XIV Jornadas de Ingeniería Telemática.
JITEL 2019.
Universidad de Zaragoza.

Actas de las XIV Jornadas
de Ingeniería Telemática
(JITEL 2019),
Zaragoza (España),
22-24 de octubre de 2019.

ISBN: X-XXXX-XX-XXX-X
DOI: XXXXXX

Combinación de Network Coding Sistemático y Solapamiento en escenarios IoT.

Mihail Zverev[1], Pablo Garrido[1], Ramón Agüero[2], Josu Bilbao[1].
[1]Information and Communication Technologies Area, Ikerlan Technology Research Center,
Paseo José María Arizmendiarrieta, 2, 20500 Mondragón, Gipuzkoa, España.
[2]Dpto. Ingenería Comunicaciones, Universidad de Cantabria,
Avenida Los Castros, s/n, 39005, Santander, Cantabria, España.
[1]{mzverev, pgarrido, jbilbao}@ikerlan.es, [2]ramon@tlmat.unican.es

***Resumen*—Las aplicaciones de las arquitecturas IoT son cada vez más numerosas, y abarcan desde dispositivos *wearables* hasta comunicaciones en entornos vehiculares. Con la consolidación de la Industria 4.0, los entornos IIoT (*Industrial IoT*) son cada vez más comunes. Los enlaces inalámbricos sobre los que se sustentan son generalmente susceptibles a sufrir pérdidas en las transmisiones, y su recuperación puede incrementar notablemente el *retardo*. La fiabilidad (robustez), así como el retardo, son requisitos fundamentales en las comunicaciones IIoT. Una posible estrategia para mejorar el comportamiento de las comunicaciones es la utilización de técnicas *Network Coding* (codificación de red). Su utilización para la mejora del comportamiento de comunicaciones en entornos IIoT no ha recibido hasta la fecha suficiente atención, y no existe ninguna implementación que cuente con una amplia aceptación. En este trabajo se analiza una alternativa interesante para comunicaciones IIoT, basada en la combinación de un esquema *Network Coding* sistemático y una estrategia de solapamiento. Se lleva a cabo un análisis exhaustivo del comportamiento de esta solución, extendiendo notablemente los estudios previos, y se obtienen una serie de conclusiones prácticas de cara a su implementación.**

***Palabras Clave*—TF, CCRS, RD, Network Coding, codificación sistemática, IIoT**

I. INTRODUCCIÓN

Las casas inteligentes, los dispositivos *wearables* y las comunicaciones vehiculares son sólo algunos de los ejemplos de las crecientes aplicaciones IoT. A medida que las fábricas están evolucionando hacia la Industria 4.0, el entorno IIoT (*Industrial IoT*) es cada vez más común. Un despliegue IIoT tiene típicamente múltiples dispositivos interconectados a través de una red, que puede estar caracterizada por sufrir pérdidas de información, siendo la fiabilidad y el bajo retardo requisitos estrictos.

Es bien sabido que los enlaces inalámbricas son propensos a inducir errores en la comunicación. El reenvío de la información perdida (para ser recuperada) supone un aumento de la carga de la red y aumenta el retardo, no sólo en la transmisión que genera el reenvío, sino en todas las comunicaciones de la red.

Existen múltiples técnicas que permiten aumentar la robustez de la comunicación y reducir su retardo, previniendo el reenvío de la información perdida. Dichas técnicas suelen consistir en la codificación del mensaje de tal manera que el receptor pueda recuperar la información original a pesar de las pérdidas. Hasta la fecha la codificación preventiva en IoT no ha recibido suficiente atención. El primer paso en la búsqueda de una implementación de las técnicas de codificación preventiva optimizada para IoT debería ser un estudio de los esquemas de codificación existentes. Este trabajo se centra precisamente en un esquema de codificación que permite responder a estas dos necesidades de las comunicaciones inalámbricas en entornos IIoT.

El resto del trabajo se estructura tal y como sigue. La Sección II proporciona un breve resumen de diferentes esquemas de codificación que se han propuesto en la literatura relacionada, y se identifican aspectos no cubiertos hasta la fecha, que son los que se estudian en este trabajo. En la Sección III se describe la implementación del esquema de codificación que se propone. La Sección IV resume los resultados obtenidos tras las simulaciones realizadas, mientras que en la Sección V, se concluye el trabajo, identificado los aspectos que aparecen como líneas de investigación futuras.

