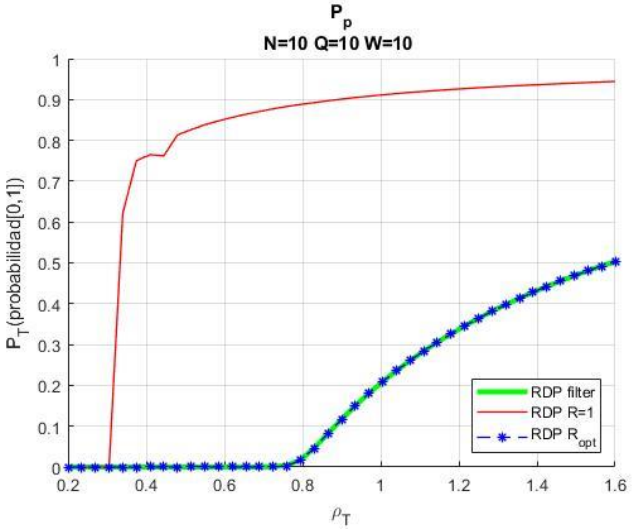

Figura 41. Probabilidad de pérdidas con 10 nodos.

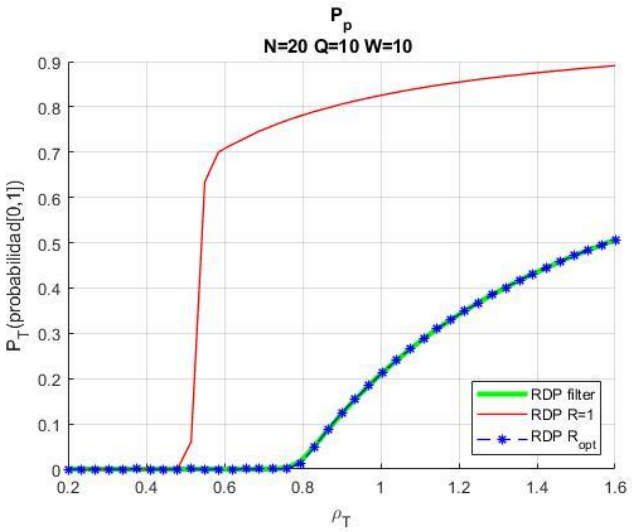

Figura 42. Probabilidad de pérdidas con 20 nodos.

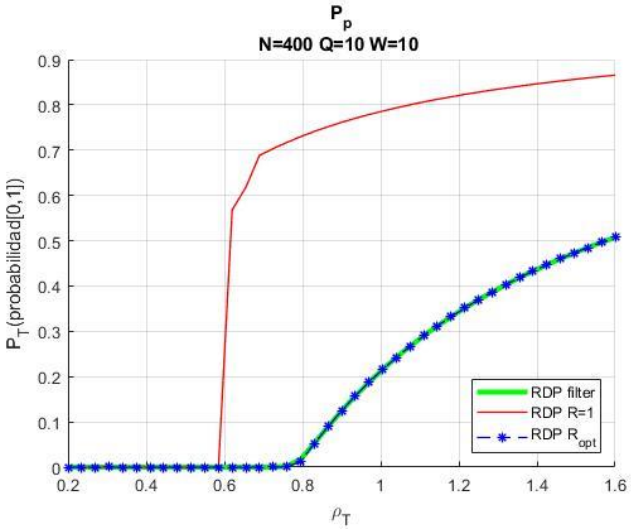

Figura 43. Probabilidad de pérdidas con 40 nodos.

8.1.3 Caudal cursado.

El funcionamiento del protocolo añadiendo el filtro adaptativo es homogéneo en todos los casos respecto al caudal cursado.

En todos los escenarios de simulación (Figura 44, Figura 45 y Figura 46) el caudal cursado es el mismo para el caso de FSA-RDP con el filtro que con r_{opt}. Siendo así el funcionamiento de FSA-RDP con el filtro adaptativo muy superior al de FSA-RDP sin el mecanismo de probabilidad de acceso (respecto al caudal cursado).

