(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 926 238**

(51) Int. CI.:

H04L 1/00 (2006.01)
H04L 1/18 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **29.03.2018 PCT/US2018/025168**

(87) Fecha y número de publicación internacional: **04.10.2018 WO18183694**

(96) Fecha de presentación y número de la solicitud europea: **29.03.2018 E 18720064 (7)**

(97) Fecha y número de publicación de la concesión europea: **06.07.2022 EP 3602871**

(54) Título: **Sistema y técnica para la generación de paquetes basados en codificación de red de ventana deslizante**

(30) Prioridad:

29.03.2017 US 201762478114 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
24.10.2022

(73) Titular/es:

MASSACHUSETTS INSTITUTE OF TECHNOLOGY (33.3%)
77 Massachusetts Avenue
Cambridge, MA 02139, US;
CODE ON NETWORK CODING, LLC (33.3%) y
TECHNISCHE UNIVERSITÄT DRESDEN (33.3%)

(72) Inventor/es:

MEDARD, MURIEL;
WUNDERLICH, SIMON;
PANDI, SREEKRISHNA;
GABRIEL, FRANK y
FOULI, KERIM

(74) Agente/Representante:

ELZABURU, S.L.P

DESCRIPCIÓN

Sistema y técnica para la generación de paquetes basados en codificación de red de ventana deslizante

Antecedentes

Los codificadores de codificación de red de ventana deslizante son conocidos en la técnica. La ventana deslizante se refiere a una técnica en la que se aplica codificación lineal a una ventana de paquetes para generar un paquete codificado. En general, un tamaño de ventana utilizado en tal codificador varía dinámicamente entre 1 y un tamaño de ventana máximo de N donde N puede, teóricamente, ser infinitamente grande. Los expertos en la técnica apreciarán cómo seleccionar el valor de N en sistemas prácticos.

En general, un codificador de ventana deslizante puede funcionar de la siguiente manera: primero, el codificador de ventana deslizante se alimenta con nuevos datos. En respuesta a los datos proporcionados al mismo, el codificador aumenta el tamaño de la ventana del codificador de ventana deslizante. Un remitente envía un paquete a un receptor y, tras la recepción del paquete, el receptor envía un paquete de ACK de vuelta al remitente. El codificador de ventana deslizante recibe el paquete de ACK y, en respuesta al mismo, puede reducir el tamaño de la ventana deslizante (de ahí la razón de que se haga referencia a la técnica como técnica de ventana deslizante, es decir, se añaden nuevos paquetes al codificador y se eliminan los antiguos). Una ventana deslizante se puede aplicar sobre una generación (es decir, sobre un grupo de paquetes) o puede ser sin generación.

Como también se sabe, algunos codificadores de ventana deslizante aplican codificación de red lineal aleatoria (RLNC) como un código convolucional. Un estudio general de tal código de ventana deslizante y sus propiedades fue proporcionado por M. Karzand, D. J. Leith, J. Cloud y M. Medard, "Low Delay Random Linear Coding Over A Stream", versión preliminar de arXiv arXiv:1509.00167, 2015. Se estudió una posible implementación de SATCOM en J. Cloud y M. Medard, "Network coding over satcom: Lessons learned", en la Conferencia Internacional sobre Sistemas Inalámbricos y Satelitales. Springer, 2015, páginas 272-285 y para múltiples trayectos combinados en J. Cloud y M. Medard, "Multi-path low delay network codes", preimpresión de arXiv arXiv: 1609.00424, 2016. Estos enfoques de ventana deslizante muestran propiedades de retardo superiores para aplicaciones de difusión en forma continua en comparación con códigos basados en generación. Sin embargo, tales enfoques de la técnica anterior generalmente suponen una ventana deslizante infinita, o una ventana deslizante de tamaño dinámico, que se cierra solo por realimentación.

Algunas técnicas de la técnica anterior (por ejemplo, consulte V. Roca, B. Teibi, C. Burdinat, T. Tran y C. Thienot, "Block or Convolutional AL-FEC codes? a performance comparison for robust low-latency communications", 2016) sugieren el uso de códigos convolucionales sobre códigos de bloque para aplicaciones sensibles al retardo, en base a una comparación de Reed-Solomon y un código lineal aleatorio (RLC) de ventana deslizante finita. Sin embargo, tales técnicas no describen cómo se puede implementar de manera eficiente tal proceso de codificación y no consideran la realimentación.

TCP/NC se describe en J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher y J. Barros, "Network coding meets TCP", en INFOCOM 2009, IEEE. IEEE, 2009, páginas 280-288. Esta técnica añade una capa intermedia de codificación de red entre TCP e IP, incluyendo realimentación en el proceso y utiliza una forma general de ventana deslizante donde los paquetes se combinan de una manera sin restricciones y la codificación no está limitada por estructuras tales como bloques y generaciones. En tales formas de ventana deslizante, la ventana de codificación (los paquetes que actualmente se almacenan temporalmente para la codificación potencial en el codificador) solo se cierra mediante señales de acuse de recibo (ACK), y posiblemente puede ser muy grande - al menos en el rango del producto de retardo de ancho de banda, ya que la codificación siempre se realiza en toda la ventana de congestión (es decir, los paquetes que actualmente se almacenan temporalmente para una potencial retransmisión en el remitente).

Otra técnica descrita en Karafillis et al. mejora la técnica de TCP/NC anterior enviando paquetes redundantes adicionales - véase P. Karafillis, K. Fouli, A. ParandehGheibi y M. Medard, "An algorithm for improving sliding window network coding in TCP", en la 47ª Conferencia Anual sobre Ciencias y Sistemas de la Información (CISS), 2013. IEEE, 2013, páginas 1-5. Con la realimentación, el receptor señala el número de grados de libertad perdidos. El remitente utiliza esta información para enviar paquetes adicionales a intervalos específicos, por ejemplo, cada 13 paquetes.

Kim et al. describe TCP Codificado ("CTCP") véase M. Kim, J. Cloud, A. ParandehGheibi, L. Urbina, K. Fouli, D. Leith y M. Medard, "Network Coded TCP (CTCP)", preimpresión de arXiv arXiv:62.2291, 2012. La técnica de CTCP es una integración de codificación de red dentro de TCP. Este enfoque utiliza números de secuencia para paquetes y un código sistemático. La realimentación se envía solo al final de cada bloque. El tamaño del bloque se puede establecer en el producto del retardo del ancho de banda, lo que puede dar como resultado generaciones muy grandes.

Los códigos de bloque tienen baja complejidad pero introducen retardos, ya que el decodificador necesita esperar a que llegue un número suficiente de paquetes para decodificar cada bloque. Las técnicas de ventana deslizante convencionales, por otro lado, ofrecen baja latencia dado que pueden recibir paquetes y decodificar

simultáneamente. Esto tiene el precio de una complejidad potencialmente alta de las operaciones de sobrecarga, realimentación y decodificación que involucran potencialmente muchos paquetes. Esta invención combina las ventajas de los métodos tanto de bloque (baja complejidad) como de ventana deslizante (baja latencia). Esto se hace limitando estrictamente el número de paquetes en la ventana deslizante, utilizando codificación sistemática y adaptando la arquitectura de codificación de ventana deslizante (por ejemplo, formato de paquete, matriz de coeficientes) para reducir el número de operaciones lineales requeridas en las operaciones de codificación (es decir, menor complejidad). El resultado es un código de ventana deslizante (baja latencia en comparación con los códigos de bloque), pero con un tamaño de ventana estático que permite la implementación utilizando las herramientas de codificadores/decodificadores de bloques existentes. Esto conduce a propiedades de pérdida de paquetes similares y un rendimiento muy alto de los codificadores y decodificadores, ya que los codificadores/decodificadores de bloques existentes ya se han optimizado para el rendimiento en diversas arquitecturas de CPU. Otra ventaja de la ventana deslizante limitada es reducir las necesidades de almacenamiento temporal en todos los nodos de codificación.

Compendio

En contraste con las técnicas de la técnica anterior, se describe en la presente memoria un enfoque de ventana deslizante finita de tamaño fijo que se puede implementar de manera eficiente. Los sistemas y técnicas descritos en la presente memoria combinan la baja latencia de los códigos de ventana deslizante con las características de eficiencia y fiabilidad de los códigos basados en generación. Las ventanas deslizantes reducen la latencia de los paquetes en comparación con los códigos de bloque, pero son potencialmente más complejas (véase el comentario anterior). Las ganancias de latencia provienen del hecho de que los decodificadores de ventana deslizante pueden decodificar mientras que reciben datos codificados, mientras que los decodificadores de bloque no pueden realizar esas operaciones simultáneamente. Esta técnica permite el uso de una ventana deslizante mientras que se controla la complejidad. Esta técnica también mantiene las ventajas de ajuste y flexibilidad de los métodos de ventana deslizante basados en codificación lineal convencional, por lo que parámetros tales como los tamaños de ventana de codificación/decodificación o la tasa de código se pueden modificar dinámicamente. Esto permite múltiples compromisos (por ejemplo, retardo frente a fiabilidad/rendimiento/complejidad).

Además, en contraste con las técnicas de la técnica anterior, los sistemas y técnicas descritos en la presente memoria utilizan un código sistemático dentro del diseño de ventana finita. Los códigos sistemáticos envían datos no codificados primero, reduciendo la complejidad de la decodificación y la recodificación en comparación con los sistemas de codificación no sistemáticos.

Además, en contraste con las técnicas de la técnica anterior, los sistemas y técnicas descritos en la presente memoria pueden utilizar una ventana de código de borrado directo (es decir, una "ventana de FEC"). La ventana de FEC restringe aún más el número de símbolos que se pueden codificar y recodificar. Esto reduce el número de operaciones lineales requeridas para recodificar o decodificar símbolos, reduciendo por tanto aún más la complejidad de cálculo de las operaciones y permite una recodificación más fácil. El uso de una ventana de FEC que es más pequeña que la ventana de codificación permite el uso de esquemas de realimentación para volver a incluir paquetes perdidos de la ventana de codificación en los paquetes codificados de salida.

Las características de rendimiento de los sistemas y técnicas descritos se pueden mejorar en relación con las técnicas de la técnica anterior. Por ejemplo, los sistemas y técnicas descritos se pueden mejorar en relación con las técnicas de la técnica anterior en términos de retardo, dado que de acuerdo con los conceptos descritos en la presente memoria, los paquetes redundantes se pueden enviar a menudo en lugar de solo al final de (posiblemente grandes) generaciones como se hace en técnicas de la técnica anterior.

Además, los sistemas y técnicas descritos podrían reemplazar el enfoque codificado por bloques en TCP codificado ("CTCP") y, por tanto, ser incorporados a CTCP. Si bien proporcionar fiabilidad de entrega absoluta puede ser costoso, a diferencia de los enfoques de codificación de ventana deslizante tradicionales (por ejemplo, en CTCP), los conceptos descritos se pueden usar para proporcionar baja latencia y baja complejidad.

También se debería apreciarse que el enfoque descrito en la presente memoria también se podría implementar en una Capa de Enlace, asumiendo otro protocolo como TCP que proporciona fiabilidad en la parte superior. En tal caso, la técnica descrita en la presente memoria ayudaría a reducir las pérdidas observadas por TCP y mejoraría el retardo en comparación con otros esquemas de solicitud de repetición automática (ARQ).

Con esta disposición, se proporciona un código de ventana deslizante sin generación. Diversas realizaciones proporcionan, entre otras ventajas: un diseño modular que permite una fácil integración con bibliotecas y aplicaciones de software comercial mientras que se mantiene el alto rendimiento de los códigos de bloque de biblioteca de software comercial; hacer frente a diversos requisitos de red; simplificar o eliminar el manejo de múltiples generaciones de paquetes; proporcionar bajo retardo y alta fiabilidad para las capas de protocolo superiores; y proporcionar compromisos entre retardo, eficiencia y fiabilidad.

Por lo tanto, una primera realización de la invención se define en el método de decodificación de la reivindicación 1 adjunta. Se definen detalles adicionales en las reivindicaciones 2-12 adjuntas.

El decodificador puede proporcionar además realimentación a un codificador a través de un canal de datos, permitiendo así que el codificador transmita, a través del canal de datos, uno o más paquetes recodificando cualquier dato que el decodificador no decodificó debido a un borrado de paquetes.

Otra realización de la invención se define en la reivindicación 12 adjunta.

Otra realización más de la invención es un decodificador como se define en la reivindicación 13 adjunta.

Las personas con experiencia ordinaria en la técnica pueden reconocer otras formas de incorporar los conceptos, sistemas y técnicas descritos en la presente memoria.

Breve descripción de los dibujos

La manera y el proceso de fabricación y uso de las realizaciones descritas se pueden apreciar con referencia a los dibujos, en los que:

la Fig. 1 es un diagrama que ilustra un sistema ejemplar en el que se pueden incorporar los principios en la presente memoria;

la Fig. 2 es un diagrama que ilustra los paquetes codificados tal como se podrían transmitir por un codificador;

la Fig. 3 es un diagrama de flujo para la operación de un codificador de acuerdo con una realización;

la Fig. 4 es un diagrama que ilustra los componentes funcionales de un codificador según una realización de los conceptos, sistemas y técnicas descritos en la presente memoria;

las Figs. 5 y 5A son un diagrama de flujo para la operación de un decodificador de acuerdo con una realización;

las Figs. 6, 6A y 6B son una serie de diagramas que ilustran tres casos diferentes para insertar coeficientes con relación a un nuevo símbolo codificado en una matriz de coeficientes;

las Figs. 7 y 7A son diagramas que ilustran diferentes estrategias de eliminación gaussiana;

la Fig. 8 es un diagrama que ilustra los componentes funcionales de un decodificador según una realización de los conceptos, sistemas y técnicas descritos en la presente memoria;

la Fig. 9 es un diagrama de flujo para la operación de un recodificador de acuerdo con una realización;

la Fig. 10 es un diagrama que ilustra los componentes funcionales de un recodificador según una realización de los conceptos, sistemas y técnicas descritos en la presente memoria;

las Figs. 11, 11A y 11B son una serie de diagramas que ilustran ejemplos de esquemas de codificación comparados que tienen una tasa de código R = 2/3;

la Fig. 12 es un diagrama que ilustra una S-ARQ generalizada;

la Fig. 13 es un diagrama que ilustra una propuesta de S-ARQ de ventana deslizante; y

las Figs. 14, 14A, 14B y 14C son una serie de diagramas que ilustran una matriz de coeficientes para diferentes esquemas de retransmisión.

Descripción detallada

Los conceptos, sistemas y técnicas descritos en la presente memoria se pueden usar para realizar la codificación de borrado entre codificadores usando una ventana deslizante finita. Antes de describir estos conceptos, sistemas y técnicas en detalle, se explican conceptos introductorios y la terminología que se usa en la presente memoria.

Un "símbolo" es un vector de elementos en el campo finito GF(q). En general q es una potencia prima; para la comunicación de datos binarios, q es una potencia de 2.

Un "número de secuencia" es un número que identifica de manera única un símbolo de origen en una secuencia de símbolos de origen.

Una "ventana deslizante" es una secuencia de símbolos consecutivos que tienen un tamaño de ventana fijo. Una "ventana deslizante del codificador" tiene un tamaño de ventana fijo indicado como w_e, mientras que una "ventana deslizante de decodificador" tiene un tamaño de ventana fijo indicado como w_d. Los tamaños de las ventanas deslizantes en el codificador w_e y en el decodificador w_d pueden ser diferentes.

Un "símbolo codificado" es un vector de elementos en GF(q) que es una combinación lineal de como máximo w_e símbolos de entrada, donde cada coeficiente de la combinación lineal es un elemento en GF(q).

Un "paquete no codificado" es una unidad de datos para ser intercambiada entre dos codificadores que incluye un número de secuencia y un símbolo no codificado.

Un "paquete codificado" es una unidad de datos para ser intercambiada entre dos codificadores que incluye un número de secuencia, un vector de coeficientes de longitud w_e, y un símbolo codificado, codificado como una combinación lineal utilizando los coeficientes incluidos en el vector de coeficientes.

Un "borrado" es un elemento que falta en una secuencia.

"Codificación de borrado" significa codificar uno o más símbolos de entrada en una secuencia de paquetes que puede incluir borrados.

Con referencia a la Fig. 1, un sistema incluye varios codificadores 2a a 2N conectados por una red 9. Los codificadores se pueden implementar como ordenadores, o usando hardware, microprograma, software de propósito especial o una combinación de estos. Por tanto, por ejemplo, el codificador 2a se puede implementar como un circuito integrado de aplicaciones específicas (ASIC) o una agrupación de puertas programables en campo (FPGA) o tecnología similar. Los codificadores pueden estar presentes en cualquier nodo en la red y no están restringidos al borde de la red. Cada codificador 2a a 2N puede estar hecho de componentes idénticos o diferentes, y cualquier número de tales codificadores puede estar presente en un sistema de acuerdo con una realización de los conceptos, sistemas y técnicas descritos en la presente memoria.

