Monitorización de redes 802.11 con Linux

Luis Rodero Merino, Miguel Ángel Ortuño Pérez,
Jesús M. González Barahona, Vicente Matellán Olivera
Grupo de Sistemas y Comunicaciones
Departamento de Informática Estadística y Telemática, Universidad Rey Juan Carlos
C/ Tulipán s/n, 28933, Móstoles, Madrid
Telf: 91-6647400, Fax: 91-4887049
E-mail: {lrodero, mortuno, jgb, pheras}@gsyc.escet.urjc.es

Abstract

Este artículo presenta un API desarrollado en Linux para la monitorización de redes inalámbricas 802.11b. Este API puede ser utilizado por aplicaciones que necesiten monitorizar la red inalámbrica en la que se encuentra el nodo, información que es útil por ejemplo para realizar roaming entre puntos de acceso. El API depende del manejador del dispositivo inalámbrico que se esté usando, y de que éste implemente las llamadas extensiones inalámbricas para Linux. Por ello incluimos también un pequeño resumen de los manejadores disponibles para Linux, incidiendo en si tienen o no capacidades de monitorización.

1. Introducción

La monitorización es un concepto fundamental en redes inalámbricas, incluidas las redes 802.11[1]. A partir de la monitorización de los PA (puntos de acceso) en rango se puede obtener información de la calidad de la señal desde cada uno de esos puntos, lo que tiene importantes aplicaciones:

- Crear mapas de cobertura.
- Estimar la posición del nodo a partir de la información de la calidad de señal de distintos PA.
- Implementar y probar distintas políticas de *roaming* basadas en los datos sobre la calidad de señal de los PA.

Aunque existen numerosas soluciones para realizar la monitorización, éstas son soluciones propietarias y difíciles de integrar con otras aplicaciones. Para crear nuevas aplicaciones que necesiten monitorizar la red se necesita un API (*Application Programmer Interface*, Interfaz de Programa de Aplicación) que haga de intermediario entre la aplicación y el dispositivo inalámbrico.

En este artículo presentamos un API desarrollado en Linux[2] cuyo objetivo es proporcionar una forma sencilla de obtener información de monitorización. Se comenzará con una breve descripción de las extensiones incluidas en Linux para el manejo de dispositivos inalámbricos y una enumeración de los manejadores más importantes. Después se introducirá el API. Finalmente resumiremos cuales son las limitaciones presentes aún en Linux para el trabajo con dispositivos inalámbricos.

2. Extensiones inalámbricas para Linux

Las extensiones inalámbricas son básicamente APIs orientadas al manejo de dispositivos inalámbricos en Linux. Extienden las APIs ya existentes para el manejo de redes.

Al trabajar con estas extensiones podemos distinguir tres niveles. El primer nivel serían las modificaciones hechas al kernel de Linux para definir y soportar esas extensiones. El segundo serían los manejadores, que han de implementar esas extensiones. El tercero serían las aplicaciones que utilizan las extensiones. Mencionamos ahora brevemente cuales son las modificaciones al kernel. Los manejadores son descritos en el punto 3.

2.1. Modificaciones al kernel

Las modificaciones al kernel han sido incluidas desde las versiones 2.0.30 y 2.1.17. Distinguimos dos:

- `/proc/net/wireless` Los accesos a este 'fichero' nos dan información acerca del estado de la red.
- `/usr/include/linux/wireless.h` En este fichero se definen las llamadas y estructuras de datos para el manejo de un dispositivo inalámbrico.

3. Manejadores para tarjetas inalámbricas 802.11 en Linux

Cualquier software que utilice un dispositivo inalámbrico dependerá de la funcionalidad implementada en el manejador. Se presentan a continuación algunos de los manejadores disponibles para Linux. Para una información más detallada consultar el *Linux Wireless LAN HowTo* [3].

No es fácil tener una visión completa de todos los manejadores existentes en Linux. Aquí intentaremos describir algunos de los importantes. También indicaremos si incluyen funcionalidad para la

monitorización. Si no se indica lo contrario, estos manejadores son *libres*[4].

3.1. Manejadores

Distinguiremos según los tipos de dispositivos más extendidos: los Wavelan y Wavelan IEEE de Lucent, y los basados en el *chipset* PrismII de Intersil.