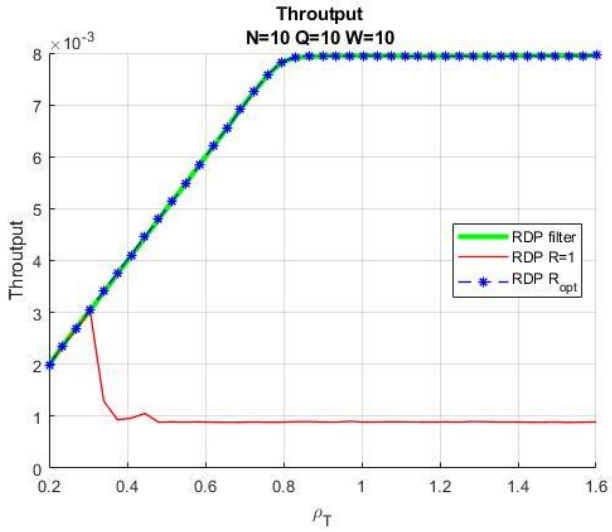

Figura 44. Caudal servido con 10 nodos.

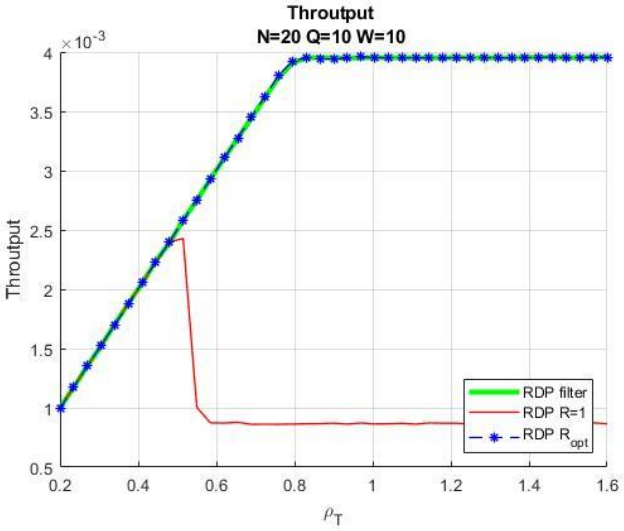

Figura 45. Caudal servido con 20 nodos.

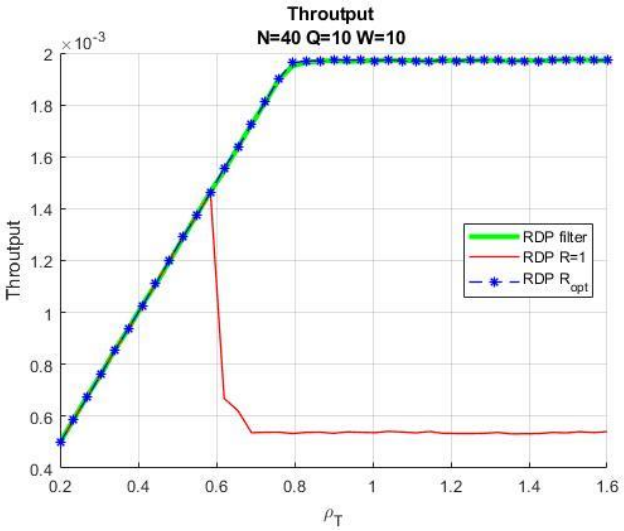

Figura 46. Caudal servido con 40 nodos.

8.1.4 Uso del canal.

El funcionamiento del protocolo añadiendo el filtro adaptativo es homogéneo en todos los casos respecto al uso de canal.

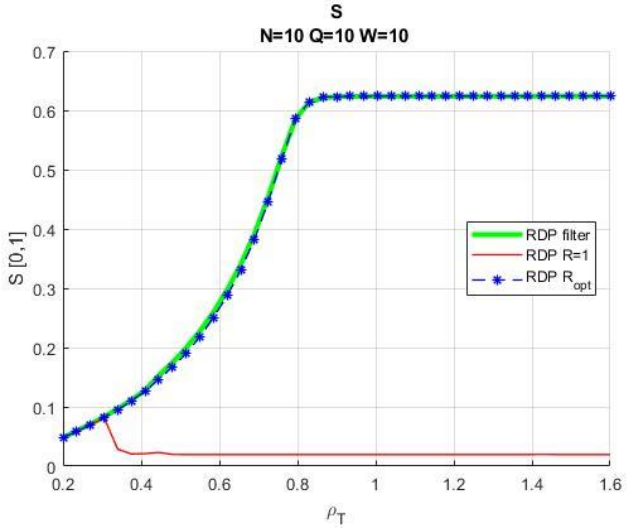

Figura 47. Uso de canal con 10 nodos.

