



**UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**

**UNA INFRAESTRUCTURA DE APOYO A LA COLABORACIÓN NO PREVISTA
ENTRE WORKSPACES COLABORATIVOS MÓVILES**

**TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN
CIENCIAS MENCIÓN COMPUTACIÓN**

FRANCISCO ARIEL CASTRO VERA

**PROFESOR GUIA:
SERGIO OCHOA DELORENZI**

**MIEMBROS DE LA COMISION:
JOSE A. PINO URTUBIA
LUIS GUERRERO BLANCO
YADRAN ETEROVIC SOLANO**

**SANTIAGO DE CHILE
NOVIEMBRE 2009**

Resumen

Este trabajo de tesis aborda la problemática de la colaboración no prevista en ambientes de trabajo colaborativo móvil. Esto es, trabajo asistido por dispositivos computacionales móviles, apoyados en un sistema de groupware que corre sobre una red inalámbrica con o sin infraestructura. Por colaboración no prevista entendemos a aquellas situaciones no planificadas en las que dos o más usuarios se encuentran y deciden colaborar utilizando sus dispositivos, por ejemplo, para compartir un documento.

Inicialmente se definió un conjunto mínimo de servicios requeridos para que un dispositivo móvil pueda interactuar con otro permitiendo la colaboración no prevista. Estos son: detección de usuarios en la red, servidor de archivos, servicios web, herramientas de sincronización de documentos, y notificaciones de actividad y cambios en el entorno de trabajo.

Se implementó una infraestructura de software que integra una serie de módulos que dan soporte a los servicios definidos. Dicha solución permite realizar tareas tales como generar sesiones de trabajo, intercambiar archivos, intercambiar aplicaciones, comunicarse vía servicios web y sincronizar documentos XML. La infraestructura desarrollada es, además, un framework completo para el desarrollo de aplicaciones colaborativas móviles.

La infraestructura propuesta fue probada en forma preliminar con dos aplicaciones colaborativas móviles. La primera fue un Chat ad hoc diseñado para ser utilizado en un entorno móvil, y la segunda fue una aplicación de asistencia a situaciones de emergencia. Los resultados obtenidos son muy alentadores y se espera que este trabajo de investigación sea un aporte a la colaboración entre los usuarios que operan dispositivos móviles.

Agradecimientos

Agradezco enormemente a mi profesor guía, Don Sergio Ochoa, por su constante apoyo y por brindarme la oportunidad de realizar este trabajo.

A Andrés Neyem quien, desde los inicios de este trabajo me brindó su ayuda y conocimientos.

A Danilo Carrasco, Cristian Moya y Claudio Oyarzún por su tiempo y colaboración a la hora de explicarme sus respectivos trabajos.

Agradezco también a mi familia y a todas las personas que durante este tiempo me han ayudado con comentarios, ideas o han escuchado mientras les hablo de mi tesis.

Finalmente quiero agradecer a mi esposa Carolina por estar siempre a mi lado.

Este trabajo de tesis ha sido parcialmente financiado por el proyecto Fondecyt Nro: 11060467, y el proyecto LACCIR Nro: R0308LAC005.

Índice de Contenido

1.	Introducción.....	1
1.1.	Motivación.....	2
1.2.	Marco Teórico	7
1.2.1.	Workspaces.....	7
1.2.2.	Redes MANET	8
1.2.3.	Colaboración no Prevista.....	11
1.3.	Problema a Resolver.....	12
1.4.	Hipótesis de Trabajo.....	14
1.5.	Objetivos.....	15
1.5.1.	Objetivo General.....	15
1.5.2.	Objetivos Específicos	15
1.6.	Resultados Esperados y Obtenidos.....	16
2.	Trabajo Previo	18
2.1.	Manejo de Sesiones Cerradas para Ambientes Colaborativos Móviles	18
2.2.	Servicios Web en Dispositivos Móviles para el Soporte de Aplicaciones Colaborativas	19
2.3.	Módulo para la Sincronización Automática de Documentos XML	19
3.	Solución Propuesta	22
3.1.	Requisitos de la Solución	22
3.1.1.	Mecanismo de Detección de Usuarios Dentro del Entorno de la Red.....	22
3.1.2.	Mecanismos para Proveer y Recibir Información sobre Posibles Formas de Colaboración con Otros Usuarios.....	24
3.1.3.	Mecanismos de Colaboración.....	25
3.2.	Arquitectura de la Solución	26
3.3.	Detección de Peers y Manejo de Usuarios	30
3.4.	Administración de Sesiones.....	33
3.4.1.	Sesión Ad-hoc.....	34
3.4.2.	Sesión Suscribible.....	34
3.5.	El Micro-servidor Web y el Disconnected Proxy Agent	35
3.6.	Comunicación entre Procesos (TCPInterProcessCommunicator).....	37
3.7.	Sincronización de Documentos (XMLSync).....	37
3.8.	Soporte para Notificaciones.....	39

4.	El Framework Desarrollado.....	41
4.1.	Estructura de una Aplicación Colaborativa	42
4.2.	Implementación de una Aplicación Colaborativa	44
4.2.1.	Implementación de la Interfaz ISharedApplication.....	44
4.2.2.	Implementación de Servicios Web Asociados	46
4.3.	Pruebas Realizadas	48
4.3.1.	Detección de Usuarios, Sesiones y Recursos Compartidos.....	49
4.3.2.	Traspaso de Archivos y Aplicaciones	52
4.3.3.	Comunicación entre Aplicaciones	52
4.3.4.	Comunicación no Prevista usando MapaMovil.....	54
4.3.5.	Pruebas de Desarrollo Usando el Framework	57
5.	Conclusiones y Trabajo Futuro.....	60
6.	Bibliografía y Referencias	63
	Anexo A: Código Fuente de los Principales Componentes del Framework	68
	Anexo B: Código Fuente de Aplicación MobileChat.....	81
	Anexo C: Iconografía de la Interfaz de Usuario del Manejador de Sesiones.....	86

Índice de Ilustraciones

Figura 1 Búsqueda y rescate asistido por computador	4
Figura 2 Trabajo móvil en un hospital.....	6
Figura 3 Sesiones de trabajo en un escenario colaborativo móvil.....	8
Figura 4 Arquitectura de una MANET Flat-routed.....	10
Figura 5 Capacidad de Interacción Actual	13
Figura 6 Capacidad de Interacción a Futuro.....	14
Figura 7 Componentes de la Solución.....	29
Figura 8 Mecanismo de detección de Peers.....	31
Figura 9 Atributos que generan diferencias en sincronización.....	39
Figura 10 Notificación de conexión de un usuario a la red	40
Figura 11 Notificación de invitación a una sesión recibida por el usuario local.....	40
Figura 12 Usuarios y sesiones detectadas por la herramienta	50
Figura 13 Gráfico Tiempo de prueba v/s Tamaño de adjuntos	52
Figura 14 MobileChat.....	53
Figura 15 PASIR Chat.....	53
Figura 16 Aplicaciones Colaborativas Móviles.....	58

1. Introducción

En los últimos años hemos visto cómo los dispositivos móviles se han vuelto parte de la vida cotidiana, y se han ido convirtiendo en herramientas cada vez más imprescindibles. Dispositivos como PDAs (Personal Digital Assistants) y teléfonos celulares aumentan cada día más sus capacidades en lo que respecta a almacenamiento, cómputo, visualización y conectividad. Este es un nicho de investigación abierto en el que se buscan soluciones a los crecientes desafíos que plantea la integración de las tecnologías móviles en las distintas actividades humanas. Esto va desde los escenarios de educación, hasta el apoyo a actividades en procesos productivos, e incluso su uso durante situaciones de emergencia.

Diversos avances tecnológicos están afectando el futuro del trabajo móvil, especialmente cuando éste es realizado con apoyo computacional utilizando comunicación inalámbrica [Brugnoli, 2005; Vuolle, 2008; Dutta, 2009]. Las tecnologías de telecomunicaciones que serán dominantes para apoyar este tipo de trabajo, se espera que sean la telefonía móvil y las evoluciones de WiFi (tales como WiMax y WiMax Mobile) [Brugnoli, 2005]. En el corto plazo, esto no sólo involucrará a los países desarrollados sino también a países con buena infraestructura de comunicaciones, como es el caso de Chile.

Hace unos pocos años se creía que la próxima generación de tecnología móvil para computación establecería el concepto de “siempre conectado”, en combinación con tarifas planas ofrecidas por los operadores [Schaffers, 2006]. Hoy vemos como este concepto está llegando a países como Chile, a través de redes de tipo 3.5/4G y WiMax [Emol, 2008; Subtel, 2009]. Esto estimula el uso de nuevos tipos de sistemas de información, enfocados a dar soporte a procesos de negocio extendidos. Estos procesos de negocios están presentes en las nuevas formas de operación de las organizaciones, como los modelos de organizaciones virtuales o descentralizadas [Franco, 2005; Vuolle, 2008]. Si a lo anterior sumamos el incremento del ancho de banda y al aumento de la capacidad de procesamiento,

es esperable que la inversión en el desarrollo de aplicaciones complejas, centradas en el trabajo del usuario, sea cada vez mayor.

Este trabajo de tesis aborda la problemática de la colaboración, dentro de un ambiente de trabajo móvil, desde la perspectiva de las situaciones de colaboración no previstas. Vale decir, donde existen dos o más usuarios con la necesidad de intercambiar información, pero no existe un mecanismo preestablecido específico para realizar dicho proceso. Más aún, los dispositivos que cada usuario posee, pudieran eventualmente tener características diferentes de software, hardware, y particularmente sistema operativo. Este trabajo de tesis busca apoyar la colaboración entre trabajadores nómadas, aunque esta heterogeneidad esté presente.

1.1. Motivación

Según un estudio de la IDC, entidad dedicada a estudios de mercado globales en el área de tecnologías de la información, en el 2004 el número de trabajadores móviles ascendía a 650 millones, concentrados principalmente en el Asia Pacífico, Estados Unidos y Europa Occidental [Drake, 2005]. Según el mismo artículo, se esperaba que para el 2009, este número ascendiera a 850 millones de trabajadores. Nuevos estudios de la IDC [BNet, 2008] indican que el número de trabajadores móviles ascenderá a más de mil millones para el 2011. Cifra que correspondería al 30% de la fuerza laboral del mundo.

En el futuro inmediato, las aplicaciones móviles serán más inteligentes y se acoplarán en forma segura, de acuerdo a los perfiles del usuario, al contexto de trabajo y a la disponibilidad del servicio de comunicación. Además, la reducción del costo de los dispositivos y del acceso a la red, el conocimiento del contexto circundante (inteligencia ambiental), y la comunicación basada en mensajería instantánea y audiovisual, serán importantes catalizadores del uso de aplicaciones colaborativas móviles. Por lo anterior, entre otras cosas, se espera una tendencia del desarrollo hacia los ambientes de trabajo

(workspaces) colaborativos móviles durante los próximos años [Nah, 2005; Shaffers, 2006; Andriessen, 2006; Iansiti, 2007; Vuolle, 2008].

Estos workspaces permitirán a los trabajadores móviles interactuar con otras personas o sistemas en forma ad-hoc (a través de mecanismos de tipo plug & play), adaptándose ellos mismos, en función de la información de contexto que cada dispositivo sea capaz de sensar [Tarasewich, 2003]. Estos workspaces representan un espacio de trabajo que puede ser visto como una porción de una oficina (que contiene información, procesos y servicios), la cual está disponible a través del dispositivo móvil de un trabajador nómada. Este tipo de herramientas permite a los trabajadores realizar sus actividades, casi en cualquier momento y en cualquier lugar, logrando de esta manera un impacto positivo en la productividad y la calidad del trabajo realizado. Con el fin de ilustrar el rol de los workspaces colaborativos móviles y su visión para el futuro próximo, a continuación se describen brevemente dos posibles escenarios de aplicación.

Escenario I – Asistencia a Desastres

Las actividades de búsqueda y rescate en escenarios de desastres (o grandes emergencias), tales como huracanes, tsunamis, terremotos, ataques terroristas, derrames químicos, o grandes incendios, son altamente dinámicas y requieren la colaboración eficaz entre un amplio espectro de organizaciones. Esta colaboración es necesaria debido a que cada entidad está especializada en resolver una parte del problema. Típicamente, la policía se hace cargo de aislar y asegurar el área afectada, los bomberos son los responsables de proteger a los ciudadanos y la infraestructura física, el personal médico es responsable de la atención sanitaria de las personas afectadas, y las autoridades del gobierno son responsables de coordinar los esfuerzos entre las organizaciones participantes y tomar las mejores decisiones para reducir el impacto del desastre sobre la sociedad [Comfort, 2004]. Los equipos de respuesta a estos incidentes (policía, bomberos y personal médico) suelen desplegarse a lo largo de toda el área de trabajo y necesitan mantener información actualizada sobre el área afectada (por ejemplo: mapas, planos de edificios y probables ubicaciones de víctimas), las rutas de ingreso/evaluación definidas, los recursos

desplegados en el área, los recursos disponibles en la zona y la asignación de tareas, entre otros. Algunas personas pertenecientes a las organizaciones participantes, podrían utilizar dispositivos de computación móvil (por ejemplo PDAs: Personal Digital Assistants) interconectados a través de redes inalámbricas ad-hoc [Monares, 2009a]. De esta manera, dichas personas podrán mantener comunicación con sus pares y podrán organizar el trabajo a realizar a través de múltiples workspaces [Ochoa, 2006; Monares, 2009a].



Figura 1 Búsqueda y rescate asistido por computador

Cada bombero a cargo de un grupo de búsqueda y rescate (líderes de equipo) podrá utilizar un PDA adosado a su brazo (ver Figura 1). Ésta le permitirá recuperar la información entregada por el puesto de mando, actualizar el estado de las actividades asignadas, y organizar el trabajo de los miembros del grupo. Cada líder de grupo usa un espacio de trabajo, que se conecta a los espacios de trabajo de los otros líderes desplegados en el área, y también se conecta al espacio de trabajo del jefe de bomberos (si es que éste está en el área). Por lo tanto, la información (órdenes, mapas, notificaciones y el estado de trabajo) puede circular en forma automática entre las personas involucradas [Ochoa, 2007].

Esta estrategia para compartir información podría ser utilizada por otras organizaciones, también involucradas en la asistencia a situaciones de emergencia, como por ejemplo el ejército o defensa civil. Típicamente, las autoridades del gobierno encargadas de las macro-decisiones son capaces de acceder a varios espacios de trabajos compartidos (de policías, bomberos y personal médico), a fin de poder monitorear la evolución del proceso de respuesta a la emergencia y transmitir las órdenes hacia las

distintas organizaciones participantes. Diversos mecanismos de advertencia, que se adaptan según la información de contexto que poseen, son usados para disparar alarmas y proveer percepción de la evolución del trabajo entre las personas que están utilizando los dispositivos móviles. Este tipo de herramienta también puede ser aplicado a otro tipo de escenarios, por ejemplo a la industria manufacturera y la construcción.

Escenario II - Servicios de Salud

El sector de servicios de salud está avanzando en el *cambio interno* y en el *externo*. Los cambios internos son producto de la continua innovación tecnológica que amplía los métodos y herramientas disponibles, y también del incremento de la complejidad organizacional. Por otra parte, el cambio externo está asociado al incremento de la demanda de los ciudadanos con mayor poder adquisitivo, a la necesidad de reducir costos, y a las implicancias de la evolución debido a cambios demográficos (como por ejemplo, el envejecimiento).

Estos cambios han llevado a que las organizaciones dedicadas a proveer servicios de salud investiguen nuevas formas de organizar y entregar dichos servicios. Las tecnologías de información y comunicaciones podrían tener un papel importante en la reorganización de los servicios de salud, facilitando y permitiendo nuevas formas confiables de trabajo, de colaboración y de compartir la información [Tentori, 2008]. Las actividades que llevan a cabo los trabajadores nómades en terreno (por ejemplo durante campañas de salud en poblaciones) podrían ser apoyadas por tecnología móvil, a través del uso de espacios de trabajo virtuales. Éstos les permitirían a los profesionales, tener acceso a información relevante para diagnosticar o determinar tratamientos a los pacientes. También permitiría, tanto a los pacientes como a los profesionales, realizar inter-consultas a fin de obtener una segunda opinión, dejando disponible los datos del paciente a través de diversos formatos (incluyendo información multimedial). En los servicios dedicados a la salud, los pacientes generalmente están distribuidos en diferentes ubicaciones (habitaciones, pisos y edificios) y el equipamiento del centro de salud está en lugares fijos. Diferentes workspaces que se

ejecutan en Tablet PCs o PDAs pueden ser usados por los médicos y enfermeras para supervisar y actualizar el estado del tratamiento de un paciente o de su historia clínica.



Figura 2 Trabajo móvil en un hospital

Un espacio de trabajo colaborativo móvil permitiría asistir a varios pacientes, cada uno con su propia información asociada (historias clínicas, resultados de exámenes, etc.). El estado del espacio de trabajo puede ser replicado en los sistemas de información del hospital, y también en los dispositivos de computación móvil de las personas encargadas del cuidado de tales pacientes. Los médicos se pueden reunir para analizar un caso complejo, utilizando como apoyo la información disponible en sus workspaces. Si necesitan más información, a través de sus espacios de trabajo podrían acceder a las bases de datos del centro de salud, donde podrían encontrar la información de casos similares. Todo esto se hace con la finalidad de hacer diagnósticos y tomar decisiones más rápidas y certeras. Los diagnósticos, las decisiones y sus consecuencias deberían también ser registradas en los sistemas de información del hospital y en los espacios de trabajo de los médicos, para que puedan luego ser utilizados en casos similares en el futuro.

