Escola Politècnica Superior
de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FIN DE CARRERA

TÍTULO DEL TFC: Agrupamiento de MANETS

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Telemática

AUTOR: Anabel López Toré

DIRECTOR: Roc Messeguer Pallarès

FECHA: 11 de diciembre de 2009

Título: Agrupamiento de MANETS

Autor: Anabel López Toré

Director: Roc Messeguer Pallarès

Fecha: 11 de diciembre de 2009

Resumen

En este trabajo se presenta un algoritmo capaz de realizar grupos automáticos entre dispositivos activos dentro de una red de tipo MANET. Este algoritmo utilizará como herramienta de posicionamiento un software libre llamado PlaceLab.

Tras el desarrollo del algoritmo de agrupamiento realizamos un estudio del comportamiento del algoritmo en diferentes entornos controlados, para poder evaluar su funcionalidad.

Veremos por tanto de qué manera se realiza la agrupación, los diferentes grupos obtenidos, y cómo utilizando las tecnologías WiFi y Bluetooth, seleccionamos los nodos de la red que van a formar parte del proceso de agrupamiento.

A partir del análisis de las diferentes pruebas, se han extraído una serie de conclusiones que se pueden resumir en que el algoritmo desarrollado es capaz de realizar la agrupación de diversos nodos de una misma red MANET.

Title: Clustering of MANETS

Author: Anabel López Toré

Director: Roc Messeguer Pallarès

Date: November 11th 2009

Overview

The goal of this paper is introduce an algorithm which is able to make automatic clustering among different users involved in a MANET. This algorithm uses as a positioning tool free software called PlaceLab.

After the development of the clustering algorithm we studied his behavior in different context in order to evaluate his functionality.

We'll see then, how we do that clustering and the different behavior of the obtained groups, and how using WiFi and Bluetooth technologies we select the nodes which are going to be part of the process of clustering.

Once analyzed the results of the different proves we obtained some conclusions that can be summarized like the developed algorithm is able to make clustering among different users of the same network.

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

En este apartado se definirán cuáles son las razones por las cuales se ha decidido llevar a cabo este proyecto. Asimismo, veremos también cuáles son las necesidades en relación a la situación actual en la que nos encontramos, que son las que se deberán cubrir con el sistema que se va a desarrollar durante este proyecto de fin de carrera.

1.1 Marco del proyecto

Si por algo se ha caracterizado el mundo tecnológico tras el increíble éxito de Internet, ha sido por el auge de los servicios móviles de cualquier tipo. Cada vez más, los usuarios demandamos poder estar interconectados mientras nos desplazamos libremente y sin la necesidad del uso de cables. La era de la tecnología inalámbrica desde sus inicios en los años 90 ha avanzado de manera exponencial, haciendo cada vez más importante el desarrollo de nuevos protocolos, nuevos servicios o nuevas formas de interconectar los dispositivos.

Hoy en día, una de las topologías que se está estudiando e intentando mejorar son las redes MANET (Mobile Ad-hoc Network). Las redes MANET son redes ad-hoc móviles sin infraestructura, donde cada nodo actúa simultáneamente de nodo final y de router. De ésta forma, continuamente se están introduciendo nuevos nodos a la red y, al mismo tiempo, otros desaparecen de la misma.

Debido a la topología variable y dinámica de este tipo de redes, frecuentemente se da el caso que la cantidad de nodos interconectados llega a ser muy grande. Sin embargo, puede ser interesante poder diferenciar distintos grupos más pequeños dentro de una misma MANET. Por ejemplo, una empresa puede estar interesada en que se formen grupos por cada departamento, o a una universidad podría interesarle que se formasen grupos de trabajo en las clases.

Con este proyecto se quiere automatizar este proceso de agrupamiento de los distintos nodos de la MANET. Para ello se diseñará un algoritmo que, utilizando sistemas de posicionamiento y basándose en la posición de cada uno de los nodos, será capaz de realizar la agrupación de los mismos.

1.2 Objetivos

El objetivo principal de este proyecto es el estudio y el diseño de un algoritmo integrable en PlaceLab, capaz de realizar el agrupamiento automático de dispositivos activos en una red de tipo MANET, de manera que podamos actualizar dinámicamente los grupos previendo las bajas o las nuevas incorporaciones a nuestra red.

En relación a lo comentado en el párrafo anterior, para detectar los PCs que van a formar parte de nuestra agrupación, se realizan dos procesos distintos. Por un lado, mediante tecnología WiFi detectaremos aquellos nodos que forman parte de nuestra red MANET, y por otro, utilizando Bluetooth puesto que es de corto alcance, realizaremos un filtrado de los PCs que se encuentran fuera de la zona de acción.

La ventaja de poder generar grupos automáticamente es que la agrupación se realiza de forma totalmente transparente al usuario, de esta forma no tendrá que realizar ninguna acción para formar parte de cualquier grupo.

El agrupamiento se basa en la optimización de la calidad - distancia entre los nodos, y en virtud de esto se escoge la forma más apropiada en la que los nodos deben agruparse.

1.3 Estructura de la memoria

En el siguiente capítulo del trabajo, el capítulo dos, se explica de forma general los conceptos previos necesarios para la mayor asimilación de la tesis del trabajo, como funciona un sistema de localización, más concertantemente uno basado en WiFi, sus características, objetivos y aplicaciones.

En el tercer capítulo se hará un breve análisis de los requisitos que queremos cubrir con el desarrollo del algoritmo y el estudio de su comportamiento.

En el capítulo cuatro se detalla todo el desarrollo técnico del proyecto, tanto la implementación de las funciones como las pruebas realizadas en los diferentes escenarios que se han escogido. De este modo finalmente podremos establecer las diferencias existentes al agrupar en los diferentes entornos estudiados.

En quinto capítulo figuran las conclusiones respecto a los objetivos definidos, la comparación de los resultados obtenidos y por último se hace una relación de los posibles trabajos futuros y ampliaciones del proyecto.

Como parte de los Anexos de este trabajo, figuran los códigos fuente de varias clases específicas de PlaceLab que se han usado para la función de agrupamiento.

CAPÍTULO 2. CONCEPTOS PREVIOS Y HERRAMIENTAS

2.1 Conceptos básicos

En los últimos años, el uso de la tecnología inalámbrica ha crecido de forma considerable, y día a día se desarrollan y comercializan nuevos dispositivos pensados para este tipo de tecnología, tales como ordenadores portátiles, PDAs, GPS, teléfonos móviles, etc. Esta tecnología permite al usuario una alta movilidad con un rendimiento eficiente y económico.

Existen dos tipos de redes móviles inalámbricas: redes con infraestructura y redes sin infraestructura, estas últimas son más conocidas como redes ad hoc.

Las redes con infraestructura también conocidas como redes centralizadas, son redes en las que la comunicación entre los dispositivos de la red es controlada y gestionada mediante un nodo central o Access Point.

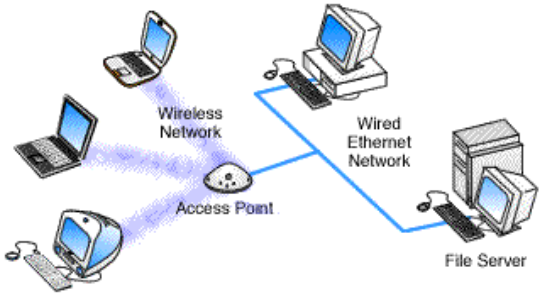


Figura 2.1. Ejemplo de red con infraestructura

En cambio, las redes sin infraestructura (o redes ad hoc), son redes formadas por elementos móviles que están conectados entre sí de forma arbitraria y dinámica, y se caracterizan porque no necesitan de un nodo central para gestionar la comunicación. Un ejemplo de redes sin infraestructura que se están estudiando hoy en día son las redes MANET.

Figura 2.2. Ejemplo de red ad hoc

2.1.1 Redes Móviles Ad-Hoc (MANET)

Las redes móviles ad hoc, conocidas por sus siglas en inglés, MANET ("Mobile ad hoc networks"), son redes inalámbricas y móviles, sin infraestructura, sin puntos de acceso, donde cada nodo desempeña simultáneamente las funciones de nodo final y de router. En este nuevo entorno los nodos participan en la toma de decisiones, realizando funciones de mantenimiento y encaminamiento de la red.

Figura 2.3. Ejemplo de MANET

Las principales características de este tipo de red ad-hoc son las que vamos a ver a continuación:

- *Topología variable*: en redes ad-hoc, se asume que una mayoría de los nodos son móviles, desplazándose libremente por el entorno. Además la tipología de la red cambia constantemente ya que los enlaces entre los nodos se crean y se destruyen dinámicamente.

- *Enlaces inalámbricos*: la comunicación en este tipo de redes se lleva a cabo de manera inalámbrica, caracterizadas por anchos de banda reducidos y más propensos a errores que los enlaces cableados.

- *Baterías limitadas*: debido a esta limitación, para un buen uso de estas, las transmisiones son de baja potencia y de alcances limitados.

- *Seguridad limitada:* Las redes móviles inalámbricas son generalmente más propensas a las amenazas de seguridad física que las redes cableadas.

Puesto que la topología de este tipo de redes está descentralizada, utilizando redes MANET se eliminan las restricciones de movilidad que pueden tener una infraestructura, que obliga a todos los usuarios a estar dentro del alcance del Access Point. En una MANET la red la conforman todos y cada uno de los dispositivos que forman parte de ella, de manera que se podría decir que cada nodo actúa como "centro" de la red.

2.1.2 Sistema de localización: Objetivo y aplicaciones

En los últimos años se ha producido un crecimiento importante en la demanda de servicios de posicionamiento, lo que demuestra la gran aceptación y proyección de futuro de los sistemas de localización.

Un sistema de localización se basa en la combinación de tres elementos: tecnologías de posicionamiento, un medio de comunicación para el intercambio de información entre unidades móviles y el centro de control, y un software capaz de procesar información cartográfica.

El objetivo principal de un sistema de localización es poder visualizar en tiempo real la posición del usuario sobre un mapa, de esta forma se puede utilizar el sistema para cualquier tipo de aplicación en la que se necesite la ubicación exacta de un usuario. Por esta razón, dado que se trata de un sistema en tiempo real, es muy importante mantener al usuario constantemente actualizado sobre su posición.

La localización de los usuarios se logra gracias a los aparatos que empleamos: los PC usan direcciones IP o punto de acceso WiFi; los teléfonos móviles, las antenas; y hoy en día los localizadores GPS son los más frecuentes.

