Universidad Politécnica de Cartagena
E.T.S. de Ingeniería de Telecomunicación

Espacio-Tele□o n^{0}2 (2011)
Revista de la ETSIT-UPCT

Códigos *Fountain* y su aplicación a las comunicaciones a través de las redes de baja tensión

Pedro J. Piñero Escuer y Pilar Manzanares López

Departamento de Tecnologías de la Información y las Comunicaciones.
Universidad Politécnica de Cartagena

pedrop.escuer@upct.es; pilar.manzanares@upct.es

Resumen

Los códigos *Fountain* son un tipo de codificación de fuente muy útil en canales de comunicación unidireccionales y en comunicaciones multicast. Además, presentan una serie de características que hacen pensar que pueden ser muy útiles en las redes de comunicaciones PLC (*PowerLine Communications*). A lo largo de este trabajo se describen detalladamente los diferentes tipos de códigos *Fountain* existentes actualmente y se describen dos aplicaciones de este tipo de códigos a las redes PLC.

Proyecto/Grupo de investigación:
Grupo de Ingeniería Telemática. Investigación financiada por la subvención de proyecto TEC2010-21405-C02-02 "Evaluation, Planning and Improvement of Key Technologies for the Future Internet: Knowledge and Transfer (CALM)" y también desarrollada en el marco del "Programa de Ayudas a Grupos de Excelencia de la Región de Murcia, de la Fundación Séneca, Agencia de Ciencia y Tecnología de la RM (Plan Regional de Ciencia y Tecnología 2007/2010)". Pedro J. Piñero Escuer también agradece a la Fundación Séneca la concesión de una beca predoctoral FPI (Exp. 13251/FPI/09).

Líneas de investigación: *Power-line Communications*; *Códigos Fountain*; *Redes in-home*.

1. Introducción

La extensa y amplia gama de aplicaciones existentes en el mundo de las comunicaciones genera diferentes escenarios, con diferentes características. Uno

de estos posibles escenarios sería aquel en el que la transmisión bidireccional entre emisor y receptor es imposible o símplemente indeseable. Muchos de los canales que se utilizan para la distribución de contenidos multimedia son unidireccionales; es decir, son canales en los que el receptor no puede enviar paquetes de control al transmisor. Algunos ejemplos de estos sistemas de telecomunicación son la televisión (en cualquiera de sus variantes: analógica, TDT o IPTV) o la radio digital. Otro tipo de servicios o aplicaciones en las que sería deseable eliminar los paquetes de control serían las aplicaciones multicast. En este caso, un transmisor desea enviar un mensaje a varios receptores al mismo tiempo, y como cada receptor experimenta unas pérdidas independientes a los demás, el número de paquetes de control puede llegar a ser muy elevado.

Teniendo en cuenta estas consideraciones, se consideró muy interesante la utilización de códigos *Fountain* [2] para la transmisión de la información digital a través de este tipo de redes. La idea principal en la que se basan este tipo de códigos puede plantearse mediante el siguiente símil: el transmisor sería una fuente de agua que es capaz de producir una cantidad infinita de gotas de agua. El receptor sería un recipiente que necesita recoger un cierto número de esas gotas para poder obtener la información. La ventaja principal que presentan este tipo de códigos es que el receptor puede recuperar la información sin importarle cuales de esas gotas haya recogido, tan solo necesita recibir la cantidad adecuada de gotas. Tenemos por tanto que los códigos Fountain deben cumplir las siguientes características:

1. El transmisor debe ser capaz de generar una cantidad potencialmente infinita de paquetes codificados a partir de la información que desea transmitir.

2. El receptor debe poder decodificar un mensaje formado por K paquetes a partir de cualquier conjunto de K' paquetes codificados, para un valor de K' ligeramente superior a K.

Las tres implementaciones más importantes que existen en la actualidad de este tipo de códigos son los códigos LT [1], los códigos *Raptor* [4] y los códigos *Online* [3]. Los códigos LT fueron los primeros en definirse, mientras que los códigos *Raptor* son una evolución de los códigos LT que consiguen reducir su coste computacional. Finalmente, los códigos *Online* son una alternativa basada en software libre a los códigos *Raptor*.

En este trabajo se estudia la aplicación de los códigos *Fountain* a las comunicaciones a través de los cables de baja tensión existentes en cualquier edificio. Las redes PLC (*Powerline Communications*), y concretamente su estándar más extendido, Homeplug AV [7], presentan algunos problemas que pueden ser resueltos con la ayuda de este tipo de códigos.