A veces los pacientes requieren ser transportados, con sus historias clínicas, hacia otros hospitales o centros de salud para seguir con sus tratamientos. En tales casos, la información contenida en el espacio de trabajo asociado al paciente debería ser transferida al nuevo centro de salud.

1.2. Marco Teórico

A continuación se presenta el marco teórico involucrado en este trabajo de tesis.

1.2.1. Workspaces

Entendemos por workspace, o espacio de trabajo, una infraestructura compuesta de una serie de herramientas que permiten a un usuario desarrollar una determinada actividad o trabajo. Más particularmente podemos definir un workspace como un conjunto de herramientas de software y hardware, que cuenta con una organización orientada a permitir y/o facilitar el trabajo de un usuario sobre un dispositivo computacional.

Yendo un poco más lejos podemos hablar de un workspace colaborativo como aquel orientado a dar soporte a tareas cooperativas. Estos proporcionan a los usuarios un espacio virtual dentro del cual la información puede ser compartida e intercambiada [Neyem 2007]. Dicha información compartida puede corresponder a documentos, mensajería, o incluso aplicaciones y datos.

La figura 3 muestra cuatro usuarios utilizando un workspace colaborativo, que les permite compartir recursos y asignarlos a diferentes sesiones de trabajo. En este ejemplo, cada usuario cuenta con un workspace local en su dispositivo. En él pueden realizar sus tareas de manera independiente de lo que ocurra con los demás usuarios. Ahora bien, existe un espacio virtual de intercambio de información denominado “Entorno compartido”. Allí cada sesión de trabajo posee un “espacio público”, el cual está rotulado como “Sesión A” o “Sesión B” en el centro de la Figura 3.

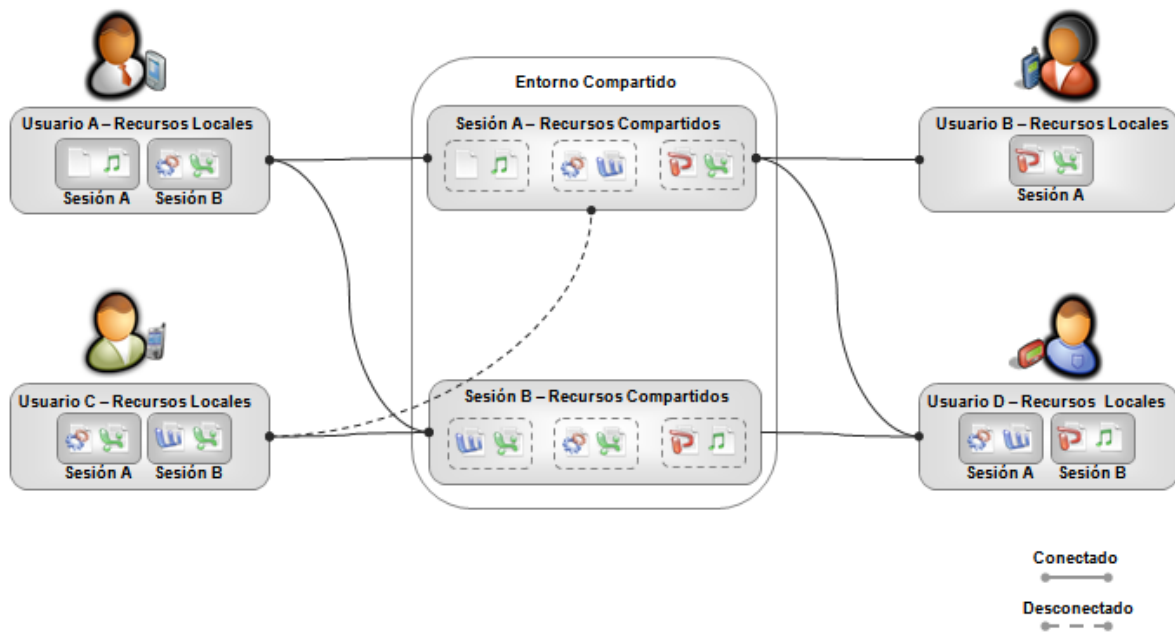


Figura 3 Sesiones de trabajo en un escenario colaborativo móvil

El espacio compartido de cada sesión de trabajo, puede ser accedido por cualquier miembro de la sesión. De esa manera, un usuario puede enviar, recibir y sincronizar datos con los demás usuarios de la sesión, convirtiendo de esta forma su workspace privado en un workspace colaborativo. Ahora bien, si consideramos que cada dispositivo es móvil y se encuentra interconectado con otros dispositivos a través de una red inalámbrica, entonces podemos hablar de un workspace colaborativo móvil. Ese es particularmente el ámbito en el cual se desarrolla este trabajo de tesis.

1.2.2. Redes MANET

Cuando hablamos de un workspace colaborativo móvil nos referimos a un sistema de apoyo al trabajo colaborativo, el cual se encuentra montado sobre una infraestructura de red móvil o no cableada, vale decir, una red inalámbrica. Nos centraremos en un tipo particular de estas redes inalámbricas, las cuales son conocidas como redes MANET (Mobile Ad-Hoc Network) [Perry, 2001; Wang, 2005]. Esta clase de redes está formada por un conjunto de nodos (o peers) conectados entre sí, sin necesidad de contar con uno o más

entes centralizados (por ejemplo, Access Points) que los controle y/o almacene información. Nos centramos en este tipo de redes en particular porque el escenario de colaboración que proveen corresponde a un ambiente improvisado y cambiante en el tiempo, que, al ser además descentralizado, debe necesariamente ser capaz de permitir la colaboración no prevista para soportar el trabajo colaborativo.

Las MANET son redes peer-to-peer (P2P), complementarias a las redes de infraestructura fija, que generalmente utilizan el protocolo IEEE 802.11x o Bluetooth para acceder al medio. Esta infraestructura ad-hoc de comunicaciones permite a usuarios móviles comunicarse usando diversos dispositivos, tales como: PDAs, Smart phones y Notebooks, entre otros. La diferencia esencial que tienen las redes MANET respecto a las redes con infraestructura fija, es que son sistemas autónomos y auto-organizados compuestos de nodos inalámbricos que cooperan para establecer una comunicación dinámica. Estos nodos pueden moverse libre y aleatoriamente a través de un escenario físico real. De esta manera, la topología de una MANET puede cambiar rápidamente y de forma imprevisible. Por ejemplo, una MANET podría dividirse en varias sub-redes, y luego volver a integrarse en un lapso de unos pocos minutos. Además, el desplazamiento de los nodos a través de un escenario físico real los expone a interferencias producidas por señales externas u obstáculos como paredes y pisos, entre otros.

La figura 4 muestra una arquitectura de una MANET típica (conocida en inglés como *Flat-routed*), en la cual todos los nodos son idénticos en términos de responsabilidades, y no existe el concepto de *gateways* especiales. Cada nodo tiene un área de cobertura, que constituye la distancia máxima a la que es capaz de captar y enviar señales, superada esta distancia, el nodo queda aislado del resto. Un tema importante a resolver para la colaboración, es lograr que un nodo aislado mantenga su capacidad de trabajo en forma asíncrona, de manera que cuando el nodo vuelva a interactuar con otros nodos de la MANET, el sistema que soporta la colaboración móvil debe sincronizar las operaciones en ambos sentidos, hasta obtener un estado compartido coherente.

La combinación de sistemas móviles y redes MANET lleva a las soluciones hacia sistemas distribuidos móviles, compuestos de nodos que continuamente cambian su localización física y establecen conexiones de pares, basándose en la proximidad de otros nodos. Los nodos en ocasiones interactúan durante breves encuentros físicos y en ese momento intercambian datos. Para mantener un cierto orden en este medio altamente inestable, estas interacciones se llevan a cabo a través de sesiones implementadas sobre la red.

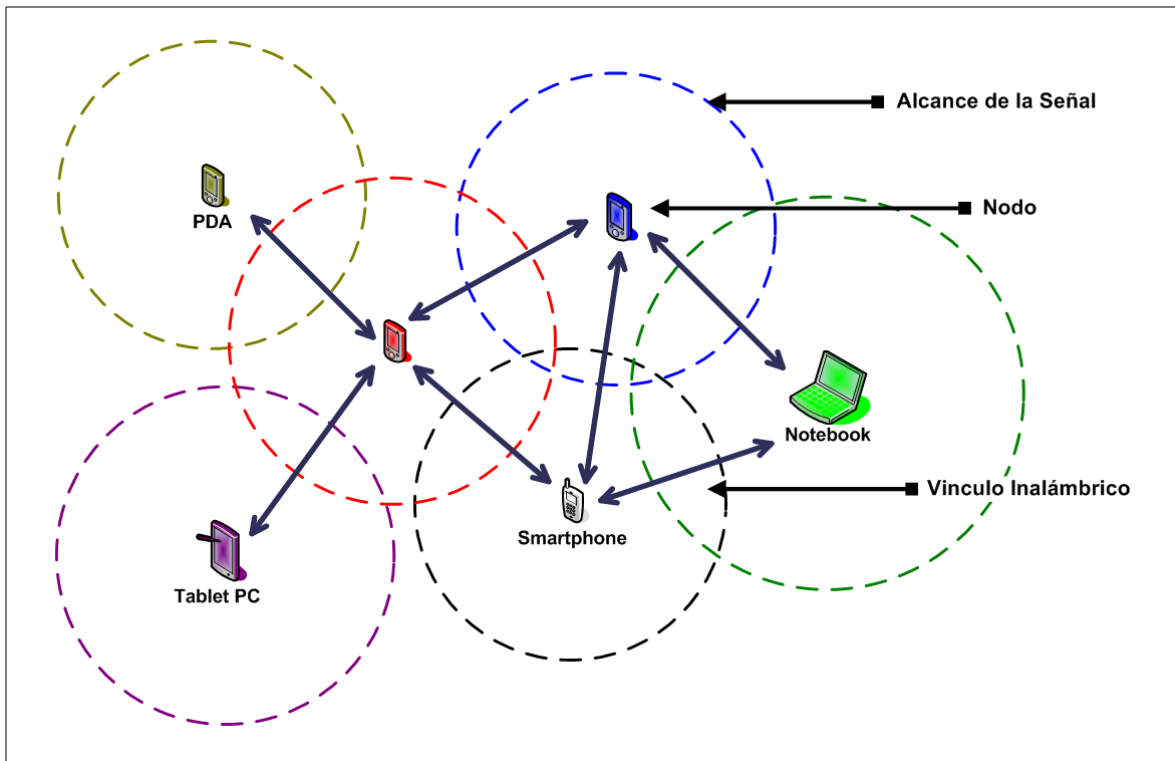


Figura 4 Arquitectura de una MANET Flat-routed

Si vemos una red MANET como una especie de nube generada por un conjunto de nodos (dispositivos móviles), podemos vislumbrar situaciones que no ocurren en las redes cableadas o en aquellas mediadas por *Access Points*. Un nodo maneja información sobre todos los nodos que se encuentran a su alcance y desconoce a los nodos que no lo están. Cada nodo se encuentra en una situación y ubicación única dentro de la red, y la topología que cada uno percibe es distinta en cada caso. Indudablemente, en un lugar con condiciones ideales, todos los nodos de una red se verán unos a otros, pero esto no es lo que ocurre en la

realidad. Constantemente los nodos desaparecen y reaparecen en la red, ya sea porque se alejaron demasiado, surgieron interferencias en el medio, o simplemente porque se les acabó la batería.

Todas las particularidades mencionadas sobre las redes MANET, y por tanto, de un ambiente en que se espera que exista colaboración no prevista, definen una serie de funciones mínimas requeridas para soportar el trabajo colaborativo. Estas funciones son las siguientes: detección de peers, mantención de estados a lo largo del tiempo, e intercambio y sincronización de datos. Estas funciones se explican en detalle a lo largo de este trabajo de tesis.

1.2.3. Colaboración no Prevista

En los escenarios de trabajo móvil podemos clasificar los tipos de colaboración en dos categorías: prevista (*foreseen*) y no prevista (*unforeseen*). El primer tipo de colaboración ocurre cuando los trabajadores siguen un proceso previamente definido para llevar a cabo la colaboración. Esto es, un proceso que ha sido considerado durante el diseño del espacio de trabajo usado como intermediario para colaborar. Este tipo de colaboración está presente, por ejemplo, en un escenario de asistencia a catástrofes, cuando dos compañías de bomberos se reúnen para apoyarse mutuamente en tareas de búsqueda y rescate. En ese caso, la colaboración entre los usuarios de los espacios de trabajo móviles fue previamente considerada en el diseño de la aplicación que ellos usan. Siguiendo con el ejemplo, la colaboración no prevista puede ser realizada por el jefe de la compañía de bomberos que trata de compartir información con el personal de la policía o del gobierno. Estas personas no poseen un espacio de trabajo diferente para poder interactuar con los trabajadores de cada organización participante. Típicamente ellos tienen un único workspace, que es el que conocen y usan para llevar a cabo sus tareas. Por lo tanto, ese workspace necesita proporcionar mecanismos de apoyo a la colaboración no prevista. De esa manera, trabajadores de distintas organizaciones podrán interactuar entre ellos, sin necesidad de tener que contar con diferentes workspaces en cada dispositivo móvil.

Actualmente los workspaces móviles no permiten la interacción no prevista entre ellos. Una de las principales razones es la falta de compatibilidad entre los frameworks que apoyan el desarrollo de estos productos, como por ejemplo LaCOLLA [Marques, 2005], iClouds [Heinemann, 2003], YCab [Buszko, 2001] y la plataforma de Nokia [Hirsch, 2006]. Por otra parte, estos frameworks no adhieren a soluciones estándares (como por ejemplo, servicios web), que podrían ser un camino para solucionar estos problemas. Específicamente, un enfoque como el planteado en el artículo “A Semantic Web Services GIS based Emergency Management Application” [Tanasescu, 2006] podría servir para identificar semánticamente los servicios que exponen las distintas organizaciones participantes. Sin embargo, debido a la alta tasa de desconexión que tienen las redes ad-hoc móviles (MANET), sería recomendable la replicación y/o migración de servicios Web entre los dispositivos que colaboran, y la ejecución dinámica de estos servicios. De esa manera, la funcionalidad disponible en el escenario de trabajo podría extenderse, permitiendo aumentar la capacidad de colaboración con que cuentan las personas.

1.3. Problema a Resolver

En el escenario actual, si dos o más usuarios desean colaborar entre ellos, sus workspaces deben estar diseñados para trabajar juntos. Supongamos, retomando el caso de la asistencia a un desastre, que el grupo de bomberos utilizan workspaces de tipo A y la policía, workspaces de tipo B (Figura 5). Ambas partes pueden interactuar con unidades que pertenecen a la misma organización, pero no pueden interactuar con unidades pertenecientes a una organización distinta a la de ellos, pues sus workspaces son estancos. Esto significa que no son capaces de interoperar con workspaces heterogéneos.

Si bien está perfecto que cada usuario móvil pueda interactuar con los miembros de su organización, también se requiere que lo puedan hacer con usuarios de otras organizaciones. La emisión de alarmas y la distribución de información son dos servicios que es importante que estén disponibles más allá de los límites de una organización. Con

esto se podría lograr una mejor coordinación, capacidad de reacción y eficiencia en el trabajo realizado por las diversas organizaciones participantes.

Soportar la colaboración no prevista involucra abordar diferentes desafíos, que van desde mantener la sensibilidad respecto del contexto (*context-aware*), hasta poder ejecutar y migrar código en forma dinámica (*mobile code*), incluyendo el soporte de datos asociado. Si se considera el compartir información como uno de los servicios importantes a brindar en este escenario, hay que considerar no sólo la transferencia de archivos entre máquinas que se están moviendo (topología de la red cambiante), sino también la posibilidad de sincronizar servicios y datos entre dos o más usuarios. Este desafío aumenta, si además se pretende que las sincronizaciones sean automáticas, para que de esa forma se vuelvan transparentes para el usuario.

En este trabajo de tesis se aborda una parte de esta problemática. Para ello se diseñó e implementó una infraestructura que pueda ser agregada a un workspace colaborativo móvil (como una especie de interfaz de servicios), a fin de obtener servicios de interacción básicos entre espacios de trabajo heterogéneos (Figura 6). Utilizando esta interfaz, los servicios propietarios de cada workspace deberían poder replicarse o migrar entre máquinas a fin de extenderlos, tanto como los usuarios lo permitan.