A continuación podemos ver diferentes aplicaciones para un sistema de localización:

- Información de tráfico y turismo: esta quizá es la aplicación más extendida hoy en día. Cada vez más personas recurren a los sistemas de localización para determinar su posición en la carretera y conocer el detalle del tráfico en cada momento.

- Servicios sanitarios y de emergencia: los sistemas de localización facilitan el seguimiento de pacientes y equipos médicos en instalaciones sanitarias; así como el seguimiento de ancianos en una residencia, discapacitados, invidentes o personas con problemas de movilidad.

- Servicios de activación automática: Actualmente se tiende a la automatización de todos los sistemas, esta automatización está llegando a nuestros hogares de mano de la domótica. Los sistemas de localización y posicionamiento permiten servicios de activación automática para hogares domóticos en los que la activación de algunos servicios depende de la localización del usuario, como por ejemplo el encendido de la calefacción cuando el usuario está próximo a su casa.

2.1.3 Sistemas de localización basados en WiFi

El GPS es la solución “casi universal” para obtener localización precisa y rápida en cualquier punto del planeta. Desgraciadamente, en determinados entornos exteriores, y en la mayoría de los interiores, el GPS no funciona tan bien. Sin embargo, en dichos lugares transcurre gran parte nuestra actividad diaria por lo que actualmente se están desarrollando sistemas de posicionamiento basados en otros protocolos (WiFi, Bluetooth, etc.) que faciliten la localización en interiores. A continuación se detalla en qué consiste un sistema de posicionamiento basado en WiFi y cuáles son sus ventajas y desventajas.

En los sistemas de localización basados en WiFi, el mecanismo de posicionamiento se basa en las mediciones que los puntos de acceso de la red hacen de la potencia emitida por los dispositivos inalámbricos que se conectan a la red.

Para establecer un sistema de posicionamiento basado en WiFi sólo necesitamos puntos de acceso (APs) que dan servicio a los clientes, y éstos deben incorporar tarjetas WiFi de tal modo que puedan trabajar con dicha tecnología.

Este sistema generalmente consiste en una red de sensores (en el caso de WiFi, estos sensores son puntos de acceso 802.11) que obtienen información de los usuarios que se encuentran en su área de obertura. Con dicha información, y utilizando algoritmos más o menos complejos, el sistema es capaz de posicionar a esos usuarios en algún punto de esa zona de cobertura.

Las ventajas que destacamos del uso de un sistema de localización basado en WiFi son las siguientes:

- Es un sistema que requiere poca infraestructura y la red se configura de forma rápida y sencilla. Además permite una gran movilidad al eliminar las conexiones cableadas.

- La localización se puede hacer en el propio dispositivo móvil (ya que no se necesitan receptores específicos como sucede con GPS) y gracias a la red inalámbrica, podemos enviar la información deseada de forma sencilla.

Como desventaja principal podemos decir que la presencia de elementos fijos tales como paredes, viga, etc., elementos móviles como mesas, ordenadores… y personas distorsiona la señal potencia.

2.2 Herramientas utilizadas

2.2.1 PlaceLab

Como hemos visto anteriormente, la localización en sistemas móviles se está convirtiendo cada vez más en un punto importante y necesario a desarrollar, sin embargo, existen ya en el mercado dispositivos y software que cumplen con creces los requisitos deseados pero que son excesivamente caros. PlaceLab ofrece un software de bajo coste y de fácil uso para el cálculo de la posición de un dispositivo en medios inalámbricos.

Este software de posicionamiento desarrollado por Intel y la Universidad de Washington permite a sus usuarios la localización de los dispositivos mediante el monitoreo y la escucha de diversos emisores de señales de radio: puntos de acceso 802.11 para tecnología WiFi, torres GSM de telefonía móvil y dispositivos Bluetooth.

2.2.1.1 Arquitectura

Existen tres elementos clave en la arquitectura de PlaceLab: beacons, las bases de datos y los clientes.

Beacons

Como ya hemos comentado PlaceLab utiliza cuatro emisores de señales radio diferentes: GPS, torres GSM de telefonía móvil, 802.11 Access Points y dispositivos Bluetooth; también conocidos como “radio beacons”. La clave de estos emisores es que cada uno de ellos tiene un identificador único que lo identifica y lo diferencia del resto. En el caso de los Access Point, se identifican gracias a la dirección MAC.

Bases de datos

Otro elemento importante en la arquitectura de PlaceLab son las bases de datos. PlaceLab depende de la localización de los Access Points para funcionar correctamente, por lo que necesita una base de datos que contenga las coordenadas de los APs y su dirección MAC. Para que el sistema de localización funcione de manera eficiente, la cantidad de APs mapeados debe ser lo más extenso posible.

Clientes de PlaceLab

Los clientes de PlaceLab utilizan tres tipos de elementos para llevar a cabo su objetivo: spotters, mappers y trackers.

- Spotters: Traducido como observador, este elemento es el encargado de escuchar constantemente el medio físico en busca de beacons. Su función es monitorear las señales de radio y compartir los IDs de los beacons (ya sean Access Points, dispositivos Bluetooth, etc.) con otros componentes del sistema.

- Mappers: La función de este elemento es reconocer lo que se ha recibido gracias al spotter y buscarlos en la base de datos para comprobar si éstos han sido mapeados.

- Trackers: La finalidad del tracker es combinar las informaciones recibidas por el spotter y por el mapper y estimar la posición del usuario.

2.2.1.2 Cálculo de posición

Una vez conocidos los elementos implicados en el sistema de PlaceLab, podemos hacernos a la idea de cómo se realiza el cálculo de la posición de un usuario.

Todos los APs tienen un identificador único que se corresponde con la dirección MAC. Poniendo como ejemplo que disponemos de un PC donde se está ejecutando PlaceLab y que dispone de dispositivos Bluetooth y WiFi, el sistema calculará su posición utilizando los elementos comentados en el punto anterior.

En primer lugar, instanciando un Spotter por cada tipo de protocolo (en nuestro caso para el IEEE 802.11) el sistema monitorizará los IDs de los distintos emisores de señal (beacons) que se detectan en el ambiente; posteriormente, utilizando la información generada por los Spotters, el Mapper buscará en la base de datos de mapeo los dispositivos detectados para obtener su posición; finalmente, con toda la información recopilada por los anteriores, el Tracker intentará estimar la posición des usuario.

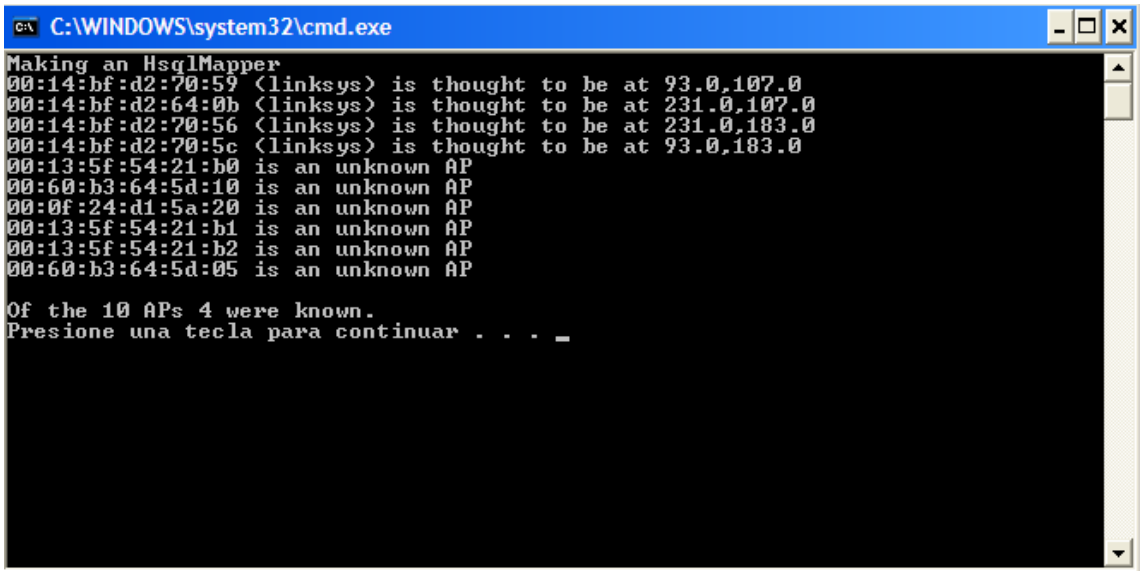


Figura 2.4. Lista de Access Points conocidos y desconocidos (Función del Mapper)

2.2.1.3 Ventajas y desventajas de PlaceLab

Una de las principales ventajas de PlaceLab es su eficacia tanto en ambientes interiores como en exteriores, que comparada con otros sistemas de localización es un hecho importante, ya que la mayoría de ellos sólo son capaces de posicionar en exteriores.

Otra gran ventaja de PlaceLab es que un software libre, por lo que no supone ningún coste adicional para el usuario; además, al ser código abierto, permite a cualquier persona con los conocimiento suficientes realizar modificaciones o ampliaciones en el código.

Por otro lado, PlaceLab sólo se basa en señales de radio que se están extendidas a nivel mundial y que utilizan protocolos de comunicación estándar. Esto facilita el uso ya que no requiere la compra de un hardware que implemente un protocolo concreto (como GPS) sino que se puede utilizar cualquier dispositivo que sea capaz de detectar señales de radio estándar, concretamente Bluetooth, IEEE 802.11 WiFi y señales GSM.

Además, gracias al boom de las tecnologías inalámbricas cada vez hay más dispositivos (móviles, ordenadores…), por lo que el número de Access Points está creciendo muy rápidamente, lo que ayuda en gran medida a la finalidad de PlaceLab, ya que con mayor número de APs mayor es la eficiencia en el posicionamiento.

Como desventaja destacable de PlaceLab, podemos decir que aunque funciona igual tanto en exteriores como en interiores, la precisión en la localización en exteriores no es tan buena como con otros sistemas como el GPS. Además, el radio de cobertura de PlaceLab todavía no es suficientemente alto como para ser usado de manera popular, por lo que aún se necesita un tiempo para poder desarrollar todo su potencial.

2.2.2 JPcap

Una necesidad del proyecto es el poder comunicarnos con todos los nodos de la red (ver capítulo 4). Para ello necesitamos conocer las direcciones IP de cada uno de los nodos.

Una de las características de las redes ad-hoc que vimos al intentar detectar los nodos que formaban parte de la red, es que al formarse la red en lugar de identificar cada dispositivo que forma parte de ella, se identifica por un único SSID que representa el conjunto de toda la red ad-hoc.