Los codificadores 2a a 2N se comunican unos con otros utilizando paquetes codificados y no codificados. La manipulación de elementos en GF(q) además de tal comunicación se puede realizar utilizando cualquier hardware o software conocido en la técnica. Por ejemplo, la biblioteca Fifi publicada por Steinwurf ApS de Dinamarca se puede usar con este propósito.

Cada codificador 2a a 2N se puede usar para codificar paquetes, decodificar paquetes, recodificar paquetes o cualquier combinación de estos. El caso más general es que un codificador realiza todas de estas tres funciones. Por lo tanto, la operación del codificador ejemplar 2a se describe a continuación en detalle. Sin embargo, varias realizaciones pueden incluir un codificador (por ejemplo, el codificador 2c) que solo transmite paquetes y, por tanto, solo requiere un codificador (por ejemplo, el codificador 115c, omitiendo el decodificador 1104c y el recodificador 1106c). Otras realizaciones pueden incluir un codificador (por ejemplo, el codificador 2c) que solo recibe paquetes y, por tanto, solo requiere un decodificador (por ejemplo, el decodificador 1104c, omitiendo el codificador 115c y el recodificador 1106c). Aún otras realizaciones pueden incluir un codificador (por ejemplo, el codificador 2c) que solo reenvía paquetes y, por tanto, requiere solo un recodificador (por ejemplo, el recodificador 1106c, omitiendo el codificador 115c y el decodificador 1104c).

El codificador 2a tiene un codificador 4a, que toma como entrada una secuencia de símbolos de origen y genera como salida una secuencia correspondiente de paquetes codificados o no codificados. Por tanto, el codificador 2a es adecuado para transmitir información como origen de canal. La Fig. 2 es un diagrama que ilustra los paquetes codificados como se podrían transmitir por el codificador 4a. La operación del codificador 4a se ilustra en detalle en la Fig. 3. Una realización del codificador 4a se ilustra en detalle en la Fig. 4.

El codificador 2a también tiene un decodificador 6a, que toma como entrada una secuencia de paquetes codificados o no codificados y genera como salida una secuencia correspondiente de símbolos de origen, opcionalmente en orden. Por tanto, el codificador 2a es adecuado para recibir información como sumidero de canal. La operación del decodificador 6a se ilustra en detalle en la Fig. 5 con referencia a las Figs. 6 y 7. Una realización del decodificador 6a se ilustra en detalle en la Fig. 8.

El codificador 2a también tiene un recodificador 8a, que toma como entrada una secuencia de paquetes codificados o no codificados y genera como salida una secuencia correspondiente de paquetes codificados y no codificados, donde los paquetes codificados de salida pueden ser paquetes recodificados; es decir, paquetes codificados cuya codificación es diferente de los paquetes codificados de entrada. Por tanto, el codificador 2a es adecuado para reenviar información desde un canal de origen a un canal sumidero utilizando, por ejemplo, un código de red lineal. La operación del recodificador 8a se ilustra en detalle en la Fig. 9. Una realización del recodificador 8a se ilustra en detalle en la Fig. 10.

Haciendo referencia ahora a la Fig. 2, los detalles de una realización se ilustran con referencia a algunos paquetes codificados 20a - 20e como se podrían haber enviado por un codificador (tal como el codificador 4a). Cada paquete codificado incluye una carga útil 12, tal como un símbolo codificado, y una cabecera de codificación 14, 16, 18. La cabecera de codificación incluye un número de secuencia de paquete ("PSN") 14. El PSN 14 representa el número de secuencia más grande de un símbolo de entrada utilizado en la codificación del símbolo codificado como una combinación lineal de símbolos de entrada. Por tanto, según una secuencia de transmisión, el primer PSN 14a puede ser 0, el segundo PSN 14b puede ser 1, y así sucesivamente. Sin embargo, de acuerdo con varias realizaciones, el decodificador puede recibir los paquetes codificados fuera de orden, o el codificador puede transmitir paquetes codificados que codifican de manera redundante los símbolos de entrada utilizando el mismo PSN (por ejemplo, a través de un canal de datos sujeto a muchos borrados).

En algunas realizaciones, la cabecera de codificación incluye opcionalmente un vector de coeficiente 16, 18. Es decir, en algunas realizaciones, cuando se transmite un paquete no codificado, el vector de coeficiente 16, 18 se omite o se reemplaza por otros datos para proporcionar un uso más eficiente del canal de datos entre el codificador y el decodificador. Alternativamente, el vector de coeficientes 16, 18 se puede transmitir pero contiene una única entrada distinta de cero igual a 1 en GF(q). Independientemente, cuando se transmite el vector de coeficientes, tiene un tamaño fijo predeterminado igual a la longitud de la ventana w_e. El vector de coeficientes se puede usar para recuperar símbolos de entrada en presencia de borrados de paquetes, como se describe a continuación.

De acuerdo con realizaciones, los índices del vector de coeficientes se interpretan de una forma especial por el decodificador. Se utiliza un índice de decodificador para realizar un seguimiento de qué números de secuencia están dentro de la ventana deslizante y cuáles no. Para ayudar a comprender este concepto, la Fig. 2 muestra el índice de decodificador y divide el vector de coeficientes en un componente 16 más antiguo y un componente 18 más nuevo. El índice de decodificador se mueve de izquierda a derecha a través del vector de coeficientes, por lo que el índice de coeficientes asociado con el número de secuencia más antiguo (es decir, el menor) en la ventana deslizante es 16N, y los índices de coeficiente 16a a 16e se refieren a los números de secuencia más nuevos de izquierda a derecha. El siguiente número de secuencia más nuevo está asociado con el índice de coeficiente 18a, y estos llegan a ser aún más nuevos desde 18b-18k, siendo este último el índice de decodificador.

En varias realizaciones, el índice de decodificador rastrea el símbolo de origen con el mayor número de secuencia recibido por el decodificador en un paquete de datos. En este caso, el índice de decodificador se puede calcular como este número de secuencia más grande módulo el tamaño de ventana fijo w_e del codificador. Como se señaló anteriormente, los otros índices representan símbolos más antiguos. Esto significa que un índice se reutiliza cada w_e símbolos de origen, y se debe tener especial cuidado para manejar la reutilización correctamente.

Los últimos índices de coeficientes presentes en el componente 18 más nuevo del vector de coeficientes se pueden dividir además para reconocer una ventana de corrección de errores sin canal de retorno ("FEC") 22 que tiene un ancho "f". La ventana de FEC 22 comprende el último símbolo transmitido y f - 1 símbolos antes de eso. Estos símbolos se utilizan para la codificación directa activa, por ejemplo, codificación oportunista incluso sin realimentación. Si la ventana de FEC 22 comprende 4 símbolos, entonces los símbolos asociados a los coeficientes 18g-18k se pueden usar para realizar la FEC. La ventana de recuperación de realimentación 24, 26 contiene coeficientes asociados a símbolos más antiguos, que solo se deberían usar cuando se requiera, por ejemplo, después de que el decodificador detecte que uno o más de estos símbolos faltan o no se pueden decodificar.

Es ventajoso dividir la recuperación en una ventana de FEC 22 y una ventana de recuperación de realimentación 24, 26. Hacerlo así mantiene la ventana de FEC lo suficientemente pequeña para manejar las ráfagas esperadas típicas de paquetes transmitidos (por ejemplo, 16 paquetes), mientras que se mantiene la ventana de recuperación de realimentación 24, 26 lo suficientemente grande como para ser capaz de recuperar los paquetes transmitidos durante al menos un tiempo de ida y vuelta ("RTT"), incluyendo todo el almacenamiento temporal (por ejemplo, 128 - 256 paquetes). Por tanto, el codificador y el decodificador pueden negociar el tamaño de la ventana deslizante w_e intercambiando datos para medir el RTT, predeterminando así un tamaño fijo para w_e antes de que el codificador transmita el primer paquete codificado. A continuación se describe un método de decodificación que utiliza la ventana de FEC 22 y se contrasta con la técnica anterior en conexión con las Figs. 11-14C.

Una aplicación particularmente útil de estos conceptos ocurre cuando la red 9 comprende una red en malla u otra red para la cual existen múltiples trayectos o múltiples canales entre los codificadores 2a a 2N. El concepto general es usar una ventana deslizante, con o sin realimentación, en múltiples canales o trayectos disjuntos, por ejemplo, LTE + WiFi. Para aplicar multitrayecto, es importante incluir disposiciones que consideren diferentes latencias de paquetes, entrega fuera de orden, tamaño de ventana deslizante, etc.

En una operación multitrayecto, el codificador o cualquier recodificador puede decidir enviar paquetes a través de más de un trayecto al decodificador de destino o a un nodo intermedio. El nodo que inicia la operación multitrayecto, típicamente el nodo de origen que contiene el codificador, se denomina en lo sucesivo "nodo multitrayecto". La decisión sobre qué paquetes transmitir sobre qué trayecto se puede basar en la información que el nodo multitrayecto ha recibido o inferido en los diversos trayectos. Algunos mecanismos de multitrayecto incluyen, pero no se limitan a, los siguientes.

Los nodos multitrayecto pueden asignar paquetes a trayectos en base a la pérdida de trayecto o la determinación de carga de trayecto. Tal determinación se puede basar en la información de estado comunicada por otro nodo (por ejemplo, destino, nodos aguas abajo, gestión de red) o inferida por el nodo multitrayecto en sí mismo a partir del tráfico de origen-destino (por ejemplo, monitorización de realimentación y deducción de paquetes perdidos en cada trayecto) o alguna otra métrica disponible (por ejemplo, CSI).

El nodo multitrayecto puede enviar paquetes sistemáticos (es decir, de origen, no codificados) a través del trayecto más rápido o más fiable para disminuir la complejidad de la decodificación y acelerar la conexión, ya que esto asegura que más paquetes de origen alcancen el decodificador en un tiempo más corto, requiriendo menos operaciones lineales para decodificar y permitir una entrega más rápida de paquetes de origen.

El nodo multitrayecto puede usar información inferida (por ejemplo, CSI) o recibida sobre la latencia del trayecto (por ejemplo, realimentación) para abstenerse de usar los trayectos cuando se considere que su latencia asociada hace que los paquetes sean inútiles tras la recepción por el destino (es decir, quedan fuera la ventana de decodificación tras la recepción).

Los nodos intermedios en diferentes trayectos pueden ser selectivos en sus transmisiones de paquetes recibidos, en sus retransmisiones de paquetes recibidos, o en su transmisión de paquetes codificados/recodificados. Se puede aplicar un proceso de muestreo de paquetes (es decir, enviar un paquete por cada M paquetes recibidos), implementando una tasa de muestreo de destino (es decir, 1/M aquí).

Esta tasa de muestreo de paquetes se puede comunicar por otro nodo (por ejemplo, de origen, destino, gestión de red) o inferir por el nodo multitrayecto en sí mismo a partir de su posición en la transferencia (por ejemplo, los nodos más cercanos al destino envían a una tasa más baja), del tráfico de origen-destino, de la realimentación de los nodos vecinos o de alguna otra métrica disponible.

Las Figs. 3, 5, 5A y 11 son diagramas de flujo que ilustran el procesamiento que se puede implementar dentro del sistema 2 (FIG. 1). Los elementos rectangulares (tipificados por el elemento 30 en la FIG. 3), algunas veces se denominan en la presente memoria "bloques de procesamiento", representan instrucciones de software informático o grupos de instrucciones. Los elementos en forma de diamante (tipificados por el elemento 40 en la FIG. 3), algunas veces se denominan en la presente memoria "bloques de decisión" y representan instrucciones de software de ordenador, o grupos de instrucciones, que afectan a la ejecución de las instrucciones de software de ordenador representadas por los bloques de procesamiento.

Alternativamente, los bloques de procesamiento y decisión pueden representar funciones realizadas por circuitos funcionalmente equivalentes tales como un circuito procesador de señal digital o un circuito integrado de aplicaciones específicas (ASIC). Los diagramas de flujo no representan la sintaxis de ningún lenguaje de programación en particular. Más bien, los diagramas de flujo ilustran la información funcional que requiere un experto en la técnica para fabricar circuitos o generar software informático para realizar el procesamiento requerido de un aparato o dispositivo en particular. Se debería señalar que no se muestran muchos elementos del programa de rutina, tales como la inicialización de bucles y variables y el uso de variables temporales. Se apreciará por los expertos en la técnica que, a menos que se indique de otro modo, la secuencia particular de bloques descrita en los diagramas de flujo es solo ilustrativa y se puede variar sin apartarse de los conceptos, estructuras y técnicas descritas. Por lo tanto, a menos que se indique de otro modo, los bloques que se describen a continuación no están ordenados, lo que significa que, cuando sea posible, las funciones representadas por los bloques se pueden realizar en cualquier orden conveniente o deseable.

Volviendo ahora a la Fig. 3, un método de codificación de acuerdo con una realización de codificador (por ejemplo, el codificador mostrado en la Fig. 1) comienza con un proceso de inicialización 30. Las realizaciones ilustrativas utilizan este proceso de inicialización 30 para restablecer un número de secuencia de codificador a cero, por ejemplo, o para intercambiar datos con un decodificador correspondiente para determinar un tiempo de ida y vuelta ("RTT") y, por lo tanto, predeterminar un tamaño de ventana fijo w_e. Un experto en la técnica apreciará otras inicializaciones que se pueden realizar y los métodos convencionales para hacerlo así.

El método de codificación continúa con un proceso de recepción 32 que inicia el bucle de codificación principal. En este proceso de recepción 32, el codificador recibe una nueva entrada o símbolo de origen. Por ejemplo, el símbolo de origen se puede recibir como datos de una red informática o de un programa de aplicación utilizando una interfaz de programación de aplicaciones (API) o utilizando otros medios convencionales.

Entonces, el proceso de aumento 34 aumenta el número de secuencia de codificador, y el codificador asocia el símbolo de origen recién recibido con el nuevo número de secuencia. La asociación puede incluir almacenar el símbolo de origen en un almacenador temporal de entrada en una ubicación de memoria según el número de secuencia, como se describe a continuación con más detalle. El proceso de aumento 34 permite que el codificador genere símbolos codificados.

A continuación, un proceso de generación de paquetes 36 genera un paquete no codificado adjuntando una cabecera que contiene el número de secuencia de codificador actual. En una realización, la mayoría de los paquetes transmitidos por el codificador no están codificados; es decir, carecen de un vector de coeficientes porque incluyen un símbolo no codificado. Esta realización permite que tasas de código relativamente altas sean logradas. Sin embargo, en otra realización, incluso los paquetes no codificados incluyen una cabecera de codificación que tiene un vector de coeficientes. En esta realización, el vector de coeficientes tiene una longitud igual al tamaño de ventana deslizante finita w_e. Debido a que el símbolo no está codificado, el vector de coeficientes contiene exactamente un elemento distinto de cero del campo finito GF(q), por ejemplo, el valor 1. Ilustrativamente, el elemento distinto de cero se sitúa en la posición (ESN mod w_e), donde ESN es el valor del número de secuencia de codificador. Se pueden usar medios convencionales que incluyen un almacenador temporal u otra memoria para generar el paquete no codificado.

El método continúa con un proceso de puesta en cola 38 que coloca el nuevo paquete no codificado así formado en

un almacenador temporal de salida. En realizaciones ilustrativas, el método de la Fig. 3 se realiza de manera asíncrona; es decir, los nuevos paquetes se generan tan rápido como se reciben los símbolos de origen, en lugar de esperar a que los paquetes existentes se entreguen al canal de datos. Por tanto, el proceso de puesta en cola 36 transfiere el nuevo paquete no codificado a un almacenador temporal de salida compartido accesible a otro proceso para su eventual entrega a la red. Por supuesto, una realización alternativa puede operar de una manera síncrona, esperando hasta que se confirme que cada paquete ha sido transmitido hacia el decodificador antes de continuar. Tales transferencias asíncronas se pueden realizar usando medios convencionales.

De acuerdo con la realización ilustrada en la Fig. 3, el método continúa con un proceso de comprobación de tasa de código 40 que comprueba si se requiere o desea la transmisión de uno o más paquetes codificados. El proceso de comprobación de tasa de código 40 se puede realizar rastreando una relación de paquetes no codificados con respecto al total generado (es decir, tasa de código) para asegurar que se alcance una tasa de código dada. En una realización, el proceso de comprobación de tasa de código 40 devuelve un resultado SÍ cuando el número de secuencia de codificador actual es un múltiplo de un número N de paquetes no codificados a ser enviados para cada paquete codificado, y NO de otro modo. En este caso, la tasa de código se calcula como que es N / N+1. Por supuesto, debería quedar claro cómo se podrían lograr otras tasas de código utilizando diferentes pruebas.