II. ESTADO DEL ARTE

Una técnica que tradicionalmente se ha usado para incrementar la fiabilidad de las comunicaciones es FEC (*Forward Error Correction*). Con ella, los fragmentos de información, o *símbolos*, que envía un transmisor son protegidos por otros símbolos -de reparación, redundantes o codificados- que se envían junto con los originales. Así, las pérdidas se pueden recuperar en el receptor con los símbolos redundantes. Otra forma de aumentar la robustez de la comunicación es el uso de NC (*Network Coding*, codificación de red). Una de las configuraciones

más empleadas en soluciones NC es la conocida como *Intra-flow* (definido en [1]), que comparte varias de las características del FEC, ya que con esta técnica también se envían símbolos codificados para proteger los originales, sólo que en este caso los símbolos codificados podrían ser generados, además de por la fuente. por los nodos intermedios (routers). Así, NC podría generar los símbolos codificados necesarios para cada enlace entre routers consecutivos, mientras que FEC lo hace para toda la comunicación, extremo a extremo. El uso de NC en lugar de FEC supone una reducción de sobrecarga de la red, especialmente apreciable en las redes inalámbricas malladas.

Una técnica NC que ha suscitado mucho interés es RLNC (*Random Linear NC*), introducida en [2] y posteriormente extendida en [3]. Los símbolos originales o *fuente* se agrupan en bloques conocidos como *generaciones*, para posteriormente ser combinados linealmente con el uso de coeficientes aleatorios en un Campo de Galois $GF(2^q)$. Cada combinación lineal de los símbolos fuente se conoce como *símbolo codificado*, y se generan tantos como sean necesarios para cubrir las pérdidas en el próximo enlace. Al recibir los nodos intermedios los símbolos codificados suficientes para recuperar la información original, pueden volver a hacer una combinación lineal con coeficientes aleatorios para generar los símbolos necesarios en el próximo enlace, incluso sin tener que decodificarlos. Es decir, los nodos pueden *recodificar sin decodificar*. El receptor podrá recuperar el mensaje original siempre y cuando reciba suficientes símbolos codificados, independientemente del número de recodificaciones por las que ha pasado el mensaje. Esta técnica de codificación también puede aplicarse a FEC (codificación en los extremos de comunicación, sin recodificación), siendo este esquema conocido como RLC (*Random Linear Coding*).

RLNC introduce una complejidad computacional que sería interesante evitar en el ámbito IoT, por las limitaciones de los dispositivos. Una manera de reducir dicha complejidad es bajar la densidad de la codificación, dando lugar a técnicas SNC (*Sparse NC*). Para codificar se usan sólo *algunos* símbolos fuente en lugar de todos, lo que implica que un símbolo codificado sólo incluye una parte de toda la generación [4]. Una de las técnicas SNC es el NC sistemático (SysNC) [5] que, a diferencia del RLNC clásico, envía los símbolos fuente sin codificar junto con los símbolos codificados necesarios. Gracias a ello, gran parte de la información puede usarse nada más ser recibida, ya que no es necesario decodificarla.

Se dice que el RLNC clásico es un esquema de codificación por bloques, puesto que divide la información en unos bloques (generaciones) que no se solapan. El *solapamiento* tiene sus ventajas, fáciles de ver con SysNC: al solapar las generaciones, se reduce el periodo existente entre símbolos codificados, pudiéndose recuperar los símbolos perdidos con mayor frecuencia. Al solapar las generaciones se puede definir además una ventana deslizante de codificación, que cubra los símbolos que pertenecen a la generación más nueva. Un ejemplo de SysNC con una ventana de codificación y decodificación deslizante es el protocolo *Caterpillar RLNC* (CRLNC), presentado en [6] y extendido con ARQ (*Automatic Repeat reQuest*) en [7]. CRLNC ofrece una probabilidad de decodificación muy cercana a SysNC por bloques, pero con un retardo significativamente menor.

Un enfoque semejante al solapamiento es el *intercalado*. Un bloque de generaciones (*bloque de intercalado*) se envía símbolo a símbolo con multiplexación temporal. Esto es, en primer lugar se envía el primer símbolo de cada generación del bloque, luego el segundo, y así sucesivamente hasta enviar el bloque de intercalado completamente, incluyendo los símbolos codificados. La ventaja del intercalado es su robustez a pérdidas de símbolos a ráfagas. La multiplexación temporal introduce un retardo que se puede evitar según se plantea en [8], utilizando la técnica denominada *Interleaving with On-the-fly Coding* (IOC). En esta técnica los símbolos fuente son asignados a cada generación del bloque al estilo Round-Robin, lo que permite enviarlos en su orden natural. Su principal desventaja es que puede requerir una memoria elevada, tanto para la codificación como para la decodicación.