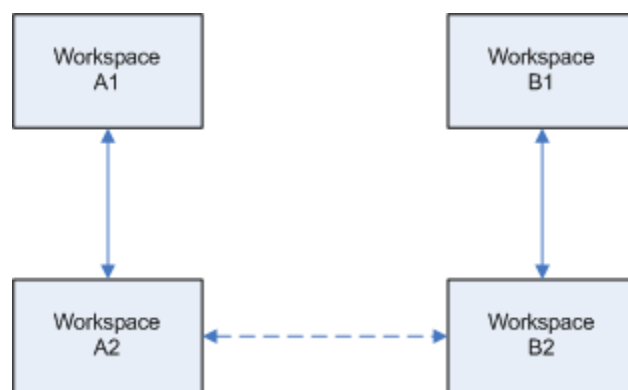


Figura 5 Capacidad de Interacción Actual

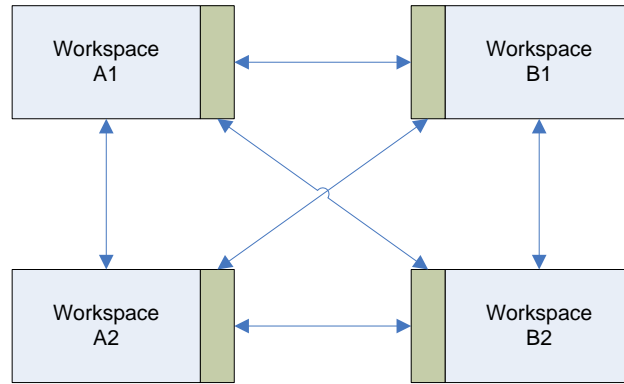


Figura 6 Capacidad de Interacción a Futuro

Esta infraestructura deberá funcionar de forma tan autónoma y transparente (para los usuarios finales) como sea posible, de manera que no se requieran procesos de configuración para que dos o más workspaces diferentes puedan interactuar. Esta característica de la infraestructura hará a la solución más usable y atractiva.

Actualmente los workspaces colaborativos móviles son aplicaciones con poca historia de uso, pues son una consecuencia de los más recientes avances en computación móvil y comunicaciones inalámbricas. Particularmente, los ámbitos de asistencia a desastres, servicios de salud y la industria de la construcción, son algunos de los nichos donde se ha visto la potencial ventaja que el uso de estos workspaces podrían tener [Schaffers, 2006; Tentori, 2008; Ochoa, 2009].

1.4. Hipótesis de Trabajo

Las hipótesis de trabajo definidas son las siguientes:

H1: La definición de un conjunto de servicios básicos, que corran en forma transparente para los usuarios, permitirá llevar a cabo la colaboración no prevista entre usuarios móviles que utilizan equipos heterogéneos.

H2: El uso de la infraestructura de software que implemente estos servicios básicos, permitirá reducir el esfuerzo de desarrollo de aplicaciones colaborativas móviles y aumentar su probabilidad de éxito.

1.5. Objetivos

Esta sección presenta el objetivo general, y los objetivos específicos que fueron definidos para este trabajo de tesis.

1.5.1. Objetivo General

El objetivo de este trabajo de tesis es diseñar, implementar y probar una infraestructura de software que facilite la interoperabilidad entre workspaces colaborativos móviles heterogéneos.

1.5.2. Objetivos Específicos

Los objetivos específicos que se derivan del objetivo general son los siguientes:

1. Determinar el conjunto de servicios necesarios para permitir interoperabilidad básica entre workspaces heterogéneos. La interoperabilidad planteada cubre los servicios de mensajería, de sincronización de información y transferencia de archivos (incluyendo servicios web), entre dos o más colaboradores.
2. Diseñar e implementar la infraestructura de software que permite la interoperabilidad planteada. Para esto se llevó a cabo un desarrollo basado en prototipos, utilizando PDAs y TabletPCs como dispositivos de computación móvil.

3. Probar la infraestructura desarrollada en un escenario simulado, y ajustar la solución en caso de ser necesario. Estas pruebas se llevaron a cabo en el Departamento de Ciencias de la Computación, utilizando dos aplicaciones: *MobileChat* y *MobileMap*.

La primera es una aplicación de Chat diseñada para ser usada en entornos móviles e implementada por el autor de esta tesis. La segunda aplicación fue desarrollada por el Ing. Álvaro Monares, como trabajo de memoria para obtener el título de Ingeniero civil en computación [Monares, 2009a]. Esta aplicación implementa un workspace colaborativo móvil, llamado *MobileMap* [Monares, 2009b], de apoyo a la labor de bomberos durante una emergencia. Actualmente esta herramienta está siendo probada por la segunda compañía de bomberos de Ñuñoa.

1.6. Resultados Esperados y Obtenidos

Los resultados de las pruebas realizadas fueron altamente satisfactorios (en la sección 4.3 se explica en detalle el proceso de experimentación y los resultados obtenidos). Sin embargo, éstas aún son insuficientes y preliminares. Por lo tanto se espera que las principales contribuciones de este trabajo apunten al mejoramiento de las actividades de los usuarios móviles en terreno. Desafortunadamente, debido a la importancia y criticidad del trabajo de los bomberos, el prototipo experimental usado para validar este trabajo, recién se está comenzando a utilizar como soporte en algunas situaciones de emergencias reales. Por lo tanto, la evaluación del proceso de colaboración no prevista en un escenario real, es aún una incógnita.

Como resultado de este trabajo se diseñó e implementó una infraestructura de soporte, que permite la interoperabilidad entre distintos espacios de trabajo móviles, y por consiguiente también permite la colaboración no prevista. Tanto el diseño como la implementación de la solución pueden ser usadas como base para agregar interoperabilidad a otras aplicaciones de apoyo al trabajo colaborativo móvil. Particularmente, algunas áreas de aplicación como la salud, la construcción y la industria manufacturera podrían verse beneficiadas con la solución de propuesta por este trabajo.

En el caso puntual del escenario de emergencias, se espera que la infraestructura construida ayude a:

1. *Mejorar la coordinación entre las organizaciones participantes.* Actualmente estas organizaciones utilizan 2 o 3 canales de radio para comunicarse entre ellos y tratar de mantener algún tipo de coordinación entre sus actividades [Ochoa, 2006]. Claramente esto es insuficiente si se pretende coordinar a más de 6 o 7 equipos de trabajo. Se espera que la infraestructura propuesta ayude a coordinar el esfuerzo de los participantes, a través de servicios de información y notificaciones comunes entre ellos. Estos servicios permiten parte de la interoperabilidad propuesta en términos de mensajería.
2. *Mejorar la toma de decisiones.* En el escenario de desastre hay mucha improvisación, especialmente por parte de aquellas personas que necesitan actuar, pero carecen de la información necesaria para tomar una decisión correcta y a tiempo [Ochoa, 2007]. Se espera que la infraestructura construida ayude a la transferencia de información entre los nodos participantes, y de esa manera se tomen decisiones más certeras y oportunas.
3. *Mejorar la propagación de información.* Debido a que los workspaces tienen mecanismos para proveer interoperabilidad de mensajes, se espera que la propagación de la información entre organizaciones sea más fluida. Lo mismo se espera que ocurra con las señales de alarma que son emitidas ante una situación de alto riesgo para los socorristas.

2. Trabajo Previo

Parte de la solución de software desarrollada en esta tesis se basa en soluciones anteriores creadas en tres memorias de ingeniero, realizadas durante el año 2007; una de las cuales corresponde a la del autor de esta tesis. Cada una de esas soluciones fue integrada, en forma de módulos, a la infraestructura aquí propuesta. Dichos módulos corresponden a un administrador de usuarios y sesiones [Castro, 2007], un micro-servidor Web [Carrasco, 2007] y un sistema de sincronización de documentos XML [Moya, 2007]. A continuación se describen brevemente las características de cada solución utilizada.

2.1. Manejo de Sesiones Cerradas para Ambientes Colaborativos Móviles

El trabajo desarrollado corresponde a un administrador de usuarios y sesiones de trabajo para ambientes colaborativos móviles [Castro, 2007]. La solución provee mecanismos para manejo y detección de usuarios y administración de diferentes tipos de sesiones de trabajo colaborativo, todo esto apuntado al trabajo sobre redes MANET.

Cada usuario tiene la posibilidad tanto de crear distintos tipos de sesiones, como de suscribirse a sesiones creadas por otros usuarios. Además, la solución provee diferentes tipos de sesiones que van desde el tipo de sesión más básica (una sesión compartida por todos los miembros de la red móvil), hasta sesiones cerradas con diversos mecanismos para brindar privacidad a sus usuarios. Un mecanismo de privacidad sería, por ejemplo, la necesidad de una invitación para suscribirse o mecanismos de encriptación en el traspaso de información. El caso más elaborado de sesiones cerradas, es aquel que implementa sesiones cerradas con múltiples roles, en las que un usuario actúa como administrador. Este administrador tiene la capacidad de crear diversos roles (con distintas atribuciones), y asignarlos a los usuarios de la sesión.

2.2. Servicios Web en Dispositivos Móviles para el Soporte de Aplicaciones Colaborativas

Entre los trabajos previos sobre los que se apoya esta tesis se encuentra el desarrollo de un micro-servidor Web, el cual fue optimizado para poder funcionar (exponer y consumir servicios Web) en dispositivos móviles pequeños [Carrasco, 2007]. La solución apunta a utilizar el mínimo posible de recursos de hardware, ateniéndose estrictamente a los estándares existentes para servicios Web. Esta solución además implementa WS-Attachments, un estándar de servicios Web que permite el envío de datos adjuntos dentro de un mensaje SOAP. Este micro-servidor fue desarrollado utilizando .Net Compact Framework y librerías de OpenNETCF, las cuales agregaron la capacidad de manejar datos adjuntos en la mensajería del servicio.

El trabajo consta también de una serie de mediciones que muestran las ventajas de mandar, como respuesta a una solicitud, datos adjuntos versus enviarlos como parámetros de la respuesta. Las mediciones muestran una notoria diferencia de rendimiento entre ambas alternativas. De esta forma, el micro servidor es capaz de transmitir no sólo documentos de texto, sino que también fotos, videos, e incluso aplicaciones.

Por último, el micro-servidor implementa también el estándar WS-Security, utilizando un UsernameToken; proveyendo así un cierto nivel de seguridad a los servicios Web.

2.3. Módulo para la Sincronización Automática de Documentos XML

Este trabajo consistió en la implementación de una serie de algoritmos que permiten la sincronización de documentos XML [Moya, 2007]. La solución fue también programada en .Net Compact Framework y pensada para funcionar en dispositivos móviles pequeños.

Mediante el uso de árboles de diferencias, generados a partir de dos o más árboles correspondientes a documentos XML distintos, la solución es capaz de sincronizar estos archivos. Los archivos a sincronizar tienen que tener una base común, o sea, deben haberse derivado a partir de un documento XML común.

El algoritmo de sincronización es capaz de encontrar nuevos nodos o ramas agregados en un árbol (representación de un documento XML), así como también es capaz de detectar nodos que hayan tenido modificaciones en sus valores internos. Luego, mediante el uso de resolutores (o algoritmos de resolución de conflictos), la solución es capaz de generar un documento unificado de las distintas versiones existentes, el cual corresponde al documento sincronizado.

En lo que respecta a los resolutores, éstos permiten múltiples criterios para la sincronización de los datos, tales como fecha de modificación, valor mayor, y orden alfabético. Más aún, se deja abierto el sistema de manera tal que el usuario desarrollador de aplicaciones pueda incorporar sus propios algoritmos resolutores. O sea, sincronización a la medida de las necesidades.

2.4. Trabajos Relacionados

Además de los trabajos en los que forman la base de esta tesis, existen otros trabajos que guardan cierta relación con el tema propuesto, al estar orientados al trabajo colaborativo móvil. Aunque la gran mayoría de los trabajos existentes en la actualidad para trabajo móvil están sujetos a la condición de que exista un ente central que regula la colaboración y maneja toda la información necesaria, existen líneas de investigación orientadas al trabajo colaborativo en redes móviles adhoc, o MANETs, y así, también hay ejemplos de softwares que actualmente proveen diversos niveles de colaboración, entre estos se pueden citar los siguientes:

- YCab e YCab.net [Buszko, 2001; Procopio, 2002]: Es un sistema para el trabajo colaborativo que consta de sesiones asociadas a un IP y puerto Multicast. En YCab

los usuarios colaboran mediante diferentes aplicaciones y servicios predefinidos en el software (visualizador de imágenes, mensajería, pizarra, traspaso de archivos, etc.). Además YCab posee una API para el desarrollo de nuevas aplicaciones móviles. YCab no tiene permisos asociados a usuarios.

- iClouds [Heinemann, 2003]: Este sistema de colaboración no maneja sesiones, los usuarios pueden intercambiar recursos por medio de las listas denominadas *iHaveList* y *iWishList* que son comparadas cuando dos usuarios se encuentran. Es un sistema que sirve básicamente para intercambiar archivos.

3. Solución Propuesta

Como parte de este trabajo de tesis se diseñó e implementó una infraestructura de software enfocada a resolver el problema de la colaboración no prevista entre espacios de trabajo (workspaces) colaborativos móviles. La arquitectura de esta solución está basada en PASIR (*Platform for Ad-hoc Sharing Information Resources*)[Neyem, 2005] y en adelante nos referiremos con el mismo nombre a la solución desarrollada, que es, una ramificación del concepto original de PASIR. Cuando se habla de interconectar workspaces móviles sin saber a priori qué elementos son los que se desean compartir, se pueden establecer una serie de funcionalidades mínimas con las que una infraestructura de soporte debe contar. Estas funcionalidades son las siguientes:

- Mecanismo de detección de usuarios dentro del entorno de la red.
- Mecanismos para proveer y recibir información sobre posibles formas de colaboración con otros usuarios.
- Mecanismos para realizar la colaboración en sí misma.

A continuación se explican brevemente cada uno de estos requisitos, y la forma en que se abordaron para cumplir los objetivos propuestos.

3.1. Requisitos de la Solución

Vale aclarar que los requisitos a los que se refiere en esta sección corresponden sólo a aquellos que apoyan la colaboración no prevista.

3.1.1. Mecanismo de Detección de Usuarios Dentro del Entorno de la Red

Ya sea que se trate de una red cableada y centralizada, o de una red móvil distribuida, siempre que se habla de trabajo colaborativo se vuelve obligatoria la existencia de usuarios que llevan a cabo una actividad persiguiendo una meta común. En particular, para el escenario presentado en este trabajo, se debe contar con un sistema que permita detectar a otros usuarios (en la vecindad) dentro de una red de topología altamente variable. Lo anterior implica, no sólo detectar la presencia de otros usuarios dentro de la red, si no también, ser capaz de proveer mecanismos que permitan la interacción a pesar de que, dadas las características de una red MANET, no siempre los usuarios están conectados el uno con el otro.

La solución propuesta consta de un mecanismo de detección de usuarios (o detección de peers) basado en mensajería UDP multicast. Cada peer o dispositivo móvil envía periódicamente mensajes de identificación al medio, estos mensajes pueden o no ser recibidos por otros dispositivos. En el caso de que se reciba un mensaje de identificación se sabrá que existe otro dispositivo dentro de la red y, a partir de esto, será posible entablar comunicación para obtener mayor información si se desea.

Un segundo paso en la detección de usuarios es la mantención en el tiempo del espacio colaborativo virtual en el que interactúan uno o más usuarios. Llamamos a este espacio “sesión”, y lo definimos como un espacio de trabajo compartido por uno o varios usuarios dentro de la red. Para efectos de esta tesis, entendemos el espacio colaborativo dentro de una sesión como un espacio virtual de trabajo definido por la interacción de los distintos usuarios, lo que es diferente al concepto de workspace de trabajo que se entiende como el espacio particular de cada usuario.

En lo que respecta a sesiones, la solución contempla una serie de mecanismos y protocolos orientados a establecer sesiones que perduren en el tiempo, permitiendo de esta forma, que una vez que se ha iniciado una interacción entre uno o más usuarios, ésta pueda ser retomada en cualquier momento, e incluso el usuario pueda trabajar en forma aislada, para luego, una vez restablecida la comunicación con otros usuarios, restablecer la sesión y sincronizar información.

Finalmente se definen distintos tipos de sesiones para diferentes escenarios colaborativos. Es así como existe una sesión general, a la que todos los usuarios pertenecen, sesiones públicas, a las que cualquier usuario puede acceder, y sesiones privadas, que requieren la invitación de otro usuario para acceder. La existencia de diferentes tipos de sesiones y la posibilidad dada a los usuarios de pertenecer a una o más sesiones a la vez, permite mantener la información de los distintos procesos colaborativos en que participa el usuario. Estos procesos se realizan en el marco de una cierta sesión de trabajo, lo cual permite manejar políticas particulares para cada situación. Un usuario puede, además de pertenecer a una sesión, decidir cuándo estar conectado a dicha sesión y cuándo no. En resumen, ahora se cuenta con un sistema de sesiones que permite a los usuarios decidir cómo y en qué momento entablar una relación colaborativa con los demás usuarios de la red.

3.1.2. Mecanismos para Proveer y Recibir Información sobre Posibles Formas de Colaboración con Otros Usuarios

Una vez establecidos los mecanismos básicos de detección de usuarios y manejo de sesiones, el siguiente paso es establecer una serie de funcionalidades que permitan a cada usuario conocer, y dar a conocer, las opciones de interacción que éste tiene respecto del resto de los usuarios de la red.