De esta manera, se puede interpretar como si fuese un único Access Point. Puesto que la red la conforman todos sus nodos y no un único punto de acceso, el sistema no tiene una dirección MAC única, de manera que lo que sucede es que le asigna una dirección MAC aleatoria e irreal.

Esta característica de las redes MANET supone que, al ejecutar PlaceLab, en lugar de detectar un punto de acceso por cada nodo de la misma, sólo detecta el punto de acceso imaginario que el sistema genera y su dirección MAC falsa. Esto provoca que no seamos capaces, a priori, de identificar todos los nodos de la red.

En el cuadro siguiente podemos ver el resultado de ejecutar una de las funciones de PlaceLab donde nos muestra los Access Point que ve y donde también nos muestra nuestra red MANET con SSID *“prueba1”*.

```
14 APs were seen

BSSID               MAC Address         SSID                RSSI
00:1a:e3:d1:2e:95   00:1a:e3:d1:2e:95   eduroam-web         -94
00:1a:e3:d1:2e:93   00:1a:e3:d1:2e:93   cttc-voip-sip       -92
00:1a:e3:d1:2e:91   00:1a:e3:d1:2e:91   SatConxion          -95
00:13:5f:54:36:71   00:13:5f:54:36:71   SatConxion          -94
00:18:39:ae:93:44   00:18:39:ae:93:44   lab323              -86
00:18:f8:f1:7f:ba   00:18:f8:f1:7f:ba   ISILABW-123         -73
00:13:5f:54:36:a5   00:13:5f:54:36:a5   eduroam-web         -90
00:13:5f:54:36:a4   00:13:5f:54:36:a4   eduroam             -94
00:13:5f:54:36:a3   00:13:5f:54:36:a3   cttc-voip-sip       -92
02:e0:53:1e:71:ec   02:e0:53:1e:71:ec   prueba1             -45
00:13:5f:54:36:a2   00:13:5f:54:36:a2                       -90
00:23:69:1a:c1:3b   00:23:69:1a:c1:3b   ICARUS              -92
00:13:5f:54:36:a0   00:13:5f:54:36:a0                       -92
00:0c:41:3a:5a:94   00:0c:41:3a:5a:94   Lab_tel_rec         -92
```

Para solucionar este inconveniente, se decidió utilizar un Sniffer de redes, que nos permitiese capturar los paquetes de datos de la red, y así conseguir saber que nodos forman parte de la red y por consiguiente las direcciones IP de todos ellos, JPcap.

JPcap es un proyecto basado en Java que utiliza la librería pcap, que proporciona una API para la captura de tráfico de una red.

A continuación se comenta de forma resumida cuales son las funcionalidades de esta librería, que datos podemos capturar de la red, como podemos tratar estos datos, etc.:

- Nos muestra una lista de interfaces de red de qué disponemos y nos facilita información de las mismas como el nombre y la dirección MAC.

```
\Device\NPF_{55EB6FFA-2631-4DC2-9288-4297DC424C45}=00:1C:23:36:17:C4
\Device\NPF_{A0B23B14-0E76-4F72-9974-4BEDE3DEEF48}=00:1F:3A:4B:D0:82
```

- Prepara la interfaz para la captura de paquetes. Una vez se ha escogido la interfaz que se va a usar para la captura, se establecen los parámetros que determinaran la captura: tamaño máximo de paquetes admisible, tiempo máximo de espera y el modo de captura, modo promiscuo, con el cual se puede obtener todos los paquetes que lleguen a nuestra interfaz de red, aunque la máquina que los captura no sea su destinatario o sin modo promiscuo con el que solo podremos capturar paquetes que se envíen o se reciban desde nuestra máquina.

- Filtra paquetes de acuerdo con reglas específicas, de modo que podemos descartar los paquetes que no nos son necesarios en la captura.

- Guarda y lee paquetes capturados de ficheros. Esta opción permite poder trabajar con los paquetes capturados desde otras aplicaciones que permitan la lectura de estos ficheros.

- Identifica automáticamente los tipos de los paquetes y genera los objetos Java correspondientes a cada tipo (para paquetes Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP e ICMPv4)

- Permite el envío de paquetes a la red.

2.2.3 OLSR

Para asegurarnos que todos los nodos de la red, que queremos agrupar, se comuniquen entre ellos, de modo que el Sniffer sea capaz de detectar todos los nodos que forman parte de la MANET, utilizaremos el protocolo OLSR (Optimized Link State Routing).

El protocolo OLSR utiliza el enrutamiento proactivo del estado del enlace. La ventaja del uso de protocolos proactivos es que permite mantener las tablas de enrutamiento actualizadas en todo momento. Sin embargo, la desventaja de un protocolo de este tipo consiste en que se proporciona una carga adicional a la red debido a la transmisión periódica de mensajes de control. Este protocolo está pensado específicamente para conexiones inalámbricas, especialmente para MANETs, aunque también se puede implementar sobre redes cableadas, como Ethernet.

Un nodo OLSR envía mensajes de tipo “HELLO” cada cierto intervalo de tiempo (por defecto 5 segundos) para que los nodos vecinos detecten su presencia.

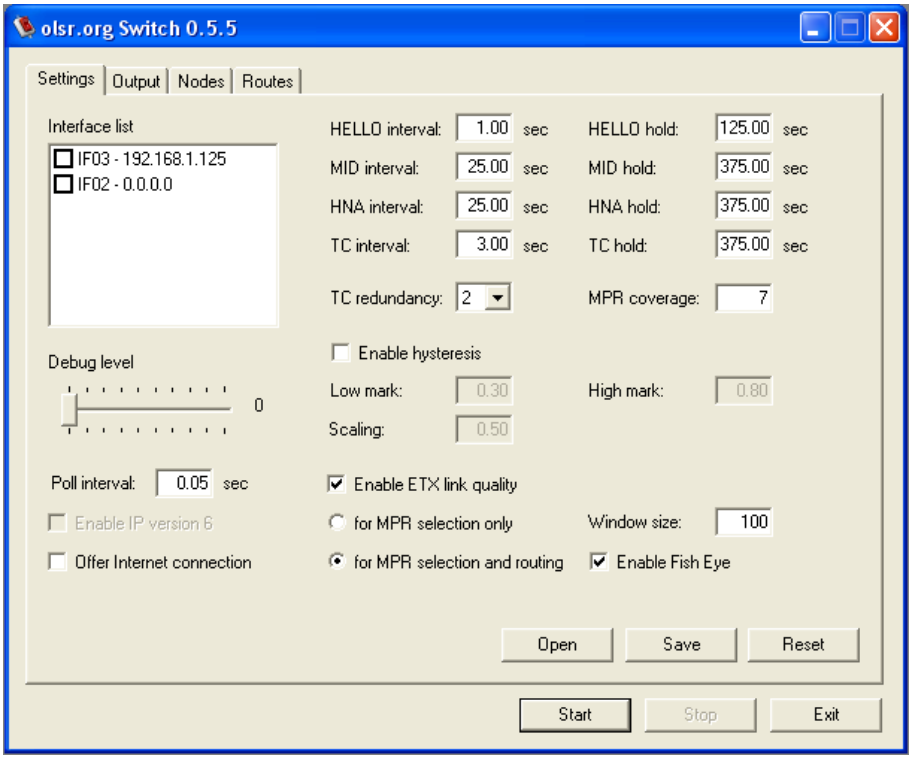


Figura 2.5. Pantalla principal de Olsr Switch.

En nuestro caso, hemos modificado el fichero de configuración de los parámetros de OLSR de tal modo que se envíen paquetes cada segundo, con lo que reduciremos el tiempo de esnifado de la red.

CAPÍTULO 3. ANÁLISIS DE REQUISITOS

En este capítulo, se realiza un análisis de los requisitos que debe cumplir nuestro sistema de agrupamiento.

Como hemos apuntado en la introducción, nuestro objetivo principal es el estudio y el desarrollo de un algoritmo que agrupe de manera automática nodos de una red MANET. Para conseguir con éxito nuestro objetivo la aplicación debe cumplir los siguientes requisitos:

- Para facilitar la agrupación, se decidió que cada grupo se formaría a partir de un PC "líder", de manera que el sistema establece los grupos por afinidad (ver capítulo 4) con cada líder. Para ello el sistema necesita saber cuáles de los PCs son líderes.

- Además, para facilitar la tarea de identificar los PCs líderes que están implicados en el proceso de agrupamiento, el sistema debe leer esta información a partir de un archivo de tipo XML, de manera que el sistema sea capaz de procesar su estructura y utilizarla como parámetros de entrada.

- Una vez conocidos los PCs implicados y los líderes, el sistema que desarrollaremos debe ser capaz de realizar una agrupación automática de los mismos. Para ello, a priori debe asignar cada PC al nodo líder con el que tenga más afinidad, pero si fuese necesario que los grupos fuesen equitativos será capaz de construir la mejor distribución posible.

- Finalmente, con el objetivo de facilitar la posterior integración con otras aplicaciones externas, el sistema debe escribir los resultados del agrupamiento en un archivo XML con el formato que se ha establecido (ver capítulo 4).

Para ejemplificar una posible distribución de grupos, en la figura siguiente podemos ver una serie de nodos separados en diferentes grupos donde los nodos marcados con una L son líderes de grupo.

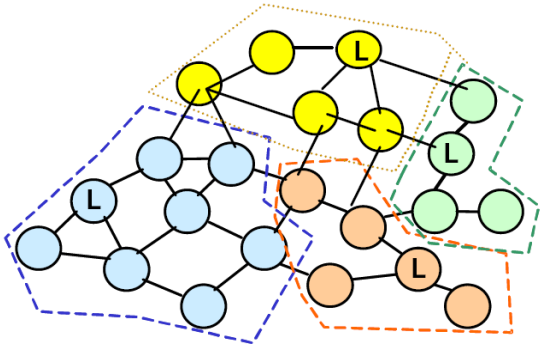


Figura 3.1. Ejemplo de agrupamiento

CAPÍTULO 4. DESARROLLO TÉCNICO Y PRUEBAS REALIZADAS

En este capítulo comentaremos detalladamente todos los pasos que se han llevado a cabo para conseguir el objetivo que se ha planteado. Veremos, por tanto, cómo se han ido desarrollando cada una de las etapas, las decisiones que se han tomado, las tecnologías utilizadas, y cómo se ha diseñado e implementado finalmente el algoritmo de agrupamiento.