Si no se requiere la transmisión de un paquete codificado, el bucle principal vuelve a su estado inicial, esperando otro símbolo de origen en el proceso de recepción 32. Sin embargo, si se requiere la transmisión de un paquete codificado, el método continúa con un proceso de generación de paquetes codificados 42 que genera un paquete codificado. En realizaciones ilustrativas, el codificador usa o bien codificación sistemática (es decir, incluye el último símbolo de origen en un símbolo codificado generado) o bien envía paquetes redundantes desde dentro de la ventana de FEC ilustrada en la Fig. 2. El proceso de generación de paquetes codificados 42 por lo tanto incluye generar un vector de coeficientes que tiene un coeficiente en GF(q) por símbolo de origen en la ventana deslizante a ser codificada.

Dado que el codificador utiliza una ventana finita, el número de paquetes combinados linealmente normalmente es igual a w_e, excepto en las etapas iniciales de la transmisión cuando hay menos símbolos de origen en la ventana finita (es decir, cuando el número de secuencia de codificador es menor que w_e). El proceso de generación de paquetes codificados 42 luego combina linealmente los símbolos de origen en la ventana deslizante finita usando los coeficientes generados y forma el paquete codificado añadiendo una cabecera de codificación al símbolo codificado.

La cabecera de codificación contiene un número fijo w_e de coeficientes y un número de secuencia de paquete (que puede diferir del número de secuencia de codificador). En una realización, la ubicación del coeficiente se reutiliza de una manera que hace implícito el símbolo de origen S asociado con cada coeficiente. Esto se hace colocando el coeficiente asociado al símbolo de origen S en la posición (ESN(S) mod w_e), donde ESN(S) es el número de secuencia de codificador asociado con el símbolo de origen S. Cualquier intervalo de coeficientes vacío obtiene un coeficiente cero. Los vectores de coeficientes así formados llegan a ser las filas de la matriz de coeficientes (R) en el decodificador, donde la matriz de coeficientes (R) es la matriz utilizada por el decodificador para realizar la decodificación, como se detalla a continuación.

Una vez que se ha generado el paquete codificado, un proceso de cola 44 lo coloca en el almacenador temporal de salida. Este proceso de puesta en cola 44 es como el proceso de puesta en cola 38 y puede funcionar de manera similar. El método continúa determinando si transmitir un paquete codificado adicional en el proceso de comprobación de tasa de código 40. Si no se requiere ningún paquete codificado adicional, entonces el método concluye su bucle regresando al proceso de recepción 32, en el que espera el siguiente símbolo de origen.

Haciendo referencia ahora a la Fig. 4, una realización de los conceptos descritos en la presente memoria es un codificador 50 que tiene varios componentes funcionales que incluyen un módulo de control de ventana finita 52, un almacenador temporal de símbolos 54, un registro de número de secuencia de codificador 56, un codificador de codificación de red lineal aleatoria ("RLNC") 58, un generador de paquetes 60 y un almacenador temporal de salida 62. Ahora se describe la operación de estos componentes.

El módulo de control de ventana finita 52 es el mecanismo principal para controlar el codificador 50. El módulo de control de ventana finita 52 recibe símbolos de origen y los almacena en un almacenador temporal de símbolos 54. El almacenador temporal de símbolos 120 contiene w_e símbolos, donde w_e es el tamaño de la ventana deslizante de codificador finito, y el módulo de control de ventana finita 52 desaloja los símbolos obsoletos cuando pasan fuera de la ventana. Esto se hace para respetar el tamaño de la ventana de codificación y controlar la complejidad de la decodificación. Se pueden recibir realimentación de un decodificador que fue incapaz de decodificar un símbolo a pesar de usar las técnicas descritas en la presente memoria. Cuando se recibe realimentación, generalmente se refiere a un símbolo que tiene un número de secuencia que es menor que el guardado por el codificador. En este caso, la información de realimentación de los símbolos fuera del tamaño de la ventana del codificador se descarta. El módulo de control de ventana finita 52 también almacena y actualiza el número de secuencia de codificador en un registro de número de secuencia de codificador 56, como se describe en conexión con la Fig. 3.

El codificador 50 también incluye un codificador de RLNC 58 para calcular los símbolos codificados encontrados en un paquete codificado, por ejemplo, como se describió anteriormente en el proceso 42. Por ejemplo, la biblioteca

Kodo publicada por Steinwurf ApS de Dinamarca se puede usar con este propósito bajo la dirección del módulo de control de ventana finita 52.

Un generador de paquetes 60 crea paquetes codificados y no codificados para su transmisión a un decodificador o recodificador. Para paquetes no codificados, el generador de paquetes 60 une un símbolo no codificado obtenido del módulo de control de ventana finita 52 al número de secuencia asociado almacenado en el registro de número de secuencia de codificador 56. Para paquetes codificados, el generador de paquetes 60 une un símbolo codificado recibido del codificador de RLNC 58 a una cabecera de codificación. La cabecera de codificación incluye el vector de coeficientes utilizado para codificar el símbolo codificado, obtenido del codificador de RLNC 58. La cabecera de codificación también incluye un número de secuencia de paquete, es decir, el número de secuencia más grande de cualquier símbolo de entrada utilizado en la codificación del símbolo codificado como una combinación lineal de símbolos de entrada.

Finalmente, el codificador 50 incluye un almacenador temporal de salida 62 que realiza funciones de salida (es decir, transmisión, almacenamiento temporal, entrega a una capa de protocolo inferior, etc.). El almacenador temporal de salida 62 recibe paquetes generados desde el generador de paquetes 60. Se debería apreciar que cualquiera o todos los componentes funcionales en el codificador 50 se pueden implementar en hardware, microprograma, software o cualquier combinación de tales tecnologías.

Haciendo referencia ahora a la Fig. 5, un método de decodificación de acuerdo con una realización de decodificador (por ejemplo, el decodificador mostrado en la Fig. 1) comienza con un proceso de inicialización 91 que inicializa una matriz utilizada en la decodificación, como se describe con más detalle a continuación. Al igual que con el proceso de inicialización del codificador 30, el proceso de inicialización del decodificador 91 también puede inicializar variables de estado, incluido el establecimiento de un número de secuencia de decodificador en cero. Además, el decodificador puede intercambiar datos con un codificador correspondiente para determinar un tiempo de ida y vuelta ("RTT") y, por lo tanto, determinar el tamaño de ventana deslizante fijo w_e antes de intercambiar datos utilizando las técnicas de ventana deslizante finita descritas en la presente memoria. Un experto en la técnica puede apreciar otras inicializaciones que se pueden realizar y los métodos convencionales para hacerlo así.

La matriz utilizada en la decodificación tiene w_d filas que incluyen cada una un vector de coeficientes que indica los coeficientes utilizados para codificar un símbolo codificado diferente, donde w_d tiene un valor fijo y w_d es mayor o igual que w_e. Las primeras w_e columnas de tal matriz forman la matriz de coeficientes (R) y se puede reducir a una matriz de identidad realizando operaciones de fila (por ejemplo, multiplicando todos los coeficientes en una fila por el mismo número y sumando o restando filas). Como se sabe en la técnica, si se realizan operaciones de fila correspondientes en los símbolos codificados asociados, este proceso da como resultado símbolos de origen decodificados.

Algunas realizaciones llevan a cabo operaciones de fila usando una matriz de símbolos w_d por 1 y una matriz de coeficientes w_d por w_e separada (R) cuyas w_d filas son vectores de coeficientes (de longitud w_e) de los símbolos codificados correspondientes. Otras realizaciones coordinan operaciones de fila usando una matriz w_d por (w_e+1) “aumentada”, en la que las primeras w_e columnas forman la matriz de coeficientes (R) y comprenden los w_e vectores de coeficientes (para cada uno de los w_d símbolos), mientras que la última columna (índice w_e+1) comprende los símbolos asociados. Independientemente de su forma, se hace referencia a esta matriz en lo sucesivo como la "matriz de coordenadas".

El método de decodificación de la Fig. 5 continúa con un proceso de recepción 92 que inicia el bucle de decodificación principal. En este proceso de recepción 92 el decodificador recibe un nuevo paquete. Por ejemplo, el paquete se puede recibir como datos desde una red informática o desde un programa de aplicación utilizando una interfaz de programación de aplicaciones (API) o utilizando otros medios convencionales.

A continuación, el método de decodificación determina en los procesos de decisión 93, 95 y 96 una relación entre el paquete recibido y el estado de decodificación. El decodificador debe determinar, entre otras cosas, si un paquete recibido se refiere a símbolos que tienen números de secuencia que son más nuevos, más antiguos o iguales a los símbolos identificados recientemente. Realizaciones ilustrativas resuelven este problema incluyendo un número de secuencia de paquete ("PSN") en la cabecera de codificación de cada paquete codificado y manteniendo como parte del estado del decodificador un número de secuencia de decodificador ("DSN"). El DSN corresponde al PSN más grande recibido por el decodificador y se puede comparar con el PSN recibido más recientemente.

Por tanto, en el proceso de decisión de novedad 93, el decodificador determina si el PSN del paquete recibido es mayor que el DSN actualmente mantenido por el decodificador. Esto ocurre cuando el paquete recibido hace referencia a un símbolo de origen que anteriormente era desconocido por el decodificador. Si este es el caso, entonces el método pasa a un proceso de actualización 94 en el que el número de secuencia de decodificador se actualiza para reflejar que el paquete recibido tenía un número de secuencia más alto.

El proceso de actualización 94 realiza dos tareas importantes, que se pueden apreciar más completamente con referencia a la Fig. 6A, que ilustra un vector de coeficiente recibido 13 que tiene un índice de coeficiente 104 y la matriz de coeficiente 103 con índice de decodificador 106, donde $w_d = w_e = 8$. El índice de coeficiente 104 se calcula

como (PSN mod w_e) y el índice de decodificador 106 se calcula como (DSN mod w_e). Los recuadros sombreados o rayados en las Figs. 6, 6A y 6B representan coeficientes distintos de cero, y los recuadros en blanco o sin rayar en estas Figuras representan coeficientes cero.

En primer lugar, el proceso de actualización 94 actualiza el DSN para igualarlo al PSN recibido. El decodificador debe dar este paso porque el paquete recibido hace referencia a un símbolo de origen (que tiene el PSN recibido) que no se conocía previamente por el decodificador. En la Fig. 6A esto se muestra en la matriz de coeficientes actualizada 108, que ha movido el índice de decodificador 106 dos columnas a la derecha para que coincida con el índice de paquete 104.

Sin embargo, aumentar el DSN "desplaza" la ventana deslizante finita, que cubre un rango de números de secuencia que depende del DSN. Por tanto, la matriz de coeficientes ahora puede incluir filas cuyos coeficientes se refieren a símbolos de origen cuyos números de secuencia ahora se encuentran fuera de la ventana deslizante finita, por lo que la segunda tarea importante del proceso de actualización 94 es desalojar de la matriz de coeficientes cualquier fila de coeficientes que se refieran a símbolos obsoletos (es decir, filas con un coeficiente distinto de cero en una columna asociada con un símbolo de origen obsoleto). Eliminar estas filas puede reducir la probabilidad de decodificar el nuevo símbolo. Pero este efecto se puede limitar realizando una eliminación gaussiana en cada oportunidad, como se describe a continuación. La eliminación gaussiana reducirá el número de entradas distintas de cero y, por esto, la probabilidad de desalojar el símbolo.

Para ilustrar el desalojo, con referencia nuevamente a la Fig. 6A, el estado del decodificador antes de recibir el nuevo vector de coeficientes 13 era una matriz de coeficientes con tres filas de coeficientes y un índice de decodificador igual a 2 (por ejemplo, DSN = 138). La primera fila incluye coeficientes para los símbolos que tienen todos los ocho números de secuencia anteriores (por ejemplo, los números de secuencia 131 a 138, inclusive), por lo que su símbolo codificado correspondiente es una combinación lineal de todos los símbolos de origen respectivos. Sin embargo, la segunda y tercera filas incluyen coeficientes distintos de cero solo para los seis símbolos anteriores (por ejemplo, los números de secuencia 133 a 138, inclusive). Por tanto, después de que el decodificador recibió el nuevo vector de coeficientes 13 (incluido en un paquete que tiene PSN = 140), el DSN aumentó en dos (por ejemplo, a 140, moviendo la ventana deslizante a los números de secuencia 133 a 140, inclusive). Dado que la fila 41a depende de símbolos con números de secuencia 131 y 132, que ahora están fuera de la ventana deslizante, la fila 41a se desaloja de la matriz de coeficientes como se indica en la matriz de coeficientes actualizada 108. Sin embargo, dado que las filas 41b y 41c dependen solo de símbolos con números de secuencia 133 a 138, no se desalojan. Una vez que el decodificador ha actualizado su estado como se indica, el método pasa a la Fig. 5A, que ilustra una decodificación algebraica.

Volviendo al proceso de decisión de novedad 93, si el PSN del paquete recibido no es mayor que el DSN, entonces el método continúa con el proceso de decisión de igualdad 95, en el que el decodificador determina si el PSN del paquete codificado recibido es el mismo que el DSN. Si es así, entonces no es necesario actualizar el DSN o el índice de decodificador 15 y la decodificación algebraica puede comenzar inmediatamente, por lo que el método pasa a la Fig. 5A como anteriormente. Esta situación se ilustra en la Fig. 6.

Si el PSN del paquete recibido no es al menos tan grande como el DSN actualmente conocido por el decodificador, entonces el paquete puede hacer referencia a símbolos de origen obsoletos, por lo que el método pasa a un proceso de decisión de obsolescencia 96, en el que el decodificador determina si el vector de coeficientes recibido incluye coeficientes distintos de cero que se refieren a símbolos de origen cuyos números de secuencia se encuentran fuera de la ventana deslizante actual determinada por el DSN. Por tanto, el proceso de decisión de obsolescencia 96 es como la tarea de desalojo del proceso de actualización 94, excepto que el vector de coeficientes recibido no es (todavía) una fila en la matriz de coeficientes.

Si el paquete recibido incluye un símbolo codificado que es una combinación lineal de solo símbolos que todavía están en la ventana deslizante, como se indica por el vector de coeficientes recibido, entonces el método puede pasar a la Fig. 5A como se indica. De otro modo, en el proceso de descarte 97, el decodificador descarta el paquete codificado recibido como si fuera un borrado y vuelve al proceso de recepción 92 para esperar la llegada del siguiente paquete. Esta situación se ilustra en la Fig. 6B. Para completar, tenga en cuenta que mientras que el vector de coeficientes 110 recién recibido tiene un PSN 47 que es menor que el DSN 112 (como se muestra por el índice de paquete 47 que está a la izquierda del índice de decodificador 112), sin embargo el vector de coeficientes 110 tiene coeficientes distintos de cero solo para símbolos cuyos números de secuencia son como máximo seis números menores que su PSN, y esos seis números de secuencia están dentro de los ocho números de secuencia, no mayores que el DSN, que definen la ventana deslizante. Por lo tanto, el vector de coeficientes 110 no se descarta.

Haciendo referencia ahora a la Fig. 5A, un proceso de decodificación algebraica comienza con un proceso de suma 91A, en el que el vector de coeficientes de un paquete recibido se suma como una fila a la matriz de coeficientes R para formar una matriz de coeficientes temporal R', como se indica en las Figs. 6, 6A y 6B. La matriz de coeficientes R' con la nueva fila de coeficientes se puede almacenar en una memoria temporal para permitir que se realice el álgebra lineal sin perder el estado R del decodificador.

Si el paquete recibido es un paquete no codificado, puede carecer de un vector de coeficientes. En tal situación, el símbolo no codificado contenido en el mismo se puede representar como una combinación lineal de (cero veces cada uno de los otros w_e-1 símbolos) más (1 veces el índice de paquete no codificado), con un vector de coeficientes correspondiente. El proceso de suma 91A suma este vector de coeficientes a la matriz de coeficientes R para formar la matriz de coeficientes temporal R'. Este proceso de suma 91A se realiza incluso para paquetes no codificados, porque sus símbolos se pueden usar para decodificar otros símbolos codificados (por ejemplo, como se describe en conexión con las Fig. 11-14C). Además, como el paquete recibido contenía un símbolo no codificado, el método puede enviar una copia del símbolo no codificado inmediatamente, como se indica, al proceso de entrega 94A, que se discute a continuación.

El método continúa con un proceso de eliminación gaussiana (reducción de filas) 92A, en el que la matriz de coeficientes temporales R' se somete a una reducción de filas con pivote, en la medida que esa técnica se conoce en la técnica. Sin embargo, a diferencia de la técnica anterior 120 que comienza a pivotar desde la primera columna (como se muestra en la Fig. 7), una realización ilustrativa 124 comienza a pivotar desde la columna de índice de decodificador 122 correspondiente al DSN módulo el tamaño de la ventana w_e (como se muestra en la Fig. 7A). Así, el resultado del proceso de reducción de filas 92A es una matriz de coeficientes en forma escalonada de filas "desplazadas".