En primer lugar, la solución provee un mecanismo de alertas (*awareness*) que proveen información de eventos y recursos presentes en la red en la que se encuentra el usuario. Estas alertas pueden ser, por ejemplo, entrada o salida de usuarios a sesiones, solicitudes para iniciar algún proceso colaborativo (como podría ser la descarga o envío de algún documento), información sobre nuevos recursos o aplicaciones disponibles, etc. Existen dos mecanismos que determinan la visibilidad de las alertas, estos son: (1) el usuario debe pertenecer a la sesión en la que se está generando la alerta la sesión, esto es, el usuario sólo recibe notificaciones de las sesiones a las que pertenece, y (2) existe la

posibilidad de que el usuario opte por ignorar cierto tipo de mensajes aplicando filtros sobre ellos.

Ahora bien, al margen de las alertas internas por sesión, existen las alertas que corresponden a toda la red a la que los usuarios pertenecen. Éstas básicamente se refieren a alertas de invitaciones recibidas por el usuario para unirse a determinadas sesiones. En este punto se debe tener el cuidado de no confundir lo que es una colaboración no prevista, con lo que es una colaboración no deseada. Ésta última podría resultar una molestia para el usuario, ya que eventualmente estaría recibiendo alertas sobre eventos que no le interesan. Para solucionar este tema se proponen dos alternativas: (1) que una vez recibida una alerta, el usuario pueda optar por ignorar cualquier nueva alerta relacionada con la sesión correspondiente, y (2) que el usuario pueda optar por un modo en el cual no se reciban alertas para toda la red provenientes de ningún usuario, a no ser que dicho usuario pertenezca a una lista de personas autorizadas para enviar alertas. Esta lista sería manejada por la aplicación y es configurable por el usuario, quién puede agregar o eliminar a otros usuarios a su lista. Para el caso de la colaboración no prevista es más apropiada la alternativa (1), ya que en la (2) se requiere saber a priori con quien se interactuará. En cualquier caso ambos filtros pueden coexistir, dejando en manos del usuario la configuración de que filtros desea usar en determinados momentos.

3.1.3. Mecanismos de Colaboración

Además de los mecanismos de alerta descritos, la solución provee una serie de protocolos que apuntan a dar a conocer los distintos recursos compartidos por los usuarios dentro de cada sesión. Un recurso compartido puede ser cualquier objeto con el cual uno o más usuarios puedan interactuar, esto es, archivos de datos, documentos, aplicaciones, servicios, entre otros. Se define un recurso compartido, como aquel recurso que, estando físicamente presente en el dispositivo de un usuario, es compartido dentro de una sesión volviéndose accesible a los demás usuarios de la sesión y convirtiéndose así en un recurso de la sesión.

Finalmente, teniendo manejo de usuarios, sesiones y manejo de información del contenido de estas sesiones, el último paso es el de la colaboración misma. Si bien, el hecho de colaborar en sí es algo que corresponde a los usuarios, la solución provee una serie de servicios orientados a lograr el éxito de esta colaboración en un ambiente construido sobre una red MANET. Estos servicios son los siguientes:

- Asignación de roles a usuarios dentro de cada sesión.
- Transferencia de documentos y aplicaciones.
- Sincronización de documentos XML.
- Llamadas a funciones remotas gracias a el micro-servidor web embebido en la solución.
- Persistencia y encolamiento de solicitudes hacia servicios web, que permite realizar llamadas “offline” que subsanan la inestabilidad de la red.

Con todo lo anterior se conforma una solución cuya infraestructura posee la capacidad de actuar como mediador entre workspaces de igual o distinto tipo. El uso de la infraestructura no es del todo inmediato y requiere que cada workspace integre la solución, o al menos, se comuniquen a través de ella con los demás workspaces presentes en la red.

Para este trabajo de tesis se construyó una aplicación que provee un workspace de trabajo dentro del cual es posible ejecutar distintas aplicaciones colaborativas. Dicho workspace utiliza la infraestructura creada para colaboración no prevista, de manera que, permite mostrar y usar la solución propuesta.

3.2. Arquitectura de la Solución

La solución desarrollada se compone de una serie de módulos interconectados entre sí. Cada uno de estos módulos cumple una función específica entre las que podemos

mencionar la detección de usuarios, la exposición y consumo de servicios web y la sincronización de recursos. La interconexión de los distintos módulos es posible gracias a la definición de APIs para cada módulo, permitiendo que eventualmente existan diferentes implementaciones para un mismo componente. Las componentes que coordinan el sistema (control de usuarios, sesiones, recursos compartidos) han sido implementadas siguiendo el patrón de diseño Singleton para proveer un acceso global a la información contenida y mantener la consistencia del sistema al existir una sola instancia para cada grupo de objetos. También se usaron los patrones de diseño Adapter y Facade para la construcción de las APIs de cada capa y algunos componentes.

La arquitectura consta de tres capas: base (capa inferior), de coordinación (capa intermedia) y de aplicación (capa superior). La capa inferior provee servicios de red y persistencia de datos. La capa intermedia es el núcleo de la aplicación, y es quien controla el comportamiento de los distintos módulos. Finalmente, en la capa superior se sitúan las interfaces gráficas de usuario y/o las distintas aplicaciones que corren dentro del workspace.

La *capa base* de la solución cuenta con un módulo capaz de enviar y recibir mensajes UDP multicast, los cuales conforman mensajería destinada a enviar y recibir información del medio. Esta capa cuenta también con un micro-servidor Web destinado al intercambio de información entre aplicaciones, una vez que el medio es conocido. Esto ocurre, por ejemplo, entre dos aplicaciones colaborativas de dos usuarios que se reúnen físicamente, puesto que cada uno ha detectado la presencia del otro en la cercanía del lugar donde se encuentran. Finalmente, se incluye un módulo de persistencia de datos que utiliza una base de datos XML para leer y escribir información sobre el estado, sesiones y documentos compartidos del usuario.

En la segunda capa, que denominamos *capa de coordinación*, encontramos toda la infraestructura que permite a la solución soportar la colaboración móvil en ambientes ad-hoc. Esta capa contiene el sistema de detección de usuarios (peers) cercanos, el sistema de administración de sesiones (junto con sus recursos compartidos), los módulos de descarga y

sincronización de documentos, y los componentes que proveen acceso a los mecanismos de persistencia de datos y comunicación con otros dispositivos. En el núcleo de la solución, se encuentra la componente denominada “CoordinationManager” (Coordinador de Componentes, en la figura 7), cuya función es coordinar el comportamiento del resto del sistema. Es aquí donde se manejan los diferentes estados de la aplicación, y las reglas mediante las que se rige el resto de la solución. Directamente ligado al módulo de coordinación, se encuentra el módulo de detección de usuarios y de manejo de sesiones. En torno a estos componentes se encuentran los demás módulos que proveen servicios a los usuarios y sesiones.

Finalmente, capa superior contiene la interfaz gráfica de usuario, el sistema de notificaciones y las distintas aplicaciones compartidas. Estos tres elementos conforman, desde el punto de vista del usuario del dispositivo, su workspace colaborativo móvil. Si bien, una aplicación compartida es una aplicación que en sí puede tener su propia arquitectura de capas o su propia interfaz gráfica, situamos a estas aplicaciones en la capa superior dado que de todas formas se encuentran en un nivel superior a los demás componentes. Esto es fundamentalmente porque utilizan información y funcionalidad provista por las capas inferiores de la arquitectura.

En la figura 7 se muestra un esquema simplificado de los componentes de la solución. Tal como se indicó en los párrafos anteriores, la figura muestra las tres capas de componentes de la solución. En los siguientes capítulos se detallan las características de los componentes más importantes de la solución propuesta.

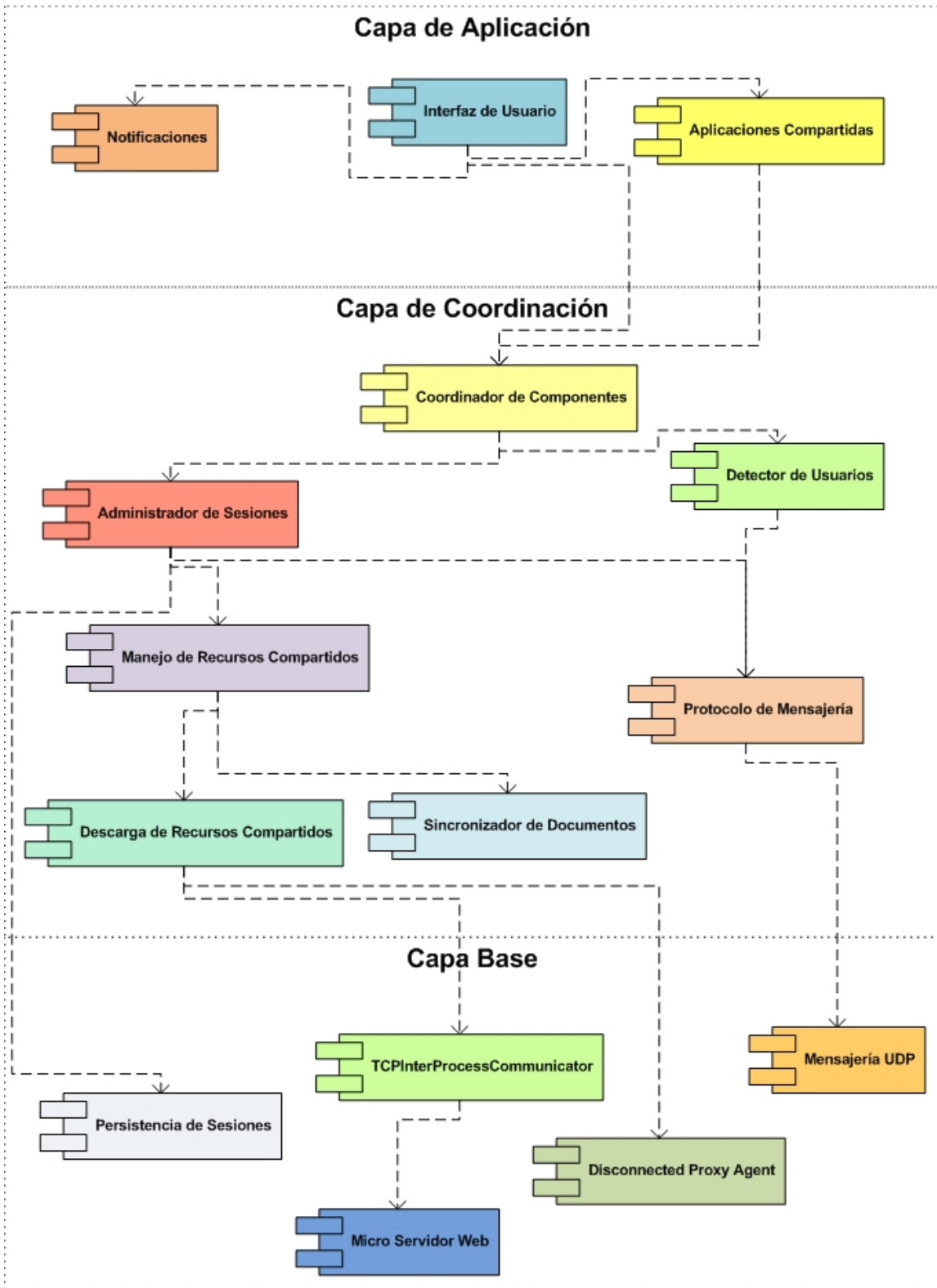


Figura 7 Componentes de la Solución

3.3. Detección de Peers y Manejo de Usuarios

Un punto crucial en la solución implementada, es el mecanismo de detección de peers. Cada peer, correspondiente a un usuario en la red MANET, desconoce la topología de la red en que se encuentra. A raíz de esto, el protocolo de comunicación debe ser capaz de enviar y recibir mensajes en un medio desconocido. Es por esto que se optó por el protocolo UDP, el cual no requiere saber quién será el receptor del mensaje, ni si este llegó con éxito o no. Cada nodo de la red está en condiciones de enviar señales al medio, pero no es posible saber si estas señales llegaron a algún destino. En este escenario, usar un protocolo que garantice que los mensajes son recibidos (como por ejemplo TCP) no tiene sentido, ya que ni siquiera se sabe quién se espera que reciba los mensajes. Aunque eventualmente se supiera quién es el destinatario, por ejemplo conociendo su dirección IP, puede ser que éste no esté temporalmente en condiciones de recibir mensajes por una u otra razón. En ese caso, el emisor se quedaría esperando las respuestas a los mensajes, aunque éstos no serán procesados por nadie.

El sistema implementado para la detección de peers funciona en un solo sentido; vale decir, los nodos envían señales al medio pero no reciben nunca una respuesta al mensaje que enviaron. En el otro lado, cada nodo que recibe mensajes, los procesa para determinar qué nodos están a su alcance. A partir de la información recopilada, sólo es posible conocer los nodos presentes en un determinado instante, ya que resulta imposible determinar los nodos que no están al alcance. La aplicación no almacena información de ningún nodo, salvo del que la está ejecutando. Entre las razones que hay para esto, podemos mencionar la capacidad de almacenamiento, y procesamiento requerida para recolectar y administrar esta información. Debido a que estamos pensando en dispositivos móviles con escasos recursos, cualquier ineficiencia podría causar problemas importantes, sin embargo, la principal razón es que al no tener contacto permanente con un nodo, la información almacenada no es certera y sólo correspondería a una “fotografía” del instante en el que el nodo fue detectado. Esto significa que a futuro una nueva detección del mismo nodo, puede ser entendida como un nodo distinto al contener información diferente.

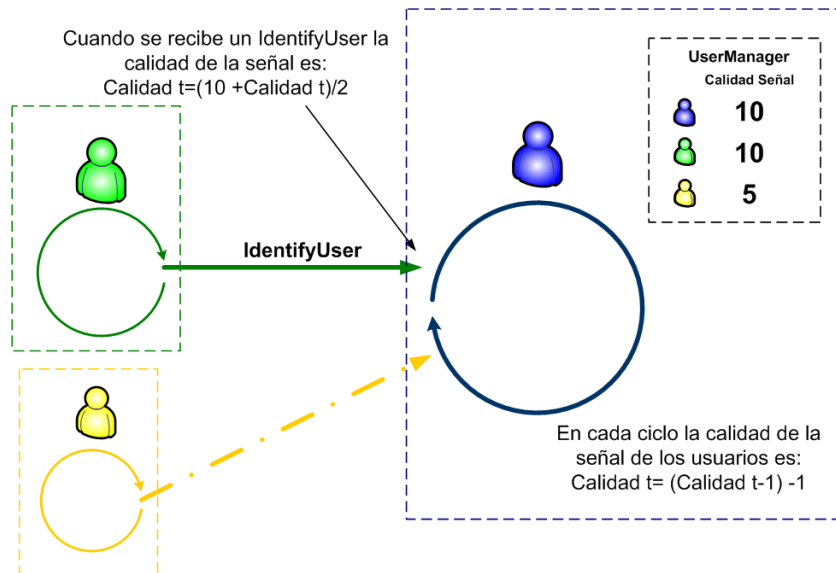


Figura 8 Mecanismo de detección de Peers

La figura 8 muestra el mecanismo de detección de peers que se implementó en el marco de esta tesis. Este mecanismo se basa en ciclos de duración constante, en los que se envía (una vez por ciclo) un mensaje de identificación (*IdentifyUser*) al medio. Por otro lado, cada *peer* está permanentemente escuchando cualquier mensaje que llegue por el canal multicast. Con esto, cada vez que un nodo recibe un mensaje de identificación, lo procesa, detectando así la presencia de los demás *peers* en su radio de alcance.

Un mensaje de identificación contiene la información mínima para reconocer a un usuario, esto es: un identificador único, un nombre y un número de versión que permite saber cuando han ocurrido cambios en el resto de la información del usuario (sesiones, recursos, etc.). El siguiente XML es el generado por el protocolo de la solución, y corresponde al mensaje que viaja a través de la MANET por un peer determinado:

```
<?xml version="1.0" encoding="utf-8"?>
<PASSIR>
  <Type>IdentifyUser</Type>
  <User>
    <Id>6ef9a09f-c13a-4f2e-9209-fa2c3df7ffe0</Id>
    <Name>Francisco</Name>
    <Version>4</Version>
  </User>
</PASSIR>
```

```
</User>  
</PASSIR>
```

El procesamiento de los mensajes de identificación recibidos, involucra las siguientes actividades:

- Si el Id del usuario no se encuentra en la lista de usuarios, se agrega el nuevo usuario a la lista y se le asigna la calidad de señal máxima (en la figura 8, la calidad máxima corresponde a 10).
- Por el contrario, si el Id del usuario se encuentra en la lista de Usuarios, entonces se le asigna una calidad de señal de la siguiente forma:

$$\text{calidad de señal} = (\text{calidad de señal actual} + \text{calidad de señal máxima})/2.$$

- Cada vez que se completa un ciclo de identificación, se le asigna a todos los nodos una calidad de señal de esta forma:

$$\text{calidad de señal} = \text{calidad de señal actual} - 1$$

Vale decir en cada ciclo la calidad de la señal se disminuye en 1. Bajo condiciones óptimas debiera llegar, por cada nodo, un mensaje por ciclo. Esto mantendría la calidad de la señal en su valor máximo.

- Si la calidad de la señal llega a 0, el usuario se da por perdido y se elimina de la lista de usuarios.