Tanto en el caso de 10 nodos (Figura 47), 20 nodos (Figura 48) y 40 nodos (Figura 49) el uso del canal con el filtro es el mismo que cuando se configura FSA-RDP con r_{opt} en cargas bajas y altas. En cargas medias el uso del canal es superior cuando se configura FSA-RDP con el filtro adaptativo que cuando se configura con r_{opt}. Respecto a FSA-RDP sin el mecanismo de probabilidad de acceso, la mejora es total dado que con el filtro se obtiene el uso óptimo del canal.

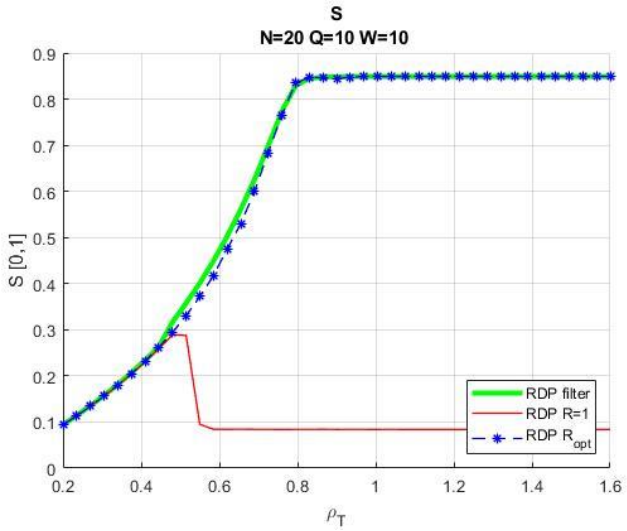

Figura 48. Uso de canal con 20 nodos.

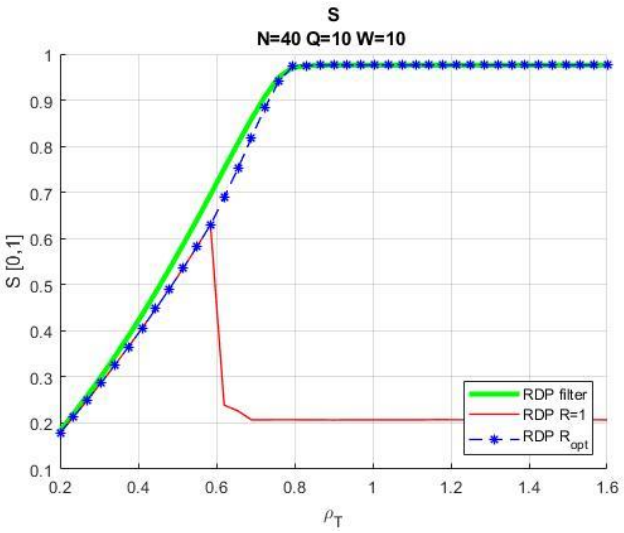

Figura 49. Uso de canal con 40 nodos.

8.2 Conclusión del funcionamiento general del filtro.

En general se trata de una mejora positiva al funcionamiento de FSA-RDP al automatizar un proceso necesario para que el protocolo funcione en las mejores condiciones.

Se trata de una implementación sencilla y que requiere poca memoria en el *gateway*. Solo necesita la memoria necesaria para los valores *double* (8 bytes) de los índices del filtro digital y del *buffer*. En nuestro caso de estudio son 40 unidades de memoria (20 para el filtro y 20 para el *buffer*), lo que daría un total 320 bytes de memoria para guardar los valores. En cuanto a las operaciones, para calcular la entrada del filtro se trata de un cálculo sencillo. La salida se trata de un filtro de activación con una función lineal de sencilla implementación. Por tanto, el nuevo mecanismo adaptativo propuesto no representaría un coste adicional significativo para el *gateway*, pudiendo implementarse totalmente por *software*.