El método continúa con un proceso de determinación de decodificación 93A, en el que el decodificador determina si cualquier nuevo símbolo de origen se decodificó por el proceso de reducción de filas 92A. El resultado es SÍ cuando, como se muestra en la Fig. 7A, la matriz de coeficientes temporales reducidos por filas R' incluye una submatriz de identidad m por m, desplazada a la derecha de la columna de índice de decodificador, que es mayor que una submatriz de identidad n por n desplazada correspondiente en la matriz de coeficientes original R. En este caso, el proceso de reducción de filas 92A decodificó con éxito m-n símbolos de origen en orden de sus números de secuencia, y el método continúa con el proceso de entrega 94A. De otro modo, el vector de coeficientes recién recibido no decodificó símbolos de origen adicionales en orden, y el método continúa con el proceso de almacenamiento 95A.

Este algoritmo proporciona una ganancia de eficiencia sustancial sobre la técnica anterior. El escenario más probable para una realización que tenga una tasa de código alta es que un paquete recién recibido no esté codificado y tenga un PSN que sea exactamente uno más grande que el DSN actual. Esto implica que el proceso de actualización 94 moverá el índice de decodificador una columna a la derecha y desalojará la primera fila de coeficientes. Como se puede apreciar a partir de la Fig. 7A, realizar estas dos acciones conserva la forma escalonada de fila "desplazada" de la matriz de coeficientes. Además, si el paquete recibido no está codificado, entonces es probable que el símbolo no codificado contenido en el mismo tenga el número de secuencia más grande recibido por el decodificador y se pueda entregar inmediatamente. Por el contrario, realizar la eliminación gaussiana de la forma tradicional daría como resultado una matriz de coeficientes con una forma escalonada de filas "no desplazadas ", y tal eliminación tendría que ser realizada de nuevo después de cada paquete recién recibido.

En el proceso de entrega 94A, el decodificador entrega símbolos de origen recién decodificados a un consumidor de símbolos. Tal consumidor puede ser, por ejemplo, una aplicación de software, un almacenador temporal de símbolos o algún otro dispositivo, sistema o proceso. Finalmente, la decodificación algebraica concluye con un proceso de almacenamiento 95A, en el que la matriz de coeficientes reducidos de fila "desplazada" R' se almacena como la matriz de coeficientes R.

Haciendo referencia ahora a la Fig. 8, una realización de estos conceptos es un decodificador 160. El decodificador 160 incluye un módulo de control de ventana finita 162, un registro de número de secuencia 164, un almacenador temporal de codificación 166, un decodificador de código de red lineal aleatorio ("RLNC") 168, y un módulo de entrega 170. Ahora se describe la operación de estos componentes.

El módulo de control de ventana finita 162 es el mecanismo principal para controlar el decodificador 160. El módulo de control de ventana finita 162 recibe paquetes no codificados y codificados y mantiene un número de secuencia de decodificador (DSN) en un registro de número de secuencia de decodificador 164. En varias realizaciones, mantiene el DSN según los procesos descritos anteriormente en conexión con la Fig. 5 y especialmente el proceso de actualización 94.

El módulo de control de ventana finita 162 almacena los símbolos recibidos, ya sean no codificados o codificados, en un almacenador temporal de codificación 166. El almacenador temporal de codificación 166 contiene w_d símbolos, donde w_d es el tamaño de la ventana deslizante de decodificador, y el módulo de control de ventana finita 162 desaloja los símbolos obsoletos a medida que pasan fuera de la ventana. Esto puede suceder, por ejemplo, de acuerdo con los procesos descritos anteriormente en conexión con el proceso de actualización 94.

En algunas realizaciones, cuando el decodificador 160 es incapaz de decodificar un símbolo a pesar de usar las técnicas descritas en la presente memoria, puede proporcionar una realimentación adecuada a un codificador respectivo. Cuando se envía realimentación, se referirá a un símbolo no decodificado por número de secuencia.

El decodificador 160 también incluye un decodificador de RLNC 168 para calcular los símbolos de origen a partir de

los símbolos codificados encontrados en un paquete codificado, por ejemplo, como se describió anteriormente en conexión con la Fig. 5A. Por ejemplo, la biblioteca Kodo publicada por Steinwurf ApS de Dinamarca se puede usar con este propósito bajo la dirección del módulo de control de ventana finita 162.

Finalmente, el decodificador 160 incluye un módulo de entrega 170 que realiza funciones de salida (es decir, almacenamiento temporal, entrega a una capa de protocolo superior, etc.). El módulo de entrega 170 recibe símbolos de origen no codificados del almacenador temporal de codificación 166 y símbolos de origen decodificados del decodificador de RLNC 168. Se debería apreciar que cualquiera o todos los componentes funcionales en el decodificador 160 se pueden implementar en hardware, microprograma, software o cualquier combinación de tales tecnologías.

Varias realizaciones de la invención también realizan recodificación. La recodificación es la operación de crear nuevos paquetes codificados a partir de paquetes "de origen" previamente codificados y no codificados sin necesariamente decodificar primero los paquetes codificados. Esto típicamente implica (1) aplicar la combinación lineal a la carga útil, (2) calcular los nuevos coeficientes codificados a partir de los coeficientes de los paquetes de origen y (3) añadir los nuevos coeficientes codificados al paquete recién recodificado. La recodificación generalmente ocurre en los nodos intermedios para mejorar el rendimiento de la conexión a través de (1) el aumento de la diversidad de paquetes codificados (es decir, el número de símbolos de origen que son parte de la combinación lineal) o (2) inyectar nuevos paquetes codificados para cubrir pérdidas en enlaces posteriores. La recodificación encuentra aplicaciones importantes en las comunicaciones multitrayecto y la unión de canales (por ejemplo, LTE + Wi-Fi), redes de malla y enrutamiento oportunista para enlaces en serie.

En general, un recodificador debería aplicar los mismos mecanismos de filtro para los paquetes de realimentación y de origen. Cuando se combinan múltiples símbolos con vectores de coeficientes dispersos, el vector de coeficientes resultante contiene más y más elementos distintos de cero. Si esos elementos distintos de cero superan el tamaño de la ventana, los paquetes llegan a ser inútiles. Para evitar ese problema, los recodificadores pueden recurrir a los siguientes mecanismos:

Los paquetes recodificados deberían estar dispersos, por ejemplo, los coeficientes distintos de cero deben estar en una ventana de FEC. Esto se puede hacer cumplir aplicando una regla que prohíba la recodificación entre paquetes con números de secuencia que estén demasiado separados (es decir, la diferencia excede el tamaño de la ventana). El tamaño de la ventana puede ser diferente del tamaño de la ventana de origen/destino y se puede comunicar por otro nodo (por ejemplo, de origen, destino, gestión de red) o inferir por el recodificador en sí mismo a partir del tráfico de origen-destino o alguna otra métrica disponible.

Los recodificadores pueden ajustar la tasa de paquetes recodificados inyectados en el flujo (es decir, hacia el destino) en respuesta a la tasa de pérdida de paquetes en el siguiente salto o en el resto de la red. La tasa de pérdida se puede comunicar por otro nodo (por ejemplo, de origen, destino, gestión de red) o inferir por el recodificador en sí mismo a partir del tráfico de origen-destino o alguna otra información disponible, tal como la Información de Estado de Canal (CSI).

En general, un recodificador puede basar su decisión de transmitir paquetes adicionales y la forma en que mezcla los paquetes en la información de estado recibida o inferida sobre la ventana deslizante, el estado de la transferencia, así como la calidad del canal, del enlace o del siguiente salto. Un recodificador también puede considerar la complejidad de la decodificación cuando se codifican paquetes juntos, evitando así recodificar o crear un paquete que haya alcanzado un número objetivo de paquetes mixtos nativos, incluso si todos los símbolos todavía están dentro de la ventana deslizante de codificación. Un recodificador puede informar a sus vecinos aguas arriba de los símbolos/grados de libertad recibidos/que faltan/vistos, informándoles así de sus símbolos recibidos así como de la calidad del enlace, donde los símbolos vistos son símbolos de origen que aún no se han decodificado, que forman parte de una combinación lineal recibida, y los grados de libertad que faltan representan el número de símbolos codificados independientes que se requieren para la decodificación de una generación o un símbolo de origen dado.

Un recodificador puede interceptar y terminar una solicitud de nodos aguas abajo (por ejemplo, de destino) de paquetes o grados de libertad adicionales si es capaz de satisfacer esa solicitud a través de la transmisión de un símbolo de origen, codificado o recodificado. Los recodificadores pueden decodificar de manera oportuna y sobre la marcha para hacer sus símbolos almacenados más dispersos. Esto solo es posible si los recodificadores reciben suficientes paquetes. Y los recodificadores pueden permitir una recodificación 'fácil' sin decodificación parcial mediante la combinación de paquetes posteriores entrantes mientras que la ventana de FEC permanece constante o aumenta marginalmente. La recodificación puede permitir la adaptación a la tasa de codificación requerida del siguiente enlace, que puede ser diferente de las tasas de codificación de enlaces anteriores (tanto más altas como más bajas).

Por tanto, haciendo referencia ahora a la Fig. 9, un método de recodificación de acuerdo con una realización de recodificación (tal como el recodificador mostrado en la Fig. 1) incluye los procesos de decodificación 130 a 142 ilustrados en la Fig. 5, los procesos de decodificación 144 a 152 ilustrados en la Fig. 5A, y los procesos de codificación 154 a 158 ilustrados en la Fig. 3. En detalle, los procesos 130, 132, 134, 136, 138, 140 y 142 de la Fig. 9

corresponden respectivamente al proceso de inicialización 91, proceso de recepción 92, proceso de decisión de igualdad 95, proceso de decisión de novedad 93, proceso de actualización 94, proceso de decisión de obsolescencia 96 y proceso de descarte 97 de la Fig. 5. Por lo tanto, estos procesos 130 a 142 realizan colectivamente los procesos de decodificación de la Fig. 5, en el mismo orden, con la diferencia no funcional de tomar la decisión de igualdad 134 antes de 136 en lugar de después. La única diferencia funcional es que después del proceso de descarte 142, el método de la Fig. 9 no vuelve al proceso 132 en correspondencia directa con el método de las Figs. 5 y 5A, sino que en su lugar pasa a los procesos de codificación 154 a 158.

Además, los procesos 144, 146, 148, 150 y 152 de la Fig. 9 corresponden respectivamente al proceso de suma 91A, al proceso de reducción de filas 92A, al proceso de determinación de decodificación 93A, al proceso de entrega 94A y al proceso de almacenamiento 95A. Por lo tanto, estos procesos 144 a 152 realizan colectivamente los procesos de decodificación de la Fig. 5A, en el mismo orden, con las únicas diferencias funcionales que el proceso de determinación 148 y el proceso de entrega 150 se realizan opcionalmente en algunas realizaciones de decodificación y reenvío (como se indica por sus apariciones en líneas discontinuas), y que el proceso de almacenamiento 152 no vuelve al proceso 132 en correspondencia directa con el método de las Figs. 5 y 5A, sino que en su lugar pasa a los procesos de codificación 154 a 158.

Los procesos 154 a 158 de la Fig. 9 corresponden respectivamente al proceso de comprobación de tasa de código 40, el proceso de generación de paquetes codificados 42 y el proceso de puesta en cola 44 de la Fig. 3. Estos procesos 154 a 158, por lo tanto, realizan colectivamente los procesos de generación de paquetes codificados de la Fig. 3 (pero no los procesos de generación de paquetes no codificados), en el mismo orden, sin diferencias funcionales.

Haciendo referencia ahora a la Fig. 10, una realización de los conceptos descritos en la presente memoria es un recodificador 180 que incluye un módulo de control de ventana finita 182, un registro de número de secuencia 184, un almacenador temporal de codificación 186, un recodificador de código de red lineal aleatorio ("RLNC") 188, y un módulo de entrega 190. Ahora se describe la operación de estos componentes.

El módulo de control de ventana finita 182 es el mecanismo principal para controlar el recodificador 180. El módulo de control de ventana finita 182 recibe paquetes no codificados y codificados y mantiene un número de secuencia de recodificador (RSN) en un registro de número de secuencia de recodificador 184. En varias realizaciones, mantiene el RSN según los procesos descritos anteriormente en conexión con la Fig. 5 y especialmente el proceso de actualización 94 para actualizar un número de secuencia de decodificador (DSN).

El módulo de control de ventana finita 182 almacena los símbolos recibidos, ya sean no codificados o codificados, en un almacenador temporal de codificación 186. El almacenador temporal de codificación 186 contiene hasta w_d símbolos, donde w_d es el tamaño de la ventana deslizante (parcial) de decodificador, y el módulo de control de ventana finita 182 desaloja los símbolos obsoletos a medida que pasan fuera de la ventana. Esto puede suceder, por ejemplo, de acuerdo con los procesos descritos anteriormente en conexión con el proceso de actualización 94.

El recodificador 180 también incluye un recodificador de RLNC 188 para reducir por filas los símbolos codificados encontrados en los paquetes codificados respectivos (por ejemplo, como se describió anteriormente en conexión con la Fig. 5A) y generar y reenviar paquetes codificados como se describió anteriormente en conexión con la Fig. 3. Por ejemplo, la biblioteca Kodo publicada por Steinwurf ApS de Dinamarca se puede usar con este propósito bajo la dirección del módulo de control de ventana finita 182.

De acuerdo con algunas realizaciones, el recodificador de RLNC 188 no realiza decodificación parcial en absoluto, sino que, en su lugar, realiza codificación de red acumulativa transmitiendo paquetes codificados cuyos símbolos codificados sucesivos son combinaciones lineales de muchos símbolos de entrada cada vez mayores. Por tanto, el recodificador de RLNC 188 puede generar un nuevo símbolo codificado S_{n+1} a partir de un símbolo codificado anterior S_n, un coeficiente C y un símbolo de entrada recién recibido S utilizando la fórmula iterativa $S_{n+1} = S_n + C \cdot S$, donde la suma y la multiplicación se realizan en el campo finito GF(q). De esta forma, el recodificador 180 puede aumentar o disminuir el número de símbolos de entrada usados para formar un símbolo codificado; es decir, el tamaño de la ventana deslizante finita w_e. Este cambio permite ventajosamente que el recodificador 180 reenvíe paquetes entre canales de datos para los que diferentes características de canal requieren diferentes tasas de código.

Finalmente, el codificador 180 incluye un módulo de entrega 190 que realiza funciones de salida (es decir, almacenamiento temporal, entrega a una capa de protocolo superior, etc.). El módulo de entrega 190 recibe paquetes recodificados del recodificador de RLNC 188 y los reenvía a un codificador o nodo aguas abajo. Se debería apreciar que cualquiera o todos los componentes funcionales en el codificador 180 se pueden implementar en hardware, microprograma, software o cualquier combinación de tales tecnologías.

Haciendo referencia ahora a las Figs. 11, 11A y 11B, se dibuja una comparación entre los contenidos de los paquetes en las técnicas de la técnica anterior de códigos de bloques sistemáticos simples (Fig. 11) y ventana deslizante infinita (Fig. 11A), y la técnica de ventana deslizante finita descrita en la presente memoria (Fig. 11B). Cada una de tales figuras indica los paquetes transmitidos (codificados) en filas por orden de transmisión. Cada fila

indica mediante puntos qué símbolos de origen (o paquetes de origen) están codificados por combinación lineal en el paquete codificado respectivo. Además, cada figura proporciona la transmisión de dos símbolos no codificados por cada símbolo codificado (redundante) y, por lo tanto, ilustra un código que tiene una tasa de código de 2/3.

El código de bloques sistemático de la Fig. 11 muestra tres generaciones 190a, 190b, 190c de símbolos. Los primeros cuatro paquetes transmitidos no están codificados e incluyen símbolos de origen numerados 1, 2, 3 y 4 en orden. Los paquetes quinto y sexto están codificados y contienen combinaciones lineales de estos símbolos de origen que se pueden usar para recuperar los símbolos de origen en presencia de como máximo dos borrados de entre cualquiera de los seis paquetes transmitidos. Tal recuperación se discute a continuación en conexión con la Fig. 12. Habiendo transmitido la primera generación de símbolos numerados del 1 al 4 en los primeros seis paquetes, los siguientes seis paquetes transmiten una segunda generación de símbolos numerados del 5 al 8, los siguientes seis paquetes transmiten una tercera generación de símbolos numerados del 9 al 12, y así sucesivamente.