Cronológicamente, podríamos dividir el desarrollo de este proyecto en las siguientes etapas:

- Desarrollo técnico
- Escenario de trabajo y pruebas realizadas

4.1 Desarrollo técnico

En este apartado se explica el desarrollo técnico de cada una de las partes del sistema:

4.1.1 Parametrización de la información

Uno de los requisitos de nuestro sistema de agrupamiento es el de poder leer la información que luego utilizaremos como parámetro de entrada, de un fichero XML. Para ello, hemos utilizado una librería externa al PlaceLab que permite la serialización de archivos XML a objetos Java y viceversa. Esta librería se llama XStream.

Antes de decidir utilizar XStream como herramienta de serialización, barajamos diversas alternativas (como Jakarta Commons Digester). Finalmente nos decantamos por la librería XStream por su facilidad de uso y porque permite la posibilidad de cargar un archivo XML y pasarlo a un objeto Java, cosa que no permitían otros serializadores.

La clase XStream de la librería XStream contiene múltiples funciones para la serialización de archivos XML a objetos Java y viceversa. La funcionalidad de parseo del XStream que hemos utilizado se basa en tres métodos básicos, contenidos dentro de la clase XStream: “toXML”, “fromXML” y “alias”. A continuación se muestran estos métodos:

public String toXML (Object obj)

Esta función permite transformar un objeto Java a un archivo XML.

public Object fromXML (String xml)

Esta otra función permite leer un fichero de tipo XML y traspasarlo a un Object.

public void alias (String name, Class type)

Esta función nos permite poner nombres más intuitivos a las clases que serializamos de un XML y viceversa. De esta manera, para que el serializador sepa a qué clase se refiere cada “tag” XML no es necesario que aparezca con el nombre completo con package incluido, sino que es suficiente con que aparezca el alias que le hemos asignado.

A continuación se muestra la estructura del archivo XML donde podemos ver que parámetros del PC hemos utilizado:

```
<pcs>
      <pc>
        <macBT>00191566CD30</macBT>
        <ip>192.168.1.3</ip>
        <longitud>107.0</longitud>
        <latitud>93.0</latitud>
        <esLider>true</esLider>
      </pc>
        <macBT>00191566hf80</macBT>
        <ip>192.168.1.15</>
        <longitud>183</longitud>
        <latitud>204</latitud>
        <esLider>false</esLider>
    </pc>
</pcs>
```

Los PCs líderes y los que no lo son tienen los mismos atributos, IP, MAC del Bluetooth, longitud y latitud. Sin embargo para poder diferenciar unos de otros se decidió añadir un atributo más, esLider.

4.1.2 Arquitectura del sistema

En este apartado se intenta detallar la arquitectura de todo el sistema desarrollado durante el proyecto para conseguir nuestro objetivo principal de agrupamiento automático.

4.1.2.1 Recopilación de la información de los nodos

Para conseguir los objetivos propuestos de agrupar los nodos de la red, primero es necesario obtener cierta información individual de cada uno de los ellos; una vez hecho esto, el algoritmo ya tendrá la información necesaria para poder realizar su función.

Los datos necesarios para esto, son principalmente: la posición en la que se encuentra en un momento dado cada uno de los nodos, las direcciones IP de cada uno de ellos, la dirección MAC del dispositivo Bluetooth local y los dispositivos Bluetooth que el nodo en cuestión detecta.

Para ello utilizamos la herramienta PlaceLab, anteriormente comentada. Este software ejecutándose de forma local, es capaz de determinar la posición (latitud – longitud) del nodo en cuestión basándose en la posición mapeada de los distintos Access Points. Al mismo tiempo, nos informa de la potencia que detecta de cada AP parametrizado. Así mismo, PlaceLab nos proporciona herramientas para la obtención de la dirección MAC del dispositivo Bluetooth, y para detectar paquetes Bluetooth y así determinar los vecinos que se detectan con esta tecnología.

Puesto que PlaceLab se ejecuta de forma local, y sólo es capaz de darnos la información del nodo en el que se está ejecutando; y dado que necesitamos la información de todos los nodos para realizar el agrupamiento, será necesario que PlaceLab se ejecute en cada uno de ellos.

4.1.2.2 Comunicación entre nodos

Como consecuencia de lo explicado en el primer apartado, nos encontramos el siguiente problema: que la información de la posición de los distintos nodos de la red está dispersada por todos ellos. Por lo tanto necesitamos algún sistema que concentre toda esta información para que el algoritmo pueda funcionar.

Para solucionar este problema hemos utilizado una de las herramientas de las que dispone PlaceLab: los “servlets”, que permiten la comunicación entre los distintos nodos de una red. De esta manera, aunque la información de un nodo esté guardada en él mismo, los servlets nos permiten hacer una petición a dicho nodo para que nos dé sus datos de posición.

Para poder realizar esta comunicación a través de los servlets, la única información que se requiere acerca del nodo destino con el que nos queremos comunicar, es su dirección IP; única en la red. En el próximo apartado veremos cómo obtener estas direcciones.

Servlets

Los servlets son módulos que extienden los servidores orientados a petición-respuesta, como los servidores web compatibles con Java. Un servlet puede manejar múltiples peticiones concurrentes y puede sincronizarlas, cosa que permite a los servlets soportar sistemas como conferencias on-line.

PlaceLab incluye un proxy HTTP y un motor servlet, que permite a las aplicaciones tanto crear interfaces web simples como incluir los servicios de PlaceLab en servidores Web remotos. PlaceLab comunica la información de la ubicación de los nodos a los servicios web a través de un proxy web que los se ejecuta en el puerto 2081. El proxy se ejecuta al crear un objeto de tipo PlacelabWithProxy y mediante la llamada a la función createProxy().

El proxy añade la información de la localización utilizando una petición HTTP llamada X-PlaceLab.location que contiene los valores de la última posición obtenida posición en términos de latitud y longitud. Se ha modificado la función ServletExample, de manera que además de la posición también sea capaz de informarnos de la MAC del dispositivo Bluetooth local y los vecinos que es capaz de detectar con esta tecnología.

PlaceLab también incluye un motor servlet sencillo que permite recoger y responder peticiones HTTP basadas en sus URLs. Para utilizar el servidor proxy o el motor servlet tan solo es necesario configurar nuestro navegador para que use http://localhost:2081\sample como nuestro proxy HTTP.

En la figura siguiente podemos ver la representación de la arquitectura de un servlet:

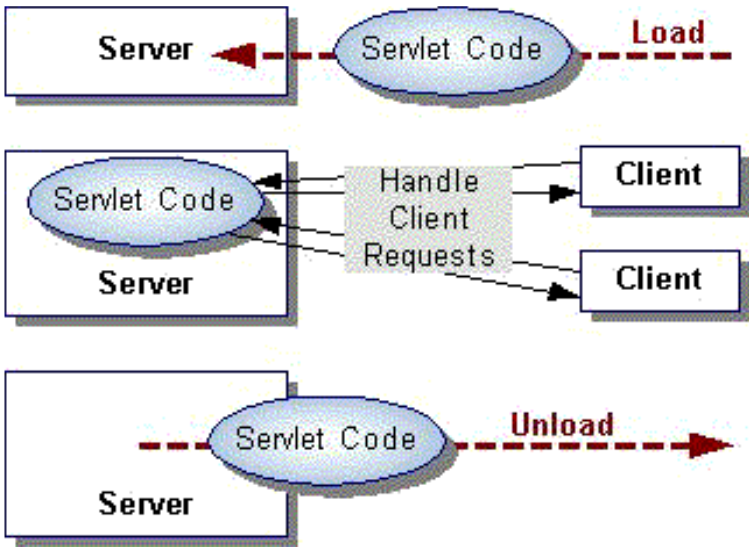


Figura 4.1. Arquitectura de un servlet

4.1.2.3 Obtención de las direcciones IP de los nodos

Como hemos comentado en el apartado anterior, para poder concentrar toda la información de las posiciones de los distintos nodos, necesitamos comunicarnos con cada uno de ellos para que nos den su ubicación. A continuación explicaremos cómo se ha solucionado este problema.

Para realizar la comunicación mediante peticiones HTTP, con la tecnología que hemos explicado en el apartado anterior, es necesario conocer la dirección IP del nodo destino, de manera que para realizar la comunicación con cada uno de los nodos de la red, necesitamos obtener el listado de todas las direcciones IP.

Para conseguir este objetivo hemos optado por esnifar el tráfico de nuestra red, de manera que podamos capturar paquetes de datos de todos los nodos, los cuales contienen su dirección IP. Para ello hemos utilizado la librería JPcap, que nos permite filtrar los paquetes que recibimos, para así asegurarnos que sólo capturamos los que necesitamos, y luego analizarlos y obtener la información que contiene.

Con el fin de obtener las direcciones IP de todos los nodos de nuestra red, necesitamos capturar paquetes que provengan (o se dirijan) a cada uno de ellos. Para asegurarnos de esto, hemos utilizado el protocolo OLSR, que haciendo repetidos envíos de paquetes de tipo “HELLO”, nos permite detectar la respuesta a estos paquetes por parte de todos los nodos de la MANET.

4.1.2.4 Nodo central

De cara a centralizar todo el proceso, puesto que necesitamos aglutinar toda la información recopilada de los distintos nodos, necesitamos de un “nodo central” que realice estas tareas. Este nodo es el que gestiona todo el proceso de agrupamiento, desde la obtención de las direcciones IP hasta la ejecución del algoritmo de agrupamiento.

Las tareas concretas que realiza este nodo son: en primer lugar, se ocupa de realizar la captura de todos los paquetes que genera el protocolo OSLR, para conseguir el listado de IPs de los nodos; en segundo lugar, envía una petición HTTP a cada nodo para recuperar todos los datos necesarios (posición, MAC del Bluetooth local, y listado de vecinos BT); y finalmente, con toda esta información, lanza el algoritmo de agrupación que genera el fichero XML con los grupos formados.

Con el objetivo de realizar un filtrado previo de los PCs que van a formar parte de la agrupación hemos utilizado la tecnología Bluetooth, puesto que es de corto alcance, para excluir aquellos PCs de la red (detectados mediante OLSR) que no se detecten desde ningún otro nodo. Para ello, se fusionan todas las listas de vecinos BT de cada nodo en una única lista de vecinos Bluetooth, y se excluyen del agrupamiento aquellos PCs que no aparezcan en dicha lista.

Para poder realizar este filtrado de Pcs cada uno de ellos incorpora un dispositivo Bluetooth como el siguiente:

Figura 4.2. Bluetooth Kensington Modelo K33348B

4.1.3 Diseño del algoritmo

Como hemos dicho anteriormente, el algoritmo que queremos diseñar deberá agrupar de forma automática una serie de nodos de una red MANET.