El resto del trabajo se estructura de la siguiente manera: en la sección 2 se describen detalladamente los tipos de códigos *Fountain* existentes actuálmente. La sección 3 muestra las aplicaciones estos códigos a las redes PLC y, finalmente, la sección 4 resume las conclusiones más destacadas de este trabajo.

2. Códigos *Fountain*

2.1. Códigos LT

Los códigos LT fueron los primeros que cumplían todas las características de los códigos *Fountain*. Fueron creados por Luby en 1998 y publicados por primera vez en 2002. La principal característica de estos códigos es que utilizan como mecanismo de codificación y decodificación un algoritmo muy simple basado en operaciones XOR.

Si utilizamos un código LT para transmitir un mensaje formado por k paquetes de entrada, cada símbolo codificado se genera mediante $O(\log(k\,\delta))$ operaciones. Para poder recuperar el mensaje original con una probabilidad $1-\delta$ se necesitan $k + O(\log^2(k\,\delta)\sqrt{k})$ símbolos codificados empleando un total de $O(k\,\log(k\delta))$ operaciones de media.

El procedimiento para crear un símbolo codificado se describe de la siguiente forma:

1. Se selecciona de manera aleatoria el grado d del símbolo que vamos a codificar utilizando para ello una distribución estadística determinada. El grado de un símbolo se define como el número de símbolos originales por los que estará formado. El diseño y análisis de la distribución de grado se explica posteriormente.

2. Se escogen de manera aleatoria d símbolos de entrada distintos.

3. El valor del símbolo codificado será el resultado de la operación XOR entre los d símbolos de entrada elegidos.

Para que el receptor pueda recuperar el mensaje original debe conocer el grado de cada símbolo codificado y los símbolos de entrada por los que esta formado. Por tanto, se debe implementar algún mecanismo para transmitir esta información al receptor. El procedimiento de decodificación que utilizan estos códigos se puede describir también de manera muy sencilla:

1. Se busca un símbolo codificado que solo esté formado por un único símbolo de entrada (es decir, de grado $d = 1$).

2. Eliminamos este símbolo de entrada de los demás símbolos codificados de los que forma parte realizando una operación XOR. De esta forma, reduciremos en una unidad el grado de dichos símbolos, por lo que es muy probable que se obtengan nuevos símbolos de grado uno. Al conjunto de símbolos de grado uno que están a la espera de ser procesados se denominan *Ripple*.

3. Se repite esta operación hasta que recuperemos todos los símbolos de entrada. Destacar que la decodificación falla si, antes de obtener todos los símbolos de entrada, en algún paso no hay símbolos codificados de grado uno. Este fenómeno puede minimizarse eligiendo convenientemente la distribución de grado a utilizar.

Una distribución de grado adecuada debe cumplir que el tamaño del *Ripple* no sea demasiado pequeño ni demasiado grande. Un tamaño excesivo puede provocar que varios símbolos correspondan al mismo símbolo de entrada, y que por tanto sean redundantes. Al contrario, si el *Ripple* es demasiado pequeño, puede llegar a desaparecer y provocar la interrupción del proceso de decodificación. Una buena solución a este problema es imponer como condición de diseño que los símbolos sean añadidos al *Ripple* a la misma velocidad que son procesados, es decir, que el tamaño medio del *Ripple* sea uno. Basándose en esta condición, Luby propone como distribución de grado ideal la *"Ideal Soliton Distribution"*, $\rho(\cdot)$ que tiene la siguiente forma:

$$\rho(i) = \begin{cases} \dfrac{1}{k} & \text{para } i = 1, \\ \dfrac{i}{i(i-1)} & \text{para } i = 2, \cdots, k. \end{cases} \tag{1}$$

2.2. Códigos Raptor

Los códigos *Raptor* son una extensión de los códigos LT que ofrecen tiempos de codificación y decodificación lineales. Estos códigos son capaces de recuperar un conjunto k de símbolos de entrada a partir de $k(1+\varepsilon)$ símbolos codificados con una probabilidad $1-\delta$, donde $\varepsilon = 4\delta/(1-4\delta)$. Cada símbolo codificado se genera mediante $O(\log(1/\varepsilon))$ operaciones, y los símbolos de entrada se recuperan con $O(k \cdot \log(1/\varepsilon))$ operaciones.