El código de ventana deslizante infinita 68 de la Fig. 11A repite un patrón de dos paquetes no codificados seguidos de un paquete que contiene un símbolo codificado que es una combinación lineal de todos (o algún subconjunto dinámico de todos) los símbolos de origen anteriores. Por tanto, los primeros dos paquetes transmitidos no están codificados e incluyen símbolos de origen numerados 1 y 2. El tercer paquete está codificado y contiene una combinación lineal de todos los símbolos de origen anteriores, numerados 1 a 2. Los siguientes dos paquetes no están codificados e incluyen símbolos de origen numerados 3 y 4. El sexto paquete está codificado y contiene una combinación lineal de todos los símbolos de origen anteriores, numerados del 1 al 4, y así sucesivamente.

Haciendo referencia ahora a la Fig. 11B, se describe el concepto básico del enfoque de Ventana Deslizante. Todavía no consideramos la realimentación o el multitrayecto, y solo consideramos un solo enlace. Se omite el tamaño de la ventana de FEC, ya que la FEC es más útil cuando los paquetes se envían a través de diferentes trayectos y se espera reordenar, o cuando está disponible la realimentación. Por tanto, la codificación ocurre durante toda la generación.

De acuerdo con realizaciones de los conceptos y técnicas descritos en la presente memoria, el código de ventana deslizante finita de la Fig. 11B repite un patrón de dos paquetes no codificados seguido de un paquete que contiene un símbolo codificado que es una combinación lineal de, como máximo, un número de ventana fijo w_e de los símbolos de origen anteriores. El código de ventana deslizante finita de la Fig. 11B combina la simplicidad ventajosa y los límites de almacenamiento de los códigos de bloque sistemáticos que faltan en los códigos de ventana deslizante infinita, con la latencia de decodificación reducida ventajosa de los códigos de ventana deslizante infinita con respecto a los códigos de bloque sistemáticos.

La Fig. 12 ilustra la decodificación de símbolos de origen usando una técnica de solicitud de repetición automática selectiva (S-ARQ) con un código de bloque sistemático que tiene generaciones como en la Fig. 11. En esta figura ejemplar, se transmitieron 24 paquetes por un codificador, pero solo se recibieron 15 paquetes por un decodificador debido a borrados de paquetes, indicados por filas cuyos puntos están tachados. Los símbolos de origen se proporcionan en generaciones de tres.

La Fig. 12 se puede entender de la siguiente manera. Se recibió el paquete no codificado 0 por el decodificador que contenía el símbolo 1, permitiendo la entrega en orden del símbolo 1 al codificador local para su procesamiento adicional (por ejemplo, en una capa de protocolo superior). Se recibió el paquete no codificado 1 que contenía el símbolo 2, permitiendo la entrega en orden del símbolo 2. Se borró el paquete no codificado 2 que contenía el símbolo 3, como lo fue el paquete 3 codificado que contenía un símbolo codificado. Por tanto, el símbolo de origen 3 no se pudo entregar en orden.

En la segunda generación, se recibió el paquete no codificado 4 que contenía el símbolo 4, pero el símbolo 4 no se pudo entregar en orden porque el símbolo 3 estaba pendiente de entrega. Asimismo, se borró el paquete no codificado 5 (símbolo 5), se recibió el paquete no codificado 6 (símbolo 6) y se recibió el paquete codificado 7, permitiendo la recuperación del símbolo 5 borrado.

Sin embargo, ninguno de los símbolos decodificados 110 se pudo entregar en orden porque el símbolo 3 aún estaba pendiente de entrega. Por lo tanto, el decodificador solicitó en el primer punto de realimentación (indicada por la línea horizontal después del paquete 7) que el codificador transmitiera un nuevo símbolo codificado, linealmente independiente, para la primera generación.

Para la tercera generación, el codificador transmitió este símbolo codificado en el paquete codificado 8, pero este paquete también se borró. El codificador procedió a transmitir los paquetes no codificados 9, 10 y 11 (símbolos de origen 7, 8 y 9; todos borrados) y el paquete codificado 12 que se recibió. Cuando se borró el paquete codificado 8, en el segundo punto de realimentación, el decodificador solicitó de nuevo la transmisión de un símbolo codificado para la primera generación.

Para la cuarta generación, el decodificador finalmente recibió su símbolo codificado necesario en el paquete codificado 13. La recepción del paquete codificado 13 permitió la recuperación (y entrega) del símbolo 3. Como los símbolos de origen 4, 5 y 6 ya estaban decodificados, también fueron entregados entonces. Luego, el decodificador

recibió los paquetes no codificados 14, 15 y 16 que contenían respectivamente los símbolos 10, 11 y 12. Sin embargo, estos símbolos no se pudieron entregar, a la espera de la entrega de los símbolos borrados 7, 8 y 9. Dado que el decodificador obtuvo un símbolo codificado en el paquete 12 de una generación de tres, solicitó 3-1=2 símbolos codificados más linealmente independientes para la tercera generación.

Para la quinta generación, el decodificador recibió sus símbolos codificados necesarios para la generación tres en los paquetes codificados 18 y 19. Dados los tres símbolos codificados linealmente independientes de los paquetes 12, 18 y 19, la eliminación gaussiana (como se discute a continuación) permitió que el decodificador recuperase los tres símbolos de origen 7, 8 y 9, que se entregaron inmediatamente con los símbolos 10, 11 y 12 obtenidos previamente. Finalmente, el decodificador fue capaz de entregar inmediatamente los símbolos de origen 13, 14 y 15 después de recibir los paquetes no codificados 20, 21 y 22 correspondientes. El borrado del paquete codificado 23 no afectó a la entrega de estos símbolos.

La Fig. 13 ilustra la decodificación de símbolos de origen usando una técnica de S-ARQ modificada de acuerdo con una realización de los conceptos de ventana deslizante finita descritos en la presente memoria. El decodificador procesa la secuencia de paquetes utilizando tanto la ventana deslizante de codificador como una ventana de FEC deslizante más pequeña contenida en el mismo, como se muestra en la Fig. 2. En la secuencia de paquetes ejemplar de la Fig. 13, una ventana de FEC de seis símbolos (es decir, dos generaciones) se utiliza para proporcionar corrección de errores sin canal de retorno. El diseño de la ventana de FEC mostrado en la Fig. 1 permite que una realimentación se reciba por el codificador mientras que la ventana deslizante todavía permite retransmisiones porque los paquetes solicitados todavía están en la ventana deslizante del codificador.

Un decodificador que procesa la secuencia de paquetes de la Fig. 13 recibe los paquetes no codificados 0, 1, 4 y 6 y el paquete codificado 7 (que contiene una combinación lineal de los símbolos 1 a 6), mientras que los paquetes 2, 3 y 5 se borraron. Por tanto, se entregaron los símbolos de origen 1 y 2, se desconocían los símbolos 3 y 5 pendientes y se conocían los símbolos 4 y 6 pendientes, pero no se pudieron entregar. En el primer momento de realimentación después de la segunda generación, el decodificador tenía cinco paquetes para los seis símbolos en las dos primeras generaciones. Por tanto, solicitó la transmisión de 6-5 = 1 paquete adicional que contenía una combinación lineal de los símbolos desconocidos de la primera generación (es decir, el símbolo de origen 3).

Para la tercera generación, se borró el paquete no codificado 8 que contenía el símbolo 3, como lo fueron los paquetes no codificados 9, 10 y 11, mientras que se recibió el paquete 12 que contenía una combinación lineal de los símbolos 4 a 9. Por tanto, los símbolos 3, 5, 7, 8 y 9 pendientes eran desconocidos y los símbolos 4 y 6 pendientes seguían siendo conocidos pero no se podían entregar. Como el decodificador todavía tenía solo cinco paquetes para los seis símbolos en las dos primeras generaciones, solicitó la transmisión de 6-5 = 1 paquete adicional que contenía una combinación lineal de los símbolos desconocidos de las dos primeras generaciones (es decir, los símbolos 3 y 5).

Para la cuarta generación, el decodificador recibió el paquete codificado 13. Combinado con los paquetes recibidos 0, 1, 4, 6 y 7, el paquete codificado 13 hizo seis paquetes ($w_d = 6$) asociados con seis símbolos de origen ($w_e = 6$). Dado que $w_d \geq w_e$, según el álgebra lineal básica, las combinaciones lineales en estos paquetes produjeron un sistema de ecuaciones que se podría resolver para recuperar los símbolos de origen 1 a 6. Así, los símbolos pendientes desconocidos numerados 3 y 5 fueron recuperados y entregados con los símbolos numerados 4 y 6. Los símbolos 7, 8 y 9 pendientes eran desconocidos y los símbolos 10, 11 y 12 pendientes eran conocidos pero no se pudieron entregar.

Después de recibir los paquetes no codificados 14 a 16 pero no el paquete codificado 17, el decodificador tenía cuatro paquetes (numerados 12, 14, 15 y 16) para los seis símbolos en las generaciones tres y cuatro. Por lo tanto, solicitó la transmisión de 6-4 = 2 paquetes adicionales que contienen combinaciones lineales (linealmente independientes) de los símbolos desconocidos de estas generaciones (es decir, los símbolos 7, 8 y 9). Tenga en cuenta que el paquete codificado 12 contó para la generación tres, porque su símbolo codificado abarcó las generaciones dos y tres, pero todos los símbolos de origen en la generación dos eran conocidos y se podían restar de la combinación lineal.

Para la quinta generación, el decodificador recibió los paquetes codificados 18 y 19, completando los seis paquetes necesarios para decodificar los seis símbolos de origen en las generaciones tres y cuatro. Por lo tanto, recuperó los símbolos desconocidos 7, 8 y 9, que entregó con los símbolos conocidos 10, 11 y 12. Finalmente, entregó los símbolos de origen 13, 14 y 15 inmediatamente tras recibir los respectivos paquetes no codificados 20, 21 y 22.

Tenga en cuenta que la realización de la Fig. 13 es generacional, mientras que las realizaciones de las Figs. 6 y 7 pueden ser sin generación, demostrando que las técnicas de ventana deslizante finita descritas en la presente memoria se pueden aplicar en cualquier caso. También tenga en cuenta que después de cada generación, se proporciona realimentación solo con respecto a los símbolos que son parte de la ventana deslizante finita. Por ejemplo, el paquete 12 es una combinación lineal de los símbolos de origen 4-9, los seis últimos símbolos de la ventana deslizante (es decir, los símbolos dentro de la ventana de FEC). Tras recibir la realimentación, el codificador envía el paquete 13, una combinación lineal de los paquetes que faltan 3 y 5. En este caso, aunque el símbolo 3 está fuera de la ventana de FEC, todavía está dentro de la ventana deslizante del codificador, por lo que se puede

incluir en el paquete 13.

Las realizaciones según la Fig. 13 tienen diferencias y ventajas sobre la codificación de borrado de la técnica anterior de la Fig. 12. La secuencia de paquetes de la Fig. 13 borra los mismos paquetes numerados que la secuencia de la Fig. 12, pero el contenido de cada paquete es diferente. Tenga en cuenta también que las Figs. 12 y 13 ambas indican la entrega del símbolo de origen 1 después del paquete 0, el símbolo 2 después del paquete 1, los símbolos 3 a 6 después del paquete 13, y así sucesivamente, y los decodificadores de las Figs. 12 y 13 ambos tienen límites de complejidad y almacenamiento similares. Sin embargo, ventajosamente, el decodificador de la Fig. 13 se ejecuta más rápido porque tiene el retardo de decodificación de un código de ventana deslizante en lugar de un código de bloque sistemático.

Las Figs. 14, 14A, 14B y 14C muestran cuatro aplicaciones diferentes del mecanismo de realimentación con los conceptos de ventana deslizante finita descritos en la presente memoria, donde el tamaño de la ventana de decodificación es $w_d = 5$ y la matriz de coeficientes de decodificación contiene al menos dos vectores de coeficientes recibidos (1,a,b,0,0) y (0,0,0,1,c). En cada figura, los cinco símbolos de origen indicados por las etiquetas de las columnas a, b, c, d y e corresponden a los símbolos transportados por los paquetes p_1, p_2, p_3, p_4, y p_5, respectivamente, y la línea horizontal indica una solicitud de realimentación, como en las Figs. 12 y 13. Las Figs. 14A y 14C ilustran aplicaciones para las que se espera una menor latencia y más flexibilidad, en forma de compromisos de rendimiento/eficiencia frente a retardo.

En la Fig. 14, ARQ se aplica en una capa superior. Dado que ninguno de los símbolos de origen se entregó a la capa de ARQ, la realimentación del decodificador solicita los cinco paquetes de origen, por lo tanto, la retransmisión de p_1, p_2, p_3, p_4, y p_5. Las Figs. 14A, 14B y 14C muestran combinaciones más eficientes de realimentación con los conceptos de ventana deslizante finita descritos en la presente memoria, donde la realimentación se integra con la capa de codificación.

En la Fig. 14A, en lugar de solicitar todos los símbolos de origen, el proceso de realimentación del decodificador solicita tres paquetes completamente codificados (es decir, tres paquetes donde se combinan todos los símbolos en la ventana deslizante). Los tres paquetes recibidos permiten la decodificación de todos los símbolos de origen tras la recepción del tercer paquete codificado, porque el decodificador tiene cinco paquetes que contienen cinco combinaciones linealmente independientes de estos cinco símbolos de origen.

La Fig. 14B muestra una configuración en la que los paquetes retransmitidos son sistemáticos (es decir, no codificados). A diferencia de la Fig. 14, los paquetes retransmitidos en la Fig. 14B se pueden utilizar por el decodificador. Por lo tanto, el decodificador puede entregar p_3 junto con p_1 después de decodificar su primer paquete almacenado (1,a,b,0,0). Sin embargo, los paquetes redundantes se entregan después de que se reciba el tercer paquete.

La Fig. 14C muestra una configuración óptima en la que la realimentación del decodificador solicita paquetes sistemáticos (es decir, no codificados) pero es consciente de los paquetes presentes en la matriz de coeficientes del decodificador. Por lo tanto, solo los paquetes p_1, p_2, y p_4 se solicitan, permitiendo la entrega de todos los paquetes por la tercera retransmisión recibida. Mientras que ambos de los esquemas de las Figs. 14A y 14B pueden entregar todos los paquetes en la tercera retransmisión recibida, el esquema en la Fig. 14C funciona mejor ya que entrega algunos de los paquetes antes (por ejemplo, p_1, p_2, y p_3) y requiere menos operaciones de decodificación debido a la naturaleza sistemática de las retransmisiones.

Por tanto, se puede apreciar que el concepto de ventana finita descrito en la presente memoria permite un código sistemático que inserta paquetes codificados o redundantes de vez en cuando, permitiendo un retardo mínimo. Especialmente en el caso de realimentación, el decodificador puede solicitar paquetes aumentando la probabilidad de decodificación si el codificador los enviara sistemáticamente. Por supuesto, el codificador puede enviar además paquetes codificados o redundantes, como se representa en la Fig. 14A.

Las técnicas y estructuras descritas en la presente memoria se pueden implementar en cualquiera de una variedad de formas diferentes. Por ejemplo, las características de los conceptos, sistemas y técnicas descritos en la presente memoria se pueden incorporar dentro de varias formas de dispositivos de comunicación, tanto por cable como inalámbricos; equipo de televisión; receptores multimedia digitales; dispositivos de audio/vídeo; ordenador portátil, de mano, de escritorio y tabletas con o sin capacidad inalámbrica; asistentes digitales personales (PDA); teléfonos; buscapersonas; comunicadores por satélite; cámaras que tienen capacidad de comunicación; tarjetas de interfaz de red (NIC) y otras estructuras de interfaz de red; estaciones base; puntos de acceso; circuitos integrados; tales como instrucciones y/o estructuras de datos almacenadas en medios legibles por máquina; y/o en otros formatos. Ejemplos de diferentes tipos de medios legibles por máquina que se pueden usar incluyen disquetes, discos duros, discos ópticos, memorias de solo lectura de discos compactos (CD-ROM), discos de video digital (DVD), discos Blu-ray, discos magnetoópticos, memorias de solo lectura (ROM), memorias de acceso aleatorio (RAM), ROM programables borrables (EPROM), ROM programables borrables eléctricamente (EEPROM), tarjetas magnéticas u ópticas, memoria flash y/u otros tipos de medios adecuados para almacenar instrucciones electrónicas o datos.

En la descripción detallada anterior, varias características de los conceptos, sistemas y técnicas descritas en la

presente memoria se agrupan entre sí en una o más realizaciones individuales para simplificar la descripción. Este método de descripción no se ha de interpretar como que refleja una intención de que los conceptos, sistemas y técnicas reivindicados descritos en la presente memoria requieran más características de las que se enumeran expresamente en cada reivindicación. Más bien, los aspectos inventivos pueden residir en menos de todas las características de cada realización descrita.