- *Manejadores Wavelan* Manejador para dispositivos Wavelan, ya algo anticuados. No soporta totalmente el estándar 802.11.
  - *Manejador wavelan.o* Manejador algo antiguo, tiene una versión (*wavelan_cs.o*) para PCMCIA. No implementa *roaming* ni monitorización.
- *Manejadores Wavelan IEEE(Orinoco)* WavelanIII es un producto de Lucent, perteneciente en la actualidada a Proxim. Es una evolución sobre Wavelan, aunque tiene poco que ver con su predecesor. Lucent lo ha renombrado a Orinoco. Sólo existe en dispositivos PCMCIA. Lucent creó la librería de bajo nivel HCF (*Hardware Control Functions*) para trabajar con ellos, de la que liberó una versión limitada llamada HCF-light.
  - *Manejador wvlan_cs.o* Primer manejador para dispositivos WavelanIEEE, es un proyecto discontinuado. No implementa *roaming* ni monitorización.
    1. *Manejador orinoco.o* Reescritura del *wvlan_cs.o*, sin utilizar HCF-light (que es compleja y puede provocar errores). Ganó robustez sobre *wvlan_cs.o*. Añade soporte básico para tarjetas PrismII. No implementa *roaming* ni monitorización.
    2. *Manejador wavelan2_cs.o* Manejador de Lucent, sobre la HCF (recordemos que es más completa que la *Open Source*). No implementa *roaming* ni monitorización.
    3. *Manejador mwavelan_cs.o*[5][1] Añade capacidades de monitorización al *wavelan2_cs.o.*
    4. *Manejador mwvlan_cs.o* Se basa en el manejador *wvlan_cs.o*, al que añade monitorización. Parecido al *mwavelan.o*, con la diferencia de que es totalmente *Open Source.*
    5. *Manejador morinoco.o* Añade al *orinoco.o* soporte para monitorización.
- *Manejadores PrismII* El PrismII es un *chipset* construido por Intersil, que lo ofrece a fabricantes de dispositivos wireless (Compaq, Ovislink...). Otros fabricantes (como Lucent) pueden incluir parte de este *chipset* en sus productos pero con su propio firmware, luego no se manejan de la misma forma.
  - *Manejador hostap.o* Tiene versión para PCMCIA (*hostap_cs.o*). Este manejador es muy completo y utiliza las posibilidades del PrismII que permiten configurar el dispositivo como un punto de acceso. Incluye posibilidades de monitorización y *roaming.*
  - *Manejador p80211.o* Manejador del proyecto WaveLAN Linux. Este proyecto intenta 'crear una red inalámbrica completa basada en estándares'. Da poco soporte a las extensiones inalámbricas.

[1] Hemos añadido funcionalidad a este manejador para que pueda decir a qué punto de acceso está conectado el dispositivo.

4. API de programación

Hemos construido un API en lenguaje C por encima de las extensiones inalámbricas que realiza algunas tareas de monitorización. Puede ser utilizado con cualquier manejador que implemente las extensiones inalámbricas. Proporciona la siguiente información sobre cada punto de acceso en rango:

- Dirección MAC del punto de acceso.
- Calidad del enlace (%)
- Nivel de señal (dBm)
- Nivel de ruido (dBm)
- Bandera *updated* indica si se han cambiado los datos desde la última lectura.

4.1. Estructuras de datos del API

Dentro del API definimos dos estructuras:

- Estructura que almacena estadísticas de un punto de acceso concreto.

```
struct AccessPointData{
      /* Dirección MAC del AP */
      struct sockaddr MAC;
      /* Calidad del enlace */
      int link_quality;
      /* Nivel de la señal */
      int signal_level;
      /* Nivel de ruido */
      int noise_level;
      /* Dato actualizado desde
       * última lectura */
      int update;
}
```

- Estructura que almacena estadísticas de una colección de PA.

```
struct ListAccessPointsData {
      /* Núm de Aps en la lista */
      int length;
      /* Lista de Aps */
      struct AccessPointData
listAP[IW_MAX_AP];
}
```

4.2. Funciones del API

Se han implementado las siguientes funciones para el manejo del dispositivo inalámbrico. Todas tienen como parámetro el nombre de la interfaz de red en la que está trabajando el dispositivo:

- `listAccessPoints` Devuelve información sobre todos los PA dentro del rango del dispositivo.
```
struct ListAccessPointsData*
   listAccessPoints
   (char *interface_name);
```
- `getESSID` Devuelve el ID de la red (ESSID) en la que trabaja el dispositivo en ese momento.
```
char* getESSID
        (char *interface_name);
```
- `setESSID` Para fijar el ID de la red en la que ha de trabajar el dispositivo.
```
int setESSID
        (char *interface_name,
         char *ESSID);
```
- `connectToANY` Fija el identificador de la red (ESSID) a *any*. Es necesario que el dispositivo se conecte a esta red si se desea monitorizar todos los PA en rango independientemente de la red a la que pertenezcan.
```
int connectToANY
        (char *interface_name);
```
- `connectedAP` Devuelve información sobre el punto de acceso al que está conectado el dispositivo.
```
MACaddress connectedAP
        (char *interface_name);
```

Otras funciones para fijar el nombre del nodo, etc., son fácilmente implementables. Las funciones `setESSID` y `connectToANY` necesitan privilegios de superusuario o *root*.

5. Ejemplo de resultados obtenidos

Mostramos ahora algunos resultados obtenidos con una aplicación de monitorización que utiliza el API descrito. La aplicación fue configurada para tomar medidas durante 1000 segundos en intervalos de 10 segundos.

La Fig. 1 muestra las medidas de calidad de señal obtenidas para cada uno de los PA detectados.

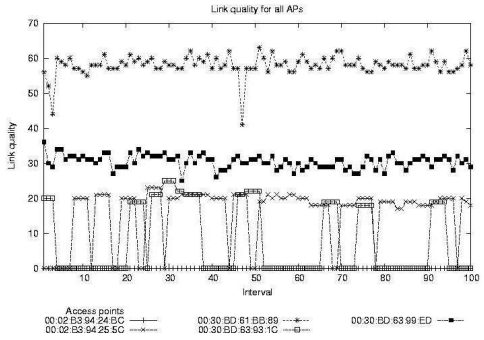

Figura 1 Calidad de la señal

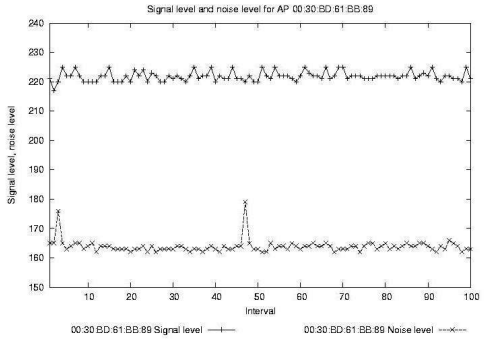

Figura 2 Niveles de señal y ruido

La Fig. 2 muestra los niveles de señal y ruido de un punto de acceso.

6. Limitaciones

Quizá la limitación más importante en la actualidad sea la imposibilidad de hacer *roaming* entre PAs a voluntad en Linux, al menos con la mayoría de manejadores disponibles. Ha de ser el hardware o firmware del dispositivo el que decida cuando hacer *roaming* según su propia política.

7. Conclusiones y trabajo futuro

Aunque el soporte para aplicaciones sobre redes inalámbricas en Linux no es completo, los últimos manejadores implementan la suficiente funcionalidad como para realizar tareas de monitorización sobre cobertura, calidad de la señal, etc., con las herramientas actuales. El API presentado en este artículo es otro paso para facilitar esas mismas tareas.

Entre las posibles tareas que pueden realizarse en el futuro se incluiría la mejora de los manejadores para aumentar su funcionalidad.

8. Referencias

[1] IEEE 802.11, 1999 Edition (ISO/IEC 8802-11:1999)

[2]Daniel Pierre Bovet Understanding the Linux Kernel O'Reilly & Associates, November 2000 ISBN: 0596000022

[3]Jean Tourrilhes Linux Wireless LAN Howto 2003, http://www.hlp.hp.com/personal

[4]Richard M. Stallman Free Software Definition 1996, http://www.gnu.org/philosophy/free-sw.html

[5]Moustafa A. Youssef A New GPL Driver for the Wavelan IEEE/Orinoco 2001, http://www.cs.umd.edu/~moustafa/mwavelan/mwavelan.html