Fulcrum Coding Performance on Fog Computing Architecture

Y. Rivera, *Member, IEEE*, I.S. Gutiérrez, and J. Márquez

***Abstract*— The proposed architecture defines an adaptation of the Fulcrum coding system towards a Fog computing architecture. The distributed coding scheme allows the adjustment of the coding density according to the complexity of each communication device in the network, enabling the system to concentrate the system's performance to those that would allow for a high complexity in the coding if it significantly affects the delay network. These changes enable eliminating redundant information in the network, thus shortening the system delay. Additionally, thanks to Fulcrum coding defining a code concatenation based on RLNC coding, it allows the system to control the coding parameters from the application layer directly. The proposed architecture is an extension of the IoT-Cloud scheme, which allows establishing distributed performance, whose efficiency is controlled by a Fog-RLNC server and a series of Microcasting-type services.**

***Keywords*— Fulcrum code, Fog computing, Random Lineal Network Coding (RLCN), IoT-Cloud, Microcasting.**

I. Introducción

El rápido avance en las comunicaciones y el aumento en las capacidades de procesamiento en dispositivos móviles conectados a la red de datos, han permitido el establecimiento de nuevos escenarios de interacción que implican un aumento masivo para el intercambio de datos entre dispositivos heterogéneos conectados a la nube[1]. Sin embargo, estas nuevas arquitecturas de interconexión contraen algunas restricciones inherentes a la solución, ya que requieren una conexión estable con un mayor costo de operación por byte. Por ejemplo, en la Fig. 1, para encender un led a través de un dispositivo IoT ubicado en la misma *Wireless Local Area Network*(*WLAN*), es necesario que el paquete generado por el dispositivo llegue a la nube o *Cloud-centric IoT*(*CIoT*), en donde se encuentra el servidor IoT, posteriormente la respuesta del paquete retorna a la LAN, lo que supone un retardo adherido a la solución final. Todo este andamiaje implica un mayor costo para la transferencia de los datos, mayores tiempos de transferencias que inciden en el rendimiento del sistema [2].

Por consiguiente, *Fog computing* (ver Fig. 2) se ha propuesto como una solución distribuida que permite mejorar el soporte y la interoperabilidad entre aplicaciones IoT sensibles al tiempo, inclusive entre dispositivos heterogéneos[3].

[1].Msc. Rivera. Julio Yair. Candidate for a doctorate in Systems Engineering and Computer Science, Department of Systems Engineering and Computer Science, Universidad del Norte, Barranquilla, CO 018000 Colombia, (e-mail: yairr@uninorte.edu.co).

Dr. rer. nat. Gutiérrez García. Ismael. Professor Department of Mathematics and Statistics, Universidad del Norte, Barranquilla, CO 018000.Colombia, (e-mail: isgutier@uninorte.edu.co).

Dr Jose Márquez. Professor Systems Engineering and Computer Science, Department of Systems Engineering and Computer Science, Universidad del Norte, Barranquilla, CO 018000 Colombia, (e-mail: jmarquez@uninorte.edu.co).

Fog computing establece algunas de las características de la extensión de la nube o *Edge Data Centers* (EDC), donde se genera un escenario colaborativo de dispositivos (A,B,C) de la Fig. 2 y servicios adaptativos más cerca de la capa del cliente, los cuales no necesariamente tienen conexión a la nube [4].

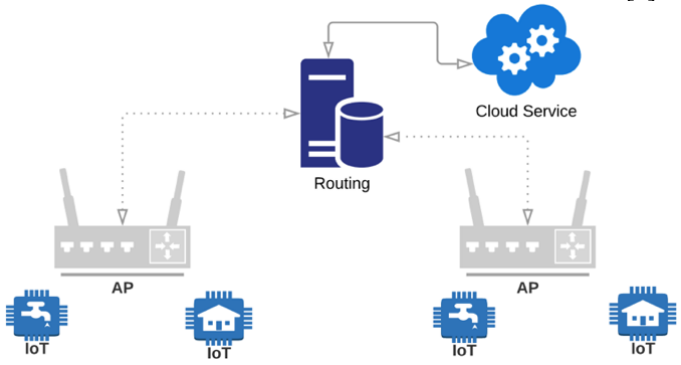

Fig. 1. Arquitectura IoT-*Cloud*.

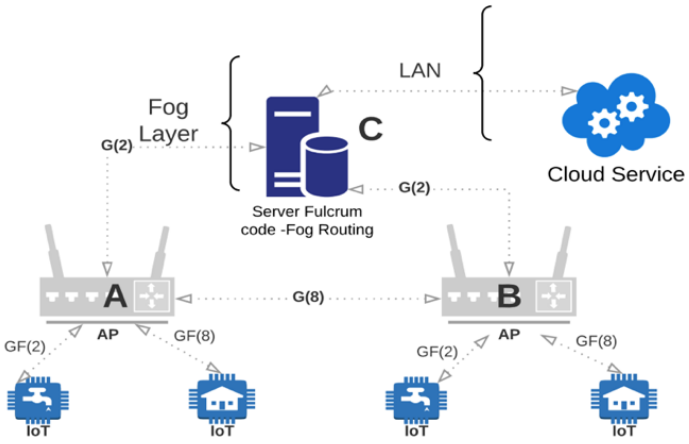

Fig. 2. Arquitectura IoT-Fog

Se han propuesto arquitecturas que formalizan este tipo de arquitectura como IF3C(I= *Internet of things layer*, F3= *Fog data collection*, *Fog Edge layer*, *Fog intermediate computation layer* and C=*Cloud Computing*), las cuales pretenden explotar una gama de servicios móviles. Estos servicios aprovechan, de alguna manera, la heterogeneidad que se presenta en una red inalámbrica con dispositivos de diferentes capacidades de procesamiento y con conexiones concurrentes inalámbricas entre diferentes estándares de comunicación [5]. En este caso se pretende establecer un soporte en la codificación *Random Linear Network Coding* (RLNC) extendida a una codificación *Fulcrum* sobre una arquitectura *Fog computer*, aprovechando el nivel de inteligencia en el procesamiento de cada dispositivo para ofrecer una nueva generación de aplicaciones y servicios;

es decir, pasar de una arquitectura *Cloud computer* establecida en la Fig. 1 a una extensión descentralizada en los servicios de codificación establecida en la Fig. 2.

La principal contribución del trabajo se encuentra resumida en la inserción de un sistema distribuido de codificación dinámica que permite adaptarse a una arquitectura Fog computing. Este es un sistema de códigos de próxima generación diseñado para optimizar la entrega de los datos entre dos puntos con un enfoque de recuperación de datos corruptos o perdidos. Asimismo, su flexibilidad adaptada a *Fog Edger Layer*, contempla reducir la señalización de paquetes y el retardo adicional que acarrea la codificación de servicios en la nube [6]. La arquitectura propuesta pretende disminuir los tiempos de respuesta, eliminar los cuellos de botellas y elevar dinámicamente el rendimiento de la red. La idea clave es utilizar un nodo de borde con las capacidades de procesamiento para definir datos codificados bajo estructuras algebraicas lineales, cuya naturaleza es aprovechada por un amplio alcance de topologías, requeridas para dar ventaja a un número de aplicaciones emergentes[7].