Habiendo descrito implementaciones que sirven para ilustrar varios conceptos, estructuras y técnicas que son el tema de esta descripción, ahora será evidente para los expertos en la técnica que se pueden usar otras implementaciones que incorporen estos conceptos, estructuras y técnicas. En consecuencia, se afirma que el alcance de la patente no se debería limitar a las implementaciones descritas, sino que más bien se debería limitar solo por el alcance de las siguientes reivindicaciones.

REIVINDICACIONES

1. Un método de decodificación de datos empaquetados en presencia de borrados de paquetes, el método que comprende, mediante un decodificador, repetidamente:

recibir (92; 132) un paquete codificado que comprende un número de secuencia de paquete, PSN, un vector de coeficientes que tiene una longitud fija w, y un símbolo codificado, codificado como una combinación lineal de w símbolos de entrada usando el vector de coeficientes, el PSN que comprende el número de secuencia más grande de un símbolo de entrada utilizado en la codificación del símbolo codificado;

determinar (93; 134, 136) si el número de secuencia del paquete está dentro de una ventana deslizante de w números de secuencia consecutivos que no son mayores que un número de secuencia de decodificador, DSN, el DSN correspondiente a un PSN más grande recibido por el decodificador, en donde la ventana deslizante tiene un tamaño fijo w determinado midiendo un tiempo de ida y vuelta, RTT; y caracterizado por que

cuando el número de secuencia del paquete está dentro de la ventana deslizante, decodificar (91A-93A; 144,146) el símbolo codificado en uno o más de los w símbolos de entrada utilizando el vector de coeficientes, incluyendo el uso de la corrección de errores sin canal de retorno, FEC, en los últimos f < w símbolos transmitidos, en donde uno o más símbolos dentro de la ventana deslizante y más antiguos que los últimos f símbolos transmitidos se usan para decodificar en respuesta a la detección de que uno o más de los últimos símbolos transmitidos faltan o no se pueden decodificar.

2. Un método según la reivindicación 1, que comprende predeterminar el tamaño fijo de la ventana deslizante según un tiempo de ida y vuelta para los datos que viajan entre el decodificador y un codificador (4A, 4C) a través de un canal de datos.

3. Un método según la reivindicación 1, en donde recibir comprende recibir un paquete codificado que tiene un número de secuencia de paquete que está fuera de orden.

4. Un método según la reivindicación 1, en donde recibir comprende recibir una pluralidad de paquetes que incluyen el paquete codificado y decodificar comprende corregir un error en uno de la pluralidad de paquetes según un código de corrección de errores sin canal de retorno.

5. Un método según la reivindicación 1, en donde decodificar comprende decodificar según un código sistemático o un código de red lineal.

6. Un método según la reivindicación 1, en donde decodificar comprende establecer el número de secuencia de decodificador igual al número de secuencia del paquete recibido cuando el número de secuencia del paquete recibido es mayor que el número de secuencia de decodificador.

7. Un método según la reivindicación 1, en donde decodificar usando el vector de coeficientes comprende generar un borrado de paquete cuando el vector de coeficientes tiene una entrada distinta de cero asociada con un símbolo de entrada cuyo número de secuencia está fuera de la ventana deslizante.

8. Un método según la reivindicación 1, en donde decodificar usando el vector de coeficientes comprende realizar (92A; 146) la eliminación gaussiana en una matriz, una fila de la cual incluye el vector de coeficientes.

9. Un método según la reivindicación 8, en donde decodificar utilizando el vector de coeficientes comprende, antes de realizar la eliminación gaussiana, eliminar cada fila de la matriz que tiene una entrada de coeficiente distinta de cero asociada con un símbolo de entrada cuyo número de secuencia está fuera de la ventana deslizante.

10. Un método según la reivindicación 8, en donde realizar la eliminación gaussiana comprende pivotar sobre la columna de la matriz cuyo índice es igual al número de secuencia de decodificador módulo el tamaño de la ventana deslizante.

11. Un método según la reivindicación 1, que comprende además que el decodificador proporcione realimentación a un codificador a través de un canal de datos, permitiendo por lo tanto que el codificador transmita, a través del canal de datos, uno o más paquetes recodificando cualquier dato que el decodificador no decodificó debido a un borrado de paquete.

12. Un medio de almacenamiento tangible, legible por ordenador, en el que se almacena de manera no transitoria un código de programa de ordenador que, cuando se ejecuta por un procesador de ordenador, hace que el procesador de ordenador lleve a cabo el método de cualquier reivindicación anterior.

13. Un decodificador (6A, 6C; 160) para decodificar datos empaquetados en presencia de borrados de paquetes, el decodificador que comprende:

una memoria para almacenar una matriz de coeficientes;

un almacenador temporal (166) para recibir un símbolo codificado de un paquete codificado que comprende un número de secuencia de paquete, PSN, un vector de coeficientes que tiene una longitud fija w, y el símbolo codificado, codificado como una combinación lineal de w símbolos de entrada usando el vector de coeficientes, el PSN que comprende el mayor número de secuencia de un símbolo de entrada utilizado en la codificación del símbolo codificado;

un registro (164) para almacenar un número de secuencia de decodificador, DSN, el DSN correspondiente al PSN más grande recibido por el decodificador;

una unidad de determinación (162) para determinar si el número de secuencia del paquete está dentro de una ventana deslizante de w números de secuencia consecutivos que no son mayores que el número de secuencia de decodificador almacenado en el registro, en donde la ventana deslizante tiene un tamaño fijo w determinado midiendo un tiempo de ida y vuelta, RTT; y

un módulo decodificador (168) para decodificar caracterizado por que cuando la unidad de determinación determina que el número de secuencia del paquete está dentro de la ventana deslizante, el símbolo codificado recibido en uno o más de los w símbolos de entrada utilizando el vector de coeficientes, incluyendo el uso de la corrección de errores sin canal de retorno, FEC, en los últimos f < w símbolos transmitidos, en donde uno o más símbolos dentro de la ventana deslizante y más antiguos que los últimos f símbolos transmitidos se usan para decodificar en respuesta a detectar que uno o más de los últimos f símbolos transmitidos faltan o no se pueden decodificar.

14. Un decodificador según la reivindicación 13, en donde el módulo de decodificación está configurado para escribir el número de secuencia del paquete del paquete recibido en el registro cuando el número de secuencia del paquete del paquete recibido es mayor que el número de secuencia de decodificador almacenado en el registro.

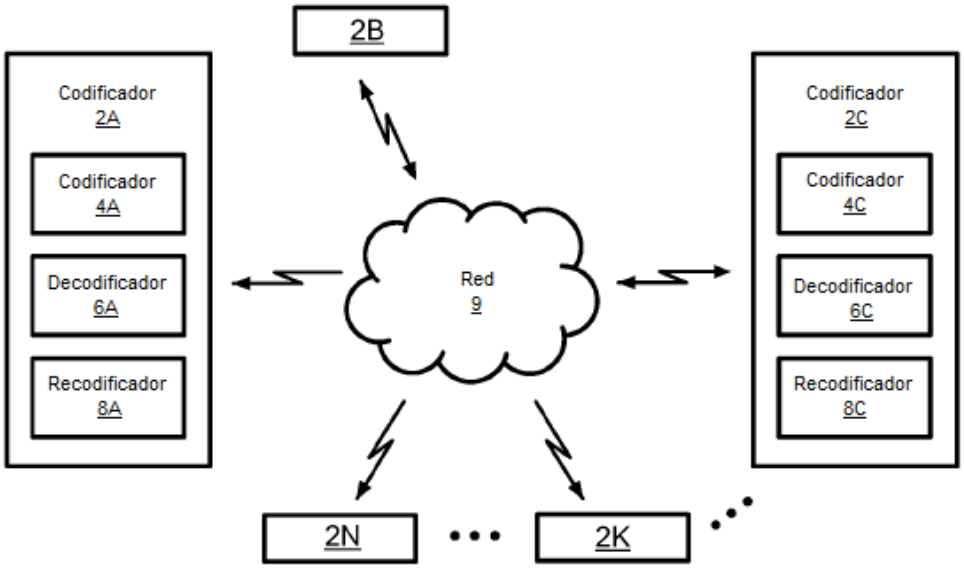
2B
Codificador
2A
Codificador
4A
Decodificador
6A
Recodificador
8A
Red
9
Codificador
2C
Codificador
4C
Decodificador
6C
Recodificador
8C
2N
2K

FIG. 1

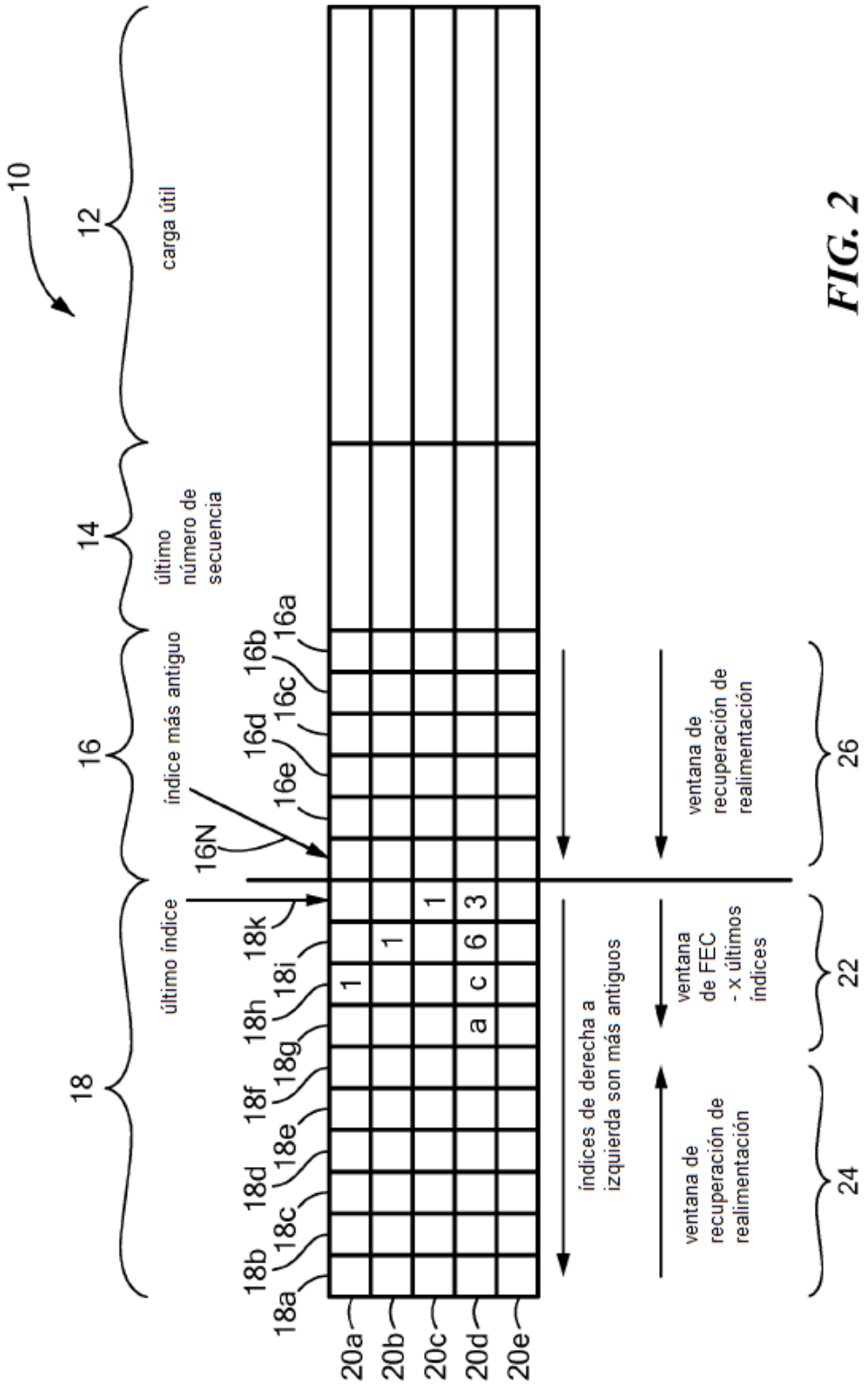
10
12
carga útil
14
último número de secuencia
16
índice más antiguo
16N
16a
16b
16c
16d
16e
último índice
18
18a
18b
18c
18d
18e
18f
18g
18h
18i
18k
20a
20b
20c
20d
20e
índices de derecha a izquierda son más antiguos
ventana de recuperación de realimentación
ventana de FEC - x últimos índices
ventana de recuperación de realimentación
22
24
26

FIG. 2

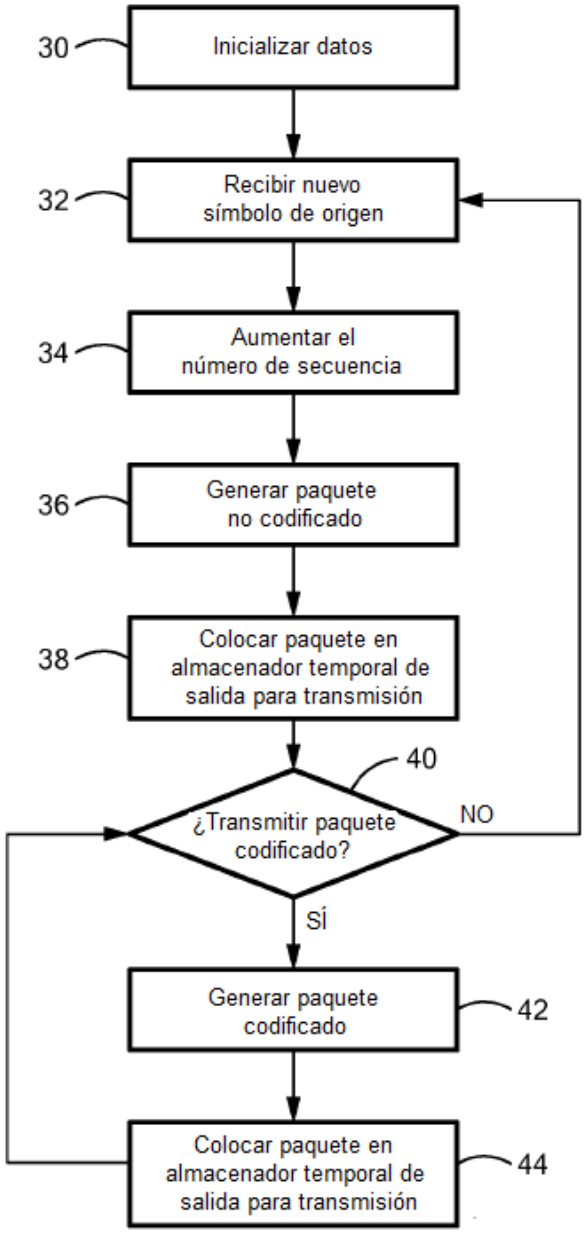
30
Inicializar datos
32
Recibir nuevo símbolo de origen
34
Aumentar el número de secuencia
36
Generar paquete no codificado
38
Colocar paquete en almacenador temporal de salida para transmisión
40
¿Transmitir paquete codificado?
NO
SÍ
Generar paquete codificado
42
Colocar paquete en almacenador temporal de salida para transmisión
44