Como se ha visto, el algoritmo trabaja generalmente bien en todos los casos, salvo en las poblaciones altas de sensores que tiende a generar un pequeño retardo para conseguir un uso mayor del canal. El mecanismo del filtro se basa en que el *gateway* solo conoce cuantos elementos en total conforman el *cluster,* el tamaño de la subrama de contienda y el estado de los *minislots* de contienda. Con estos valores busca maximizar el uso del canal reduciendo el número de colisiones en la subtrama de contienda. Al comportarse de esta manera, el filtro puede tender a reducir drásticamente el valor de *r* en casos donde hay un gran número de sensores intentando acceder a los *minislots* de contienda. Esta reducción de *r* supone que cuando un nodo se activa y calcula la probabilidad de acceso, si esta probabilidad es muy pequeña durante un número de tramas, tenderá a no acceder a la subtrama de contienda aumentado así el retardo de los paquetes que tenga en su cola. Además, el límite inferior de la función de salida del sistema de adaptación depende del tamaño total del *cluster*, por tanto, a mayor tamaño del *cluster* menor será el valor de *r* cuando haya muchas colisiones en la subtrama de contienda aumentando así más el retardo a la hora de intentar acceder a la subtrama de contienda, pero también se sucederán menos colisiones y se podrá por ello aprovechar mejor el canal.

Este funcionamiento adaptativo que busca maximizar el uso de canal podría explicar por qué se obtienen rendimientos superiores en este parámetro (Figura 48 y Figura 49) respecto a la configuración de FSA-RDP con r_{opt}, pues se adapta al estado puntual del sistema durante la simulación en vez de ser un valor fijo que simplemente maximice el caudal cursado.

Capítulo 9. Conclusiones y trabajos futuros.

9.1 Conclusiones

En este trabajo se ha realizado un análisis comparativo de las prestaciones de dos protocolos de acceso en redes de sensores inalámbricas. Este tipo de estudios es de gran interés debido al gran desarrollo que se observa del paradigma tecnológico denominado *Internet of Things* (IoT), bajo el paraguas de las redes de acceso celulares de quinta generación. Dentro del contexto tecnológico en el que se enmarca el trabajo realizado, se han presentado tanto de los desafíos de este nuevo paradigma dentro de las comunicaciones M2M, como de los avances y esfuerzos realizados por las entidades estandarizadoras para que pueda llevarse a cabo esta transformación. Además, se ha justificado que las redes inalámbricas de sensores y su organización es parte intrínseca de la transformación tecnológica que ya se está dando.

. Los protocolos de acceso estudiados están pensados para gestionar el acceso de poblaciones de sensores en este nuevo paradigma tecnológico. Tras su justificar su encaje dentro del conjunto de tecnologías necesarias para el desarrollo de IoT, se ha descrito su funcionamiento. Ambos están basados en *Frame Sloted Aloha* (FSA) y en la definición de una estructura de trama dividida en dos subtramas. Una subtrama de contienda en la que se envían pequeños paquetes y una de envío de datos libre de colisiones. La principal diferencia entre ellos es la gestión de las colisiones en la contienda:

- *Framed Slotted ALOHA with reservation data packets* (FSA-RDP) está basada en un modelo de permiso para el envío de los paquetes por parte del *gateway* del *cluster*.
- *Frame Slotted Aloha with Distributed Queueing* (FSA-DQ) hace uso de *tree-splitting* y de colas distribuidas para gestionar el envío de los paquetes.

Mediante simulación orientada a eventos se han extraído las prestaciones relativas a ambos protocolos, donde FSA-RDP ha obtenido unos resultados más positivos en todas las prestaciones estudiadas. En contra partida la variabilidad de las tramas de este protocolo añade complejidad a la hora de sincronizar los distintos dispositivos con el *gateway* una vez que se han activado. También añade trabajo extra al *gateway* dado que debe gestionar cuál de los sensores del *cluster* puede o no transmitir en la subtrama de datos.

En el trabajo también se ha propuesto y simulado un novedoso mecanismo de gestión de tráfico adaptativo para FSA-RDP. Éste permite la autoconfiguración dinámica de la probabilidad de acceso. Se ha justificado la propuesta deductivamente y tras su implementación en el modelo de simulación se han obtenido muy buenos resultados. En la mayoría de los casos el nuevo mecanismo consigue un funcionamiento óptimo del sistema, controlando de manera eficaz el acceso de los sensores.