Para mantener la información de los distintos usuarios detectados en la MANET, se trabaja con una lista de usuarios contenida dentro de una estructura que denominamos *UserManager*, y dentro de la cual se mantiene, en la memoria del dispositivo, la información del usuario local y de los demás usuarios detectados. El procedimiento descrito

permite dar persistencia a los usuarios, a pesar de las eventuales pérdidas de mensajes debido a las interferencias u otras razones. Además, es también posible determinar cuándo un usuario ya no es alcanzable por diversas causas, ya sea porque se alejó demasiado, se desconectó involuntariamente, o apagó su dispositivo. También este mecanismo provee información valiosa respecto a la calidad de la comunicación con los diferentes nodos de la red MANET, brindando la posibilidad de que la información de calidad de la señal sea accedida desde la interfaz de usuario.

Un punto que se debe mencionar, es que las salidas voluntarias de usuarios son gestionadas en forma independiente de la detección de peers. Un usuario que sale del sistema envía voluntariamente un mensaje denominado *ExitDomain* al canal multicast. Todo nodo que recibe este mensaje elimina al usuario inmediatamente de la lista de usuarios conectados. El mensaje de salida es enviado una única vez, de manera que quien no lo recibió, seguirá teniendo al usuario en su lista hasta que, al cabo de algunos ciclos, lo dé por perdido.

3.4. Administración de Sesiones

Por sesión (o sesión de trabajo) entendemos una estructura que provee herramientas necesarias para administrar un grupo de usuarios, de acuerdo a sus necesidades y, además provee herramientas para facilitar la colaboración entre estos usuarios.

Es posible pensar en diferentes tipos de sesiones y en diferentes características para clasificarlas. En particular, en este trabajo se miran los distintos tipos de sesiones desde el punto de vista de la privacidad o seguridad que ellas proveen. Se debe notar que las sesiones más simples, también resultan más “livianas”, al requerir menos procesamiento y menos cantidad de información a comunicar. De esta forma se establece una gama de opciones para el usuario a la hora de crear una sesión. Dichas opciones van desde lo más simple y carente de seguridad, como lo es una sesión abierta, hasta lo más complejo, que en este caso, es una sesión privada con roles y mecanismos de encriptación. Antes, como

primera clasificación de las sesiones en la solución creada tenemos dos grupos básicos de sesiones: ad-hoc y suscribible.

3.4.1. Sesión Ad-hoc

La sesión Ad-hoc (*AdhocSession*) es el tipo de sesión más sencillo. Todos los usuarios conectados a la red MANET pertenecen a una sesión ad-hoc siempre. La sesión provee las herramientas de mensajería que permiten a los usuarios comunicarse entre ellos, una vez que están dentro de la MANET. Existe una única instancia de sesión ad-hoc en la aplicación, y además, ésta posee un identificador fijo igual en todos los usuarios, independientemente de si éstos han estado en contacto o no. En otras palabras, un usuario conectado a una red pertenece siempre a la sesión ad-hoc y esta actúa como medio básico de comunicación para dicha red.

3.4.2. Sesión Suscribible

La sesión suscribible (*SubscribableSession*) es la definición abstracta para todas las sesiones que requieren suscripción. Por suscripción entendemos el proceso mediante el cual un usuario solicita y obtiene el permiso para participar en una sesión de trabajo.

Un usuario suscrito a una sesión podrá, en principio, conectarse a dicha sesión, saber qué usuarios de la MANET se encuentran también conectados a la sesión, compartir recursos y enviar y recibir mensajes de *chat* a la sesión. A continuación se explican brevemente las variantes de sesiones suscribibles que forman parte de esta propuesta.

- **Sesión Abierta** (*PublicSession*). Consiste en una sesión a la que cualquier usuario de la MANET puede suscribirse, sin requerir del permiso de otro usuario, sólo se requiere solicitar el ingreso a la misma. Esta es la versión más sencilla de una sesión suscribible.

- **Sesión Cerrada Básica** (*PrivateSession*). Para formar parte de este tipo de sesión, se requiere contar con una invitación para suscribirse. En estas sesiones cualquier usuario que se encuentre dentro, está facultado para invitar a otros usuarios a unirse a la sesión. Dado que no existe una jerarquía interna, no existen tampoco mecanismos de expulsión de usuarios, ni ningún otro mecanismo que suponga mayores atribuciones de un usuario sobre el resto.
- **Sesión Cerrada con Múltiples Roles** (*RoleSession*). Una sesión cerrada con roles permite la existencia de distintos niveles de jerarquía para los usuarios. La sesión cuenta con diferentes roles, cada uno de ellos con permisos asignados por el administrador de la sesión. Los permisos para cada rol definidos en esta implementación son: invitación y expulsión de usuarios, lectura, escritura y ejecución de recursos compartidos y silenciamiento de usuarios en el chat de la sesión.

3.5. El Micro-servidor Web y el Disconnected Proxy Agent

La solución implementada cuenta con un micro-servidor web [Carrasco, 2007] cuya función es recibir y enviar respuestas a peticiones remotas, las cuales son hechas desde aplicaciones que se ejecutan en otros dispositivos. Esta solución también implementa el protocolo WS-Attachments, que permite enviar documentos adjuntos en la mensajería del servicio. En forma paralela se introdujo el disconnected proxy agent, cuya función es la de proveer una interfaz para que las aplicaciones realicen llamadas al micro-servidor web en forma transparente a pesar de que eventualmente se puedan extraviar mensajes TCP entre el proxy y el servidor web.

La razón para incorporar un servidor Web a la solución, es para permitir que las aplicaciones que se ejecuten dentro del workspace tengan la capacidad de interactuar con otras aplicaciones en forma remota, invocando funcionalidades en otros dispositivos. Por ejemplo, funcionalidades que proveen o extraen información, o bien solicitan y proveen

documentos a través de un servicio. Se optó por esta solución ya que es un estándar para comunicaciones, lo que permite a futuros desarrolladores crear aplicaciones con un sistema de interconexión conocido. Además, facilita la portabilidad de aplicaciones ya existentes, hacia el framework desarrollado.

El gran inconveniente de integrar servicios web a una solución de groupware móvil es que estos servicios requieren del protocolo TCP/IP para funcionar. Vale decir, la comunicación es entre dos participantes, y entre ambos existen mecanismos para saber cuándo un mensaje fue enviado y recibido con éxito. Como se explicó antes, el uso de este protocolo resulta inadecuado si el emisor del mensaje desconoce el medio, y no sabe quién es su contraparte. Para subsanar esto se utiliza el subsistema de detección de peers cercanos. Gracias a este subsistema, la información sobre el receptor del mensaje es conocida, dado que cada usuario detectado posee un identificador y una dirección IP que hacen posible la comunicación TCP/IP. Sin embargo, si bien es cierto que al conocer la dirección IP es posible hacer invocaciones a un servicio web, aún tenemos el problema de las posibles desconexiones de peers, producto de la inestabilidad de la red. En este punto es donde entra a participar el disconnected proxy agent, como interfaz para las llamadas al servicio web que corresponda.

El módulo denominado disconnected proxy agent es básicamente un proxy para la llamada de servicios web remotos, que además consta de un sistema de encolamiento y reencolamiento de peticiones. Este módulo es capaz de reconocer a un usuario por su identificador, aunque su dirección IP haya cambiado. El sistema recibe los llamados a servicios web por parte de las aplicaciones, y los pone en una cola de solicitudes pendientes. Dichas solicitudes son despachadas cuando se detecta la presencia del destinatario en la red. Si se pierde la conexión con el destinatario y la llamada falla, ésta es reencolada en el sistema. Luego, la solicitud se despacha nuevamente cuando se vuelva a detectar la presencia del usuario destinatario en la red. Si el usuario emisor abandona la sesión o se desconecta de la misma, la solicitud será eliminada de la cola.

Los componentes antes descritos proveen una solución estandarizada para la invocación de servicios remotos, y que además, subsana los problemas de conexión que son comunes y recurrentes de una MANET.

3.6. Comunicación entre Procesos (*TCPInterProcessCommunicator*)

Inicialmente el micro-servidor web y el resto de PASIR (detección de usuarios y sesiones) eran aplicaciones independientes, y es importante mantenerlas como tal, principalmente porque esto deja al servidor web como una unidad estándar e independiente que puede ser reutilizada en otras aplicaciones. Por esta razón se creó un nuevo componente, denominado *TCPInterProcessCommunicator*, que permite la comunicación interprocesos vía un canal TCP entablado entre PASIR y el micro-servidor web.

La componente consiste básicamente de un listener (*IpTcpListener*) que recibe mensajería TCP proveniente de la máquina local en un puerto dado, y de uno o varios clientes (*IpTcpClient*) que se ejecutan dentro del servidor web, los cuales envían mensajería con información al listener.

En la capa de coordinación de PASIR se desarrolló la componente denominada *WebServiceBridge*, que se encarga de levantar el *IpTcpListener* y luego procesar y redirigir la mensajería entrante hacia la aplicación correspondiente. Por otro lado, cada aplicación del workspace que utilice servicios web debiera tener un *IpTcpClient*, el cual sería el encargado de inyectar la información en el socket para que sea leída por el listener en PASIR.

3.7. Sincronización de Documentos (*XMLSync*)

Otro aspecto relevante para el trabajo colaborativo, independiente del tipo de red en que se desarrolle, es la sincronización de documentos. Los usuarios pueden trabajar sobre el

mismo documento y generar nuevas versiones de estos debido a que se les introducen modificaciones. Por esta razón, deben existir mecanismos que permitan, a partir de los documentos modificados por cada usuario, generar un único documento unificado que contenga los cambios hechos por todos los usuarios. Esto se complica cuando entre las diferentes versiones existen conflictos o modificaciones que se superponen, en este caso deben existir reglas que permitan decidir qué actualizaciones serán las que finalmente queden en el documento unificado.

La solución incorpora un módulo para sincronización de documentos XML, el cual está basado en XMLSync [Moya, 2007]. Este módulo provee la capacidad de sincronización de documentos XML, mediante algoritmos de generación de árboles de diferencia, reconciliación y resolución de conflictos.

El sistema se basa en que cada documento XML puede ser visto como un árbol, en donde es posible asignar un identificador único a cada nodo o rama del árbol. Por ejemplo, la raíz tendrá como identificador un “0”, luego el nodo a la izquierda será “0.0” mientras que el de la derecha del anterior será “0.1” y así sucesivamente. A partir de un documento inicial los usuarios generan cambios, los cuales se ven reflejados en el árbol como nodos añadidos o modificados. Luego, para sincronizar estos documentos, se genera el árbol de diferencias, que consiste en la unión de ambos árboles, integrando los nodos añadidos por cada usuario y marcando los nodos que presenten conflictos (es decir, mismo identificador pero distinto valor). Estos últimos nodos se someten a un algoritmo de reconciliación basado en resolutores que cumplen la función de determinar qué modificación será conservada en el documento final. Un resolutor es básicamente un conjunto de reglas usadas para comparar valores; por ejemplo, fecha de modificación, menor que, mayor que. Además cada resolutor consta de un criterio a aplicar por defecto en caso de conflicto, como por ejemplo “elegir el mayor valor”.

<pre> <inventario _id="1"> ... <descripcion _id="1.34" _resolutor="timeStamp" _syncTimes="1" _timeStamp="341" _role="1" _visible="true" valor="coca cola" cantidad="215"> ... </producto> </inventario> </pre> <p style="text-align: center;">documento A</p>	<pre> <inventario _id="1"> ... <descripcion _id="1.34" _resolutor="timeStamp" _syncTimes="5" _timeStamp="256" _role="3" _visible="true" valor="coca cola" cantidad="4571"> ... </descripcion> </inventario> </pre> <p style="text-align: center;">documento B</p>
--	--

Figura 9 Atributos que generan diferencias en sincronización

La figura 9 muestra una situación en que dos nodos, con el mismo identificador, tienen valores distintos para el atributo “cantidad”. En este caso, al reconciliar los documentos se verá que el resolutor asignado es “timeStamp” y por lo tanto se escogerá como valor final para cada nodo del documento sincronizado, los valores del documento A por tener un mayor timeStamp. En el ejemplo hemos asumido que timeStamp se refiere a timeStamp mayor, dicho criterio se define en la implementación del resolutor para timeStamp.

3.8. Soporte para Notificaciones

Otro módulo de la aplicación que vale la pena destacar por su aporte a la experiencia del usuario final, es el de notificaciones. Este módulo es un sistema de alertas que dan al workspace la capacidad de proveer, al usuario, información sobre el medio (*awareness*) en forma no invasiva.

En un sistema pensado para la colaboración no prevista debe existir un método para informar al usuario, que otro usuario desea interactuar con él. Para lograr esto, el primer paso fue incluir notificaciones, utilizando los mecanismos provistos por .Net Compact

Framework. Estas notificaciones fueron implementadas a través de cuadros texto que muestran la información pertinente. Dadas las dimensiones de la pantalla de un dispositivo móvil, cualquier diálogo de texto inevitablemente interrumpirá el trabajo del usuario. Aunque la idea es llamar la atención del usuario, la porción de pantalla usada para esto podría sobrepasar un tercio del espacio total, dependiendo del dispositivo que se esté utilizando. Si además sumamos que una colaboración no prevista puede también ser una colaboración no deseada, tenemos rápidamente un problema para proveer información al usuario mientras trabaja (evitando interrupciones invasivas constantes).

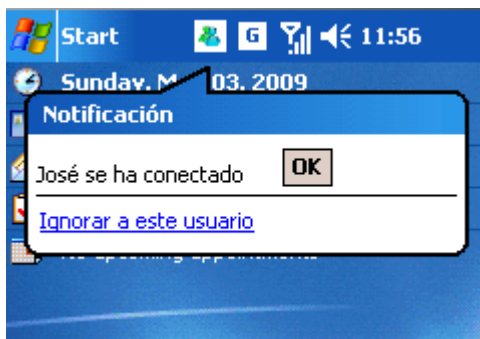


Figura 10 Notificación de conexión de un usuario a la red

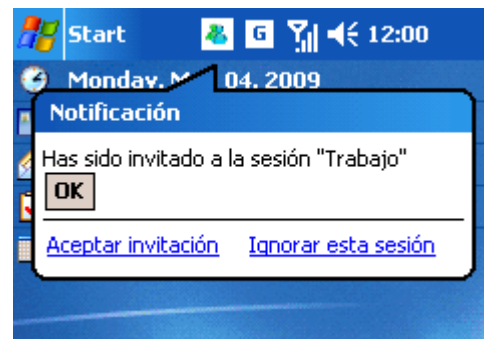


Figura 11 Notificación de invitación a una sesión recibida por el usuario local

La solución que se desarrolló incorpora un sistema de filtrado de eventos mediante el cual el usuario es capaz de configurar qué interacciones son deseadas y cuáles no. De esta forma, cuando el usuario recibe una solicitud de interacción no deseada tiene la opción de ignorar cualquier futura solicitud de ese usuario. De igual forma el usuario solo recibirá notificaciones de las sesiones de trabajo a las que pertenezca y se encuentre conectado.

El módulo, en el aspecto técnico, utiliza eventos lanzados desde la capa de coordinación a los que se suscribe la capa de aplicaciones, dichos eventos corresponden a solicitudes de usuarios, ingresos de usuarios a sesiones, nuevos recursos compartidos y otras.

4. El Framework Desarrollado

La solución creada es en sí un framework de apoyo al desarrollo de workspaces y aplicaciones colaborativas móviles, puesto que implementa una serie de mecanismos que permiten colaborar en ambientes inalámbricos (wireless) con dispositivos móviles. Ahora bien, hasta el momento no se ha profundizado en lo que es la colaboración no prevista, y la solución creada no muestra a primera vista cómo abordar el problema.

Como parte de este trabajo de tesis se diseñaron e implementaron una serie de componentes, que permiten poner en práctica la solución desarrollada. Si bien esta solución es completa por sí sola, no resulta posible visualizar sus capacidades sin la existencia de un workspace que la utilice, y de un conjunto de elementos que puedan ser compartidos por los usuarios. Para ello se desarrolló un software que permite a los usuarios, a través de una interfaz gráfica y haciendo uso de la infraestructura desarrollada, tener un entorno de trabajo sobre el cual crear sesiones de trabajo, invitar a otros usuarios a unirse a ellas, y ejecutar y compartir aplicaciones.

Cada aplicación se integra al workspace a través de un sistema de plug-ins, mediante el cual se maneja una lista de posibles aplicaciones asociadas a cada sesión. Estas aplicaciones pueden ser compartidas y ejecutadas desde el entorno del sistema principal, que corresponde a nuestro workspace. Estas aplicaciones, dado el eje de este trabajo, representan software creado para apoyar la colaboración entre usuarios, cada uno de los cuales está provisto de los mecanismos de colaboración particulares de cada aplicación, y utiliza las funcionalidades provistas por el framework. Además, cada aplicación dentro de una sesión es en sí un recurso compartido, y por tanto, es descargable y ejecutable por los demás usuarios miembros de dicha sesión. Lo anterior permite que un usuario que se conecta a una sesión y no cuenta con la aplicación requerida para trabajar en dicha sesión, pueda descargarla de algún otro usuario y ejecutarla dentro de su workspace. Con esto logramos que una misma aplicación pueda ser ejecutada simultáneamente en workspaces de distintos usuarios, e incluso en workspaces distintos, siempre que sean capaces de ejecutar

la aplicación. Así tenemos la misma aplicación colaborativa corriendo en workspaces de distintos usuarios, y permitiendo de esta forma el trabajo colaborativo no previsto entre usuarios móviles.