FIG. 3

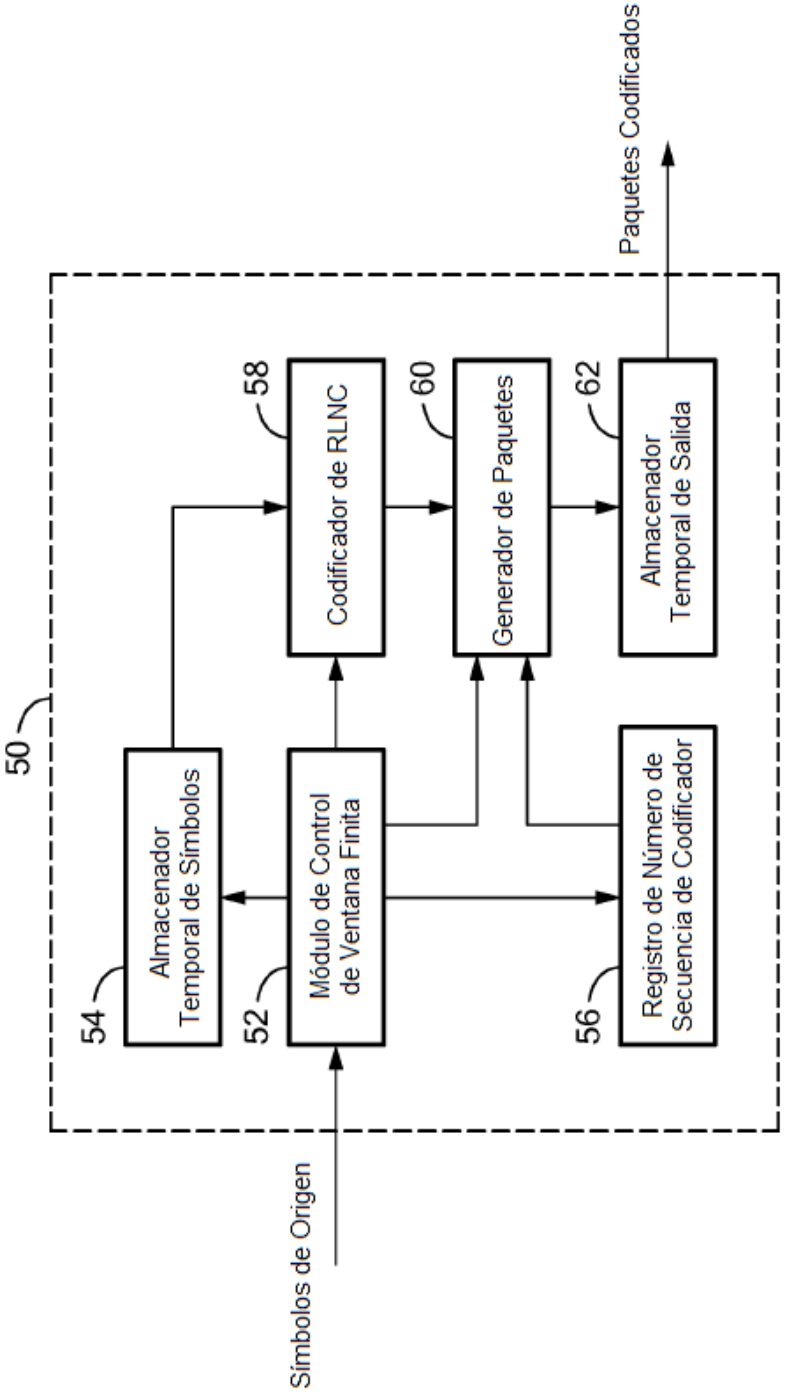
Paquetes Codificados
50
58
60
62
Codificador de RLNC
Generador de Paquetes
Almacenador Temporal de Salida
Almacenador Temporal de Símbolos
Módulo de Control de Ventana Finita
Registro de Número de Secuencia de Codificador
54
52
56
Símbolos de Origen

FIG. 4

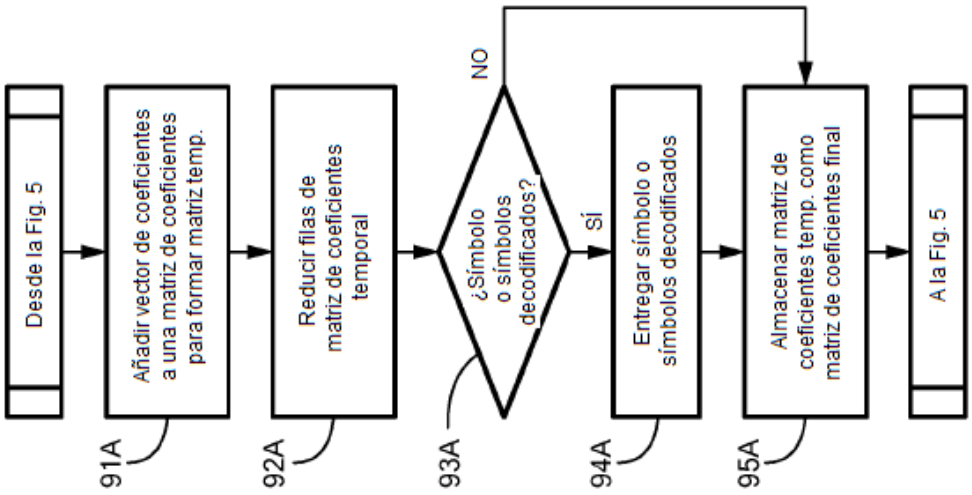
Desde la Fig. 5
Añadir vector de coeficientes a una matriz de coeficientes para formar matriz temp.
Reducir filas de matriz de coeficientes temporal
¿Símbolo o símbolos decodificados?
NO
Sí
Entregar símbolo o símbolos decodificados
Almacenar matriz de coeficientes temp. como matriz de coeficientes final
A la Fig. 5
91A
92A
93A
94A
95A

FIG. 5A

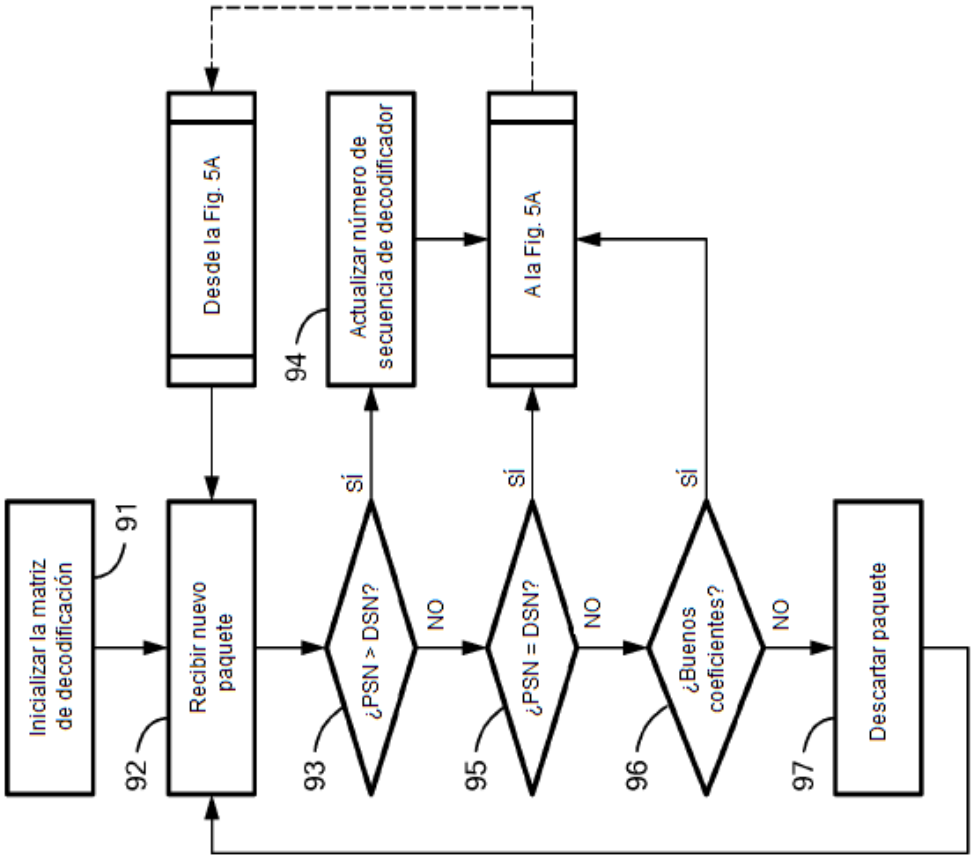
Inicializar la matriz de decodificación
Recibir nuevo paquete
¿PSN > DSN?
¿PSN = DSN?
¿Buenos coeficientes?
Descartar paquete
Desde la Fig. 5A
Actualizar número de secuencia de decodificador
A la Fig. 5A
Sí
NO
91
92
93
94
95
96
97

FIG. 5

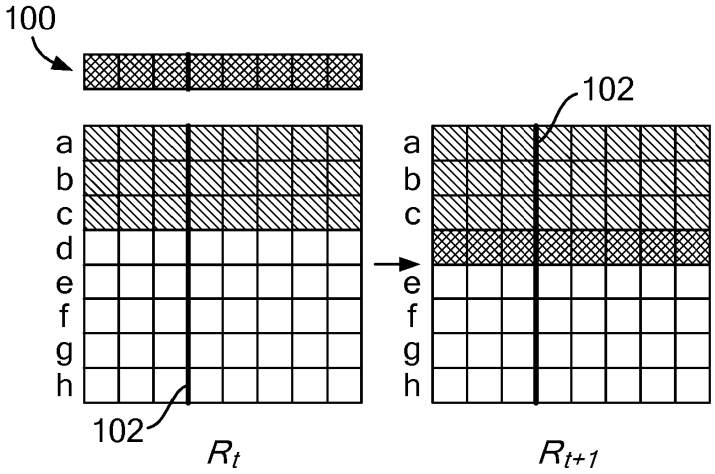

FIG. 6

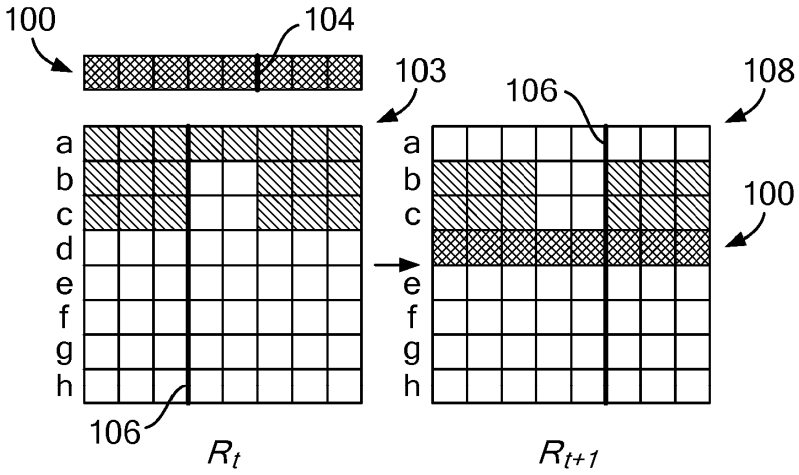

FIG. 6A

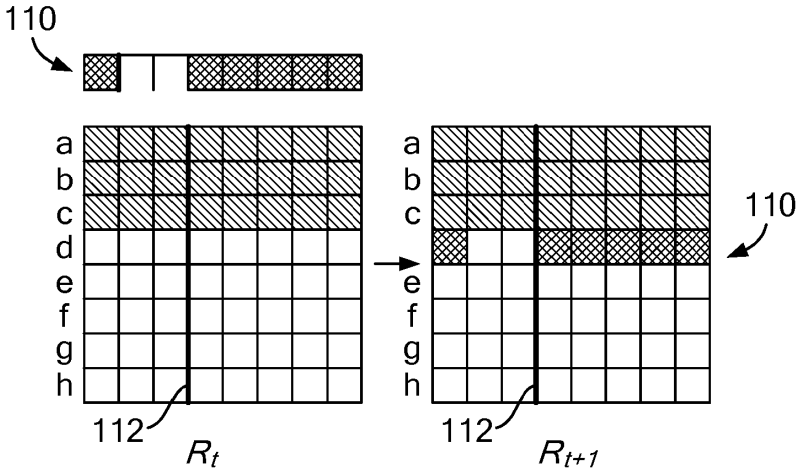

FIG. 6B

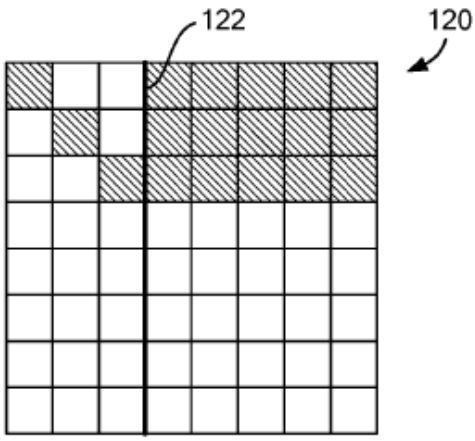

FIG. 7

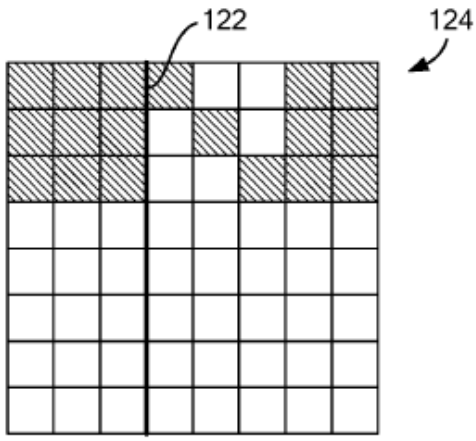

FIG. 7A

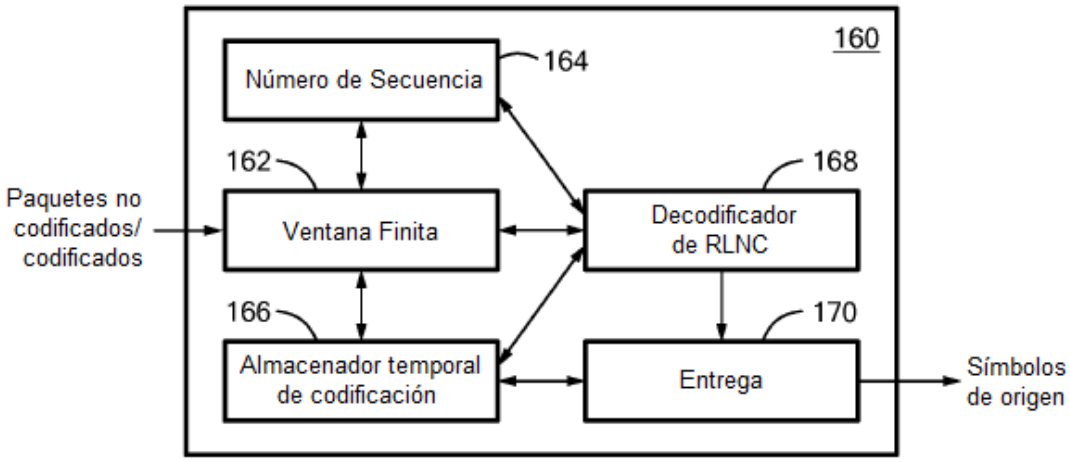

FIG. 8

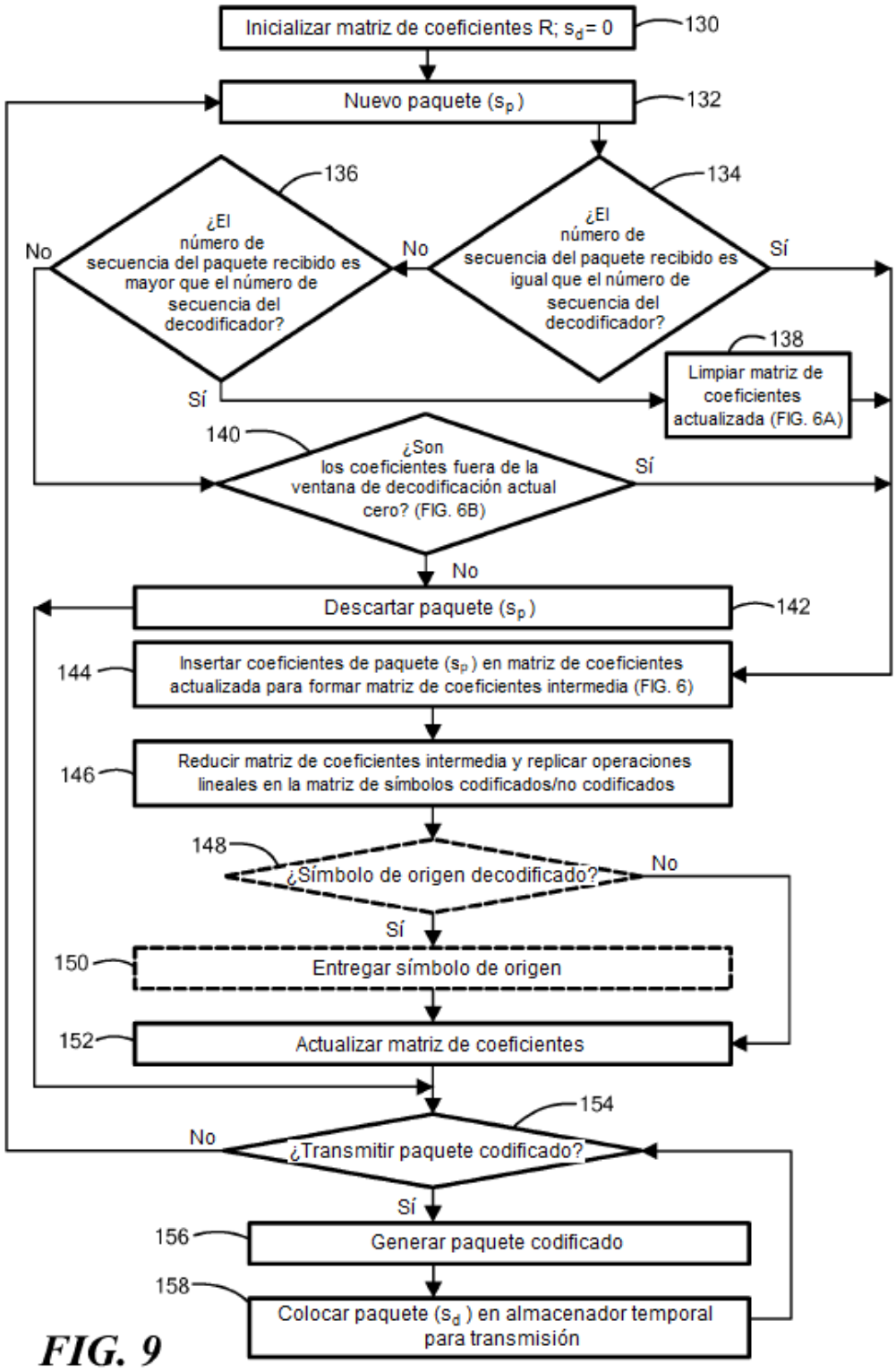
Inicializar matriz de coeficientes R; s_d = 0
130
Nuevo paquete (s_p)
132
¿El número de secuencia del paquete recibido es igual que el número de secuencia del decodificador?
134
Sí
No
¿El número de secuencia del paquete recibido es mayor que el número de secuencia del decodificador?
136
No
Sí
Limpiar matriz de coeficientes actualizada (FIG. 6A)
138
¿Son los coeficientes fuera de la ventana de decodificación actual cero? (FIG. 6B)
140
Sí
No
Descartar paquete (s_p)
142
Insertar coeficientes de paquete (s_p) en matriz de coeficientes actualizada para formar matriz de coeficientes intermedia (FIG. 6)
144
Reducir matriz de coeficientes intermedia y replicar operaciones lineales en la matriz de símbolos codificados/no codificados
146
¿Símbolo de origen decodificado?
148
No
Sí
Entregar símbolo de origen
150
Actualizar matriz de coeficientes
152
¿Transmitir paquete codificado?
154
No
Sí
Generar paquete codificado
156
Colocar paquete (s_d) en almacenador temporal para transmisión
158