Para facilitar el proceso de agrupamiento, se decidió que de entre los nodos de la red se elegirían algunos, que actuarían como nodos “líderes”. Cada uno de los grupos se formará alrededor de estos nodos “líderes”; esto significa que para cada nodo líder se asignarán un conjunto de nodos “no-líderes” que conformaran cada grupo.

Con PlaceLab podemos determinar la localización de un nodo con respecto a los Access Point que detecta, siempre y cuando estos APs hayan sido mapeados en la base de datos de PlaceLab. Para determinar la afinidad de cada nodo con respecto cada uno de los “líderes” se decidió usar sólo la distancia relativa de los nodos.

4.1.3.1 Gestión de los datos de entrada

Para el funcionamiento del algoritmo, éste necesita de varios datos de entrada, que son: el listado de PCs de la red, la dirección IP de cada PC, su posición en formato latitud/longitud, la dirección MAC del dispositivo Bluetooth, el listado de vecinos de BT, y cuáles de ellos son líderes.

El listado de PCs de la red que son líderes, se introduce manualmente antes de la ejecución del algoritmo. Esto se realiza mediante un fichero XML con un formato determinado (ver apartado 4.1.1), que el sistema leerá. Posteriormente, utilizando las herramientas correspondientes, se determinará qué nodos forman parte de la red, y toda la información necesaria de cada uno de ellos.

Una vez calculada toda esta información y almacenada en las estructuras de datos correspondientes, se enviará al algoritmo para que realice las funciones propias del agrupamiento.

4.1.3.2 Desarrollo del algoritmo

Esta función, a partir de dos PCs devuelve la distancia entre ellos.

```
private float getDistancia(Pc pc1, Pc pc2)
{
      float x=0, y=0, dist=0;
      x = pc1.getLatitud()-pc2.getLatitud();
      y = pc1.getLongitud()-pc2.getLongitud();
      dist = (float)Math.sqrt((x*x)+(y*y));

      return dist;
}
```

Esta función nos devuelve la valoración (afinidad) entre un PC y un líder, teniendo en cuenta la distancia o posición de los PCs.

```
private float getValoracion(Pc pc, Pc lider)
{
	float d = 0;
	d = this.getDistancia(pc, lider);

	return d;
}
```

Teniendo la valoración (afinidad) de todos los nodos con los líderes se hace una primera agrupación basándose en la mejor afinidad, sin tener en cuenta el número de componentes de los grupos, de tal modo que cada nodo no “líder” queda agrupado con su “mejor líder”.

Para realizar una primera agrupación “directa”, recorremos el listado de nodos y, para cada uno de ellos, recorremos el listado de líderes obteniendo la valoración con cada uno de ellos; al final del recorrido, se obtiene el líder con la mejor valoración para el PC actual. Una vez tenemos esto, no nos queda más que asignar dicho PC al grupo del líder obtenido.

La parte del algoritmo que realiza estas operaciones es la siguiente:

```
private Vector crearGrupos ()
{
	for(int i=0; i<pcs.size();i++)
	{
		int lidermin=0;
		Pc pc = pcs.get(i);
		float valmax=0;

		for (int j=0; j<lideres.size();j++)
		{
			Pc lider = lideres.get(j);
			float v=0;
			v = getValoracion(pc, lider);

			if(v>valmax)
			{
				valmax = v;
				lidermin = j;
			}
		}

		grps.get(lidermin).add(pcs.get(i));
	}
```

Una vez hecha esta primera agrupación “directa”, se redistribuyen los PCs de aquellos grupos que tengan exceso de nodos con el objetivo de conseguir una distribución uniforme, es decir, que todos los grupos contengan la misma cantidad (si es posible) de PCs. Esto sólo se realiza en caso que se desee que los grupos se distribuyan uniformemente, en caso contrario ya habremos terminado.

Para conseguirlo, en primer lugar calculamos el número de componentes que debe contener cada grupo. Una vez hecho esto, recorremos el listado de grupos, y para cada grupo que contiene menos nodos de los que corresponde, hacemos lo siguiente: recorremos el listado de grupos con más componentes de los deseados y calculamos, de entre todos los PCs, cuál de ellos tiene mejor valoración con el líder actual. Esta operación se realiza repetidamente mientras que el grupo actual no tenga nodos suficientes.

La otra parte del algoritmo, que realiza estos cálculos, es la siguiente:

```
if(uniformes)
{
	int componentes_gen = (int)pcs.size()/lideres.size();
	int componentes_lim;
	int componentes_act = componentes_gen;
	int grps_pcs_extra = (int)pcs.size()%lideres.size();
	int grps_cont_extra = 0;

	if(grps_pcs_extra>0) componentes_lim = componentes_gen+1;
		else componentes_lim = componentes_gen;

	for (int i=0; i<grps.size(); i++)
	{
		if(grps.get(i).size()>componentes_gen)
			grps_cont_extra++;
	}

	for (int i=0; i<grps.size(); i++)
	{
		if(grps_cont_extra < grps_pcs_extra)
			componentes_act = componentes_gen+1;
		else componentes_act = componentes_gen;

		while (grps.get(i).size()<componentes_act)
		{
			int minj=0, mink=0;
			float valmin=0, v=0;
			Pc lider = lideres.get(i);
			for(int j=0; j<grps.size(); j++)
			{

			if(grps.get(j).size()>componentes_lim)
				{
			valmin = getValoracion (grps.get(j).get(0),lider);

				for (int k=0; k< grps.get(j).size(); k++)
				{
					v = getValoracion (grps.get(j).get(k),lider);
						if(v<valmin)
					{
						valmin = v;
						minj = j;
						mink = k;
					}
```

```
                        }
                    }
                }
                grps.get(i).add(grps.get(minj).get(mink))
                    grps.get(minj).remove(mink);

                if(componentes_act==componentes_gen+1&&grps.get(i).size()==c
            omponentes_act)   grps_cont_extra++;
                    }
                }
            }
            for (int i=0; i<lideres.size();i++)
            {
                grps.get(i).add(lideres.get(i));
            }

            return grps;
}
```

Finalmente lo que conseguimos es un Vector que contiene todos los grupos formados. Cada uno de estos grupos los conforman el conjunto de PCs que forman parte del grupo, y su líder. La estructura donde guardamos toda esta información es la siguiente:

private Vector <Vector<Pc>>

La representación lógica de esta estructura sería la siguiente:

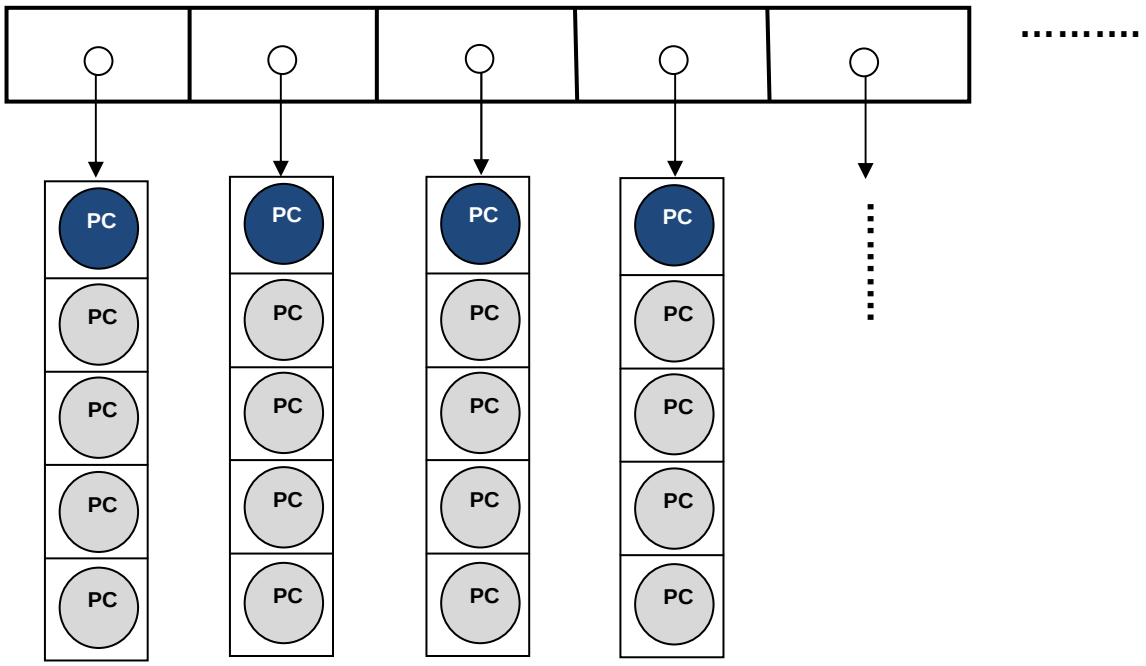


Figura 4.3. Ejemplo de estructura principal (Vector<Vector<Pc>>)

4.1.3 Detalle de clases y procedimientos

En este apartado del desarrollo técnico vamos a ver en más detalles cuáles son las clases más importantes que se utilizan y como están jerarquizadas, así como las acciones y funciones que contienen cada una de ellas.