La idea principal de los códigos *Raptor* es concatenar una codificación FEC tradicional (*reed-solomon*, turbo códigos, etc.) con un código LT "débil" [1]. La utilización de una codificación LT "débil" provoca que algunos símbolos de entrada no puedan ser decodificados correctamente. En estos casos, el código FEC debe ser capaz de solucionar las deficiencias del código LT.

Supongamos que se desea transmitir un conjunto k de símbolos utilizando un código *Raptor*. El primer paso sería codificar los símbolos de entrada mediante el código FEC y generar n símbolos intermedios. A continuación, estos n símbolos intermedios se codificarían mediante el código LT "débil" para obtener los símbolos codificados. Lógicamente, para obtener los símbolos de entrada se deberá aplicar en primer lugar el mecanismo de decodificación del código LT y a continuación el mecanismo de decodificación del código FEC.

Para el diseño de un código *Raptor*, el primer paso es escoger la distribución de grado del código LT. El autor de los códigos *Raptor* propone como distribución de grado una variante de la *Soliton Distribution* que tiene la forma mostrada en la ecuación 2, donde $\mu = \varepsilon/2 + (\varepsilon/2)^2$ y $D = [4(1+\varepsilon)/\varepsilon]$.

$$\omega(x) = \frac{1}{\mu+1}\left(\mu x + \sum_{i=2}^{D} \frac{x^i}{(i-1)i} + \frac{x^{D+1}}{D}\right). \tag{2}$$

[1] Un código LT "débil" es un código LT en el que se reduce el grado medio de los símbolos codificados para reducir la complejidad del proceso de decodificación

El siguiente paso será el diseño del código FEC. Se puede demostrar que éste debe cumplir que:

- Su tasa de información debe ser $R = (1 + \varepsilon/2)/(1 + \varepsilon)$.
- El número de símbolos intermedios que debe generar debe ser $n = [k/(1 - R)]$ donde k es el número de símbolos de entrada.

Habitualmente se utilizan como códigos FEC los Tornado, los *Hamming* o los LDPC (*Low Density Parity Check*), aunque en muchos casos lo que se utiliza es una combinación de varios tipos de códigos.

2.3. Códigos Online

Otro tipo de códigos *Fountain* que surge de manera paralela a los dos anteriores son los códigos *Online*. Éstos, al igual que los códigos *Raptor*, consiguen tiempos de codificación y decodificación que aumentan de manera lineal con el número de símbolos de entrada.

Los códigos *Online* están definidos por dos parámetros, ε y q, además de por el tamaño de bloque. Un mensaje de k símbolos de entrada, podrá ser decodificado a partir de $(1 + 3\varepsilon)k$ símbolos codificados con una probabilidad de error dada por la expresión $(\varepsilon/2)^{q+1}$.

La estructura general de estos códigos se muestra en la figura 1. Podemos ver como el proceso de codificación se divide en un código exterior y un código interior. El código exterior genera unos símbolos auxiliares, que se añaden al mensaje original para generar un mensaje compuesto. A continuación, el código interior se comporta como uno del tipo LT pero con una función de distribución del grado (d) distinta, que se escoge de forma que se cumplan las especificaciones de este tipo de códigos.

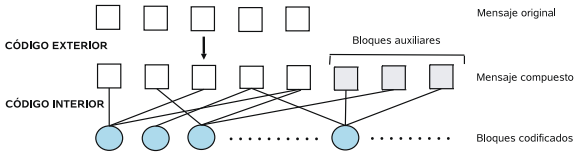

Figura 1: Estructura de los códigos Online.

En el primer paso del proceso de codificación, el código exterior genera $0,55q\varepsilon n$ símbolos auxiliares. Para cada símbolo del mensaje original se obtienen de manera uniformemente aleatoria q símbolos auxiliares. Cada símbolo auxiliar se calcula mediante la operación XOR de todos los símbolos del mensaje original que se le ha asignado. Posteriormente, estos símbolos auxiliares se concatenan al mensaje original, para formar un mensaje compuesto que tendrá un tamaño $n' = (0,55q\varepsilon n + 1)n$. Este código puede recuperar el mensaje original a partir de un porcentaje $1 - \varepsilon/2$ de los símbolos del mensaje compuesto.