El estudio de la viabilidad de IOC, o la de su combinación con solapamiento, en entornos IoT no se ha evaluado hasta la fecha. A pesar de que puede resultar eficiente, podría requerir una complejidad computacional elevada.

Teniendo en cuenta todo lo anterior, se puede decir que el esquema que mejor parece adecuarse al uso en IoT es SysNC con solapamiento. En [6], el protocolo CRLNC utiliza sólo un símbolo codificado por generación. Por su parte, los autores del trabajo original presentan resultados que muestran que para generaciones mayores la probabilidad de pérdida de símbolos se reduce, y que el retardo extremo a extremo era menor para las generaciones más pequeñas. Sin embargo, no se analiza el posible impacto sobre la eficiencia causado por la introducción de más de un símbolo redundante por generación. En este trabajo se diseña e implementa una solución SysNC con solapamiento, para ejecutar una campaña de simulaciones, a fin de analizar de manera detallada su comportamiento.

III. IMPLEMENTACIÓN DE NC SISTEMÁTICO

Dependiendo de la implementación concreta de los algoritmos de NC, se pueden incluir múltiples símbolos en un mismo paquete codificado. Sin embargo, la pérdida de un único paquete puede tener como consecuencia la pérdida de una ráfaga de símbolos. A primera vista, podría parecer por tanto que enviar varios símbolos en un paquete no es apropiado. En este estudio se consideran las transmisiones de un símbolo por paquete. Así, desde este momento se utilizarán indistintamente los términos *paquete* y *símbolo*.

A. Esquema de codificación

Se ha llevado a cabo una implementación completa del esquema SysNC con solapamiento en Python. Para ello, se hizo uso del esquema RLNC para generar los símbolos

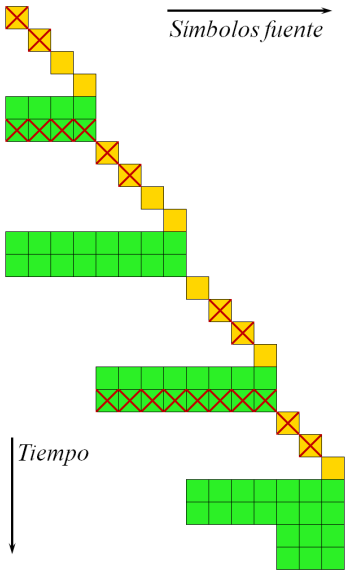


Fig. 1. Ejemplo de la implementación del esquema de codificación. Los cuadrados amarillos son los símbolos fuente, los cuadrados verdes, los símbolos codificados. Las aspas rojas marcan los símbolos perdidos.

redundantes, para lo que se combinan aleatoriamente paquetes de información, utilizando coeficientes aleatorios del campo de Galois $GF(2^8)$.

Se define una *generación* como un conjunto de símbolos originales que, al igual que en [6], se considera constante. Cada generación se protege por r paquetes codificados, y cada paquete se construye combinando símbolos de información que pertenecen a φ generaciones, lo que se define como *solapamiento*. Por otro lado, se considera un *bloque* como el conjunto de símbolos de información que se utilizan para construir un paquete codificado, es decir, compuesto por el solapamiento de φ generaciones. Los tamaños de generación y de bloque se designan con g y k, respectivamente.

Con esta definición se puede ver el esquema RLNC por bloques como un caso particular de la versión con solapamiento con $\varphi = 1$. Esta definición implica la creación de generaciones parciales adicionales para proteger los últimos símbolos fuente de la comunicación. A modo ilustrativo se presenta la Figura 1, donde los cuadrados amarillos son los símbolos fuente y los cuadrados verdes, los símbolos codificados. En este ejemplo de tan sólo 15 símbolos fuente y $r = \varphi = 2$, se puede ver claramente que sin los últimos $r = 2$ símbolos codificados, el último bloque no cumpliría con la definición del solapamiento, perteneciendo únicamente a $1 < \varphi = 2$ generación.