El trabajo desarrollado está estructurado de la siguiente manera: En la sección II se define el esquema de codificación RLNC, como un esquema de códigos de última generación que permite mapear la información a una estructura matemática. La sección III presenta un sistema distribuido, donde se explotan las ventajas de la codificación Fulcrum aplicado a una arquitectura *Fog Computer*. En la sección IV se establece un diseño de experimentos acompañado de las simulaciones y el análisis respectivo, por lo que se evidencia la efectividad del modelo de codificación Fulcrum aplicado a un entorno de conectividad distribuida. Por último, la sección V presenta las conclusiones del trabajo desarrollado y el trabajo futuro propuesto.

II. Planteamiento del Problema

Algunas arquitecturas distribuidas de última generación permiten aprovechar NC y el nivel de procesamiento extra que facilitan algunos puntos de conexión para optimizar el rendimiento del sistema, especialmente en redes heterogéneas inalámbricas[8]. Ahora bien, gracias a que ciertos dispositivos móviles pueden establecer un flujo de datos a través de una red compatible con diferentes estándares de comunicación inalámbrica, es posible crear un flujo de información consistente y compatible ubicado en un modelo híbrido de conexiones. El modelo, que mezcla las ventajas de una red ad-hoc y otra basada en un paradigma de arquitectura, permite sondear las características del rendimiento energético de cada dispositivo, con la finalidad de definir un tratamiento diferente en la dimensión de los datos en cuanto a una codificación dinámica basada en la concatenación de código [9].

Estos nuevos escenarios para la transferencia de datos digitales traen de manera implícita problemas inherentes al medio inalámbrico; por tal motivo, se han desarrollado técnicas de codificación que implementan la redundancia de información en los paquetes de datos para aumentar la probabilidad de decodificación[10]. RLNC se ha venido consolidando como un nuevo esquema de codificación basado en coeficientes lineales dentro de un campo generado normalmente en $GF(2)$, el cual permite establecer matrices de codificación para definir combinaciones lineales de paquetes, dentro de una misma generación[11]. Básicamente RLNC permite realizar un mapeo de paquetes a un espacio vectorial $\mathbb{F}_q^n$; es decir, un conjunto de elementos de longitud n definidos en un campo finito $\mathbb{F}_q$,(donde q es una potencia de un numero primo), el cual se denomina espacio ambiente[12]. Por todo lo anterior, el sistema de codificación RLNC puede verse como la creación de ecuaciones lineales basadas en los fragmentos de los paquetes de longitud n, donde cada coeficiente aleatorio es construido dentro de una matriz de codificación definida en $\mathbb{F}_q$. Cada coeficiente lineal de codificación es un escalar que acompaña a cada símbolo de información o paquete del sistema [13]. Este mapeo de paquetes pretende solucionar un sistema de ecuaciones lineales ligado a un proceso de recuperación de datos. El proceso depende en gran medida de los *buffers* de memoria de los receptores, ya que ningún símbolo del sistema se considera perdido hasta que todos los datos de la correspondiente transmisión sean almacenados y las respectivas ecuaciones estén solucionadas[14]. Para entablar un manejo eficiente en el tratamiento de los paquetes, el esquema de codificación debe ir de la mano del protocolo de comunicación, ya que es importante saber cuáles son los símbolos que se decodificaron al final de la transmisión antes que un protocolo de capa superior solicite el reenvío de este paquete al sistema. Para examinar la codificación que se genera entre los fragmentos de paquetes en todo el núcleo de la red, es necesario examinar la codificación básica Xor que se incorpora como una combinación lineal de vectores (ver Tabla I) [15].

TABLA I
DEFINICIÓN DE LA CODIFICACIÓN XOR EN EL CAMPO DE GALOIS GF(2)[16]

P1	P2	P1 or P2	P1 or (P1 or P2)=P2	P2 or (P2 or P1)=P2
0	0	0	0	0
1	0	1	1	0
0	1	1	0	1
1	1	0	1	1

Para definir la decodificación en el punto final de la comunicación o nodo sumidero, es necesario completar el rango en el buffer de memoria del nodo final o rango en la matriz de decodificación. Formalmente, cada paquete o fragmento de dato es considerado como una ecuación lineal. En total, se necesita el número de ecuaciones lineales igual o superior al rango de la matriz de codificación [17]. Definida la combinación lineal, como el resultado de la codificación a través de coeficientes generados aleatoria y uniformemente en un campo de *Galois*, implica someter la complejidad de codificación y vincularla directamente al tamaño del campo; es decir, a mayor tamaño del campo de *Galois*, mayor es el número de símbolos que se tienen a disposición para representar la información, de ahí que existe menor probabilidad de reproducir vectores linealmente dependientes para representar el espectro de la información [18]. Por otro lado, y de forma particular, existe en una red heterogénea de nodos inalámbricos, algunos dispositivos con mayor poder para procesar campos de mayor complejidad, otros con la memoria necesaria para procesar un tamaño de campo base, en este caso de orden $GF(2)$. Para el núcleo de la red se utiliza un orden de tamaño base, ya que la idea es que la conexión a este nivel debe ser compatible con todos los puntos de comunicación[11].

III. Prototipo de la Arquitectura

A. Modelo Propuesto

El modelo propuesto establece una arquitectura de codificación peer-to-peer que aprovecha la eficiencia de la codificación Fulcrum a través de la codificación RLNC para establecer un tratamiento en cada *streaming* de datos. La granularidad y el reordenamiento de paquetes flexibles a una codificación dinámica y heterogénea, permite un equilibrio de la carga adaptado a cada dispositivo de la red y a su propia ventana de congestión. Como consecuencia, *Fog computing* aprovecha la redundancia de la codificación para aumentar la eficiencia *Forward Error Correction* (FEC), y en consecuencia, también aumenta la probabilidad de decodificación al final de cada transmisión[19]. La codificación Fulcrum es un formato de codificación que concatena dos sistemas de codificación: una codificación base generalmente definida en un campo de orden $GF(2)$, llamada codificación *inner code*, y otra que es una extensión de la codificación base, presentada en un campo de orden $GF(2^n)$, llamada *outer code*. Dado los niveles de complejidad, algunos dispositivos solo pueden codificar en el ámbito de *inner code*, otros de mayor rendimiento pueden establecer la codificación en un orden mayor; es decir, establecer una concatenación completa y llegar a la codificación *outer code*. La Fig. 2 examina la arquitectura de codificación distribuida. Para esta investigación se establece una concatenación de codificación Fulcrum $GF(2^8)$ para *outer code*, y una codificación $GF(2)$ para *inner code*.[20].