9.2 Trabajos futuros.

Como trabajo futuro es pertinente trabajar e investigar en la sincronización de los sensores durante las tramas variables en FSA-RDP. Además de hacer un estudio cuantitativo en un campo de pruebas con *hardware* para obtener datos de campo que sustenten el funcionamiento de estos protocolos.

También queda abierto el estudio sobre el estado transitorio del mecanismo de adaptación de la probabilidad de acceso y su desarrollo en el tiempo. Así como la mejora del funcionamiento del mecanismo en escenarios con gran número de sensores conectados al *gateway*.

Capítulo 10. Bibliografía

[1] P. Kumar Verna, R. Verma, A. Prakash, A. Agrawal, K. Naik, P. Tripathi, M. Alsabaan, T. Khalifa, T. Abdelkader y A. Abogharaf, «Machine-to-Machine (M2M) communications: A survey,» *Journal of Network and Computer Applications,* nº 66, pp. 83-105, 2016.

[2] H. Shariatmadari, R. Ratasuk, S. Iraji, A. Laya, T. Taleb, R. Jäntti y A. Ghosh, «Machine-type communications: current status and future perspectives,» *IEEE Communications Magazine,* vol. 53, nº 9, pp. 10-17, 2015.

[3] «Wikipedia,» Fundación Wikipedia, Inc., 23 Marzo 2021. [En línea]. Available: https://es.wikipedia.org/wiki/IPv6#cite_note-5. [Último acceso: 25 Abril 2021].

[4] A. Colakovic y M. Hadžialic, «Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues,» *Computer Nerworks,* nº 144, pp. 17-39, 2018.

[5] M. Ramgir, Mayur Ramgir, Pearson Education India, 2019.

[6] J. Yick, B. Mukherjee y D. Ghosal, «Wireless sensor network survey,» *Computer Networks,* vol. 52, nº 12, pp. 2292-2330, 2008.

[7] L. Xu, R. Collier y G. M. P. O'Hare, «A Survey of Clustering Techniques in WSNs and,» *IEEE INTERNET OF THINGS JOURNAL,* vol. 4, nº 5, pp. 1229-1249, 2017.

[8] M. Miller, «eeNews,» European Business Press SA, 10 November 2017. [En línea]. Available: https://www.eenewseurope.com/design-center/evolution-industrial-wireless-sensor-networks/page/0/3. [Último acceso: 13 Abril 2021].

[9] K. Römer y F. Mattern, «The desing space of wireless sensor,» *IEEE Wireless Communications,* vol. 11, nº 6, pp. 54-61, Dec. 2004.

[10] M. Mehdi Afsar y M.-H. Tayarani-N, «Clustering in sensor networks: A literature survey,» *Journal ofNetworkandComputerApplications,* nº 46, pp. 198-226, 2014.

[11] K. Maraiya, K. Kant y N. Gupta, «Efficient Cluster Head Selection Scheme for Data Aggregation In Wireless Sensor Network.,» *International Journal of Computer Applications,* vol. 23, nº 9, pp. 10-18, 2011.

[12] M. Agiwal, A. Roy y N. Saxena, «Next Generation 5G Wireless Networks: A Comprehensive Survey,» *IEEE Communications Surveys & Tutorials,* vol. 18, nº 3, pp. 1617-1655, 2016.

[13] c. d. Wikipedia, «Frecuencia extremadamente alta,» Wikipedia Inc., 16 julio 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Frecuencia_extremadamente_alta. [Último acceso: 16 Abril 2021].

[14] «Internet of Things in the 5G era: Enablers, Architecture, and Business Models,» *IEEE Journal on Selected Areas in Communications,* vol. 34, nº 3, pp. 510-527, 2016.

[15] H. Shariatmadari, R. Ratasuk, S. Iraji, A. Laya, T. Taleb, R. Jänti y A. Ghosh, «Machine-Type Communications: Current Status and Future. Perspectives Toward 5G Systems,» *IEEE Comminications Magazine - Communications Standards Magazines,* pp. 10-17, 2015.

[16] T. Salam, W. Ur Rehman y X. Tao, «Data Aggregation in Massive Machine Type Communication: Challenges and Solutions,» *IEEE Access*, vol. 7, pp. 41921-41946, 2019.