En el ejemplo presentado en la Figura 1, se considera una comunicación con pérdida de paquetes. Los símbolos perdidos se marcan con un aspa roja. En este ejemplo es imposible recuperar los símbolos fuente de los primeros 3 bloques hasta recibir el último símbolo codificado. Este ejemplo puede expandirse a una cantidad de símbolos fuente mucho mayor con el mismo problema: los primeros símbolos perdidos pueden todavía recuperarse, pero al final de la comunicación. Dependiendo del protocolo que haga uso de este esquema de codificación, gran parte

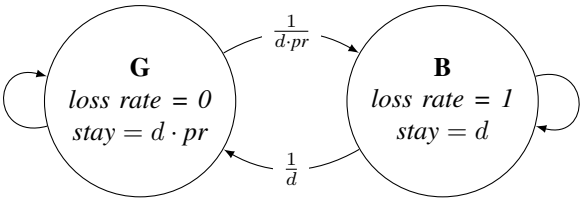


Fig. 2. Implementación del modelo Gillbert-Elliot.

de los símbolos perdidos seguramente acabarán siendo reenviados por el transmisor mucho antes de que puedan ser recuperados. Aun así, mantener todos los símbolos recibidos en memoria permitirá en algún momento recuperar los símbolos perdidos. En otras palabras, *la probabilidad de recuperación depende de la cantidad de memoria disponible en el receptor*. De aquí se llega a la conclusión que la ventana de decodificación debería ser lo más grande posible. Dado que en este estudio se pretende explorar las oportunidades que ofrece el esquema SysNC, en el receptor se define suficiente memoria para almacenar todos los paquetes fuente y codificados.

Cada vez que se recibe un símbolo de información se lanza el proceso de recuperación. Dado que el canal (descrito en la sección B) no reordena los símbolos, esta política de recuperación equivale a intentar recuperar los símbolos perdidos después de cada símbolo recibido.

B. Canal

Se considera un canal inalámbrico entre el transmisor y el receptor, en el que se pueden producir pérdidas. Además, no se reordena los paquetes que viajan por el enlace.

En trabajos previos es bastante común usar una distribución uniforme para modelar la pérdida de los símbolos (canal sin memoria), como en [5]. Sin embargo, en las redes reales los errores a ráfagas pueden ser más comunes. Este tipo de comportamiento se puede capturar con el modelo de Gillbert-Elliot, como en [6] y [8]. Con el fin de utilizar las mismas herramientas de estudios previos, en este se contemplan los dos tipos de comportamientos. A continuación se detalla la implementación que se ha llevado a cabo para el canal a ráfagas.

En el estado de la ráfaga (B, del inglés *burst* o *bad*) la tasa de pérdida es del 100%. La tasa de pérdida en el estado "tranquilo" (G, del inglés *good*) es 0%. Cada ráfaga en media tiene una duración de d paquetes, y el estado *'bueno'* tiene una duración media pr veces mayor que la de la ráfaga, $pr \cdot d$. La figura 2 muestra un esquema del modelo que se ha utilizado. En el presente trabajo se considera la proporción entre las duraciones de los estados de $pr = 10$.

C. Parámetros del modelo

Algunos de los parámetros de entrada del modelo de comunicación del presente estudio ya han sido mencionados: tamaño de generación, símbolos redundantes por generación, solapamiento, tasa de pérdidas, duración de la ráfaga de errores, y número de paquetes a almacenar en el receptor. En este punto únicamente queda un

parámetro por definir: la cantidad de símbolos a transmitir. Un planteamiento que se puede tomar en este tipo de simulaciones es enviar los símbolos fuente con los codificados hasta que el receptor consiga recibir correctamente n símbolos fuente. En el presente trabajo se sigue otro planteamiento: se envían exactamente n símbolos fuente con los correspondientes símbolos codificados al receptor.

Para ver si el es esquema de codificación funciona, es necesario conocer si el receptor ha sido capaz de recibir correctamente todos los paquetes fuente que le fueron enviados, y en caso contrario, cuántos ha recibido. Por tanto, el principal parámetro que se utilizará para evaluar el comportamiento de la solución propuesta será el número de símbolos fuente recibidos, ya sea directamente, o a través de los algoritmos de recuperación. Otra medida interesante podría ser la relación entre los símbolos recuperados y los perdidos. En este trabajo se evalúa la probabilidad de recibir (recepción normal + recuperación) todos los símbolos fuente. Dicha probabilidad se obtiene dividiendo el número total de eventos en los que se reciben o recuperan todos los paquetes fuente entre el número de simulaciones.