El modelo propuesto aprovecha, primero, la capacidad de interactuar en un proceso *cross layer* para sobrellevar múltiples flujos de transmisión con ventanas deslizantes independientes y, segundo, la capacidad de establecer una capa de ajuste definida por la codificación RLNC y un monitor de red centralizado. El monitor de red o servidor de codificación permite obtener con facilidad una visión global de las condiciones de la red; es decir, una capacidad para monitorear un grado de congestión acorde al flujo de transmisión e identificado por el *feedback* de cada transmisión [21]. Gracias a que el modelo pueda establecer una codificación variable en cuanto al orden del campo de codificación, es posible reducir los tiempos de transmisión en puntos específicos de la red, lo que hace a la red inalámbrica fácilmente escalable y tolerante a fallos [22]. Ofrecer un sistema de codificación compuesto permite determinar un balanceo entre el nivel de complejidad y redundancia que ofrece una codificación entrante y otra saliente. Vale mencionar entre la baja complejidad que ofrece una codificación sistemática definida en la codificación *inner code*, y el aumento de grados de libertad establecida por la codificación *outer code*. Sin embargo, implantar un sistema compuesto también trae problemas inherentes a su naturaleza o compatibilidad numérica, ya que la codificación *outer code* al trabajar con una extensión de orden binario, tiene menor posibilidad de generar dependencia lineal en la decodificación, aunque esto conlleve una mayor complejidad en el procesamiento de la información. Por otro lado, la codificación *inner code*, al trabajar con un sistema de orden $GF(2)$, garantiza la velocidad por su baja complejidad; aunque tenga mayor probabilidad de obtener vectores de datos codificados linealmente dependientes[23].

Teniendo fijo el balance entre complejidad y velocidad, este proceso se llevaría a cabo por un codificador establecido en el borde de la red y cuyo proceso es completado por algunos dispositivos internos de alto rendimiento. Cada *streaming* entrante es tomado para adaptarlo a un mejor rendimiento entre dispositivos con conexiones heterogéneas. Establecido de otra manera, se pueden crear conexiones eventuales en dispositivos de alto rendimiento como puentes entre diferentes estándares inalámbricos de comunicación. Siguiendo la misma política de una codificación heterogénea, cada conexión inalámbrica eventual tiene la posibilidad de elevar el orden de codificación. El modelo por consiguiente facilita establecer un *Microcasting* dinámico punto a punto orientado al aumento del rendimiento del sistema en forma colaborativa (ver Fig. 2) [24]. Gracias a la creación de estos nuevos escenarios centrados en la codificación, NC facilita soportar un flujo heterogéneo, dinámico e independiente de datos acorde al dispositivo de transmisión. Ahora bien, hay que resaltar que el tipo de operaciones dentro del hardware está definido bajo una circuitería lógica programable *Field Programmable Gate Array* (FPGA) o *Application-Specific Integrated Circuit* (ASIC). Implementar la codificación RLNC a través de una codificación *inner code* u *outer code*, permite establecer controles nativos que ayudan a disminuir considerablemente los tipos de mensajes necesarios para el control del *streaming*; entre estos: *Acknowledgements*(ACK), *Negative Acknowledgments* (NACK), *Automatic Repeat Request* (ARQ), *Request to Send* (RTS) y *Clear to Send* (CTS), en escenarios de tipo *Multicast* y *Unicast*. Por lo anterior, RLNC permite aumentar la eficiencia para la recuperación de datos a través de la codificación de coeficientes lineales, donde se realiza una codificación Xor básica sobre los paquetes recibidos para generar nuevas instancias de paquetes como lo demuestra la Fig. 3[25].

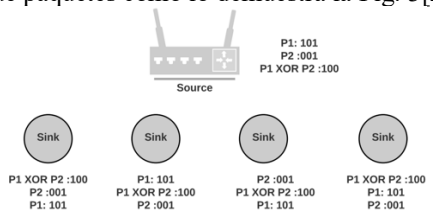

Fig. 3. Cada nodo receptor espera la llegada de segmentos de paquetes.

La arquitectura de codificación está desarrollada de tal manera que solo se pueda establecer una codificación de alta densidad entre dispositivos de alto rendimiento energético, ya que establecer una codificación de alta densidad en un dispositivo de bajo rendimiento energético, es introducir los procesos de codificación como eventos que afectarían de forma negativa en el tiempo de vida de la red y el flujo de la red [26]. Por otro lado, toda esta arquitectura de codificación es posible gracias al formato de trama de codificación propuesto por Medard Muriel, el cual permite trabajar un *streaming* de comunicación con un formato de codificación variable; es decir, gracias a los encabezados de protocolo, cada transmisión conoce qué paquete codificado corresponde a cada flujo de codificación[27]. Sin embargo, este encabezado extra incluye un costo adicional en la sobrecarga de cada paquete; del mismo modo, si ese flujo de transmisión se somete a una codificación sistemática, en cada momento se conoce el contenido exacto y la estructura exacta de la carga útil, por lo que no se generaría una sobrecarga que obedezca a la dimensión del código; es

decir, se usan menos bytes para representar los encabezados [28]. El modelo propuesto permite que cada aplicación pueda evitar transmitir o almacenar más datos de los necesarios. En otras palabras, la eficiencia en el empaquetamiento depende del códec utilizado y el tipo de codificación, ya sea sistemática o no sistemática. Básicamente para las pruebas establecidas, la codificación Fulcrum permitirá una concatenación entre sistemas diferentes de codificación *Inner code*: codificación RLNC y *Outer code*: codificación sistemática. El modelo propuesto permite trabajar distintos tipos de codificación con la restricción de que los coeficientes generados para la codificación *outer code* deben ser una extensión del campo o sistema de codificación establecido en la codificación *inner code*. Lo anterior, garantiza la recodificación o compatibilidad matemática entre los distintos coeficientes de codificación. Esta determinación es importante al momento de establecer un diseño de concatenación de código para dimensiones mayores. Para nuestra experimentación se definió la codificación *inner code* como una codificación RLNC; en cuanto a la codificación *outer code*, se seleccionó una codificación RLNC, aplicando una combinación lineal de todos los paquetes de datos[29]. Las Figs. 4 y 6 ilustran el proceso explicado.

B. Esquema de Codificación y Caracterización de los Nodos

Para entender el esquema de codificación Fulcrum es necesario establecer conceptualmente los parámetros explicados en la Tabla II.

TABLA II
DEFINICIÓN DE TÉRMINOS IMPORTANTES[16]

Término	Definición
Galois	Campo generador de los coeficientes lineales de codificación.
Generación	Número de símbolos g o paquetes en un bloque de codificación.
Nodo Fuente	Nodo origen de la transmisión.
Tamaño del campo	Definido por coeficientes generados en $GF(2)$, y sus extensiones $GF(2^4)$ y $GF(2^8)$.
Nodo Sumidero	Nodo receptor de la transmisión.
Expansión	Número de expansión r o símbolos adicionales a usar, el número de símbolos adicionales creados por el código externo (*outer code*).
Vector de codificación	Vector que guarda en cada una de sus entradas la codificación resultante, producto de la combinación lineal Xor.
Vector innovador	Vector que aumenta el rango de la matriz de decodificación.

La codificación Fulcrum establece un mapeo de código entre campos de *Galois* de $GF(2^n)$ y $GF(2)$, y viceversa. Para una condición inicial, la codificación *outer code* genera unos paquetes redundantes llamados segmentos de expansión r. Esta expansión es el resultado de efectuar una combinación lineal de paquetes originales codificados por la matriz de codificación generada en el campo $GF(2^n)$, lo que significa que cada entrada de la matriz o coeficiente c pertenece a este campo de *Galois*. Si se quiere establecer una relación entre el campo de codificación y el dispositivo transmisor, la extensión n dependería de la capacidad de procesamiento que posee este dispositivo.