4.1. Estructura de una Aplicación Colaborativa

Cada aplicación colaborativa creada para la infraestructura de PASIR consta de dos partes principales, la aplicación en sí misma y los servicios web asociados a ella. Mientras que la aplicación se inserta como un módulo o plug-in dentro del espacio de trabajo del usuario, los servicios web se insertan como nuevas funcionalidades disponibles a través del módulo de servicios web del framework propuesto.

Para lograr esto, se creó una estructura que permite definir los distintos componentes de una aplicación, así como también la información relevante que la describe. Esta estructura se almacena en un archivo de tipo .psa (PASIR Shared Application) que representa, para todos los efectos, a la aplicación que va a ser utilizada por el usuario final. Dicha estructura tiene la siguiente forma:

```
<?xml version="1.0" encoding="utf-8"?>
<PasirSharedApplication>
  <Header>
    <Name>MobileChat</Name>
    <Description>Aplicación Chat de ejemplo</Description>
    <Author>Francisco Castro Vera</Author>
    <Version>0.1</Version>
  </Header>
  <Application>
    <ApplicationFile>ChatApplication.dll</ApplicationFile>
    <File>ChatProtocol.dll</File>
  </Application>
</PasirSharedApplication>
```



```
</Application>
<Services>
  <ServiceFile>MobileChatService.dll</ServiceFile>
</Services>
</PasirSharedApplication>
```

Los campos del encabezado corresponden a información meramente descriptiva de la aplicación, con el objeto de que el usuario final tenga una idea del origen de la aplicación y la funcionalidad que ésta provee. En este caso estamos hablando de la aplicación “MobileChat”, la cual usaremos como ejemplo para describir el proceso de compartir una aplicación. A continuación, bajo el tag “<Application>”, viene la lista de archivos que componen la aplicación. El primero denominado ApplicationFile, es el archivo a partir del cual se ejecuta la aplicación. En este caso, se trata de “ChatApplication.dll”. Este componente debe implementar la interfaz ISharedApplication que se detalla más adelante en la sección 4.2. El resto de los componentes de la aplicación se listan con el tag “<File>”. Finalmente, una última sección del archivo incluye los servicios web que la aplicación utiliza.

Ahora bien, dada la estructura explicada anteriormente y siguiendo con el ejemplo de la aplicación MobileChat, cada usuario que posea dicha aplicación puede ejecutarla dentro de una sesión dada. El mecanismo para hacerlo consiste de dos pasos simples: (1) introducir o compartir la aplicación dentro de la sesión, y (2) ejecutarla mediante algún botón o menú que el workspace provea para ello. Al momento de ejecutarla, el framework verifica que todas las componentes especificadas en el archivo descriptor, estén en el lugar correspondiente. Luego, ejecuta aplicación descrita en “ApplicationFile”.

Con lo anterior tenemos una aplicación colaborativa compartida dentro de una sesión de trabajo. Cuando una aplicación es compartida, los demás usuarios dentro de la sesión son informados de la existencia de esta nueva aplicación y pueden descargarla. Para hacer efectiva la descarga, los usuarios envían una solicitud vía el Disconnected Proxy

Agent hacia el servicio web de transferencia de archivos, el cual es provisto por defecto por el framework, como un servicio del micro-servidor web. Una vez hecha la solicitud, el micro-servidor web envía primero el archivo .psa con la definición de la aplicación, que al ser recibido es procesado, gatillándose nuevas llamadas que solicitan cada uno de los archivos que componen la aplicación. Estos archivos son enviados como adjuntos siguiendo el protocolo WS-Attachments. Finalmente, el usuario que hizo la solicitud puede ejecutar la aplicación y, siguiendo con nuestro ejemplo, enviar así mensajes de chat al resto de la sesión.

En este punto se debe notar que el método de transferencia de aplicaciones es sencillo y que para garantizar la transferencia de la aplicación se usa el encolamiento provisto por el Disconnected Proxy Agent. Indudablemente, dadas las restricciones de las redes sobre las que se trabaja, y las capacidades de los dispositivos móviles, el tamaño total de una aplicación está inversamente relacionado al tiempo que un usuario tarda en poder descargar la aplicación, y al éxito que pueda tener el proceso total de transferencia. Es decir, a menor tamaño total de la aplicación, más ágil será todo el proceso y mayor será la probabilidad de éxito de la transferencia.

4.2. Implementación de una Aplicación Colaborativa

La implementación de una aplicación colaborativa es análoga a la implementación de cualquier aplicación hecha utilizando el .NET Compact Framework. Cualquier aplicación puede ser transformada en una aplicación colaborativa siguiendo los siguientes pasos: (1) implementar la interfaz *ISharedApplication*, y (2) implementar los servicios web asociados. A continuación se describen cada uno de estos pasos.

4.2.1. Implementación de la Interfaz *ISharedApplication*

La interfaz *ISharedApplication* descrita más abajo, es la que permite al framework colaborativo interactuar con aplicaciones. La interfaz posee, en primer lugar, los mismos

campos descriptores mencionados anteriormente: *Name*, *Description*, *Author*, *Version*. A continuación aparecen dos métodos: *Initialize()* y *Dispose()*. El primer método es invocado desde el framework, y su objetivo es inicializar la aplicación. Una implementación de este método correspondería a iniciar el primer formulario de una aplicación basada en Windows Forms. En el caso de nuestra aplicación MobileChat, se inicializa y muestra el formulario de envío de mensajes. El segundo método, llamado *Dispose()*, finaliza la ejecución de la aplicación.

```
public interface IApplication
{
    string Name { get; }
    string Description { get; }
    string Author { get; }
    string Version { get; }

    void Initialize();
    void Dispose();
    bool IsRunning { get; }

    void ProcessIncomingMessage(byte[] message);
    string SessionId { get; set; }
    string ApplicationId { get; set; }
}
```

Luego está la propiedad *isRunning*, que sirve como información que puede ser opcionalmente leída por el workspace para indicarle al usuario que una cierta aplicación está corriendo. Esto es muy útil en el caso que haya múltiples aplicaciones ejecutando a la vez, y algunas de ellas se encuentren minimizadas. Se dice que la lectura de este campo es opcional, puesto que su utilidad dependerá de cómo se construya el workspace colaborativo móvil, y si éste soportará múltiples aplicaciones en una sesión o cada sesión estará asociada a una sola aplicación. Un ejemplo de esto sería un workspace orientado a crear diferentes sesiones de chat entre usuarios de una red.

Finalmente se debe implementar el método *ProcessIncomingMessage(byte[] message)*, que es el encargado de recibir los mensajes provenientes del servicio Web, y encausarlos dentro de la aplicación para enviarlos a donde corresponde. En este punto se debe recordar que tanto el micro-servidor Web como el resto del framework, ejecutan en procesos diferentes. Sin embargo, estos procesos se encuentran interconectados a través de la componente denominada *TCPInterProcessCommunicator*. La principal función de esta componente es la de enviar las solicitudes hechas al servidor Web, hacia una componente encargada de redireccionar cada solicitud. Esta última redirecciona los pedidos hacia la sesión y aplicación correspondientes. Los campos *SessionId* y *ApplicationId*, indican la sesión en la que corre la aplicación y el identificador de la aplicación dentro de dicha sesión.

4.2.2. Implementación de Servicios Web Asociados

En la infraestructura propuesta, el servidor web es un módulo separado que además corre en forma independiente al resto de la solución. La implementación del sistema permite que el servidor web y la aplicación de groupware se comuniquen vía TCP. Dada esta característica, para cada aplicación que utilice servicios web, éstos deben ser desarrollados (o reutilizados) considerando que las solicitudes deben ser redirigidas hacia el módulo de coordinación. A su vez, el módulo de coordinación, redirigirá la invocación hacia la respectiva aplicación.

Hacer el proceso inverso, vale decir, enviar una respuesta desde la aplicación pasando por el módulo de coordinación es complejo e implica tener al cliente del servicio web a la espera de la resolución de su petición, lo cual puede tomar un tiempo considerable dada la poca capacidad de procesamiento de los dispositivos y la posibilidad de múltiples requerimientos simultáneos. Por esta razón, se optó por dar preferencia a utilizar métodos en que una llamada sólo pone la solicitud en el servicio web; luego la aplicación correspondiente enviará, si corresponde, la respuesta vía una nueva llamada al usuario que

hizo la solicitud. Con esto logramos mantener la lógica de la aplicación encapsulada en un solo proceso y no sobrecargamos la demanda del servicio web.

Lo anterior no excluye la posibilidad de que existan servicios web que realizan operaciones por sí solos, y/o no tienen que ver directamente con alguna aplicación. Un ejemplo de esto es el servicio web usado para descarga de aplicaciones.

Dicho todo lo anterior, durante el desarrollo de una aplicación compartida se deben desarrollar los servicios web asociados, considerando que su comunicación con la aplicación debe seguir un protocolo determinado. Este protocolo permite posteriormente manejar los requerimientos y enviarlos a cada aplicación. En forma análoga, se deben crear los proxy necesarios para la invocación de los servicios de la aplicación que pueden ser o no invocados a través del *Disconnected Proxy Agent*, explicado en la sección 3.5. El uso de esta componente queda a criterio del desarrollador de cada aplicación, quien debe sopesar qué tan crítico es que el mensaje sea enviado, versus la cantidad de recursos consumidos. El principal recurso a considerar es memoria, la cual se va a estar utilizando permanentemente al encolar los requerimientos.

En el ejemplo de la aplicación *MobileChat*, existe un servicio denominado *MobileChatService* que contiene el siguiente método:

```
[WebMethod(MessageName = "PutMessage")]
public void PutMessage(string sessionId, string applicationId, string
userId, string message)
{
    ChatMessage chatMessage = new ChatMessage();
    IpcTcpClient client = new IpcTcpClient(6968); //Comunicación entre
procesos
    client.SendMessage(chatMessage.Make(sessionId, applicationId,
userId, message));
}
```

Cada vez que un usuario en una aplicación remota desea enviar un mensaje de chat al servicio web, éste lo hará de la siguiente manera desde la aplicación:

```
proxy.PutMessage(application.SessionId, application.ApplicationId ,  
LocalUser.Instance().Id, textBoxMensaje.Text);
```

Donde, los identificadores `application.SessionId` y `application.ApplicationId` son los encargados de identificar la sesión y la aplicación (que corre dentro de la sesión) respectivamente. `LocalUser.Instance().Id` es el identificador del usuario que está enviando el mensaje, y finalmente `textBoxMensaje.Text` es el mensaje de chat capturado desde la interfaz de usuario. En este caso, no se usa encolamiento de mensajes, ya que la pérdida de estos no resulta crítica. Por el contrario, encolar mensajes de chat puede producir que éstos no lleguen en el orden deseado y que la conversación pierda sentido una vez que se comiencen a perder mensajes.

4.3. Pruebas Realizadas

En esta sección se describen las diferentes pruebas realizadas sobre la aplicación desarrollada. Durante el desarrollo del software, cada componente del sistema fue probada individualmente. En particular las macro-componentes, vale decir, servicios web [Carrasco, 2007], sincronización de archivos [Moya, 2007], detección de usuarios y manejo de sesiones [Castro, 2007] fueron probadas exhaustivamente por sus respectivos autores. Por lo anterior, las pruebas descritas en esta sección apuntan a evaluar el funcionamiento del sistema desarrollado desde la perspectiva de un usuario final, frente a la solución como conjunto.

Las pruebas realizadas se orientan a evaluar el comportamiento del mecanismo de detección de usuarios y sesiones en situaciones lo más reales posibles, y sobre todo, a

probar los mecanismos de descarga y comunicación entre aplicaciones de diferentes workspaces. Para el proceso de pruebas se utilizaron dos PDA con las siguientes características:

- Modelo iPAQ rx4540 (de HP)
- Procesador de 400 MHz, Samsung SC32442
- Memoria RAM de 64 MB
- Sistema Operativo Windows Mobile 5.0
- WiFi (WLAN 802.11b/g)
- Batería de Litio-Ion de 1200 mAh que entrega a la PDA una autonomía de 1,5 días (con la red inalámbrica activada, esta duración se reduce a entre 2 a 3 horas máximo)

Además se usaron dos notebooks equipados con tarjeta de red inalámbrica WLAN 802.11b/g, sistemas operativos Windows XP y Windows Vista, además de otras características no relevantes para las pruebas realizadas (dado que exceden con creces las necesidades de la solución). A continuación se describen cada una de los cuatro tipos de pruebas que se llevaron a cabo.

4.3.1. Detección de Usuarios, Sesiones y Recursos Compartidos

Para realizar las pruebas de los distintos mecanismos de detección, se configuraron los cuatro dispositivos de manera que todos se conectaran a una misma red adhoc inalámbrica. Una vez hecho esto, se procedió a correr la solución desarrollada y a evaluar el proceso de detección, conforme se agregaban nuevos dispositivos y se desplazaban los usuarios dentro de un ambiente semi-cerrado. La figura 12 muestra la interfaz de usuario de las ventanas de usuarios y sesiones respectivamente.

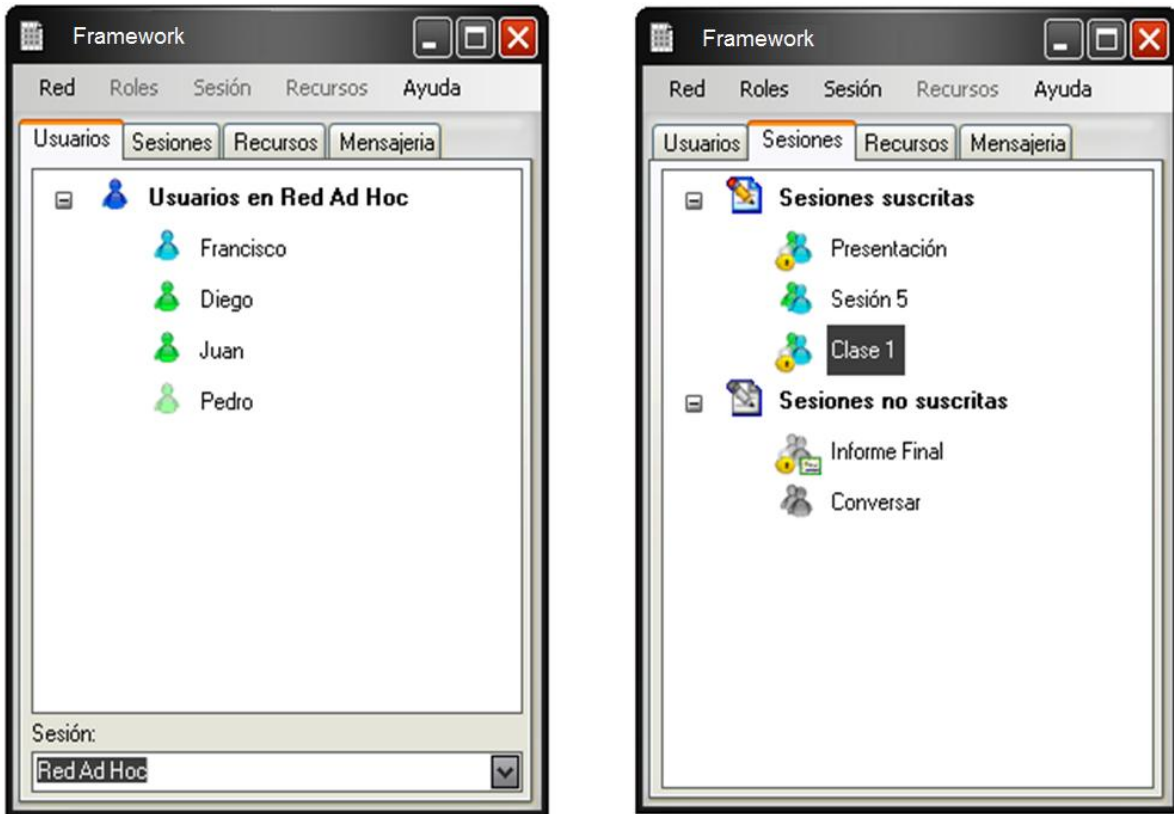


Figura 12 Usuarios y sesiones detectadas por la herramienta

Los resultados obtenidos mostraron un nivel de detección bueno, ya que en todas las situaciones donde los dispositivos se encontraron dentro del rango de comunicación, éstos fueron detectados satisfactoriamente en un tiempo inferior a los 5 segundos. Sin embargo, se pudo observar que en ciertas ocasiones un usuario era detectado débilmente durante algunos segundos (entre 1 y 3), no llegando esto a ser suficiente para que el sistema dé al usuario por perdido cuando aún estaba dentro del rango de comunicación. Esto puede explicarse debido a la alta tasa de pérdida de paquetes que es típica en las redes adhoc. Debido a eso, es posible que algunos mensajes de identificación se pierdan, no pudiendo alcanzar el dispositivo destino. Sin embargo, gracias al algoritmo usado para detección y mantención de listas de usuarios, cada usuario que no se detecta permanece siendo visto como conectado durante un tiempo tal que no desaparece, a no ser que efectivamente ya no esté dentro de la red.