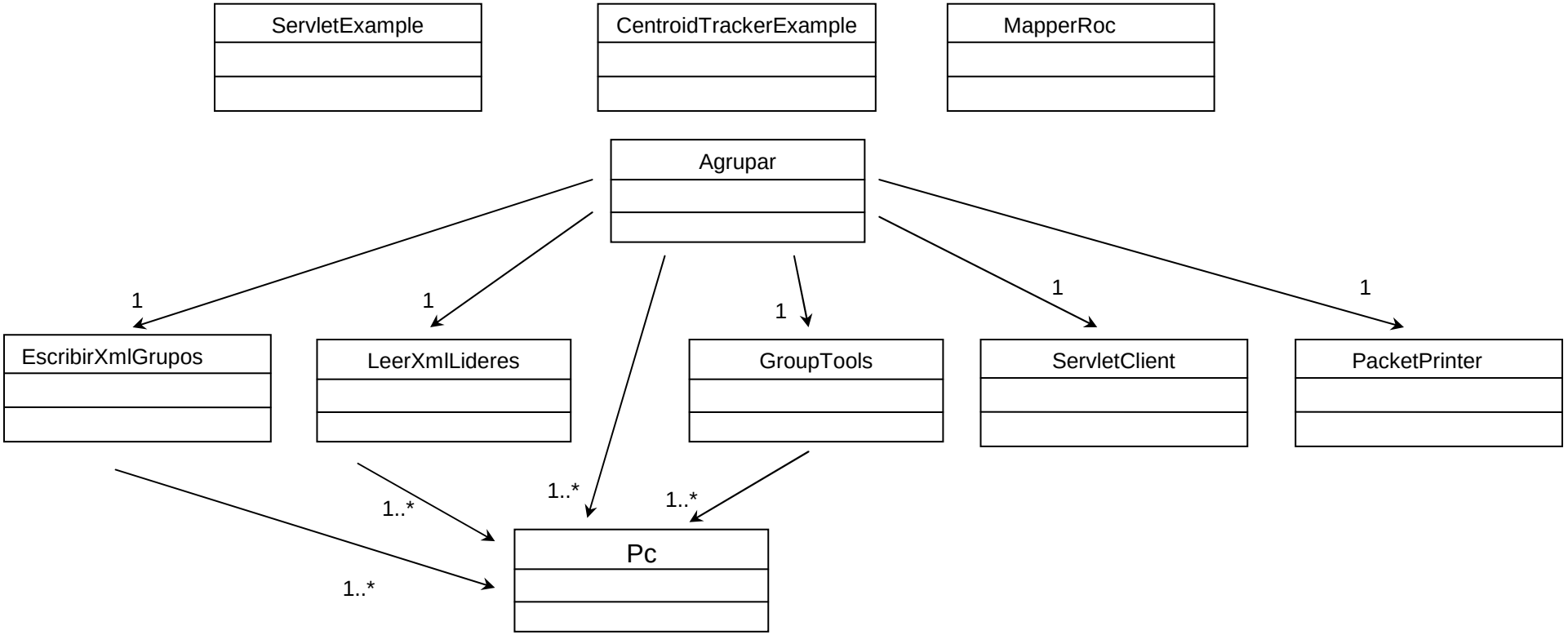


Figura 4.4. Diagrama de clases

En el siguiente apartado veremos en detalle cada una de las clases desarrolladas y todos sus métodos y atributos.

4.1.2.1 Clases

Clase GroupTools

Esta clase contiene todas las herramientas necesarias para realizar la agrupación.

Atributos

private Vector<Pc> lideres

Vector que contiene el listado de PCs “líderes”.

private Vector<Pc> pcs

Vector que contiene la lista de PCs que no son “líderes”.

private Vector<Vector<Pc>> grps

Este es el vector principal donde se guardan los grupos formados.

Métodos

public void agrupar (Vector<Pc> pcs)

La función *agrupar* a partir del listado de PCs que hemos obtenido del fichero XML, permite inicializar los vectores de líderes y los vectores de PCs no líderes. Una vez hecho esto, realiza la llamada a la función *crearGrupos* y guarda en la estructura principal (gprs) los grupos formados.

private float getValoracion (Pc pc, Pc lider)

Con esta función se obtiene el valor con el que nos basamos para agrupar un PC con un “líder”, como ya hemos comentado anteriormente este valor se basa en la distancia que separa los diferentes nodos.

private Vector crearGrupos ()

Con esta otra función se realiza la agrupación definitiva. En un primer paso agrupa los PCs con el mejor líder, es decir agrupamos el PC con el líder con mejor índice. Este paso se realiza sin tener en cuenta el número de componentes que debe tener cada grupo. Por lo que en el segundo paso de esta función, se realiza una redistribución de los PCs, donde aquellos líderes que tengan menos PCs de los que deben tener cogen de los lideres que tengan más.

Clase Pc

Esta clase PC contiene la información esencial de cada PC para poder realizar la agrupación.

Atributos

id: String

Contiene el id del PC.

macBT: String

Contiene la dirección MAC del dispositivo Bluetooth del PC.

longitud: float

Coordenada *longitud* de la posición del PC.

latitud: float

Coordenada *latitud* de la posición del PC.

ip: String

Contiene la dirección IP del PC.

esLider: Booleano

Define si el PC es de tipo líder.

Clase Agrupar

Contiene el programa principal desde donde se organizan las llamadas a las distintas funciones y clases para realizar todo el proceso de agrupamiento: la lectura del fichero XML, el cálculo de los grupos y la escritura del resultado en otro archivo XML.

Clase LeerXmlLideres

Esta clase realiza la serialización de un fichero XML a un objeto Java de tipo PC.

Métodos

public Vector getLideres ()

Esta función devuelve un vector de PCs “lideres”, los cuales se han obtenido a partir de la lectura de un fichero XML.

Clase EscribirXmlGrupos

Esta clase permite trasladar a un fichero XML los grupos ya formados.

Métodos

public void escribirGrupos (Vector<Vector<Pc>> grps)

Esta función genera un archivo de tipo XML donde se detallan los grupos que se han formado.

Clase ServletClient

Esta clase realiza la obtención de las posiciones de los nodos de la red.

Métodos

public static String getPosicionIP(String ip)

A esta función se le pasa como argumento la IP del PC del que queremos conocer la posición y devuelve un String que contiene la posición del mismo.

Clase PacketPrinter

Esta clase se encarga de realizar la captura de los paquetes generados por el protocolo OLSR y de extraer la dirección IP de cada uno de ellos.

Métodos

public void receivePacket(Packet packet)

Esta función se ejecuta automáticamente cada vez que se recibe un nuevo paquete. Tiene como argumento un objeto de tipo Packet, que representa el paquete capturado en ese momento. Se encarga de extraer la IP del paquete y añadirla al vector de direcciones IPs de los PCs.

public static Vector<String> listadoIP()

Esta función es la que abre la interfaz de red correspondiente para poder iniciar la captura. Asimismo, una vez capturados todos los paquetes, devuelve el listado de IPs (diferentes) detectadas.

Clase ServletExample

Esta clase del PlaceLab, crea un “listener” de servlet, que espera peticiones HTTP. Cuando recibe una petición, se ocupa de construir una respuesta HTTP con la siguiente información: las coordenadas (latitud y longitud) de la posición del propio nodo la dirección MAC del dispositivo Bluetooth del mismo nodo y las direcciones MAC de los dispositivos Bluetooth que es capaz de ver.

Clase MapperRoc

Esta clase realiza la función de mapeo de los Access Point de nuestra red a partir de la MAC de cada uno de ellos.

Clase CentroidTrackerExample

Esta clase a partir de las coordenadas de la posición de los APs calcula el punto medio de todas y considera que la posición del usuario es el punto calculado.

4.2 Escenario de trabajo y pruebas realizadas

En este apartado se comentan que prueba se han realizado en el escenario escogido y las diversas configuraciones que se han desarrollado para ver el comportamiento de nuestro sistema de agrupamiento.

Para que PlaceLab funcione correctamente se deben tener mapeados de entre 3 a 5 Access Points. Para saber la ubicación de los Access Points dentro de nuestro escenario de pruebas se ha usado un editor de imágenes. De esta forma se puede saber el número de píxeles que tiene la imagen, y calcular las coordenadas (latitud y longitud) en píxeles de los APs para poder mapearlos y así poder añadirlos a la base de datos de PlaceLab.

En cuanto al cálculo de la posición de los distintos nodos, vimos que en PlaceLab no están desarrolladas las aplicaciones de cálculo de posiciones mediante sus librerías propias de posicionamiento. Para solucionarlo, hemos estimado la posición de cada uno de ellos realizando un mapeo concreto de los APs para que, al utilizar la aplicación Centroid, nos dé la posición aproximada que queremos de los PCs.

En relación al párrafo anterior, y de cara a facilitar la detección y posicionamiento de los PCs, hemos desarrollado una parte de la aplicación que se ocupa de filtrar aquellos PCs que, mediante tecnología Bluetooth, no se detectan desde ninguno de los PCs de la red; es decir, que no forma parte de los PCs a agrupar. De esta manera, realizamos un filtrado previo a la agrupación para eliminar aquellos PCs que estén fuera del alcance.

Escenario de trabajo: Laboratorio 016 de la EPSC

El escenario de trabajo donde realizaremos las pruebas se muestra en el siguiente plano. El plano representa la planta baja del edificio de profesores.