Para evaluar el retardo se asumirá un canal ranurado. Para una transmisión de datos correcta y fiable, es de gran importancia la entrega a la capa de aplicación de los paquetes en orden, por lo que se estudiará el retardo extremo a extremo. Se ignora además el retardo de los paquetes no recuperados, teniéndose en cuenta únicamente los casos en los que la probabilidad de recibir (o recuperar) todos los símbolos fuente sea 1.

Como se ha mencionado anteriormente, se pretende evaluar el impacto de variar las redundancias por generación en el rendimiento de SysNC con solapamiento. Por lo tanto, los dos parámetros de entrada más importantes son el solapamiento y las redundancias por generación. Con el fin de simplificar el análisis de los resultados, las dos variables se combinan en una, la *sobrecarga*, que se define como la relación entre el número total de paquetes codificados generados, y el número total de paquetes (fuente y codificados) que envía el transmisor. Como se ha podido observar en el ejemplo de la Figura 1, al final de la comunicación se enviarán $(\varphi - 1) \cdot r$ símbolos codificados adicionales. El número total de paquetes codificados, C, que protegen S paquetes fuente se puede calcular como indica la ecuación 1.

$$C = \left(\left\lceil \frac{S}{k} \right\rceil + \varphi - 1 \right) \cdot r = \left(\left\lceil \frac{S}{\lfloor g/\varphi \rfloor} \right\rceil + \varphi - 1 \right) \cdot r \quad (1)$$

A partir de C (ec. 1) se puede calcular la sobrecarga $\hat{O}$, como se muestra en la ecuación 2.

$$\hat{O} = \frac{C}{C+S} = \frac{\left(\left\lceil \frac{S}{\lfloor g/\varphi \rfloor} \right\rceil + \varphi - 1 \right) \cdot r}{\left(\left\lceil \frac{S}{\lfloor g/\varphi \rfloor} \right\rceil + \varphi - 1 \right) \cdot r + S} \quad (2)$$

IV. RESULTADOS

Para ver el efecto de la redundancia por cada generación, con diferentes grados de solapamiento, se llevó a cabo una campaña de simulaciones con los siguientes parámetros de entrada: generaciones de $g = 64, 256$ símbolos; solapamientos de $\varphi = 1, 2, 4, 8$ generaciones; 1000 paquetes fuente; tasas de pérdidas de 1%, 5% y 10% con distribución de pérdidas uniforme y una tasa de 9% con ráfagas de pérdidas de 5 paquetes en media y 50 paquetes entre ráfagas. La memoria del receptor ilimitada, pudiéndose almacenar todos los paquetes recibidos y recuperados. En total se ejecutaron más de 1000 simulaciones independientes por cada configuración, para asegurar la validez estadística de los resultados.

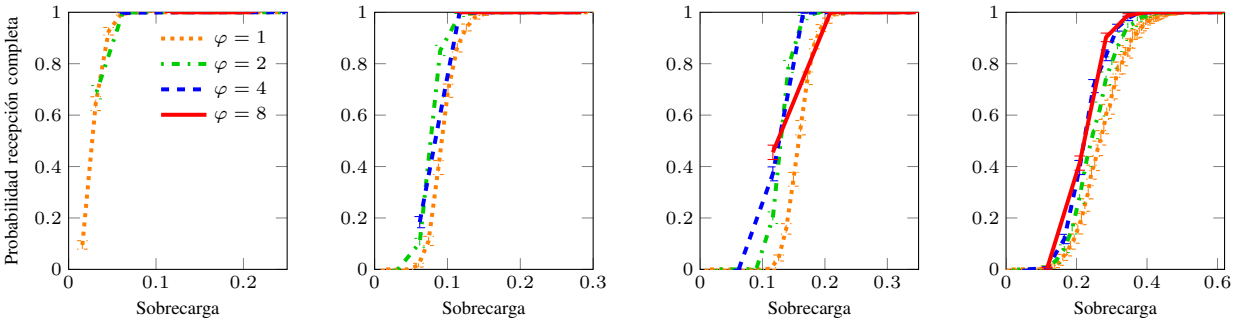
En las Figuras 3 y 4 se representan la probabilidad de recepción para diferentes solapamientos (fila de arriba) y las correspondientes latencias extremo a extremo (fila de abajo).

Dada la definición de la sobrecarga en este trabajo (ecuación 2), sus valores serán más o menos distanciados, dependiendo de φ y r.

