Utilización de códigos Fountain para la implementación de un Bonding asimétrico

P.J. Piñero Escuer, D. Montoro Mouzo, J.P. Muñoz Gea, P. Manzanares López
Departamento de Tecnologías de la Información y Comunicaciones.
Antiguo Cuartel de Antigones. Plaza del Hospital, Nº 1, 30202 Cartagena (Murcia)
E-mail: {pedrop.escuer, david.montoro, juanp.gea, pilar.manzanares}@upct.es

Resumen. *Debido a los recientes avances tecnológicos, las redes in-home están despertando mucho interés entre la industria y la comunidad científica. En la actualidad, hay muchas alternativas que pueden ser empleadas para establecer una red de este tipo y es muy común que varias de ellas estén presentes en una vivienda. Por tanto, una aplicación capaz de agrupar interfaces de diferentes tecnologías sería muy útil en este tipo de redes. El Kernel de Linux contiene un driver (o módulo), denominado Bonding, que permite agrupar diferentes interfaces Ethernet. Sin embargo, no está preparado para agrupar interfaces de diferentes tecnologías, y mucho menos para tecnologías de capacidad variable. En este trabajo se propone una variación de dicho módulo que utiliza los códigos Fountain para funcionar correctamente en escenarios asimétricos.*

1 Introducción

Se denominan redes in-home (o redes domésticas) a aquellas cuyo principal objetivo es la comunicación entre los diferentes dispositivos eléctricos o electrónicos presentes en una vivienda. Actualmente existen muchas tecnologías que se pueden emplear para establecer una red de este tipo. Estas tecnologías se pueden dividir en tres grandes grupos: cableadas, inalámbricas y No-New-Wires. Estas últimas hacen uso de las infraestructuras de cableado preexistentes en la vivienda para el intercambio de información. Entre ellas, la que más interés está despertando entre la industria y la comunidad científica es la tecnología PLC (Powerline Communications) que emplea la infraestructura de cableado de baja tensión.

En la actualidad es muy común que varias de estas tecnologías estén disponibles en una vivienda. Tenemos por tanto que sería muy útil una aplicación que fuera capaz de agrupar interfaces de diferentes tecnologías para conseguir un mayor ancho de banda, mayor tolerancia a fallos y mayores posibilidades de balanceo de carga.

El Kernel de Linux contiene un driver, denominado Bonding [1], que permite agrupar diferentes interfaces Ethernet. Sin embargo, no está preparado para agrupar interfaces de diferentes tecnologías, y mucho menos para tecnologías de capacidad variable como es el caso de PLC o de las comunicaciones inalámbricas. En este trabajo se presenta una modificación del driver Bonding que permite agrupar interfaces de diferentes tecnologías empleando los códigos Fountain [2]. Para que el módulo presentado funcione correctamente con tecnologías de capacidad variable, es también necesario algún procedimiento para medir la capacidad de las interfaces agrupadas. Por ello, en este trabajo también se ha desarrollado un sistema de medida de ancho de banda.

2 Agrupación de interfaces asimétricos: Bonding driver

El driver Bonding original proporciona un método para agrupar múltiples interfaces Ethernet en una sola interfaz lógica. Dicho driver tiene hasta 7 modos de funcionamiento distintos que proporcionan diferentes ventajas al usuario. Entre ellos, el más interesante es el modo "Round-Robin", que transmite los paquetes por las diferentes interfaces de manera secuencial consiguiendo un aumento en el ancho de banda del enlace. Sin embargo, cuando se utilizan interfaces de diferentes tecnologías, el módulo transmitirá por todas ellas a la velocidad de la interfaz más lenta, consiguiendo evitar de esta forma un desorden importante de los paquetes en recepción.

2.2 Bonding driver modificado

Para conseguir que el driver pueda funcionar con interfaces de diferentes tecnologías, se ha definido una nueva variable que almacenará la velocidad correspondiente a la interfaz más lenta de las agrupadas. Esta variable se utilizará posteriormente para calcular la cantidad de paquetes que se enviará por cada interfaz. En concreto, en cada turno, el driver transmitirá por cada interfaz un número de paquetes igual al número de veces que su velocidad de transmisión supera la velocidad de transmisión mínima almacenada en la variable.

El ancho de banda de las diferentes interfaces se medirá de manera periódica. Sin embargo, esta actualización en la distribución de los paquetes únicamente soluciona las variaciones a largo plazo de la capacidad. En el caso de tecnologías de capacidad variable es necesario algún mecanismo para solucionar los problemas asociados a las variaciones instantáneas de capacidad. Esto se consigue mediante la utilización de códigos Fountain.