La Ec (1) representa cada entrada del nuevo vector de expansión U, el cual hace parte de un nuevo vector de codificación. Este vector se añade a los paquetes nativos p; es decir, se realiza una codificación sistemática [30].

$$U_i = \sum_{j=1}^{g} c_{i,j} p_j \,, i = 1,2,\dots,r \qquad (1)$$

Congruentemente el nuevo vector codificado $P = [P\ U]$ quedaría de longitud g + r. La Fig. 4 ilustra un ejemplo de esta codificación sistemática, la cual toma una generación de paquetes y genera combinaciones lineales para establecer una redundancia en el nuevo vector de codificación. La codificación *outer code* es caracterizada por el seguimiento del Algoritmo 1[31].

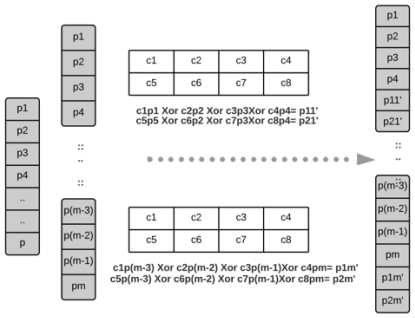

Fig. 4. Descripción *outer code*: Codificación sistemática con *m* segmentos de paquetes, *g*=4 y *r*=2 por cada bloque de codificación

Esta expansión permite trabajar vectores con mayores dimensiones, lo que teóricamente apunta a aumentar una redundancia de la información, todo a costa de un alto consumo energético debido a la complejidad que implica el número de operaciones que conlleva procesar símbolos que hacen parte del campo $GF(2^n)$.

Sin embargo, esta la redundancia de código se compensa, ya que una codificación sistemática se caracteriza por tener un bajo retardo, como consecuencia de su baja complejidad[32]. Adicionalmente, y para terminar la codificación Fulcrum, es necesario complementar el proceso a través de una codificación *inner code*; en este caso, basada en la codificación RLNC $GF(2)$ que conlleve la disminución de la complejidad al momento de la decodificación. Finalmente, el vector de codificación resultaría con dimensión $g + r$. En la Ec (2) se caracteriza el procedimiento.

Algorithm 1 Codificación Outer Fulcrum
Require: Generación $g > 0$,Expansion $r > 0$
Paquetes originales $\mathcal{P}_i = \{p_1, p_2, p_3, ..p_g\}$.
Ensure: Vector de codificado $\mathcal{U}_i = \{q_1, q_2, q_3, ..q_g\}$.
$\mathcal{U}_i \leftarrow \mathcal{P}_i$
$j \leftarrow 1$
for $j \leq r$ **do**
$i \leftarrow 1$
$\mathcal{U}_i \leftarrow \vec{0}$
$J \leftarrow \emptyset$
while $i \leq g$ **do**
$j \leftarrow J$ Random paquetes de $\{1, 2, 3..g\} \setminus J$
$c \leftarrow C$ Random coeficientes de $\{1, 2, 3..g\} \setminus GF(2)^8$
$J \leftarrow J \cup \{j\}$
$\mathcal{Q}_j \leftarrow \mathcal{Q}_j \oplus (\mathbf{c_{ij}} \cdot \mathcal{P}_i)$
$i \leftarrow i + 1$
end while
$\mathcal{U} \leftarrow \mathcal{U} \cup \mathcal{Q}_j$
end for

Cada entrada del vector de codificación resultante Q_i, es producto de una nueva combinación entre los nuevos coeficientes de codificación y cada entrada del vector de paquetes originales P_j.

$$Q_i = \sum_{j=1}^{g+r} c_{i,j} P_j,\ i = 1,2,\dots,g+r \qquad (2)$$

Este proceso es caracterizado por el Algoritmo 2, en donde se aprecia una codificación tradicional RLNC. Hay que resaltar que cada elemento presente en Q_i está conformado tanto por los segmentos de los g paquetes originales, como los resultantes de sus combinaciones lineales $\boldsymbol{Q} = \{p_1, p_2, p_3, \dots, p_g, c_{g+1}, c_{g+2}, \dots, c_{g+r}\}$[33].

Finalmente, la Fig. 5 ilustra cómo se establece el proceso de concatenación entre los dos sistemas de codificación.

Ahora bien, por ser una red de naturaleza inalámbrica y porque no todos los dispositivos de la red *Fog computing* tienen la misma capacidad de realizar una codificación *outer code*, en el medio viajarán paquetes de diferentes dimensiones tanto de $GF(2)$ como $GF(2^n)$. Por esta razón, es importante que cada nodo final al momento de realizar la decodificación pueda diferenciar entre una codificación *outer code* e *inner code*[34]. Es importante garantizar las operaciones entre los distintos elementos, o coeficientes que se generan en los respectivos campos generadores, tanto en el binario base, generados por la codificación *inner code*; como en sus diferentes extensiones, generados en la codificación *outer code*. Para considerar esta correspondencia es necesario utilizar el Lema 1.

Algorithm 2 Codificación Inner Fulcrum

Require: Generación $g > 0$, Expansion $r > 0$,
Paquetes originales $\mathcal{P}_i = \{p_1, p_2, p_3, ..p_{g+r}\}$.
Ensure: Vector de codificado $\mathcal{Q}_i = \{q_1, q_2, q_3, ..q_{g+r}\}$.
$i \leftarrow 1$
$\mathcal{Q}_i \leftarrow \vec{0}$
$J \leftarrow \emptyset$
while $i \leq g + r$ **do**
$j \leftarrow J$ Random paquetes de $\{1, 2, 3..g + r\} \setminus J$
$c \leftarrow C$ Random coeficientes de $\{1, 2, 3..g + r\} \setminus GF(2)$
$J \leftarrow J \cup \{j\}$
$\mathcal{Q}_i \leftarrow \mathcal{Q}_i \oplus (c_{ij} \cdot \mathcal{P}_i)$
$i \leftarrow i + 1$
end while

Lema 1. [35] Teorema 2.6 y [36] Teorema 6.6.
Sea $q = p^n$

a) Todo sub campo de $\mathbb{F}_q$ tiene orden p^m, donde m es un divisor de n.
b) Para cada divisor m de n, existe un único sub campo de $\mathbb{F}_q$ con p^m elementos.

Ejemplo: Los únicos sub campos de $\mathbb{F}_{16}$, son $\mathbb{F}_2, \mathbb{F}_4$ y $\mathbb{F}_{16}$.

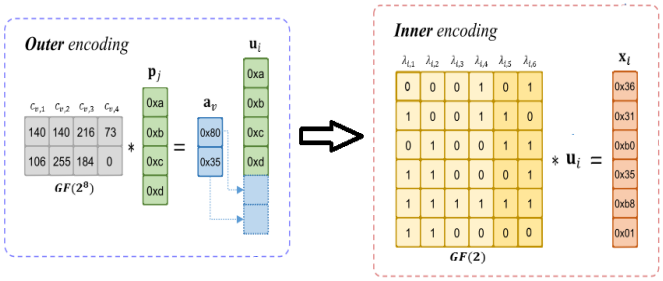

Fig. 5. Descripción de codificación Fulcrum[37].

Gracias al Lema 1, es posible configurar una decodificación dinámica por cada *streaming* de datos en una codificación Fulcrum; es decir, realizar operaciones con coeficientes binarios y ser compatibles al mismo tiempo con sus diferentes extensiones. Este cambio en el formato de codificación establece una ventaja significativa, ya que, si no hay suficientes paquetes recibidos con éxito, este puede completar el rango de la matriz en el buffer de decodificación, con otros segmentos generados por extensiones mayores. Sin embargo, se impone una alta complejidad computacional debido a las altas operaciones que exige el campo generador de los coeficientes.