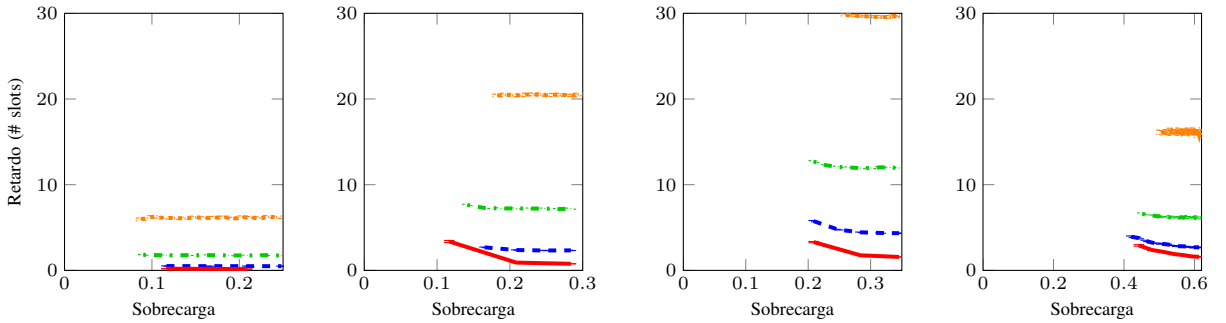
Como se puede ver en la figura 3-c, el solapamiento de 8 generaciones aparentemente no siempre ofrece la mejor probabilidad de recepción. En realidad, entre la primera y la segunda muestra hay mucha distancia, en la que los otros solapamientos tienen más muestras definidas. Para una comparación justa es necesario fijarse en los puntos de la sobrecarga para los que las muestras de las curvas de interés están definidas. Es importante tener en cuenta que *a mayor solapamiento, mayores son las sobrecargas, al aumentar las redundancias por generación*.

En las figuras 3 y 4 se puede observar que al aumentar el solapamiento, crece la probabilidad de recepción y baja el retardo. Como se puede ver, la principal ventaja de SysNC con solapamiento es la reducción de la latencia que se consigue que, según los resultados obtenidos, se puede decir que tiende a ser proporcional a φ, lo cual es de esperar, ya que los bloques (según se definen en la sección III.A) son precisamente φ veces menores con respecto a SysNC sin solapamiento.

Un detalle interesante es la sobrecarga en la que la probabilidad de recepción alcanza el valor 1. Dicha sobrecarga se puede denominar *sobrecarga de saturación*. Este parámetro resulta de especial interés a la hora de implementar NC en una red con un ancho de banda restringido. Dado que el retardo se han representado sólo para aquellos casos en los que se recibe toda la información, la sobrecarga de saturación queda marcada por la primera muestra de cada curva de latencia. En la figura 5 se representan los valores de las sobrecargas de saturación para cada caso estudiado. Como se puede ver, a mayor solapamiento, no siempre se consigue una probabilidad de saturación más baja. Esta depende tanto de la capacidad del esquema de codificación de recuperar los paquetes perdidos, como de la propia sobrecarga que introduce, por lo que no se puede establecer una conclusión clara respecto a este parámetro.

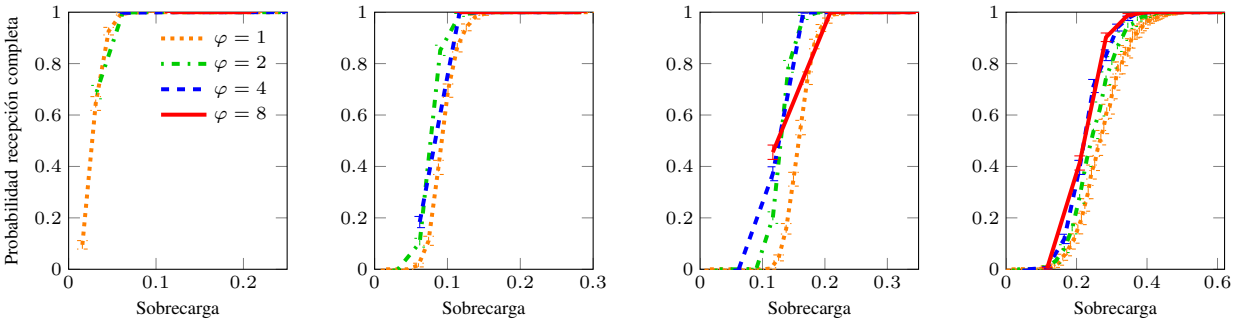


(a) 1% pérdidas uniformes. (b) 5% pérdidas uniformes. (c) 10% pérdidas uniformes. (d) 10% pérdidas a ráfagas.