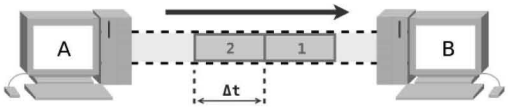

Fig. 1 Fundamentos de la dispersión de paquetes

3 Procedimiento de estimación de ancho de banda.

En este trabajo también se ha desarrollado una herramienta de medida de ancho de banda basada en técnicas de dispersión de paquetes [3]. El la Fig. 1 se muestra un ejemplo ilustrativo de este concepto cuando se quiere medir la capacidad de un enlace entre el equipo A y el equipo B. Podemos comprobar como utilizando el tiempo entre llegadas de los dos paquetes bajo estudio, se puede obtener la capacidad del enlace sin más que aplicar la expresión (1). Sin embargo, las medidas de tiempo en los sistemas informáticos están limitados por la resolución de reloj. Por lo tanto, para conseguir mejores estimaciones de capacidad se debe enviar más de dos paquetes y medir el tiempo entre llegadas del primer y último paquete respectivamente (ver expresión (2)).

$$c = \frac{L}{\Delta t} \quad (1) \qquad c = \frac{L(n-1)}{\Delta t} \quad (2)$$

También se ha añadido la posibilidad de repetir la medida un número determinado de veces para evitar posibles desviaciones estadísticas asociados a la realización de una sola medida. Estas desviaciones son muy habituales en tecnologías que utilizan un medio compartido como PLC. Tenemos por tanto que el sistema permite la configurar el número de paquetes por medida (n), el número de repeticiones de la medida (m) y el periodo de tiempo entre reconfiguraciones (T).

Finalmente, la capacidad obtenida será compartida con el driver Bonding a traves de la interfaz /proc que proporciona el sistema operativo.

4 Fountain codes

Se denominan códigos *Fountain* a aquellos que cumplen las siguientes carácteristicas:

- El transmisor debe ser capaz de generar una cantidad infinita de paquetes codificados a partir de la información que desea transmitir.
- 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, los códigos *Raptor* y los códigos *Online*.

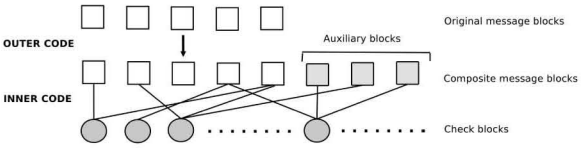

Fig. 2 Estructura de los códigos Online

4.1 Online Codes

Los códigos Online [4] están definidos por dos parámetros, ε y q, además de por el tamaño de bloque. Estos parámetros se emplean para encontrar un equilibrio entre prestaciones y complejidad de la codificación. Un mensaje de k símbolos de entrada, podría 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 esto códigos se muestra en la Fig. 2. El proceso de codificación se divide en un código exterior y un código interior. El código interior se encarga de la generación de los bloques codificados, también llamados *check blocks*. Cada *check block* se calcula como la operación XOR de d bloques del mensaje a transmitir escogidos de manera uniformemente aleatoria (d representa el grado del *check block*). La probabilidad de que d=i viene dada por una determinada distribución de probabilidad detallada en [4]. Debido a que la elección de los bloques del mensaje original es aleatoria, puede ocurrir que alguno de dichos bloques no se seleccione en la codificación. Para solucionar este problema se añade una codificación previa (código exterior).

El proceso de decodificación también se divide en dos pasos. En la primera etapa se deben recuperar un porcentaje 1-ε/2 de los bloques del mensaje compuesto. El código externo posee la propiedad de que el conocimiento de esta fracción de los bloques del mensaje compuesto permite recuperar el mensaje original gracias a la redundancia introducida. El proceso de decodificación que utiliza para obtener los bloques del mensaje compuesto a partir de los *check blocks* recibidos es el siguiente:

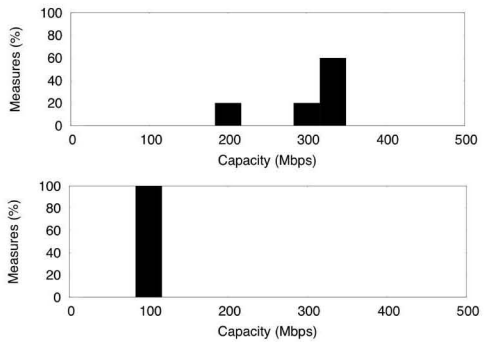

Fig. 3 Resultados medidas ancho de banda PLC y Fast-Ethernet respectivamente

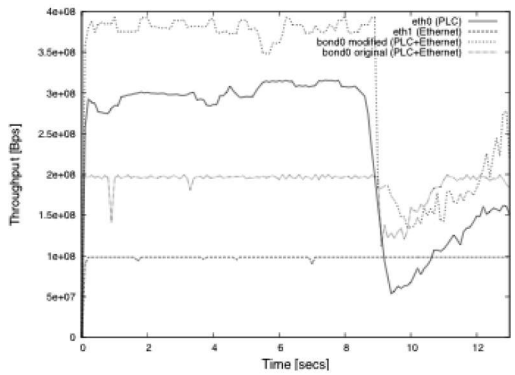

Fig. 4 Ancho de banda obtenido por cada interfaz. Fuente de ruido conectada a la red eléctrica en el segundo 8.