En lo que respecta a sesiones y recursos compartidos, se vio que una sesión era detectada cerca de dos segundos después de que se detectara al usuario que la poseía. Lo mismo se registró para los recursos compartidos. Éstos mostraron un retraso de entre 4 a 5 segundos en ser desplegados en los demás dispositivos (remotos), una vez que eran compartidos por su dueño. Esta observación es explicable dado que el algoritmo de detección revisa primero si hay nuevos usuarios y si existen cambios en estos. En caso de haber cambios, solicita las sesiones, las que llegarán durante el siguiente ciclo de detección. Finalmente, si se detectan cambios en las sesiones (en la versión de la sesión), se solicitarán los recursos compartidos. Estos recursos llegarán en el siguiente ciclo, dos ciclos después del primero. Si consideramos que cada ciclo de detección dura un segundo aproximadamente, los resultados cuadran con el algoritmo, pero no cuadran en la cantidad de tiempo esperada (cerca de 7 segundos desde que se detecta al usuario, hasta que se visualizan sus recursos). Esta diferencia de tiempo, es atribuible a la capacidad de procesamiento de las PDA, ya que al realizar las mismas pruebas pero sólo con los dos notebooks, la velocidad de detección es mayor y dentro del rango de tiempo esperado.

Un resultado importante, puesto que debe ser mejorado, fue que la velocidad de detección de recursos compartidos se veía afectada por la cantidad de recursos presentes en una sesión para el mismo dispositivo. Esto provocó demoras, tanto del lado del usuario que compartía los recursos, como de los demás usuarios pertenecientes a la sesión. El problema se presenta con listas grandes de recursos compartidos (más de 100) y se puede explicar por el tamaño del mensaje generado, y porque las PDA no son capaces de procesar la lista completa de recursos antes de que termine un ciclo de detección. En general, el comportamiento de los dispositivos con listas grandes de recursos no sólo era lento, sino que también se volvía errático, mostrando resultados diferentes en diferentes reproducciones de la misma prueba.

La solución para esto pasa, a priori, por el envío en varios paquetes que puedan ser recibidos por los demás dispositivos. Esto debiese ser analizado en la búsqueda de la mejor solución. De todas formas, podemos suponer que 100 recursos compartidos dentro de una sesión de trabajo, en general es bastante.

4.3.2. Traspaso de Archivos y Aplicaciones

Los tiempos de respuesta para el traspaso de aplicaciones y archivos, fueron en general satisfactorios. Éstos no se ven afectados por la infraestructura montada sobre el servicio web de transferencia.

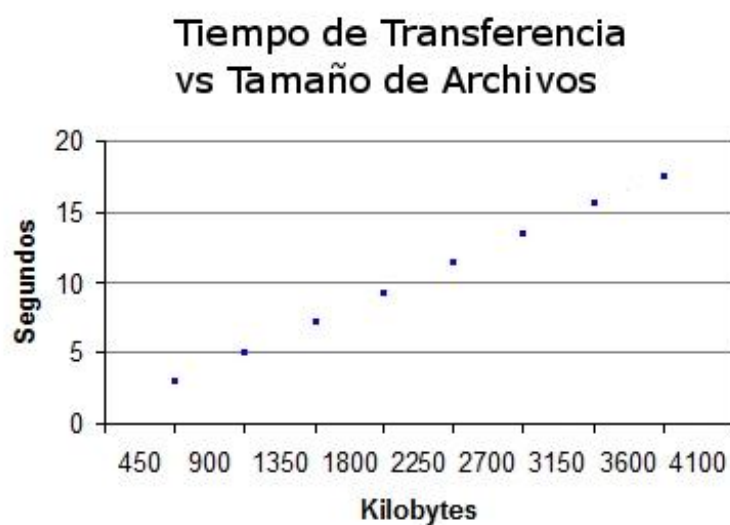


Figura 13 Gráfico Tiempo de Transferencia v/s Tamaño de Archivos

La figura 13 muestra el gráfico correspondiente a pruebas de transferencia realizadas para diferentes tamaños de archivos. Estos tiempos fueron obtenidos utilizando PDAs conectadas en una red adhoc, pero en estado estacionario; o sea ubicadas sobre escritorios. En el gráfico de resultados se observa una proporción lineal entre el tamaño del archivo y el tiempo de transferencia. Seguramente esa linealidad va a cambiar si se agrega movilidad; sin embargo, lo más importante es ver que la tasa de transferencia de datos es muy razonable.

4.3.3. Comunicación entre Aplicaciones

Para evaluar la comunicación entre aplicaciones de distintos workspaces se utilizó la aplicación MobileChat descrita en las secciones anteriores (Figura 14). Como medida de comparación usamos el mecanismo de chat inicial creado para PASIR (Figura 15), el cual se implementa usando mensajería UDP incorporada en el protocolo básico de comunicación, al mismo nivel que la mensajería de detección de peers. Este chat “nativo” es bastante rápido (prácticamente instantáneo) aunque en ocasiones se extravían mensajes. A diferencia de este chat UDP, la aplicación de MobileChat desarrollada utiliza mensajería TCP. Esta mensajería viaja a través de los servicios web expuestos por PASIR para la aplicación, y requiere conocer el destino de cada mensaje enviado. Para poder distribuir estos mensajes, la aplicación necesita contar con un sistema de envío, que recorra la lista de usuarios conectados a la sesión de chat, y le envíe a cada uno el mensaje correspondiente.

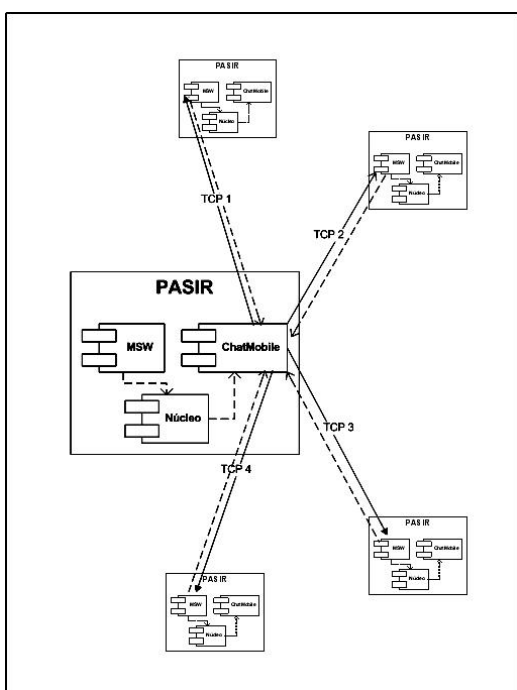


Figura 14 MobileChat

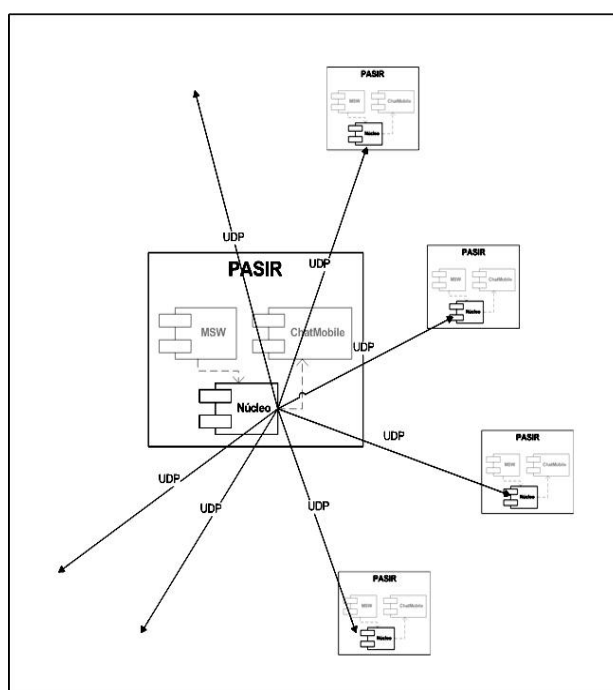


Figura 15 PASIR Chat

El resultado fue que, aunque más lento que el mecanismo UDP, la aplicación de chat creada se comporta bien y permite mantener conversaciones fluidas en tiempo real. Sin embargo, existe un tema importante que puede eventualmente representar un problema para aplicaciones donde los tiempos de respuesta deban ser muy bajos. El problema tiene su

origen en que para el .NET Compact Framework, tecnología utilizada en la implementación de la solución, no es posible realizar multi-threading y abrir múltiples conexiones TCP simultáneas. Debido a esto, una vez que un usuario envía un mensaje a los demás usuarios de la sesión, el mensaje debe ser entregado a cada dispositivo: uno a uno en orden secuencial. Por lo tanto, las demoras en recibir un mensaje por parte de un dispositivo, afectarán a todos aquellos que esperen por dicho mensaje. Esto se vió durante las pruebas, al poner tres dispositivos juntos y enviar un mensaje desde un cuarto dispositivo, siendo perfectamente apreciable el desfase con el que el mensaje llega a cada uno de los dispositivos de destino, en forma secuencial, con diferencias de entre 0,5 y 1 segundo entre cada dispositivo.

En general el uso de mensajería TCP y servicios web fue pensado para procesos que pueden tardar algunos segundos, sin que esto altere la usabilidad del sistema; como por ejemplo para descargas de documentos y sincronización de datos. Por esta razón no es necesario tener respuestas inmediatas a cada mensaje enviado, pero sí se debe tener claro que el problema descrito puede volverse serio, cuando hay muchos participantes conectados a la vez en una misma sesión.

Cabe mencionar que todas las pruebas realizadas son relativas a las condiciones creadas para el caso y pretenden dar una idea de cómo los diferentes factores afectan el desempeño de la aplicación. Diferentes dispositivos, con diferentes capacidades podrán presentar mejores o peores resultados para las distintas pruebas. Más importante aún, diferentes combinaciones de dispositivos dentro de una misma red, tendrán distinto rendimiento, y este en muchos casos vendrá dado por la capacidad de procesamiento de los dispositivos de menor capacidad dentro de la red adhoc.

4.3.4. Comunicación no Prevista usando MapaMovil

Otro caso de prueba efectuado consistió en la adaptación de la aplicación MapaMovil [Monares, 2009a, 2009b], y posterior transferencia de ésta entre dos dispositivos handhelds. MapaMovil es una aplicación orientada a dispositivos móviles que, mediante la

utilización de GPS, permite a un usuario conocer su localización y una serie de recursos cercanos, tales como cuarteles de bomberos, hospitales, cuarteles de carabineros, escuelas, etc. Esta información se encuentra almacenada en forma local en cada dispositivo móvil, y se muestra directamente sobre el mapa de la ciudad. Además, la aplicación tiene acceso a servidores remotos (ubicados en la central de alarmas), a través de los cuales puede obtener información de emergencias actualmente en proceso, alarmas en curso, recursos asignados a cada emergencia, etc. El acceso temprano a esta información permite preparar mejor, y con más anticipación, el proceso de respuesta a una emergencia. La figura 16 muestra un screenshot de esta aplicación.



Figura 16 Aplicación MapaMóvil

MapaMóvil fue utilizada para evaluar el comportamiento de la solución. La prueba se llevó a cabo entre dos dispositivos PDA; uno con la aplicación MapaMóvil (Dispositivo A) y otro sin ella (Dispositivo B). El primer paso consiste en hacer que el dispositivo A comparta la aplicación con resto de la red, en este caso sólo se trata del dispositivo B. El segundo paso es hacer que el dispositivo B realice la descarga de la aplicación y luego la ejecute.

Debemos mencionar que para realizar esta prueba se requirió un proceso de adaptación de MapaMovil, el cual consistió en la creación de su archivo descriptor para PASIR, además de la eliminación de buena cantidad de imágenes que contenían mapas completos y detallados de toda la ciudad de Santiago. Finalmente, se creó y adaptó a MapaMovil, un mecanismo que permite a cada usuario dar a conocer (al resto de la red) su posición en el mapa. La razón de la eliminación de imágenes de mapas pasa por el hecho de que se trataba de alrededor de 5.500 pequeños archivos de imágenes que en su totalidad alcanzaban los 35MB, lo que en el contexto de esta tesis hace inviable una transferencia satisfactoria para la colaboración. Para dar una idea de la demora en el proceso de transferencia, la copia de los archivos vía cable desde un notebook a una PDA tardó 1 hora con 20 minutos. Pese a lo anterior, se debe buscar a futuro un mecanismo adecuado para transferir y visualizar el mapa correspondiente a la zona en que se encuentra el usuario. Pueden existir usuarios con mapas precargados que, en caso de ser requeridos, sean transferidos según la ubicación de la persona. Los resultados de las pruebas de transferencia fueron los siguientes:

Tamaño total de aplicación transferida: 348KB

Cantidad de Archivos: 14

Tiempo de transferencia: ~3 segundos (en promedio)

Log de descarga de aplicación MapaMovil:

```
[11:26:58.504]- Download:MapaMovil.psa  
[11:26:59.365]- Download:MapaMovil.exe  
[11:26:59.441]- Download:MapaMovil.duo  
[11:26:59.528]- Download:config.xml  
[11:26:59.811]- Download:AttachmentModule.dll  
[11:27:00.017]- Download:Coordenadas.dll  
[11:27:00.105]- Download:GPS.dll  
[11:27:00.330]- Download:Lector.xml  
[11:27:00.636]- Download:Microsoft.WindowsMobile.Samples.Location.dll  
[11:27:00.730]- Download:MobileWebServerExtension.dll
```

[11:27:00.936]- Download:SecurityModule.xml

[11:27:01.132]- Download:en_EU.xml

[11:27:01.227]- Download:es_CL.xml

[11:27:01.322]- Download:imgvacia.png

De esta prueba podemos concluir que el tiempo necesario para que dos usuarios comiencen a colaborar utilizando una aplicación compleja, que cuenta con una interfaz gráfica y herramientas como servicios web para comunicarse, es más que aceptable. Esto permite pensar que muy probablemente la hipótesis 1 es verdadera. Sin embargo, no debemos perder de vista que la prueba fue realizada en condiciones ambientales óptimas, y que involucró sólo dos dispositivos. De todas maneras, esta solución parece escalar bien, por lo que si el número de dispositivos involucrados no crece en forma abrumadora, la solución debería continuar siendo un buen mecanismo para soportar la colaboración no prevista.

4.3.5. Pruebas de Desarrollo Usando el Framework

Como se ha mencionado anteriormente, la infraestructura desarrollada, es también un framework de apoyo al desarrollo de aplicaciones móviles orientadas al trabajo colaborativo. En este contexto, durante el año 2008, en el curso “CC52N: Computación para el Trabajo Grupal” dictado por el Prof. José A. Pino del DCC de la Universidad de Chile, se dio a los alumnos tres tareas consistentes en el desarrollo de aplicaciones colaborativas sobre redes MANET. La tarea 1 involucró un Chat (Fig. 16.a), la tarea 2 una pizarra compartida (Fig. 16.b), y la tarea 3 un presentador distribuido (Fig. 16.c).



Figura 17 Aplicaciones Colaborativas Móviles – Curso CC52N

Un total de 14 alumnos tomaron el curso el primer semestre del 2008. Para realizar las tareas estos alumnos formaron grupos de dos integrantes cada uno. La composición de dichos grupos fue diseñada por el equipo de cátedra del curso, quienes trataron de homogeneizar la fuerza de trabajo asociada a cada uno de ellos. Para realizar esta actividad se utilizó, como información de apoyo, las notas que los alumnos tenían en cursos de programación e ingeniería de software.

En todas las tareas hubo grupos a los cuales se les permitió utilizar el framework como apoyo al desarrollo de los productos, y otros a los que no se les permitió esto. A estos últimos se les sugirieron diversas alternativas para facilitar el esfuerzo de construir la aplicación de groupware. Semanalmente el profesor auxiliar del curso solicitaba a cada grupo, y computaba, las horas de trabajo que ellos habían asignado a la solución del problema encomendado. Finalmente, una vez entregadas las tareas, el equipo de cátedra les asignó una calificación y contó las líneas de código efectivamente escrita en cada una de las soluciones entregadas. Luego, para la tarea siguiente se formaron nuevos grupos y la dinámica del trabajo fue la misma. La tabla 1 muestra el resumen de los resultados obtenidos.