El código interno depende de la siguiente distribución estadística:

$$\rho_1 = 1 - \frac{(1+1/F)}{(1+\varepsilon)}, \tag{3}$$

$$\rho_i = \frac{(1+\rho_1)F}{(F-1)i(i-1)} \quad i = 2, 3, \cdots, F, \tag{4}$$

donde F tiene el siguiente valor

$$F = \frac{\log(\varepsilon^2/4)}{\log(1-\varepsilon/2)}. \tag{5}$$

Esta distribución estadística se utiliza para obtener el grado d del símbolo codificado. El valor de este símbolo será la operación XOR de d símbolos del código exterior elegidos al azar. Cualquier conjunto $(1+\varepsilon)n'$ de los símbolos codificados es suficiente para recuperar un porcentaje $1-\varepsilon/2$ de los símbolos del mensaje compuesto.

El proceso de decodificación del código interno es similar al proceso seguido por los códigos LT. En primer lugar, se busca un símbolo codificado de grado uno y se obtiene uno de los símbolos de entrada, posteriormente, eliminamos este símbolo de los demás símbolos codificados de los que forma parte realizando una operación XOR. Se continúa con este procedimiento hasta obtener todos los símbolos del código exterior. En este punto, para obtener los símbolos del mensaje original, habría que emplear el decodificador asociado con el codificador externo. Nótese, sin embargo, que el algoritmo de decodificación es el mismo para ambos codificadores (externo e interno).

3. Aplicación de los códigos *Fountain* a las redes PLC

Las redes PLC (*Powerline Communications*) estan despertando recientemente mucho interes en la industria y en la comunidad científica. Este tipo de redes son muy sencillas de instalar y de ampliar y además, al utilizar los cables de baja tensión instalados en el edificio, su coste de instalación es extremadamente bajo. El estándar más aceptado dentro de la tecnología PLC es Homeplug AV (HP audio-video, o simplemente HPAV). Este estándar fue presentado por la *HomePlug Powerline Alliance* [8] en 2005 y proporciona un ancho de banda de hasta 150 Mbps sobre los cables de baja tensión existentes en cualquier edificio. Para la conexión de un dispositivo a la red HPAV se utilizan los adaptadores HPAV, que disponen de un interfaz Ethernet. Los dispositivos se conectan a la red eléctrica utilizando cualquier enchufe disponible en el hogar.

En este trabajo se han estudiado diversas aplicaciones de los códigos *Fountain* a las redes PLC. En las siguientes secciones se muestran los resultados obtenidos más relevantes.

3.1. Transmisión fiable de datos

Según el estándar, la tecnología HPAV puede establecer dos modos de transferencia distintos:

- Transferencias orientadas a la conexión con requerimientos de QoS (ancho de banda garantizado, limitación del *jitter* y latencia máxima). Servicio proporcionado mediante un sistema TDMA (*Time Division Multiple Access*).

- Transferencias no orientadas a la conexión (*conectionless*) que comparten un mismo canal de comunicaciones (*contention*). Este modo de transferencia es utilizado tanto por servicios asíncronos pero con requerimiento de QoS como por servicios *best-effort*. Se proporciona mediante un esquema CSMA/CA basado en prioridades..

Sin embargo, la mayoría de los modems existentes actuálmente en el mercado únicamente implementan el modo de transferencia basado en CSMA/CA, lo que hace que el canal de comunicaciones PLC sea *broadcast*, es decir, compartido. Esta realidad lleva a pensar en el uso de los códigos *Fountain* para la transmisión fiable de datos, ya que el protocolo TCP (tradicional) utilizado en estas aplicaciones presenta problemas importantes en canales compartidos.

Para evaluar el comportamiento de los códigos Fountain sobre las redes PLC se realizaron una serie de medidas sobre un escenario con seis ordenadores conectados a la misma fase eléctrica que los dispositivos HPAV. Dos de dichos ordenadores realizan una transmisión continua de paquetes UDP a otros dos ordenadores del conjunto. Los dos ordenadores restantes actúan en un primer escenario como transmisor y receptor de códigos *Fountain* y en un segundo escenario como transmisor y receptor del protocolo TCP respectivamente. La distancia del cableado eléctrico entre estos dos últimos ordenadores es de unos 45m, que es de las mayores distancias que se pueden encontrar en una vivienda o local convencional (aprox. 100 m^2), es decir, nos acercamos al peor caso posible. Con este escenario se consiguen dos objetivos: por un lado las fuentes de alimentación y los monitores de cada uno de los ordenadores introducían ruido a la red, y por otro, las transmisiones UDP provocaban pérdidas de paquetes como consecuencia del desbordamiento en los buffers de los módems HPAV.