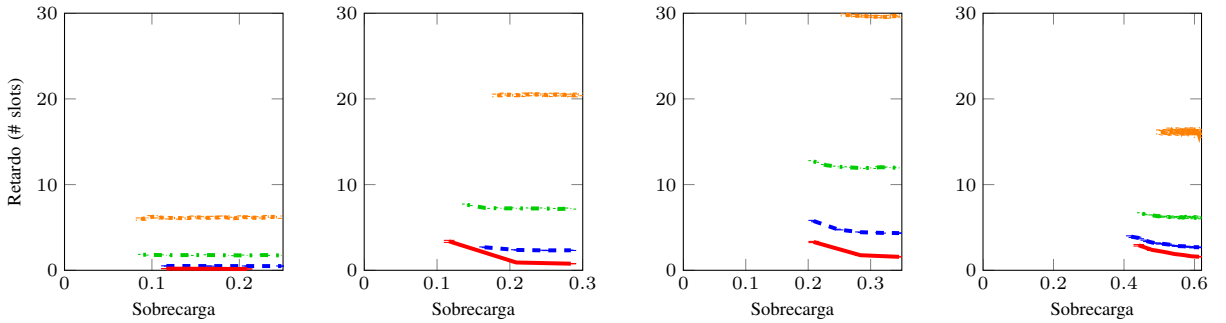


(e) 1% pérdidas uniformes. (f) 5% pérdidas uniformes. (g) 10% pérdidas uniformes. (h) 10% pérdidas a ráfagas.

Fig. 3. Probabilidad de recibir y recuperar todos los paquetes enviados (a–d) y los retardos extremo a extremo (e–h) para un tamaño de generación de 64 símbolos. Las gráficas corresponden a pérdidas uniformes de 1% (a, e), 5% (b, f) y 10% (c, g), y pérdidas en ráfagas (d, h).

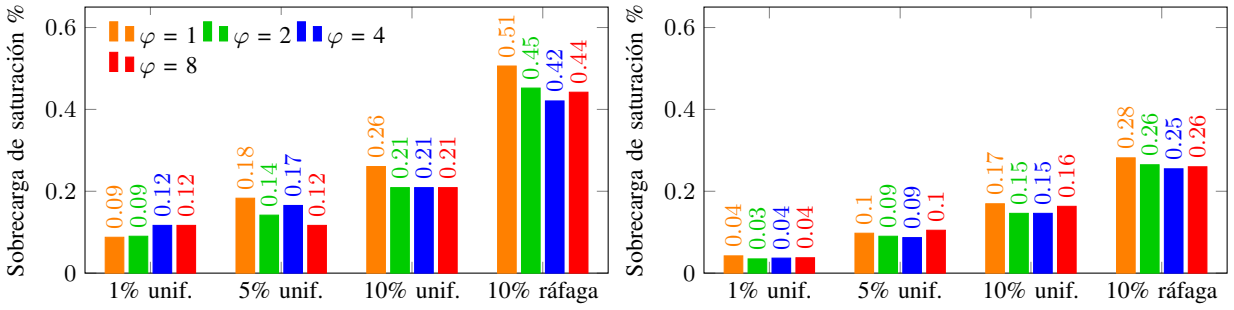


(a) 1% pérdidas uniformes. (b) 5% pérdidas uniformes. (c) 10% pérdidas uniformes. (d) 10% pérdidas a ráfagas.



(e) 1% pérdidas uniformes. (f) 5% pérdidas uniformes. (g) 10% pérdidas uniformes. (h) 10% pérdidas a ráfagas.

Fig. 4. Probabilidad de recibir y recuperar todos los paquetes enviados (a–d) y los retardos extremo a extremo (e–h) para un tamaño de generación de 256 símbolos. Las gráficas corresponden a pérdidas uniformes de 1% (a, e), 5% (b, f) y 10% (c, g), y pérdidas en ráfagas (d, h).



(a) g = 64 (b) g = 256

Fig. 5. Valores de sobrecargas de saturación para los diferentes casos de estudio contemplados en este trabajo.