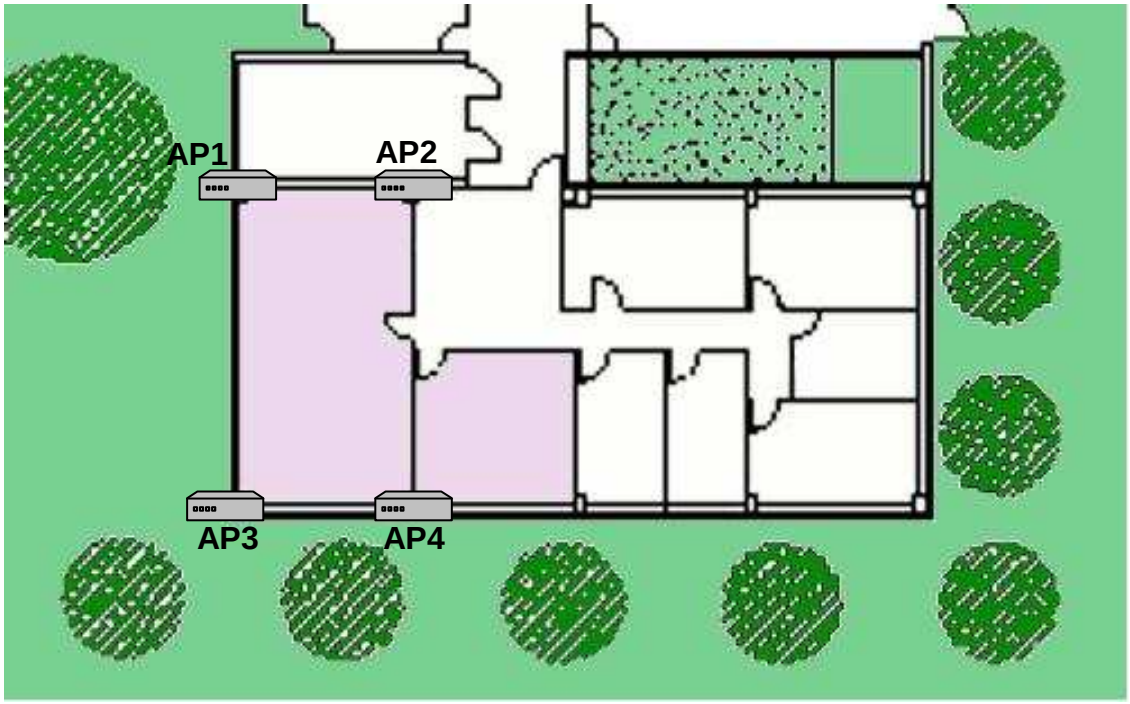


Figura 4.5. Planta del edificio de Profesores. Laboratorio 016

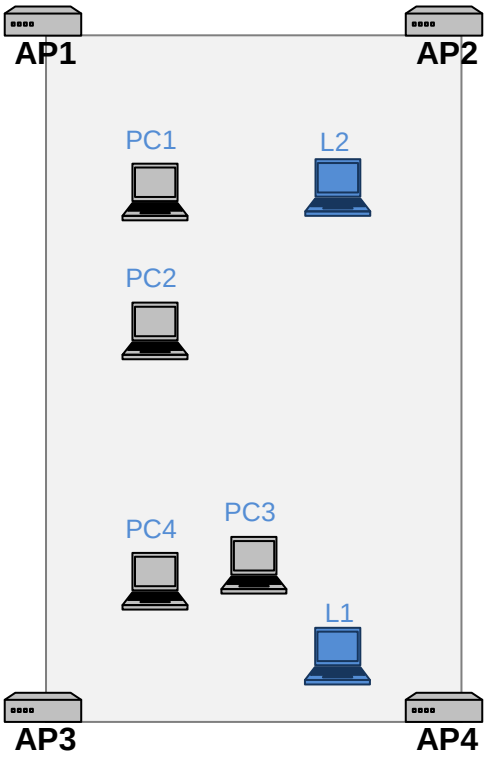
El plano entero tal como se muestra mide de ancho 563 píxeles (eje X) y 349 píxeles de alto (eje Y). EL origen de coordenadas (0,0) para realizar todas las pruebas y cálculos se sitúa en la esquina superior izquierda del plano.

En la tabla siguiente aparecen los datos necesarios para el mapeo de los Access Points que usaremos para realizar las pruebas: dirección MAC y coordenadas (Latitud y longitud). Los datos de las coordenadas se han basado en las medidas del plano anterior.

	MAC	X	Y
AP1	00:14:BF:D2:64:0B	115	90
AP2	00:14:BF:D2:70:59	203	90
AP3	00:14:BF:D2:70:56	115	255
AP4	00:14:BF:D2:70:5C	203	255

4.2.1 Resultados configuración 1

En esta primera configuración, hemos realizado la agrupación de 6 PCs de los cuáles 2 de ellos son nodos líderes y el resto nodos estándar.



	IP	X	Y
PC1	192.168.1.9	145	120
PC2	192.168.1.10	145	175
PC3	192.168.1.11	190	215
PC4	192.168.1.12	145	215
L1	192.168.1.13	230	235
L2	192.168.1.14	230	120

Al aplicar el algoritmo de agrupamiento a los nodos de esta configuración, obtenemos el siguiente agrupamiento:

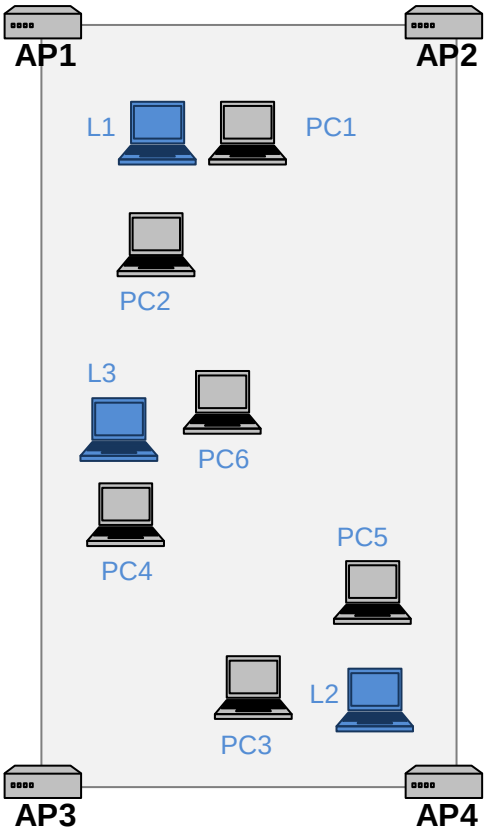
Grupos		
L1	PC4	PC3
L2	PC1	PC2

Con esta misma configuración, comprobamos el funcionamiento del algoritmo cuando uno de los PCs deja de ser visible para el resto. En este caso, decidimos alejar el PC correspondiente con el L1, de manera que obligamos al PC4 y al PC3 a agruparse con el L2.

		Grupos		
L2	**PC1**	PC2	PC3	PC4

4.2.2 Resultados configuración 2

En esta segunda configuración, hemos realizado la agrupación de 9 PCs de los cuáles 3 de ellos son nodos líderes y el resto nodos estándar.



	IP	X	Y
PC1	192.168.1.9	159	110
PC2	192.168.1.10	137	130
PC3	192.168.1.11	159	240
PC4	192.168.1.12	129	190
PC5	192.168.1.13	179	210
PC6	192.168.1.14	148	170
L1	192.168.1.15	137	110
L2	192.168.1.16	179	240
L3	192.168.1.17	126	180

A continuación, se puede ver el agrupamiento generado a partir de la ejecución de nuestro algoritmo:

Grupos		
L1	PC1	PC2
L2	PC3	PC5
L3	PC4	PC6

En el Anexo 2, se pueden ver los ficheros XML que se han generado para las configuraciones elegidas.

CAPÍTULO 5. CONCLUSIONES

En este último capítulo, haremos una valoración de los resultados obtenidos tras el desarrollo de todo el proyecto.

5.1 Objetivos alcanzados

El objetivo principal de este proyecto era el estudio y el diseño de un algoritmo integrable en PlaceLab, capaz de realizar el agrupamiento automático de dispositivos activos en una red de tipo MANET, de manera que pudiéramos actualizar los grupos previendo las bajas o las nuevas incorporaciones a nuestra red.

Tras el desarrollo de todo este trabajo, podemos decir que hemos cubierto los objetivos marcados, puesto que se ha desarrollado con éxito un algoritmo que es capaz de agrupar nodos de una red MANET.

Asimismo fueron planteados una serie de requisitos que el sistema debía cumplir (agrupar teniendo en cuenta la existencia de nodos líderes, la obtención de los líderes a través de un fichero XML y la generación de los grupos con el posterior volcado de esta información a otro fichero XML) los cuales también han podido ser cubiertos.

5.2 Problemas encontrados

Durante el proceso de realización del trabajo, nos hemos encontrado con diversos hándicaps que hemos tenido que ir resolviendo sobre la marcha.

En primer lugar, nos encontramos con la imposibilidad de que a partir de un nodo central, pudiéramos conocer las posiciones de todos los nodos de la red, ya que como vimos anteriormente, el software utilizado tiene bastantes restricciones y solo es capaz de determinar la posición del nodo donde se ejecuta.

A raíz de este problema, tuvimos que utilizar otra tecnología para poder comunicar la posición de todos los nodos de nuestra red hacia un nodo central, de manera que de ahí surgió la necesidad de los servlets.

Otra necesidad que surgió fue la de poder determinar que nodos formaban parte de la red, para descubrir esta relación de “vecinos”, utilizamos otra herramienta ajena al uso del PlaceLab, la librería JPcap, que, junto con el protocolo OLSR, nos servía de Sniffer de nuestra red y nos ayudaba a detectar a todos nuestros “vecinos”.

CAPÍTULO 6. POSIBLES MEJORAS Y AMPLIACIONES

Tras el desarrollo de este trabajo de fin de carrera, se pueden proponer posibles ampliaciones del mismo, que, ya sea por falta de tiempo, o simplemente porque no eran objetivos fundamentales del proyecto, no se han llevado a cabo durante el transcurso del mismo. A continuación comentaremos algunas de estas posibles mejoras.

Una de las ampliaciones más obvias que se extraen de este proyecto, es el hacer que la agrupación que ahora mismo generamos sólo de forma “conceptual”, sea una agrupación también a nivel “físico”. De esta manera, se podría utilizar el desarrollo de este proyecto para que, de forma automática, se formaran grupos de usuarios de la red por cercanía, para que pudiesen compartir archivos, comunicarse, etc.

De cara a conseguir el objetivo anterior, se tendría que simplificar la configuración necesaria para realizar la agrupación. Para ello se podría conseguir, utilizando únicamente la librería JPcap, inyectar paquetes de peticiones ARP en la red, para que todos los nodos de la misma contestasen. De esta manera evitaríamos el uso y la configuración del protocolo OLSR, cuya única funcionalidad era permitirnos detectar paquetes de todos los nodos.

Finalmente, una ampliación que facilitaría mucho el uso de este software sería desarrollar una interfaz gráfica que permitiese al usuario del sistema utilizar todas las herramientas interactivamente. Esto facilitaría, por ejemplo, la tarea de introducir los nodos que son líderes, o mapear las posiciones de los APs, gráficamente, sin necesidad de tener grandes conocimientos sobre el funcionamiento interno del sistema.

BIBLIOGRAFÍA

Recursos Web

[1] Página oficial de PlaceLab: http://www.placelab.org

[2] Página oficial de XStream: http://xstream.org/

[3] Página oficial de JPcap: http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html

[4] Página del protocolo OLSR: http://www.olsr.org/?q=background

[5] Página de APACHE: http://hc.apache.org/

Artículos

[1] PlaceLab: Device Positioning Using Radio Beacons in the Wild

[2] Ventajas de usar subredes en una red Ad-Hoc con nodos móviles

Otros trabajos de fin de carrera

[1] Sistemas de posicionamiento basados en WiFi de Aroa Carcavilla Sanz.

[2] Experiments about a real use of mobile ad-hoc networks de Núria Franco.