La implementación de los códigos *Fountain* utilizada tiene como base la encontrada en [6], que utiliza los códigos *Online*. A dicha implementación se le ha añadido la posibilidad de transmisión de los paquetes codificados en red mediante UDP. Para las transmisiones TCP se utilizó la aplicación SCP[5] que se basa en este protocolo para la transmisión de los datos. Las medidas realizadas consistieron en la transmisión de ficheros de tamaño comprendido entre 1 y 20 MBytes midiendo en cada caso el tiempo necesario para llevar a cabo la transmisión. Cada una de las transmisiones se repitió 5 veces y se obtuvo el intervalo de confianza al 95 % que se representa, junto con los resultados, en la figura 2.

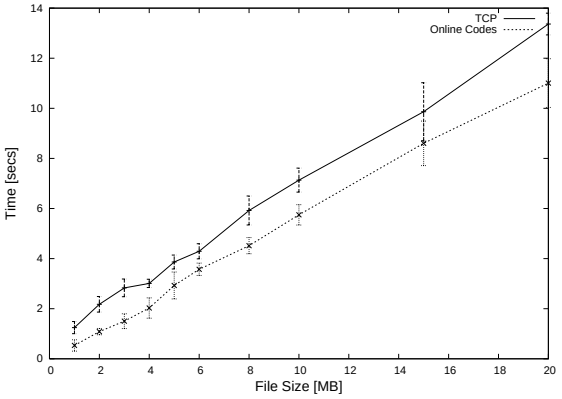

Figura 2: Duración de transmisión de datos utilizando TCP y códigos Online sobre un entorno PLC con ruido y dos transmisiones simultáneas. Intervalos de confianza al 95 %

Se puede observar como el tiempo necesario para transmitir el fichero con los códigos *Online* siempre es menor que el necesario para transmitirlo mediante la aplicación SCP. Esta diferencia de tiempos está en torno a 1 segundo para tamaños de fichero pequeños y aumenta hasta los 2 segundos cuando aumenta el tamaño del fichero. También se realizaron las pruebas con otras aplicaciones basadas en TCP como FTP, obteniendo velocidades de transmisión aún inferiores.

3.2. Implementación de un mecanismo de agregación de interfaces asimétricos

En la actualidad existen muchas alternativas para instalar una red en una vivienda, siendo además muy común que varias de ellas convivan en un momento dado. Esto a llevado a que algunas empresas estén fabricando una serie de módems *multihome* con la característica de que disponen de un conjunto de distintos interfaces: wireless, Ethernet, PLC. Una aplicación que se podría utilizar en este tipo de módems es la agrupación de los distintos interfaces.

El Kernel de Linux dispone ya de un módulo que permite agrupar interfaces Ethernet, denominado *Bonding* [9], con el que se pueden conseguir distintas ventajas: comunicación con un mayor ancho de banda, balanceo de carga, enlaces de backup,... . El problema que presenta este módulo, es que no está preparado para una conexiones asimétricas, y menos aún para conexiones cuya velocidad va variando con el tiempo, como en el caso de PLC, cuya velocidad fluctúa dependiendo del ruido del entorno. A continuación se describe una modificación del módulo *Bonding* que puede funcionar en estos entornos asimétricos, utilizando para ello una codificación *Fountain*.

Uno de los principales inconvenientes que se presenta a la hora de utilizar

un módulo *Bonding* asimétrico para la transmisión de datos es que los paquetes se recibirán casi con toda seguridad fuera de orden en el receptor, y este problema puede ser solucionado gracias a las características propias de los códigos *Fountain*.

El módulo *Bonding* presenta hasta siete modos de funcionamiento. De entre todos ellos, se va a considerar únicamente el modo *balance-rr*. En él los paquetes se transmiten en orden secuencial, desde el primer al último interfaz que forma el *Bonding*, consiguiendo balanceo de carga y tolerancia a fallos. Para conseguir que el módulo funcione correctamente con interfaces asimétricos se definió, dentro de la estructura principal del módulo, una variable utilizada para almacenar la velocidad mínima proporcionada al *Bonding*. Dicha variable es utilizada para definir el reparto de los paquetes para cada interfaz, de tal manera que la cantidad de paquetes a enviar en cada turno por cada tarjeta dependerá de la proporción de su velocidad con respecto a la velocidad mínima.