IV. Diseño de Experimentos y Resultados

A. Escenario de Simulación

El escenario principal de la simulación se basa en la Fig. 2, donde se tiene un servidor (C) de codificación Fulcrum o *Fog node*. Los enrutadores (A y B) en la red interna inalámbrica hacen la función de codificadores intermedios, lo que permitirá aprovechar las ventajas de los servicios *Fog-layers* a través de una codificación dinámica. Por tanto, se generará un trafico entre el nodo fuente y cualquier dispositivo IoT o nodo sumidero de la red interna inalámbrica, este tráfico puede ser generado según la Tabla III. Nuestra investigación gira en torno a 3 escenarios, en donde se consideran 3 tipos de ordenes de codificación: Inner code, Outer code y una combinación de esquemas.

Los datos se obtuvieron a través de bloques seleccionados aleatoriamente, y con un promedio de 100 datos por muestra para garantizar el principio de la normalidad y no correlación. De forma exploratoria se realizó una valoración inicial de la influencia del orden de la codificación en el rendimiento del sistema (ver Fig. 6a y 6b). Consecutivamente se realizó un análisis más detallado, teniendo en cuenta los errores que se presentaron en el sistema.

TABLA III
Configuración de Parámetros [16]

Parámetro	Valor
Inner code $\mathbb{F}_q$	*GF (2)*.
Outer code $\mathbb{F}_q$	*GF (2^4) y GF (2^8).*
Fulcrum code (Inner y Outer)	*GF (2), GF (2^4) y GF (2^8).*
Generación	*g*=[16,32,64,128,256,512].
Trama de paquete	2296 bytes.
Expansión	*r*=[1 2 3 4].
Número de muestras aleatorias	10000
Métrica de ancho de banda	throughput (MB/s) y goodput (MB/s)
Métrica de decodificación	Tiempo(milisec)

B. Diseño de Experimentos

Se configuró la librería Kodo y el simulador NS3 bajo la topología de la Fig. 2. Se definió un tamaño de trama máximo permitido que cuenta con una sobrecarga que contempla un encabezado LLC de 8 bytes. Cuidadosamente se seleccionó el tamaño del bloque o generación g, tanto para el vector redundante r como por cada bloque de generación; ya que seleccionar un bloque muy grande puede resultar en una gran sobrecarga de latencia, incluso más de lo que sería inducido por una retransmisión. Por otro lado, al seleccionar un bloque demasiado pequeño puede ser insuficiente para proteger contra la cantidad requerida de pérdida de paquetes.

C. Resultados Obtenidos

Se generó un tráfico con generación g=64, expansión r=[1,2,3,4], y una codificación de orden $GF(2)$, $GF(2^4)$ y $GF(2^8)$. Según la Fig. 6a, cuanto mayor es r, menor es el número de símbolos en la decodificación para generar la misma información en el nodo final; lo anterior debido a que los bytes redundantes r ayudan a generar un vector innovador con una mayor probabilidad. Sin embargo, hay que resaltar que la mayor parte del procesamiento se concentra en la decodificación *outer code*; por tanto, a medida que aumente el número de la generación para la codificación, se va haciendo despreciable el procesamiento establecido por los r bytes redundantes. Por otro lado, en la Fig. 6b, se observa que el aumento de los bytes redundantes, generan un tiempo extra en la decodificación, el cual va aumentando progresivamente a medida que aumentan las dimensiones del campo de *Galois*[38]. Adicionalmente, esta concatenación de código incurre de forma inherente en el proceso de eliminar la información redundante en el sistema, debido a las combinaciones lineales de los paquetes de datos desarrolladas por la codificación RLNC, reduciendo así cada flujo de transmisión y por consiguiente extendiendo el tiempo de vida de la red [39]. Por otro lado, se deduce que la codificación Fulcrum reduce la sobrecarga en la red originada por el reenvío de información; mantiene, además, un alto rendimiento en la codificación, al establecer un flujo de *streaming* heterogéneo y compatible al mismo tiempo con campos de *Galois* $GF(2)$ y sus respectivas extensiones $GF(2^4)$ y $GF(2^8)$.

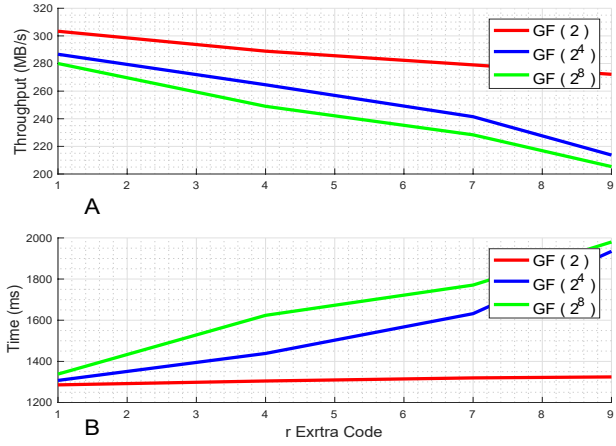

Fig. 6. (a) Throughput de la decodificación. (b) Tiempo de la decodificación.

El esquema de codificación exige un mayor nivel de procesamiento distribuido en la red, cuando trabaja con extensiones de campos superiores al binario, ya que implica un mayor nivel de complejidad; sin embargo, el mismo esquema establece una compensación entre la velocidad de procesamiento y la probabilidad de decodificación[40].

Teniendo en cuenta que este sistema de codificación es totalmente compatible con los servicios de capas superiores, un vector de codificación generado por la codificación Fulcrum, puede realizar un ajuste de expansión dependiendo del flujo de datos en el nivel de aplicación, de tal manera que cada *streaming* de datos puede realizar una calibración con base en el tipo de dispositivo *Edge Computing Network*.

Este ajuste gradual de la expansión se relaciona directamente con las características de procesamiento de cada dispositivo en la red, de tal manera que en cada punto de comunicación se puede cambiar el orden de la codificación acorde con un mejor rendimiento y las probabilidades de decodificación en los paquetes de datos[41].

Siguiendo la misma línea de análisis, la codificación Fulcrum puede aprovechar una gran variedad de dispositivos heterogéneos para aumentar la eficiencia del sistema. Por otro lado, se origina una variedad de tráfico inherente al sistema de codificación: *inner code* $GF(2)$, *outer code* $GF(2^8)$ y Fulcrum *code* (*Inner* $GF(2)$ y *Outer* $GF(2^8)$). Por tal motivo, para examinar su comportamiento se necesita observar los errores en la transmisión. En este caso, se define el *goodput* como métrica necesaria para determinar el nivel de tráfico verdaderamente útil. Se excluye la sobrecarga del protocolo debido al *overhead*, así como los datos originados por la retransmisión[42]. Con base en lo anterior, si tenemos el *throughput* y el *goodput* definidos por unidad de tiempo, tenemos una idea de los errores que se generan en el sistema debido al *overhead* de los coeficientes de codificación, y a la redundancia de tráfico debido a las dependencias lineales(ver Fig. 7 y Fig. 8).