ANEXOS

Anexo 1: Códigos de las aplicaciones en Java para PlaceLab

A.1.1 MapperRoc

```
package org.placelab.example;

import org.placelab.core.BeaconMeasurement;
import org.placelab.core.BeaconReading;
import org.placelab.core.TwoDCoordinate;
import org.placelab.core.WiFiReading;
import org.placelab.mapper.Beacon;
import org.placelab.mapper.CompoundMapper;
import org.placelab.mapper.Mapper;
import org.placelab.mapper.WiFiBeacon;
import org.placelab.spotter.LogSpotter;
import org.placelab.spotter.Spotter;
import org.placelab.spotter.SpotterException;
import org.placelab.spotter.WiFiSpotter;

/**
 * This sample is very similar to CoordinateSample with the addition
 * of a lookup in the persistent AP cache
 */
public class MapperRoc {

      public static void main(String[] args) {
          Spotter s;
          if(args.length >= 1) {
              s = LogSpotter.newSpotter(args[0]);
          } else {
              s = new WiFiSpotter();
          }
            try {
                  s.open();
                  BeaconMeasurement m = (BeaconMeasurement) s.getMeasurement();
                  Mapper mapper;
                  // This Mapper can tell us where APs are
                  // The default Mapper (set in PlacelabProperties) will be selected
                  // here.  The first argument says to exit on error, and the second
                  // says to cache Beacons in memory as they are accessed.
                  mapper = CompoundMapper.createDefaultMapper(true, true);

                  int knownAPs = 0;
                  TwoDCoordinate coor = new TwoDCoordinate(0.0,0.0);
                  for (int i = 0; i < m.numberOfReadings(); i++) {
                        BeaconReading r = (BeaconReading) m.getReading(i);
                        if (r.getType() == org.placelab.core.Types.WIFI) {
                              WiFiBeacon bTemp = new WiFiBeacon();
                        if ((((WiFiReading)r).getId()).equals("00:24:d2:23:6e:e3")) {
                                    bTemp.setSsid(((WiFiReading)r).getSsid());
                                    bTemp.setId(((WiFiReading)r).getId());
                                    coor.moveTo(93, 107);
                                    bTemp.setPosition(coor);
                                    mapper.putBeacon(bTemp.getId(), (Beacon) bTemp);
                        }
```

```
                        if ((((WiFiReading)r).getId()).equals("00:16:38:f3:56:c1"))
                        {
                                bTemp.setSsid(((WiFiReading)r).getSsid());
                                bTemp.setId(((WiFiReading)r).getId());
                                coor.moveTo(93, 183);
                                bTemp.setPosition(coor);
                                mapper.putBeacon(bTemp.getId(), (Beacon) bTemp);
                            }
                        if ((((WiFiReading)r).getId()).equals("00:22:2d:0b:9f:b3"))
                        {
                                bTemp.setSsid(((WiFiReading)r).getSsid());
                                bTemp.setId(((WiFiReading)r).getId());
                                coor.moveTo(231, 107);
                                bTemp.setPosition(coor);
                                mapper.putBeacon(bTemp.getId(), (Beacon) bTemp);
                            }
                        if ((((WiFiReading)r).getId()).equals("00:13:f7:e5:fc:50"))
                        {
                                bTemp.setSsid(((WiFiReading)r).getSsid());
                                bTemp.setId(((WiFiReading)r).getId());
                                coor.moveTo(231, 183);
                                bTemp.setPosition(coor);
                                mapper.putBeacon(bTemp.getId(), (Beacon) bTemp);
                            }

                        }
                        Beacon b = (Beacon) mapper.findBeacon(r.getId());
                        if (b == null) {
                            System.out.println(r.getId() + " is an unknown AP");
                        } else {
                            System.out.println(r.getId() + " (" +
r.getHumanReadableName()
                                    + ") is thought to be at " +
b.getPosition());
                            knownAPs++;
                        }
                    }
                    System.out.println("\nOf the " + m.numberOfReadings() + " APs "
                            + knownAPs + " were known.");
                } catch (SpotterException ex) {
                    ex.printStackTrace();
                }
            }
`
```

A.1.2. CentroidTrackerExample

```
package org.placelab.example;

import org.placelab.client.tracker.Estimate;
import org.placelab.client.tracker.Tracker;
import org.placelab.client.tracker.TwoDPositionEstimate;
import org.placelab.core.BeaconMeasurement;
import org.placelab.core.BeaconReading;
import org.placelab.core.Measurement;
import org.placelab.core.TwoDCoordinate;
import org.placelab.mapper.Beacon;
import org.placelab.mapper.CompoundMapper;
import org.placelab.mapper.Mapper;
import org.placelab.mapper.WiFiBeacon;
import org.placelab.spotter.LogSpotter;
import org.placelab.spotter.Spotter;
import org.placelab.spotter.WiFiSpotter;

/**
 * This sample is a tracker that calculates the centroid of the observed
 * readings.
 *
 * This tracker estimates the device position to be the geometric center of the
 * readings that have the same timestamp.
 */
public class CentroidTrackerExample extends Tracker {
      /** The mapper to query for information about beacons */
      private Mapper mapper;

      private TwoDPositionEstimate estimate;

      public CentroidTrackerExample(Mapper m) {
            mapper = m;
      }

      /** return an estimate based on the last set of measurements we saw * */
      public Estimate getEstimate() {
            if (estimate != null) {
                  return estimate;
            } else {
                  return new TwoDPositionEstimate(getLastUpdatedTime(),
TwoDCoordinate.NULL, 0.0);
            }
      }

      /**
       * updateEstimateImpl uses the passed measurement to compute a simple
       * geometic center.
       */
public void updateEstimateImpl(Measurement m) {
            // we only want BeaconMeasurements
            if (!(m instanceof BeaconMeasurement)) {
                  return;
            }
            BeaconMeasurement meas = (BeaconMeasurement) m;

            /* calculate the mean */
```

```
            double totalLat=0.0, totalLon=0.0;
            int count=0;
            for (int i=0; i < meas.numberOfReadings(); i++) {
                  BeaconReading reading = meas.getReading(i);
                  Beacon beacon = mapper.findBeacon(reading.getId());
                  if (beacon == null) continue;
                  TwoDCoordinate pos =
(TwoDCoordinate)((WiFiBeacon)beacon).getPosition();
                  totalLat += pos.getLatitude();
                  totalLon += pos.getLongitude();
                  count++;
            }
            TwoDCoordinate mean = new TwoDCoordinate(totalLat/count,
totalLon/count);

            /* you can also calculate the standard deviation here.  For this
sample,
             * let's ignore it.
             */

            estimate = new TwoDPositionEstimate(getLastUpdatedTime(), mean, 0.0);
      }

      //** returns true if m is a wifi measurement **/
      public boolean acceptableMeasurement(Measurement m) {
            return (m instanceof BeaconMeasurement);
      }

      public void updateWithoutMeasurement(long durationMillis) {
      }

      protected void resetImpl() {
            estimate = null;
      }

      public static void main(String[] args) {
            try {
                  Spotter spotter;
                  Mapper mapper;

                  // Create a mapper
                  mapper = CompoundMapper.createDefaultMapper(true, true);

                  // Create a new tracker
                  Tracker t = new CentroidTrackerExample(mapper);
                  Estimate e1, e2;

                  // Create either a live spotter or a log spotter
                  if (args.length == 0) {
                        spotter = new WiFiSpotter();
                  } else {
                        spotter = LogSpotter.newSpotter(args[0]);
                  }
                  spotter.open();

                  // Get readings from the spotter
                  BeaconMeasurement m =
      (BeaconMeasurement)spotter.getMeasurement();
                  System.out.println("The spotter saw " + m.numberOfReadings() +
" readings.");
                  t.updateEstimate(m);
                  e1 = t.getEstimate();
```

```
                              System.out.println("Estimated position: " +
e1.getCoord());

                    // wait around for 2 seconds
                    Thread.sleep(2000);

                    // Do it all again
                    m = (BeaconMeasurement)spotter.getMeasurement();
                    System.out.println("The spotter saw " + m.numberOfReadings() +
" readings.");
                    t.updateEstimate(m);
                    e2= t.getEstimate();
                    System.out.println("Estimated position: " +
e2.getCoord().getLatitudeAsString()
                                  + "," + e2.getCoord().getLongitudeAsString());


                    // Calculate the distance between the 2 readings
                    TwoDCoordinate c1 = (TwoDCoordinate)e1.getCoord(),
                          c2 = (TwoDCoordinate)e2.getCoord();
                    double distance = c1.distanceFrom(c2);
                    System.out.println("The distance between estimates is: "
                                  + (int) distance + " meters.");

              } catch (Exception ex) {
                    ex.printStackTrace();
              }

        }
}
```

Anexo 2. Fichero XML de grupos generados

A.2.1. Ficheros XML de la configuración 1

```
<grupos>
<grupo>
  <pc>
    <macBT>00191566CD30</macBT>
    <ip>192.168.1.9</ip>
    <longitud>145.0</longitud>
    <latitud>120.0</latitud>
    <esLider>false</esLider>
  </pc>
  <pc>
    <macBT>00191566CDD1</macBT>
    <ip>192.168.1.10</ip>
    <longitud>145.0</longitud>
    <latitud>175.0</latitud>
    <esLider>false</esLider>
  </pc>
  <pc>
    <macBT>00191566C45</macBT>
    <ip>192.168.1.13</ip>
    <longitud>230.0</longitud>
    <latitud>235.0</latitud>
    <esLider>true</esLider>
  </pc>
</grupo>
<grupo>
  <pc>
    <macBT>00191566CE80</macBT>
    <ip>192.168.1.11</ip>
    <longitud>190.0</longitud>
    <latitud>215.0</latitud>
    <esLider>false</esLider>
  </pc>
  <pc>
    <macBT>00191566CD52</macBT>
    <ip>192.168.1.12</ip>
    <longitud>145.0</longitud>
    <latitud>215.0</latitud>
    <esLider>false</esLider>
  </pc>
  <pc>
   <macBT>00191566C28</macBT>
    <ip>192.168.1.14</ip>
    <longitud>230.0</longitud>
    <latitud>120.0</latitud>
    <esLider>true</esLider>
  </pc>
</grupo>
</grupos>
```

```
<grupos>
<grupo>
  <pc>
    <macBT>00191566CD30</macBT>
    <ip>192.168.1.9</ip>
    <longitud>145.0</longitud>
    <latitud>120.0</latitud>
    <esLider>false</esLider>
  </pc>
  <pc>
    <macBT>00191566CDD1</macBT>
    <ip>192.168.1.10</ip>
    <longitud>145.0</longitud>
    <latitud>175.0</latitud>
    <esLider>false</esLider>
  </pc>
  <pc>
    <macBT>00191566CE80</macBT>
    <ip>192.168.1.11</ip>
    <longitud>190.0</longitud>
    <latitud>215.0</latitud>
    <esLider>false</esLider>
  </pc>
  <pc>
    <macBT>00191566CD52</macBT>
    <ip>192.168.1.12</ip>
    <longitud>145.0</longitud>
    <latitud>215.0</latitud>
    <esLider>false</esLider>
  </pc>
  <pc>
    <macBT>00191566C28</macBT>
    <ip>192.168.1.14</ip>
    <longitud>230.0</longitud>
    <latitud>235.0</latitud>
    <esLider>true</esLider>
  </pc>
</grupo>
</grupos>
```