FIG. 9

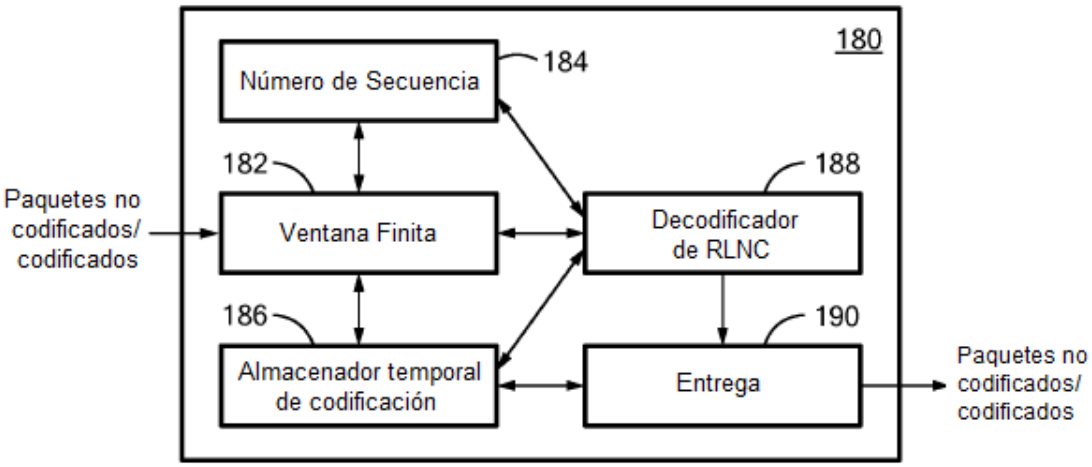
180
Número de Secuencia
184
182
188
Paquetes no codificados/ codificados
Ventana Finita
Decodificador de RLNC
186
190
Almacenador temporal de codificación
Entrega
Paquetes no codificados/ codificados

FIG. 10

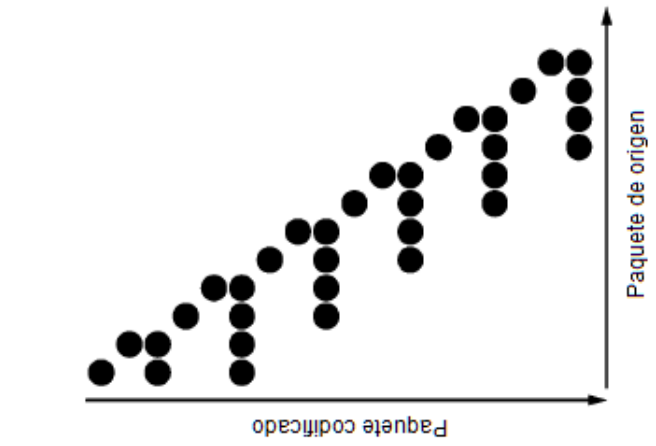

FIG. 11B

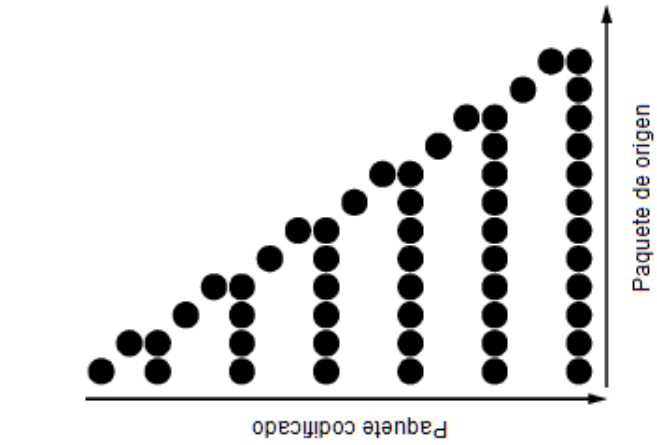

FIG. 11A

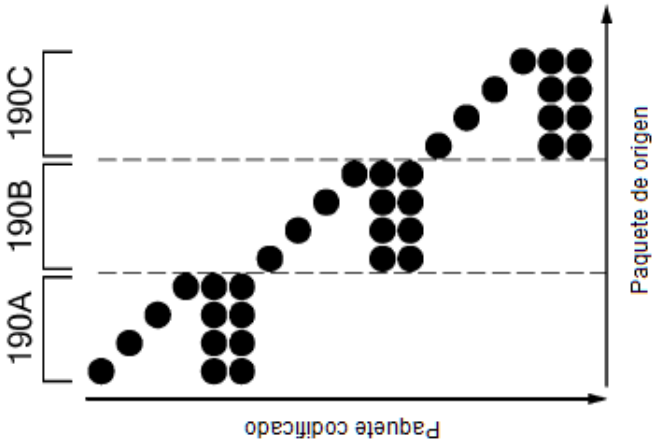

FIG. 11

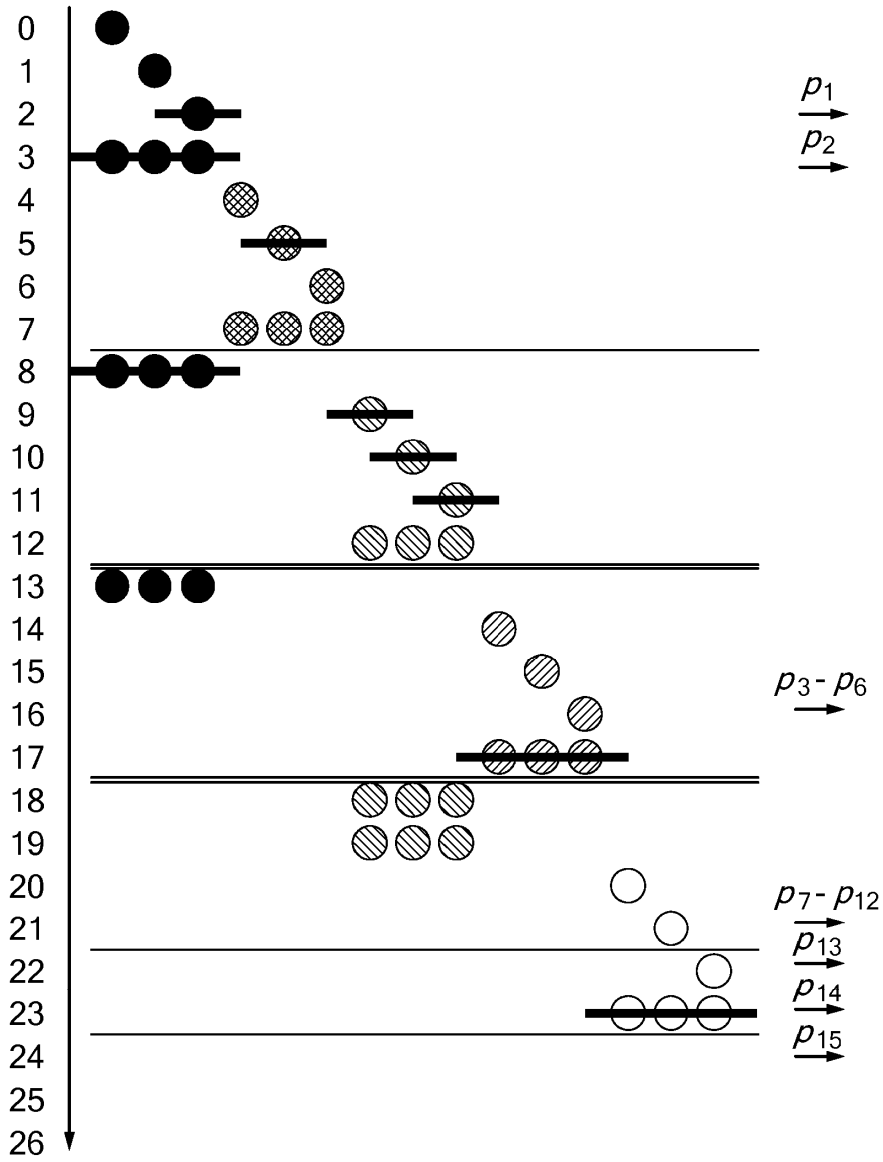
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
p_1
p_2
p_3 - p_6
p_7 - p_{12}
p_{13}
p_{14}
p_{15}

FIG. 12

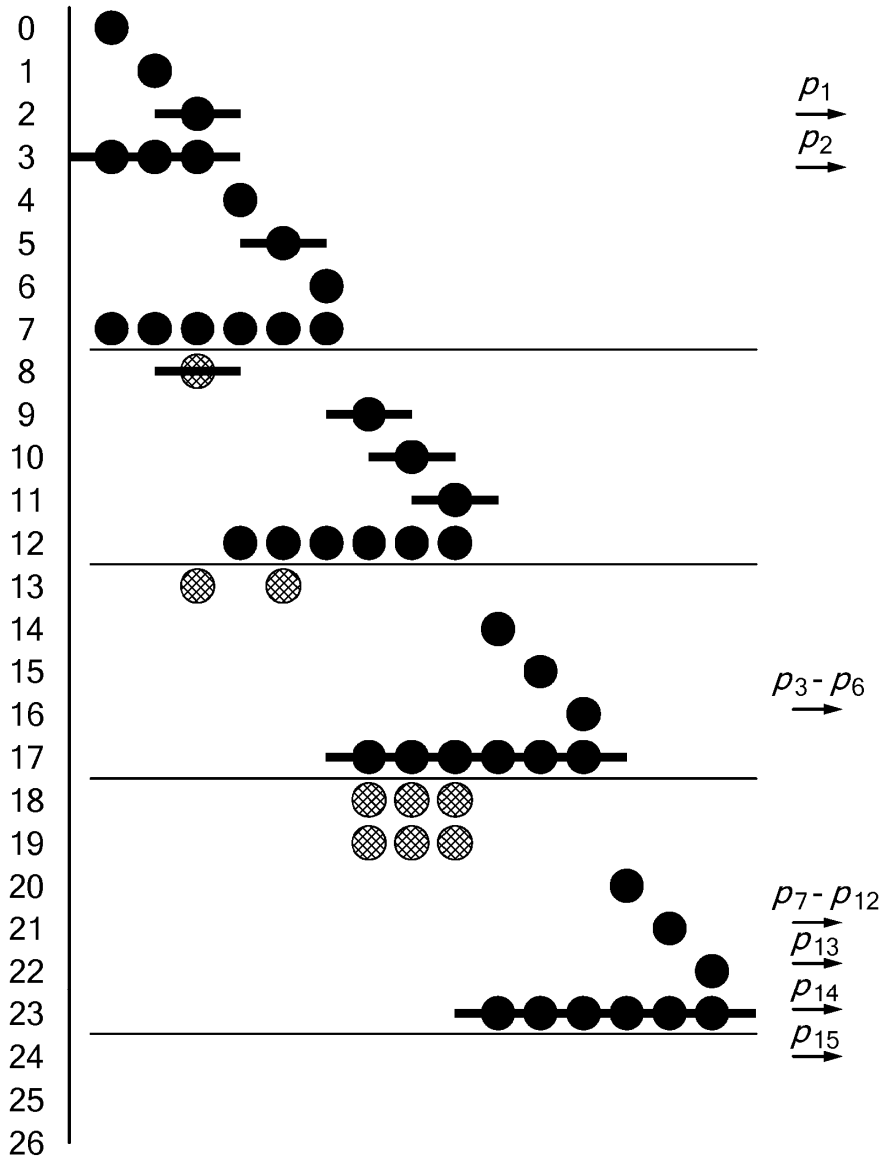
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
p_1
p_2
p_3 - p_6
p_7 - p_{12}
p_{13}
p_{14}
p_{15}

FIG. 13

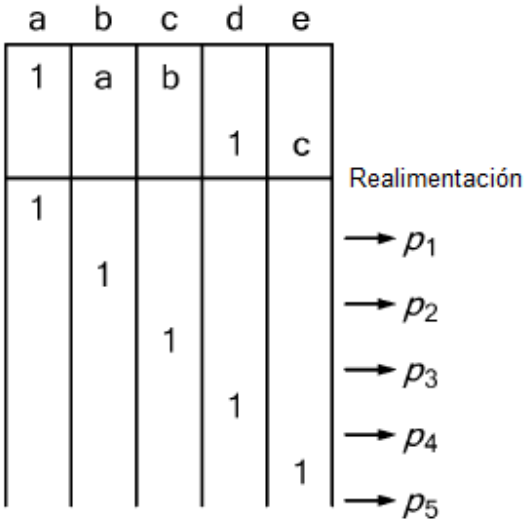

FIG. 14

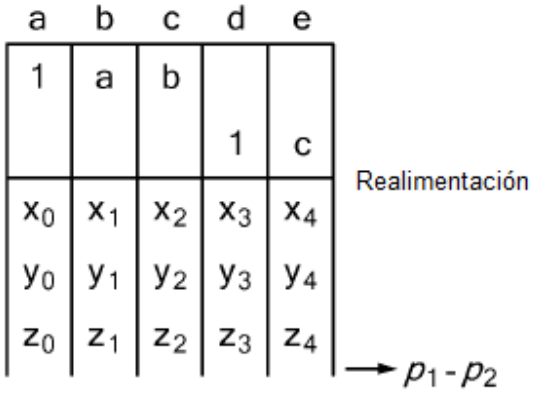

FIG. 14A

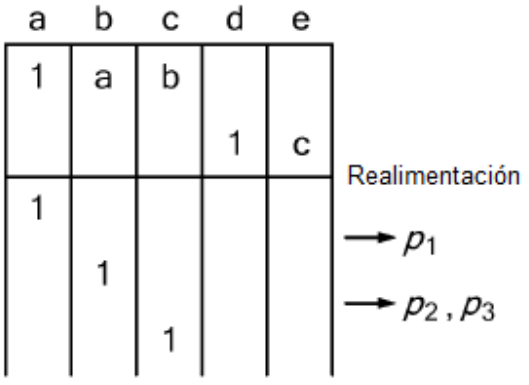

FIG. 14B

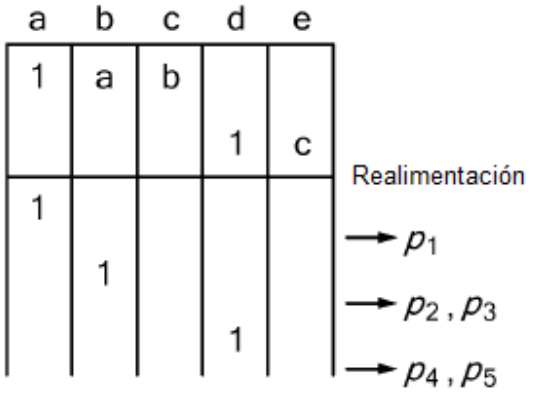

FIG. 14C