La Fig. 7 evidencia una codificación totalmente heterogénea en la red, lo que implica una complejidad adicional debido al tratamiento de coeficientes de orden de $GF(2)$ y $GF(2^8)$.

En la Fig. 8, se observa una mayor diferencia entre el *goodput* y el *throughput*, esto es consecuencia de que una codificación de orden $GF(2)$ genera más paquetes linealmente dependientes, por tanto, genera más tráfico redundante en el sistema.

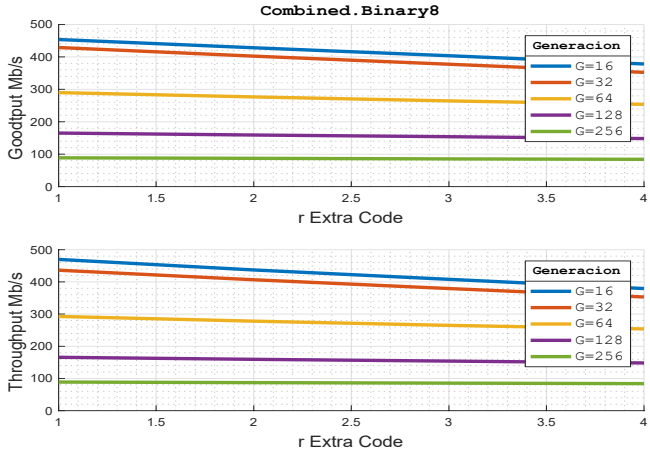

Fig. 7. Decodificación Fulcrum code (Inner $GF(2)$ y Outer $GF(2^8)$).

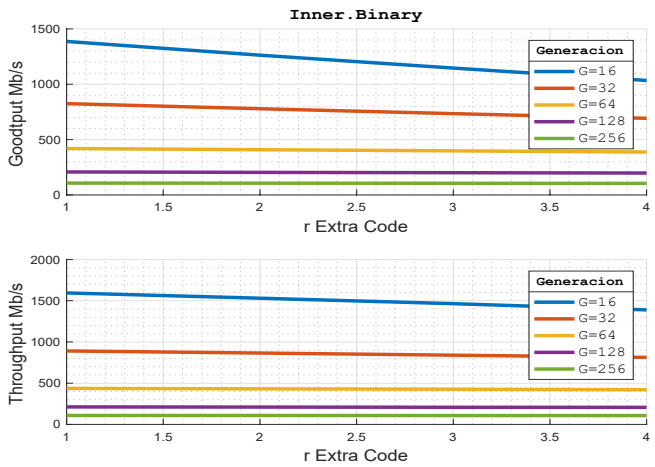

Fig . 8. Decodificación Inner code $GF(2)$.

Ahora bien, si hacemos un análisis para un tráfico exclusivo de una extensión binaria $GF(2^8)$, se observa que no hay mucha diferencia entre el *goodput* y el *throughput*. Esto es consecuencia de que esta codificación genera más grados de libertad al momento de hacer combinaciones lineales; por tanto, genera menos paquetes linealmente dependientes (ver Fig. 9). El interés particular de esta investigación se centra en un escenario de codificación dinámica Fulcrum, el cual permite aprovechar la codificación dinámica de los datos en una red con dispositivos heterogéneos. Básicamente se podría elevar el *throughput* del sistema, teniendo como consecuencia aumentar la complejidad de la codificación en puntos específicos de la red.

Por otro lado, gracias a la Tabla IV, se puede deducir la matriz de covarianza producto de los errores de transmisión y la sobrecarga que causa la codificación para varios niveles de generación *g*[16,32,64,128,256] y expansión *r*[1,2,3,4](ver Fig. 10).

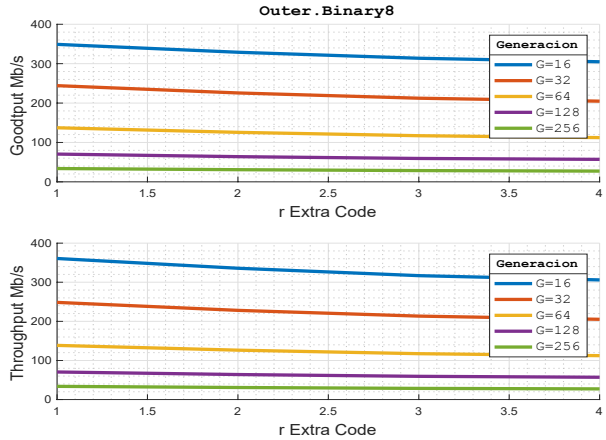

Fig . 9. Decodificación Outer Code $GF(2^8)$.

TABLA IV
RENDIMIENTO DE LA RED

G	Goodput (r extra code)				Throughput (r extra code)			
	1,0	2,0	3,0	4,0	1,0	2,0	3,0	4,0
16,0	359,2	347,3	259,6	243,0	372,7	353,4	261,3	243,6
32,0	196,2	160,2	105,9	121,0	199,4	161,3	106,4	121,3
64,0	128,3	123,5	120,8	114,8	129,4	124,1	121,0	115,0
128,0	69,6	48,3	45,0	49,6	70,0	48,4	45,1	49,6
256,0	34,7	29,3	21,4	19,3	34,8	29,3	21,4	19,3

Con base en los últimos escenarios de simulación, variando el número de generación. La matriz de covarianza evidencia un mayor número de errores en todas las transmisiones, al trabajar con un tipo de generación de menor tamaño. Recordemos que un número menor de generaciones implica un mayor volumen en los coeficientes de codificación ubicado en los encabezados de cada protocolo, lo que genera un *overhead* mayor en el sistema. No obstante, este tráfico redundante es atenuado a medida que aumenta el nivel de expansión r. Efecto contrario causa cuando trabaja con extensiones de campos superiores al binario, ya que implica un mayor nivel de complejidad; sin embargo, el mismo esquema establece una compensación entre la velocidad de procesamiento y la probabilidad de decodificación[40].

El esquema de codificación exige un mayor nivel de procesamiento distribuido en la red cuando trabaja con extensiones de campos superiores al binario, aunque genere más número de vectores innovadores, mientras que una codificación de orden binario $GF(2)$ tiene un efecto contrario. Hay que resaltar que este tipo de codificación tiene mayor probabilidad de no generar vectores innovadores, por lo tanto es el tipo de codificación que genera más vectores linealmente dependientes [43]. No hay que olvidar el hecho que el proceso de decodificación necesita una matriz de rango completo; es decir, que a medida que se presente independencia lineal entre los vectores, será más rápido completar los tiempos de transmisión. En cambio, la dependencia lineal se traduce en tráfico innecesario que no aporta al proceso de la decodificación, pero si a los errores del sistema debido a los cuellos de botella.

D. Análisis de Resultados

Acorde con los resultados, se puede apreciar que, al cambiar el orden en la codificación, es posible también reducir un flujo redundante que no aporta información para crear vectores innovadores; consiguientemente, se disminuye el número de errores en la red. Sin embargo, esto trae una consecuencia directa sobre el consumo energético, ya que, al realizar una codificación en campos de extensión binaria, esto conlleva aumentar la complejidad , por tanto, eleva con facilidad el consumo energético. Hay que tener en cuenta que los cálculos basados en campos de *Galois* tienen una complejidad asociada al número de símbolos[44]. Podemos agregar a lo anterior, que el número de generación también tiene una incidencia directa sobre los errores del sistema, ya que a medida que disminuye este factor, son más los paquetes de datos que deben enviarse para transmitir la misma información. En contrapartida, aumentar el número de generación implica aumentar el número de operaciones que conlleva la decodificación al reducir una matriz de rango completo.