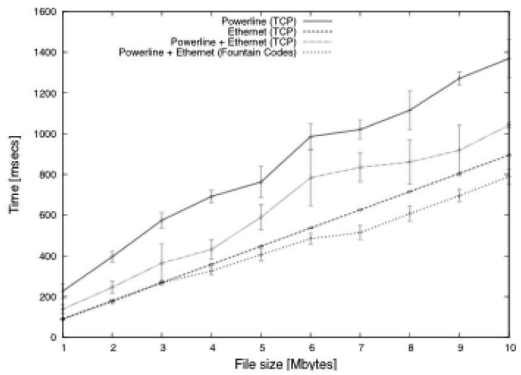

Fig. 5 Tiempo necesario para tranmitir ficheros entre 1 y 10 MB utilizando TCP y códigos Fountain. Intervalos de confianza al 95%

- Encontrar un *check block* que solo tenga un bloque adyacente (d=1) y recuperar dicho bloque.
- Eliminar el bloque recuperado de los demás *check blocks* de los que forma parte. Esto hace que el grado de los *check blocks* que contenían el bloque recuperado se reduzca en una unidad y que aparezcan nuevos bloques de grado uno.
- Continuar con este proceso hasta recuperar la fracción necesaria de los bloques del mensaje compuesto. Una vez obtenidos los bloques del mensaje compuesto, se pueden recuperar los bloques del mensaje original aplicando el mismo procedimiento.

5 Evaluación.

Para comprobar el correcto funcionamiento del sistema de agrupación de interfaces propuesto se utilizó un escenario compuesto por dos equipos conectados entre sí por una interfaz PLC y otra Fast-Ethernet respectivamente. Las dos interfaces se agruparon utilizando el módulo desarrollado. En la Fig. 3 se muestra el resultado de la estimación de ancho de banda sobre cada interfaz.

Con estos resultados la distribución de paquetes obtenida sería de 3 a 1. Utilizando esta distribución de paquetes, y transmitiendo una gran cantidad de información entre los ordenadores (con la ayuda de un socket del tipo Raw) se obtiene la capacidad mostrada en la Fig. 4 para cada caso. Vemos también como en el segundo 8 se ha conectado una fuente de ruido a la red eléctrica y se observa que en todos los casos la velocidad obtenida con el módulo desarrollado es aproximadamente la suma de las dos interfaces agrupadas. Los resultados también se han comparado con el bonding original, con el que se obtiene una capacidad igual al doble de la capacidad de la interfaz más lenta como era de esperar.

Por último, se realizó la transmisión de ficheros de diferentes tamaños entre los equipos utilizando TCP y códigos Fountain sobre ambas interfaces y TCP sobre las interfaces aisladas. El tiempo necesario para realizar cada transmisión se muestra en la Fig. 5.

En este caso se observa como los resultados obtenidos con los códigos Fountain sobre las interfaces agrupadas son siempre superiores a los obtenidos por el protocolo TCP cuando se utilizan las dos interfaces por separado.

6 Conclusiones

En este trabajo se ha desarrollado un driver de linux que permite la agrupación de interfaces de diferentes tecnologías. Este driver emplea un procedimiento de medida de ancho de banda basado en técnicas de dispersión de paquetes y codigos Fountain para evitar los problemas que se presentan cuando se emplean tecnologías de capacidad variable. El sistema propuesto se ha evaluado en un escenario con dos equipos conectados mediante dos interfaces PLC y Fast-Ethernet respectivamente. Se ha realizado la tranmisión de ficheros de diferentes tamaños entre ambos equipos y se ha comprobado que los resultados mejoran sustancialmente a los obtenidos cuando se emplea el protocolo TCP en cada interfaz por separado.

Agradecimientos

Este proyecto de investigación ha sido apoyado por la subvención de proyecto TEC2010-21405-C02-02/TCM (CALM) y también ha sido desarrollada en el marco del "Programa de Ayudas a Grupos de Excelencia de la Región de Murcia", de la Fundación Seneca. Pedro J. Piñero y David Montoro también agradecen a la Fundación Séneca la concesión de una beca predoctoral FPI (Exp. 16503/FPI/10) y una beca predoctoral asociada al proyecto FORMA (Exp. (Exp. 17541/BSCF/11) respectivamente.

Bibliografía

[1] *http://linuxfoundation.org/en/Net:Bonding*, 2010.
[2] D. J. C. MacKay, *Fountain Codes*, IEE Proc. Commun., vol. 152, no. 6, 2005.
[3] V. Jacobson, *Congestion avoidance and control*, in Proceedings of SIGCOMM 88. Stanford, CA, 1988.
[4] P. Maymounkov and D. Mazi`eres, *Rateless codes and big downloads*, In Peer-to-Peer Systems, Second International Workshop, Berkeley, CA, USA, Revised Papers, pages 247-255, 2003