[17] P. Huang, L. Xiao, S. Soltani, M. W. Mutka y X. Ning, «The Evolution of MAC Protocols in Wireless Sensor NEtworks: A Survey,» *IEEE Communications Surveys & Tutorials*, vol. 15, nº 1, pp. 101-120, 2013.

[18] c. d. Wikipedia, «ALOHAnet,» Wikipedia, La enciclopedia libre., 5 noviembre 2020. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=ALOHAnet&oldid=130663075. [Último acceso: 24 mayo 2021].

[19] c. d. Wikipedia, «Carrier sense multiple access,» Wikipedia, La enciclopedia libre, 28 noviembre 2019. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Carrier_sense_multiple_access&oldid=121646947. [Último acceso: 24 mayo 2021].

[20] W. contributors, «Carrier-sense multiple access with collision detection,» Wikipedia, The Free Encyclopedia., 6 enero 2021. [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Carrier-sense_multiple_access_with_collision_detection&oldid=998624019. [Último acceso: 24 mayo 2021].

[21] c. d. Wikipedia, «Carrier sense multiple access with collision avoidance,» Wikipedia, La enciclopedia libre., 29 noviembre 2020. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Carrier_sense_multiple_access_with_collision_avoidance&oldid=131309479. [Último acceso: 24 mayo 2021].

[22] V. Casares-Giner, J. Martinez-Bauset y C. Portillo, «Performance evaluation of framed slotted ALOHA with reservation packets and succesive interference cancelation for M2M networks,» *Computer Networks*, nº 155, pp. 15-30, 2019.

[23] A. Laya, C. Kalalas, F. Vazquez-Gallego, L. Alonso y J. Alonso-Zarate, «Goodbye, ALOHA!,» *IEEE*, vol. 4, pp. 2029-2044, 2016.

[24] M. H. MacDougall, Simulating Computer Systems Techniques and Tools, Massachusetts: Massachusetts Institute of Technology, 1987.

[25] G. Pensa, «Simulación por eventos discretos y cuándo simular,» Atlas consultora, 17 Agosto 2020. [En línea]. Available: http://www.atlasconsultora.com/simulacion-por-eventos-discretos-y-cuando-simular/#que-es-la-simulacion-por-eventos-discretos-des. [Último acceso: 6 Mayo 2021].

[26] I. Leyva Mayorga, M. A. Rodriguez Hernandez, V. Pla, J. Martinez Bauset y L. Tello Oquendo, «Adaptive access class barring for efficient mMTC,» *Computer Networks*, nº 149, pp. 252-264, 2019.

[27] R. W. Hamming, Digital filters, Mineola, New York: Dover Publications, INC, 1989.

[28] Ó. F. Corredor Camargo, L. F. Pedraza Martínez y C. A. Hernández Suárez, «Diseño e implementación de filtros digitales,» *Vis. Electrón*, vol. 3, nº 1, pp. 55-66, jun. 2009.

[29] W. contributors, «Finite impulse response,» Wikipedia, The Free Encyclopedia., 1 Julio 2021. [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Finite_impulse_response&oldid=1031458753. [Último acceso: 23 Agosto 2021].

[30] S. J. Elliott y P. A. Nelson, «Active noise control,» *IEEE Signal Processing Magazine,* vol. 10, nº 4, pp. 12-35, Oct. 1993.

[31] J. C. Sombría, «Señales aleatorias-Ingenieria Electrónica de Comunicaciones,» Universidad Complutense de Madrid, 2020. [En línea]. Available: https://www.cartagena99.com/recursos/alumnos/apuntes/210518194738-02%20-%20Senales%20Aleatorias.pdf. [Último acceso: 02 08 2021].

[32] c. d. Wikipedia, «Distribución de Poisson,» Wikipedia, La enciclopedia libre., 17 mayo 2021. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Distribuci%C3%B3n_de_Poisson&oldid=135602831. [Último acceso: 2 junio 2021].

[33] J. Chacón Sombría, «Señales Aleatorias-Ingeniería Enctrónica de Comunicaciones,» Universidad Complutense de Madrid, 2020. [En línea]. Available: https://www.cartagena99.com/recursos/alumnos/apuntes/210518194738-02%20-%20Senales%20Aleatorias.pdf. [Último acceso: 02 08 2021].