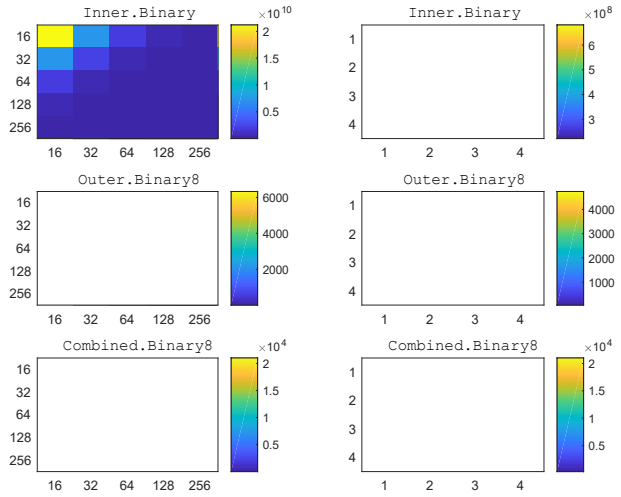

Fig. 10. Matriz de covarianza del trafico redundante e innecesario.

La codificación Fulcrum reduce los tiempos de transmisión al aumentar la probabilidad de generar vectores innovadores en el sistema. También se le puede abonar a este proceso un control dinámico que permite reducir las operaciones vinculadas al tratamiento de los paquetes, lo que se traduce al final en una extensión del ciclo de vida de la red. Todo este conjunto de mecanismos permite que el *Fog-node* pueda ejecutar servicios de codificación Fulcrum, además de fijar un balanceo en el sistema. Finalmente, se resalta el hecho de que el consumo energético en la red es linealmente proporcional a la complejidad computacional; por consiguiente, el *throughput* en la codificación y la decodificación es inversamente proporcional a esta misma complejidad computacional[45].

V. Conclusiones

La compatibilidad de la codificación *Fulcrum* con servicios de capas superiores y ligado a una arquitectura *Fog computer*, garantiza un despliegue de forma eficiente en la transmisión de los datos en un amplio escenario de configuraciones y con topologías hibridas. Un esquema distribuido de codificación que aprovecha las transmisiones de datos *Multicast* y los dispositivos con un rendimiento heterogéneo en la red para fijar un balanceo entre rendimiento y codificación. El objetivo es reducir la cantidad de datos que se envían a la nube, disminuir la latencia de la red y acortar los tiempos de transmisión para aumenta la confiabilidad de las redes Fog Computer. Esto a través de la intensa codificación *Fulcrum* y un balanceo de parámetros que influyen en el rendimiento de la red. Toda esta arquitectura de configuración se traduce en una mejora gradual de la QoS en el streaming de los datos, con un alto rendimiento sostenible y adaptable en cada punto de la comunicación.

Futuros Trabajos

Se tienen focalizadas futuras líneas de investigación relacionadas con la optimización de parámetros lineales basadas en series de tiempo, en donde se examinarán parámetros relacionados directamente con la matriz de codificación. Esto con el objetivo de definir transmisiones inalámbricas basadas en la codificación RLNC con un grado menor de complejidad, lo que se traduce en un menor consumo energético.

Agradecimientos

Agradecemos a la Universidad del Norte que ha apoyado con tiempo y recursos para el desarrollo de la investigación en Códigos de Red.

Referencias

[1] Y. E. R. Julio, "Development of a Prototype Arduino-Mobile in Area of Telemedicine for Remote Monitoring Diabetic People (MAY 2015)", *Proc. - 2015 Asia-Pacific Conf. Comput. Syst. Eng. APCASE 2015*, núm. May, pp. 36–40, 2015.

[2] A. Brogi, S. Forti, A. Ibrahim, y L. Rinaldi, "Bonsai in the Fog: An active learning lab with Fog computing", *2018 3rd Int. Conf. Fog Mob. Edge Comput. FMEC 2018*, pp. 79–86, 2018.

[3] D. You *et al.*, "Fog Computing as an Enabler for Immersive Media: Service Scenarios and Research Opportunities", *IEEE Access*, vol. 7, pp. 65797–65810, 2019.

[4] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, y A. Y. Zomaya, "Secure and Sustainable Load Balancing of Edge Data Centers in Fog Computing", *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 60–65, 2018.

[5] T. Islam y M. M. A. Hashem, "Task Scheduling for Big Data Management in Fog Infrastructure", *2018 21st Int. Conf. Comput. Inf. Technol. ICCIT 2018*, pp. 1–6, 2019.

[6] N. J. Hernández, J. Pihl, J. Heide, J. Krigslund, M. V Pedersen, y P. Vingelmann, "Wurf.it: A Network Coding Reliable Multicast Content Streaming Solution - NS-3 Simulations and Implementation", pp. 978–1.

[7] V. U. Nguyen, E. Tasdemir, G. Nguyen, D. E. Lucani, H. P. Frank, y M. Reisslein, "DSEP Fulcrum : Dynamic Sparsity and Expansion Packets for Fulcrum Network Coding", vol. 4, pp. 1–23, 2020.

[8] Y. N. Shnaiwer, S. Sorour, T. Y. Al-Naffouri, y S. N. Al-Ghadhban, "Opportunistic Network Coding-Assisted Cloud Offloading in Heterogeneous Fog Radio Access Networks", *IEEE Access*, vol. 7, pp. 56147–56162, 2019.

[9] S. Singh, N. Saxena, A. Roy, y H. S. Kim, "A Survey on 5G Network Technologies from Social Perspective", *IETE Tech. Rev. (Institution Electron. Telecommun. Eng. India)*, vol. 34, no. 1, pp. 30–39, 2017.

[10] V. Nguyen *et al.*, "Adaptive Decoding for Fulcrum Codes", en *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2018, pp. 133–139.

[11] R. Parsamehr, A. Esfahani, A. Radwan, y S. Mumtaz, "A Novel Intrusion Detection and Prevention Scheme for Network Coding-Enabled Mobile Small Cells", núm. November, 2019.

[12] I. Gutierrez-Gracia y I. M. Naizir, "Constructions of cyclic and quasi-cyclic Grassmannian codes", *IEEE Lat. Am. Trans. VOL. 17, NO. 7, JULY 2019*, vol. 17, no. 7, pp. 1180–1190, 2019.

[13] J. Bilbao, P. M. Crespo, I. Armendariz, y M. Medard, "Network Coding in the Link Layer for Reliable Narrowband Powerline Communications", *IEEE J. Sel. Areas Commun.*, vol. 34, no 7, pp. 1965–1977, 2016.

[14] J. J. (Jon. H. Park, S. C. Chen, J. M. Gil, y N. Y. Yen, "Multimedia and Ubiquitous Engineering", *Lect. Notes Electr. Eng.*, vol. 308, pp. 121–126, 2014.

[15] D. T. Hai, "On routing, spectrum and network coding assignment problem for transparent flex-grid optical networks with dedicated protection", *Comput. Commun.*, vol. 147, no. July 2018, pp. 198–208, 2019.