V. CONCLUSIONES

En este trabajo se han revisado las técnicas de codificación de redes con especial interés en identificar aquellas que son adecuadas para las comunicaciones IoT. Una de las que más interés ha suscitado es el RLNC sistemático con solapamiento, presentada y analizada en [6]. En dicho estudio se utiliza solamente 1 símbolo codificado (redundante) por cada generación de paquetes, sin que quede claro el impacto en la eficiencia del esquema de codificación al emplear varios símbolos codificados. Para evaluar este aspecto y llevar a cabo una evaluación más exhaustiva de esta técnica se ha implementado el esquema descrito en [6], y se ha extendido su operación inicial, dando la posibilidad de modificar el número de redundancias por generación. Se ha utilizado dicha implementación para llevar a cabo una extensa campaña de simulación.

Los resultados muestran que en los puntos en los que la sobrecarga de la comunicación generada por los símbolos codificados es comparable, los esquemas con mayores grados de solapamiento presentan una mayor probabilidad de recepción de todos los símbolos generados por el transmisor (mediante recepción directa o a través de los paquetes codificados). También se ha comprobado que a mayor grado de solapamiento, menor es la latencia. La conclusión de estas observaciones es que dentro de una sobrecarga determinada, *es más eficiente aumentar el solapamiento en lugar de los símbolos redundantes por generación*.

De cara a las posibles implementaciones de este esquema en el ámbito IoT, cabe destacar que si, por motivos de limitaciones de los dispositivos o requisitos de implementación, no es posible aumentar el solapamiento, es importante tener en cuenta que cuanto mayor sea el grado del solapamiento a utilizar, mayor será la sobrecarga al aumentar las redundancias por generación.

A la hora de elegir el solapamiento y las redundancias por generación, es importante identificar la sobrecarga de saturación más baja posible. En otras palabras, encontrar aquella configuración que permita asegurar la recuperación de la mayoría de los paquetes perdidos con la menor sobrecarga posible. Como se ha visto, al aumentar el solapamiento, la sobrecarga de saturación puede aumentar en lugar de bajar. Al optimizar la sobrecarga de saturación, es posible que la latencia no sea la óptima. Por tanto, es fundamental priorizar entre la eficiencia del canal (*throughput*) y el retardo.

En el futuro se implementará el esquema sobre dispositivos reales, y se analizará el impacto de las limitaciones de éstos en la configuración del esquema de codificación. También se estudiará el efecto de considerar redes con más de un salto inalámbrico y cuál es la influencia de no disponer de una capacidad (ancho de banda) ilimitada.

AGRADECIMIENTOS

Los autoress agradecen la financiación del Programa de Doctorados Industriales de la Universidad de Cantabria (Convocatoria 2018). Los autores agradecen asimismo la financiación por parte del Gobierno de País Vasco (programa Elkartek) a través del proyecto DIGITAL (KK-2019/00095) y la financiación por parte del Gobierno de España (Ministerio de Economía y Competitividad, Fondo Europeo de Desarrollo Regional, FEDER) a través del proyecto ADVICE(TEC2015-71329-C2-1-R).

REFERENCIAS

[1] Dina Katabi, Sachin Katti, and Hariharan Rahul. *Chapter 2 - Harnessing Network Coding in Wireless Systems*, pages 39–60. Academic Press, Boston, 2012.

[2] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros. The Benefits of Coding over Routing in a Randomized Setting. In *IEEE International Symposium on Information Theory, 2003. Proceedings.*, page 442, 2003.

[3] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A Random Linear Network Coding Approach to Multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.

[4] M. Wang and B. Li. How Practical is Network Coding? In *2006 14th IEEE International Workshop on Quality of Service*, pages 274–278, 2006.

[5] J. Heide, M. V. Pedersen, F. H. P. Fitzek, and T. Larsen. Network Coding for Mobile Devices - Systematic Binary Random Rateless Codes. In *2009 IEEE International Conference on Communications Workshops*, pages 1–6, 2009.

[6] S. Wunderlich, F. Gabriel, S. Pandi, F. H. P. Fitzek, and M. Reisslein. Caterpillar rlnc (crlnc): A Practical Finite Sliding Window RLNC approach. *IEEE Access*, 5:20183–20197, 2017.

[7] F. Gabriel, S. Wunderlich, S. Pandi, F. H. P. Fitzek, and M. Reisslein. Caterpillar RLNC With Feedback (crlnc-fb): Reducing Delay in Selective Repeat ARQ Through Coding. *IEEE Access*, 6:44787–44802, 2018.

[8] D. Stolpmann, C. Petersen, V. Eichhorn, and A. Timm-Giel. Extending On-the-fly Network Coding by Interleaving for Avionic satellite links. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, 2018.