Para comprobar el funcionamiento del módulo se diseño un escenario con dos equipos con dos tarjetas de red cada uno. Para conectar los dos equipos se utilizaron dos modems PLC y un cable *fast-ethernet*. En primer lugar se realizaron varias transmisiones de ficheros entre 1 y 10MB por cada uno de los dos interfaces por separado utilizando el protocolo TCP. Posteriormente se agruparon los dos interfaces y se transmitieron los mismos ficheros por ambas interfaces conjuntamente utilizando para ello la misma implementación de los códigos *Online* que en el caso anterior. En la figura 3 se muestra la duración de cada transmisión junto con su intervalo de confianza al 95 %, obtenido a partir de cinco medidas.

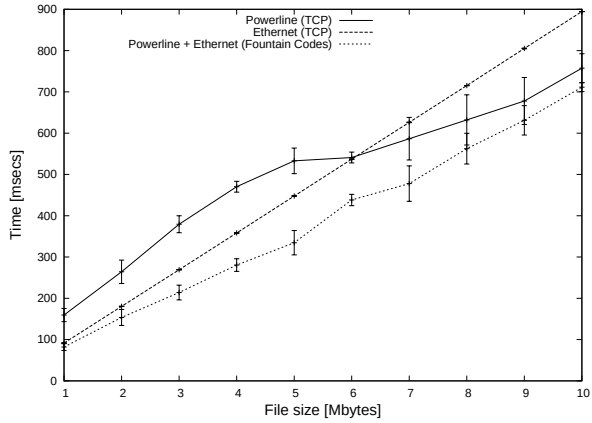

Figura 3: Duración de transmisión de datos utilizando TCP por cada interfaz por separado y códigos Online sobre los dos interfaces agregados. Intervalos de confianza al 95 %

Los resultados muestran que los tiempos obtenidos para el caso del módulo *Bonding* y los códigos *Online* son siempre menores que los obtenidos cuando

se transmiten los ficheros por cada interfaz por separado mediante TCP. Por otra parte, vemos que la ganancia obtenida disminuye a medida que aumenta el tamaño del fichero debido al alto consumo de memoria de los códigos *Online*. Para solucionar este problema sería conveniente dividir el fichero en archivos más pequeños y transmitirlos por separado.

4. Conclusiones

En este trabajo se han descrito detalladamente los diferentes tipos de códigos *Fountain* existentes. Este tipo de códigos pueden ser de gran utilidad en comunicaciones en las que no se dispone de canal de retorno o en comunicaciones *multicast*, ya que el receptor es capaz de decodificar un mensaje formado por K paquetes a partir de cualquier conjunto de K' paquetes codificados, para un valor de K' ligeramente superior a K.

Por sus características, este tipo de códigos también pueden ser muy útiles en las redes PLC. Concretamente, se ha estudiado la utilidad de los códigos *Fountain* para realizar transmisiones fiables de datos y para agregar interfaces de red asimétricos.

En el primer caso se ha comprobó que los códigos *Fountain* proporcionaban mejores resultados que el protocolo TCP a la hora de realizar transmisiones fiables de datos sobre redes PLC. Esto se debe a que el canal de comunicaciones PLC es compartido y el protocolo TCP presenta serios problemas en este tipo de canales.

En la segunda aplicacion se implementó un módulo del kernel de linux que permitía agregar interfaces de diferentes tecnologías para obtener mayores velocidades de transmisión, concretamente se unieron dos interfaces Ethernet y PLC. Debido a que la tasa de transmisión de la interfaz PLC varía según el ruido presente en la red eléctrica, los paquetes llegan al receptor desordenados. Para solucionar este problema se emplearon también los códigos *Fountain*, y se comprobó que mediante esta técnica se conseguían mayores velocidades de transmisión que empleando el protocolo TCP por cada interfaz por separado.

Referencias

[1] M. Luby, LT Codes, Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on, pages 271–280.

[2] D. J. C. MacKay. Fountain Codes. IEE Proc. Commun., vol. 152, no. 6, Dec. 2005.

[3] P. Maymounkov and D. Mazières. Rateless codes and big downloads. In Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22,2003, Revised Papers, pages 247–255.

[4] A. Shokrollahi, Raptor codes. IEEE/ACM Trans. Netw., 14(SI):2551–2567, 2006.

[5] scp-secure copy, 2009, `http://www.mkssoftware.com/docs/man1/scp.1.asp`.

[6] Implementation of Online Codes, 2009, `http://sourceforge.net/projects/onlinecodes/`.

[7] HomePlug AV Specification (2007).

[8] Homeplug Powerline Alliance. `http://www.homeplug.org`.

[9] The Linux Foundation, 2009, `http://linuxfoundation.org/en/Net:Bonding`