[16] J. H. M. Medard, K. Fouli y K. F. Steinwurf Aps, S. Shi, "Random Linear Network Coding (RLNC)-Based Symbol Representation draft-heide-nwcrg-rlnc-02", pp. 1–13, 2020.

[17] E. Tsimbalo, A. Tassi, y R. J. Piechocki, "Reliability of Multicast under Random Linear Network Coding", *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2547–2559, 2018.

[18] Y. R. Julio, I. G. Garcia, y J. Marquez, "R-IoT: An architecture based on recoding RLNC for IOT wireless network with erase channel", en *Advances in Intelligent Systems and Computing*, 2020, vol. 1137 AISC, pp. 579–588.

[19] C. Bouras y N. Kanakis, "AL-FEC Application on NGMN-Edge Computing Integrated Systems", *Int. Conf. Ubiquitous Futur. Networks, ICUFN*, vol. 2018-July, pp. 364–369, 2018.

[20] V. Nguyen, E. Tasdemir, G. T. Nguyen, D. E. Lucani, y F. H. P. Fitzek, "Tunable expansion packets for fulcrum codes", *Eur. Wirel. 2019 Conf.*

EW 2019, no. May, pp. 1–7, 2019.
[21] L. Chen, Y. Feng, B. Li, y B. Li, " Promenade : Proportionally Fair Multipath Rate Control in Datacenter Networks with Random Network Coding ", *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, núm. 11, pp. 2536–2546, 2019.
[22] D. Levonevskiy, A. Saveliev, I. Duboyskiy, y P. Drugov, "Cloud System for Distributing Multimedia Content in Cyber-Physical Systems", 2020, pp. 266–274.
[23] P. Garrido, C. W. Sorensen, D. E. Lucani, y R. Aguero, "Performance and complexity of tunable sparse network coding with gradual growing tuning functions over wireless networks", *IEEE Int. Symp. Pers. Indoor Mob. Radio Commun. PIMRC*, 2016.
[24] I. Leyva-Mayorga *et al.*, "Network-Coded Cooperation and Multi-Connectivity for Massive Content Delivery", *IEEE Access*, vol. 8, pp. 15656–15672, 2020.
[25] M. V Pedersen *et al.*, "(12) United States Patent", vol. 2, no. 12, 2017.
[26] Z. Li, M. Xu, T. Liu, y L. Yu, "A Network Coding-Based Braided Multipath Routing Protocol for Wireless Sensor Networks", vol. 2019, 2019.
[27] S. Shi y K. Fouli, "Random Linear Network Coding (RLNC)-Based Symbol Representation draft-heide-nwcrg-rlnc-background-00 draft-heide-nwcrg-rlnc-01", núm. March, pp. 1–17, 2019.
[28] N. Muangboonma, "Latency-reduced Retransmission in Multicast Network using Systematic Network code *", pp. 4–7.
[29] D. Ruano *et al.*, "Fulcrum: Flexible Network Coding for Heterogeneous Devices", *IEEE Access*, vol. 6, pp. 77890–77910, 2018.
[30] D. E. Lucani, M. V. Pedersen, J. Heide, y F. H. P. Fitzek, "Coping with the upcoming heterogeneity in 5G communications and storage using Fulcrum network codes", *2014 11th Int. Symp. Wirel. Commun. Syst. ISWCS 2014 - Proc.*, pp. 997–1001, 2014.
[31] R. Zlwk, X. Lq, y U. Luhohvv, "F M KFNetwork Coding with buffer scheme in multicast for broadbad wireless network", vol. 10, núm. 6, pp. 1022–1027, 2016.
[32] N. Muangboonma, N. Puttarak, y P. Kaewprapha, "Latency-reduced retransmission in multicast network using systematic network code", *Int. Conf. Electron. Information, Commun. ICEIC 2016*, pp. 4–7, 2016.
[33] M. Zverev y P. Garrido, "Systematic Network Coding with Overlap for IoT Scenarios", 2019.
[34] D. S. Meitis, D. S. Vasiliev, I. Kaisina, y A. Abilov, "Simulation-based Research of BATS Code Applied to Flying Ad-hoc Networks", *Moscow Work. Electron. Netw. Technol. MWENT 2020 - Proc.*, pp. 18–21, 2020.
[35] R. Lidl y H. Niederreiter, *Finite fields*. Cambridge University Press, 1996.
[36] R. J. McEliece, *Finite fields for computer scientists and engineers*. Kluwer Academic Publishers, 1987.
[37] V. Nguyen, J. A. Cabrera, D. You, H. Salah, G. T. Nguyen, y F. H. P. Fitzek, "Advanced Adaptive Decoder Using Fulcrum Network Codes", *IEEE Access*, vol. 7, pp. 141648–141661, 2019.
[38] M. S. G. Petrovic, "Galois Field Arithmetic Block Coding : DVB-FEC – Erasure Codes – (optional task for bonus points)", pp. 1–2.
[39] F. H. P. Fitzek, J. Heide, M. V Pedersen, F. H. P. Fitzek, y M. Muriel, "Perpetual Codes for Network Coding A Perpetual Code for Network Coding", núm. SEPTEMBER, 2015.
[40] A. F. Gutiérrez, "Trabajo Fin de Grado recodificación en soluciones RLNC : caracterización sobre una plataforma basada en Raspberry Pi On the benefits of recoding within RLNC solutions : characterization over a Raspberry Pi based platform", 2018.
[41] K. Matsuzono *et al.*, "Low Latency Low Loss Streaming using In-Network Coding and Caching To cite this version : HAL Id : hal-01437074 Low Latency Low Loss Streaming using In-Network Coding and Caching", 2017.
[42] J. Jansons, "Goodput analysis in short range vehicle network depends on auto traffic parameters", *2012 2nd Int. Conf. Digit. Inf. Process. Commun. ICDIPC 2012*, pp. 22–25, 2012.
[43] C. W. Sørensen, *Aalborg Universitet On Tunable Sparse Network Coding in Commercial Devices for Networks and Filesystems Sørensen , Chres Wiant Publication date :* 2017.
[44] S. Soltani *et al.*, "Cross-layer Design and SDR Implementation of DSA, Backpressure Routing and Network Coding", 2019.
[45] N. Aboutorab, P. Sadeghi, y S. Sorour, "Enabling a tradeoff between completion time and decoding delay in instantly decodable network coded systems", *IEEE Trans. Commun.*, vol. 62, no. 4, pp. 1296–1309, 2014.

Yair Rivera Julio received an MSc degree in Telematics and Telecommunications in Engineering from Universidad del Norte, Barranquilla, Colombia. His research is focused on the Wireless Network Coding with Multicasting. Current research interests are field finite soluble and their applications in transfer data digital, Network Sensors and IoT.

Ismael Gutiérrez García received the Ph.D. degree in Mathematics from Johannes Gutenberg University Mainz - Germany. His current research interests are finite soluble groups, Discrete Mathematics, and their applications.

José Márquez Díaz received the MSc degree in Computer Science from ITESM-México in agreement with Universidad Autónoma de Bucaramanga and BSc degree in Computer Science from Universidad del Norte, Barranquilla, Colombia. His main research is focused on Quality of Service (QoS) in Computer Networks, AdHoc Networks, and Network Coding in Multicasting.