Tabla 1. Resumen de notas en las tareas del curso

Estudiante	Tarea 1				Tarea 2				Tarea 3				Promedio
	Plataforma	HH	LOC	Nota	Plataforma	HH	LOC	Nota	Plataforma	HH	LOC	Nota	
1	NO	33,5	785	6,0	SI	42,0	986	6,1	NO	95,0	2544	4,3	5,5
2	NO	33,5	785	6,0	NO	90,0	1226	5,2	SI	45,5	1078	5,9	5,7
3	SI	16,5	445	5,9	SI	45,5	780	6,3	NO	19,0	0	1,0	4,4
4	SI	16,5	445	5,9	NO	9,0	0	1,0	SI	63,0	1512	5,5	4,1
5	NO	47,0	1159	5,0	NO	9,0	0	1,0	SI	63,0	1512	5,5	3,8
6	NO	47,0	1159	5,0	SI	53,0	830	6,0	NO	17,5	0	1,0	4,0
7	SI	20,0	394	6,1	NO	71,5	1555	5,6	SI	42,5	1136	6,1	5,9
8	SI	20,0	394	6,1	SI	53,0	830	6,0	NO	12,0	0	1,0	4,4
9	NO	15,0	0	1,0	NO	71,5	1555	5,6	SI	45,5	1078	5,9	4,2
10	NO	15,0	0	1,0	SI	42,0	986	6,1	NO	17,5	0	1,0	2,7
11	NO	43,0	981	5,7	SI	45,5	780	6,3	NO	95,0	2544	4,3	5,4
12	NO	43,0	981	5,7	NO	14,0	0	1,0	SI	42,5	1136	6,1	4,3
13	SI	14,5	208	6,5	NO	90,0	1226	5,2	NO	19,0	0	1,0	4,2
14	SI	14,5	208	6,5	NO	14,0	0	1,0	NO	12,0	0	1,0	2,8
												Promedio Total	4,4
Con Plataforma	Promedio		17,0	349	6,2		46,8	865,3	6,1		50,3	1242,0	5,8
	Desviación		2,5	112	0,3		5,0	96,1	0,1		9,9	210,7	0,3
Sin Plataforma	Promedio		41,2	975	5,6		80,8	1390,5	5,4		95,0	2544,0	4,3
	Desviación		6,2	167	0,5		10,7	189,9	0,2		0,0	0,0	0,0
Sin Plataforma y No Aprobados	Promedio				4,4								1,8
	Desviación				2,1				2,4				1,5

En primer lugar se pudo constatar que los resultados obtenidos, están alineados con lo expresado en la hipótesis 2. Casi en todos los casos en que el framework fue utilizado, la nota de los alumnos fue mejor, lo cual indica que la solución obtenida fue de mejor calidad y, por consiguiente, dicha solución tendría mejores probabilidades de éxito en el mundo real. Además, la cantidad de horas-hombre (HH) y la cantidad de líneas de código (LOC) de las soluciones que emplearon el framework, fue consistentemente menor que aquellas soluciones que no lo usaron. Finalmente, todos los casos en los que los alumnos reprobaron o sacaron notas bajas, correspondieron a casos donde los alumnos no utilizaron el framework propuesto.

En resumen, la experiencia resultante fue satisfactoria, puesto que se pudo comprobar (aunque en forma preliminar) la validez de la hipótesis 2. También la experimentación sirvió para evaluar las capacidades del framework, como herramienta de apoyo al desarrollo de aplicaciones colaborativas móviles.

5. Conclusiones y Trabajo Futuro

Este trabajo de tesis buscó entregar una propuesta para enfrentar el desafío de la colaboración no prevista entre usuarios de dispositivos móviles, cuando éstos llevan a cabo trabajo colaborativo. La infraestructura desarrollada aborda este desafío, e implementa una serie de servicios que permiten la interacción entre estos usuarios. Sin embargo, al tratarse de una propuesta experimental, esta infraestructura no pasa de ser un prototipo mejorable.

La infraestructura propuesta está limitada a un espectro acotado de dispositivos. Estos son aquellos dispositivos que sean capaces de usar el .Net Compact framework, lo que en la actualidad equivale a la gran mayoría de las PDA y varios teléfonos celulares de gama alta. Esta es sólo una limitante técnica, ya que todos los mecanismos implementados son factibles de ser reproducidos en otros lenguajes y para otros sistemas operativos. Por otra parte, la comunicación se realiza siempre usando mensajería XML, que en los casos de la mensajería básica del sistema (por ej. en la detección de usuarios y sesiones) corresponde a un protocolo bien definido y acotado. En el caso de mensajería más compleja entre aplicaciones, la comunicación adhiere al estándar SOAP para servicios web. Esto implica que eventualmente no será relevante considerar si dos usuarios tienen el mismo software, siempre que ambos sean capaces de entender la mensajería que reciben. Ahora bien, lograr que distintos fabricantes creen distintas implementaciones de la solución no es algo esperable, pero sí podemos esperar que usuarios con las capacidades técnicas necesarias lo puedan hacer, si se apunta a definir un protocolo estándar de reconocimiento y comunicación.

Hoy en día ya es posible acceder a planes de Internet ilimitado a precios asequibles, mientras que las velocidades y capacidades de los dispositivos aumentan cada día. Con esto cabe preguntarse qué ventaja tiene un sistema como el propuesto versus un sistema de interacción basado directamente en el uso de Internet, tal cual es usado hoy por millones de usuarios de computadores personales. Las ventajas del sistema son claras en escenarios donde las conexiones vía Internet suelen no estar disponibles (por ejemplo, en un área de

desastre, o fuera de las áreas urbanas), pero ¿dónde más sería aplicable este sistema si nos paramos en 10 años más, con los avances esperados que esto implica? A priori podemos pensar en los distintos escenarios propuestos durante la introducción de este trabajo. Otros escenarios podrían ser la investigación en terreno, donde este trabajo podría ayudar a grupos de científicos o exploradores en distintas tareas, como por ejemplo tomas de muestras o mediciones en ambientes donde las redes convencionales no llegan. Podemos ir un poco más allá y ver cómo un sistema como el planteado podría ser un aporte para la operación de sistemas robóticos o domóticos.

Tomemos por ejemplo la domótica, que busca dar al habitante de una casa, herramientas para controlar los dispositivos de su hogar desde algún otro dispositivo, como por ejemplo un control remoto central o un celular. Ahora bien, viéndolo desde la perspectiva de este trabajo de tesis, podemos dar vuelta el tema y visualizar un futuro en que sean los distintos dispositivos del hogar los que detecten el dispositivo móvil del usuario y le ofrezcan distintas aplicaciones en su teléfono móvil o PDA. Sería innovador, por ejemplo, comprarse un equipo de música, encenderlo y recibir un mensaje desde el mismo equipo queriendo compartir una aplicación para reproducción de música. Naturalmente todo esto debe tener mecanismos para que los mensajes no sean recibidos por personas ajenas al hogar, entre otras consideraciones.

Otro punto que, por el momento, es un aporte interesante de esta propuesta, radica en que el costo monetario de la comunicación entre dispositivos que utilizan el framework propuesto es casi nulo. Sin embargo, debido a la baja periódica del costo de las comunicaciones, probablemente este aspecto no será muy significativo en 4 o 5 años más. Por otra parte, tal vez este aspecto podría mantenerse siendo relevante en escenarios que involucren redes de sensores, ya que estos dispositivos podrían ser muchos y por lo tanto la diferencia en el costo global de operación podría ser significativa.

Otro escenario posible es el de la computación distribuida llevada a dispositivos pequeños. Podemos pensar en aplicaciones y toma de decisiones que impliquen cálculos relativamente grandes para un dispositivo pequeño, pero que resulten rápidos de realizar en

dispositivos grandes. Si un dispositivo es capaz de comunicarse con otros y tiene suficiente capacidad para correr un detector de dispositivos cercanos y un micro-servidor web, entonces es posible crear mecanismos que le permitan “colgarse” de otros dispositivos para realizar cálculos complejos y poder obtener resultados que de otra forma no le serían posibles. Un ejemplo sencillo de esto podría ser una herramienta de desarrollo de aplicaciones móviles, que cuente con un editor y un compilador de código. El editor podría correr perfectamente en una PDA, como las usadas para las pruebas descritas en la sección 4.3. Lamentablemente a la hora de compilar, la herramienta tardaría demasiado tiempo en realizar la tarea, si es que llegase a poder realizarla. Sin embargo, el compilador podría comunicarse con una máquina de mayor capacidad (presente en la red), traspasar los archivos de código fuente a un servicio web de compilación y recibir de vuelta el programa compilado, o bien los errores encontrados.

Como trabajo futuro, en primer lugar, debemos plantear la necesidad de crear un mecanismo que permita, dentro de una red MANET, asignar direcciones IP que no se repitan entre los distintos usuarios existentes en la red. Actualmente los usuarios deben tomar un acuerdo previo sobre qué dirección de IP utilizar antes de iniciar una sesión colaborativa. La existencia de dos IP iguales dentro de la misma red en el mismo momento, produciría colisiones entre ambos usuarios haciendo imposible para ellos comunicarse y dificultándose a los demás.

A más largo plazo sería interesante contar con diferentes implementaciones del sistema que corran en dispositivos de características variadas, y que sean capaces de interactuar entre ellas. De esta forma estaremos logrando finalmente la colaboración totalmente no prevista entre diferentes workspaces móviles.

6. Bibliografía y Referencias

- [Andriessen, 2006] Andriessen, J. H. E., Vartiainen, M. (eds.). Mobile virtual work: A new paradigm? Springer, 2006.
- [BNet, 2008] BNet. IDC Predicts the Number of Worldwide Mobile Workers to Reach 1 Billion by 2011. January, 2008. URL: http://findarticles.com/p/articles/mi_m0EIN/is_2008_Jan_15/ai_n24230213. Last visit: May, 2009.
- [Brugnoli, 2005] Brugnoli, M.C., Davide, F. Slagter, R. The Future of Mobility and of Mobile Services. In: P. Cunningham and M. Cunningham (eds.), Innovation and the Knowledge Economy: Issues, Applications, Case Studies, pp 1043-1055, IOS Press. 2005.
- [Buszko, 2001] Buszko, D., Lee, W., Helal, A.: Decentralized Ad-Hoc Groupware API and Framework for Mobile Collaboration. Proceedings of ACM International Conference on Supporting Group Work (GROUP), ACM Press, Colorado, USA, 2001, 5-14.
- [Carrasco, 2007] Carrasco, D. Servicios Web en Dispositivos Móviles para el Soporte de Aplicaciones Colaborativas. Memoria de Ingeniería Civil en Computación, Departamento de Ciencias de la Computación, Universidad de Chile. 2007.
- [Castro, 2007] Castro, F. Manejo de Sesiones Cerradas para Ambientes Colaborativos Móviles. Memoria de Ingeniero Civil en Computación, Departamento de Ciencias de la Computación, Universidad de Chile. 2007
- [Comfort, 2004] Comfort, L. Coordination in Complex Systems: Increasing Efficiency in Disaster Mitigation and Response. Annual Meeting of the American Political Science Association, San Francisco, Aug-Sept., 2004.
- [Drake, 2005] Drake, S., R. Giusto, R. Boggs, M. Sandler, K. Burden. Worldwide Mobile Worker Population 2005-2009 Forecast and Analysis. IDC, Octubre 2005.

- [Dutta, 2009] Dutta, S., Mia, I. (Eds.). The Global Information Technology Report 2008–2009: Mobility in a Networked World. World Economic Forum & INSEAD. 2009.
- [Emol, 2008] El Mercurio (On line). Entel, pionero en la implementación de tecnologías WiMax, NGN y 3.5G en Chile. El Mercurio Special Editions. Nov. 18, 2008. URL: <http://www.edicionesespeciales.elmercurio.com/destacadas/detalle/index.asp?idnoticia=0117052007021X3070069&idcuerpo=620>. Last visit: May 2009.
- [Franco, 2005] Franco, R.D., Ortiz, A., Navarro, R. Enhancing Supply Chain Co-ordination by means of a collaborative platform based on SOA. Proceedings of PRO-VE 2005: 6th IFIP Working Conference on Virtual Enterprises. Kluwer Academic Publishers. 2005.
- [Hakkila, 2005] Häkkinen, J., Mäntyjärvi, J. Collaboration in Context-Aware Mobile Phone Applications Proceedings of HICSS 2005. IEEE Press. 2005.
- [Handorean, 2003] Handorean, R., Payton, J., Julien, C., Roman, G. Coordination Middleware Supporting Rapid Deployment of Ad Hoc Mobile Systems. Proceedings of the International Workshop on Mobile Computing Middleware. pp. 363-368. May, 2003.
- [Heinemann, 2003] Heinemann, A., Kangasharju, J., Lyardet, F., Mühlhäuser, M. iClouds - Peer-to-Peer Information Sharing in Mobile Environments. LNCS 2790, Springer-Verlag. pp. 1038-1045. 2003.
- [Hirsch, 2006] Hirsch, F., Kemp, J., Ilkka, J.: Mobile Web Services: Architecture and Implementation. Nokia Research Center. John Wiley & Sons Publisher, 2006.
- [Iansiti, 2007] Iansiti, M., Jamrog, J. New Mobility: Productivity Metrics. Networking Views 9, pp. 6-7. Oct. 2007.
- [Jorstad, 2005] Jørstad, I., Dustdar, S., Van Thanh, D. A Service Oriented Architecture Framework for Collaborative Services. Proceedings of WETICE 2005. IEEE Press. 2005.
- [Kortuem, 2001] Kortuem, G., Schneider, J., Preuitt, D., Thompson, T.G.C., Fickas, S., Segall, Z. When peer-to-peer comes face-to-face: collaborative peer-to-peer

- computing in mobile ad-hoc networks. Proceedings of the First International Conference on Peer-to-Peer Computing. Aug. 27-29, 2001. Pp. 75-91.
- [Marques, 2005] Marques, J., Navarro, L. Autonomous and Self-sufficient Groups: Ad Hoc Collaborative Environments. Proceedings of the CRIWG'05, Porto do Galinhas, Brazil, September, 2005. Lecture Notes in Computer Science 3706, pp. 57-72. Springer-Verlag.
- [Monares, 2009a] Monares, A. MapaMóvil: Un ambiente colaborativo móvil para Bomberos. Memoria de Ingeniero Civil en Computación, Departamento de Ciencias de la Computación, Universidad de Chile. 2009.
- [Monares, 2009b] Monares, A. Ochoa, S. F., Pino, J. A., Herskovic, V., Neyem, A. MobileMap: A Collaborative Application to Support Emergency Situations in Urban Areas. IEEE Press, Los Alamitos, CA. Proc. of the 13th International Conference on Computer Supported Cooperative Work in Design (CSCWD'09), Santiago, Chile, April 2009, pp. 565-570.
- [Moya, 2007] Moya, C. Módulo para la Sincronización Automática de Documentos XML. Memoria de Ingeniero Civil en Computación, Departamento de Ciencias de la Computación, Universidad de Chile. 2007
- [Nah, 2005] Nah, F., Sheng, H. The Value of Mobile Applications: A Utility Company Study. Communications of the ACM, Vol. 48, No. 2. pp. 85-90. 2005.
- [Neyem, 2005] Neyem, A., Ochoa, S.F., Guerrero, L.A., Pino, J.A. Sharing Information Resources in Mobile Ad-hoc Networks. 11th Int. Workshop on Groupware (CRIWG 2005). Lecture Notes in Computer Science. Springer. Porto do Galinhas, Brazil. Sept. 2005.
- [Neyem, 2006] Neyem, A., Ochoa, S.F., Pino, J.A. A Strategy to Share Documents in MANETs using Mobile Devices. Proceedings of the IEEE 8th International Conference on Advanced Communication Technology (ICACT 2006). Phoenix Park, Korea. Feb. 2006.

- [Neyem, 2006b] Neyem, A., Ochoa, S. F., Pino, J.A. Supporting Mobile Collaboration with Service-Oriented Mobile Units. Proceedings of the 12th International Workshop on Groupware (CRIWG 2006). Medina del Campo, Spain. Springer LNCS, Berlin Heidelberg, Germany. 2006. pp. 228-245.
- [Neyem, 2007] Neyem, A., Ochoa, S. F., Pino, J.A. Designing Mobile Shared Workspaces for Loosely Coupled Workgroups. Lecture Notes in Computer Science 4715, pp. 173-190. 2007
- [Ochoa, 2006] Ochoa, S.F., Neyem, A., Pino, J.A, Borges, M.R.S. Using Context To Support Group Decision Making in Disaster Affecting Urban Areas. Proceedings of the IFIP International Conference on Creativity and Innovation in Decision Making and Decision Support. London, UK., June 2006.
- [Ochoa, 2007] Ochoa, S. F., Neyem, A., Pino, J. A., Borges, M. R. S. Supporting Group Decision Making and Coordination in Urban Disasters Relief Efforts. Journal of Decision Systems 16(2), 2007, 143-172.
- [Ochoa, 2009] Ochoa, S. F., Bravo, G., Pino, J., Rodriguez, J.F. Coordinating Loosely-Coupled Work in Construction Inspection Activities. Accepted in Group Decisions and Negotiation, 2009.
- [Perry, 2001] Perry, M., O'Hara, K., Sellen, A., Brown, B., Harper, R. Dealing with mobility: understanding access anytime, anywhere. ACM Trans. Computer-Human Interact., Vol. 8, No. 4. pp. 323-347. December 2001.
- [Procopio, 2002] Procopio, M. YCab.Net: Decentralized Collaboration Groupware for Mobile Devices Using the Microsoft .Net Framework. A Thesis Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Master of Science. University of Florida. 2002
- [Schaffers, 2006] Schaffers, H., Brodt, T., Pallot, M., Prinz, W. (eds). The Future Workplace - Perspectives on Mobile and Collaborative Working, Telematics Institute, The Netherlands. 2006.

- [Subtel, 2009] Subsecretaría de Telecomunicaciones de Chile (Subtel). Comienza Proyecto de Internet Rural que Permitirá Acceso al 92% de las Personas del País. Gobierno de Chile. URL:
http://www.subtel.cl/prontus_subtel/site/artic/20090131/pags/20090131161833.html
. Last visit: May 2009.
- [Sudan, 2007] Sudan, S., S. Ryan, S. Drake, M. Sandler, R. Boggs, and R. Giusto. Worldwide Mobile Worker Population 2007-2011 Forecast. IDC, December 2007.
- [Tanasescu, 2006] Tanasescu, V., Gugliotta, A., Domingue, J., Villarías, L., Davies, R., Rowlatt, M., Richardson, M., and Stincic, S. A Semantic Web Services GIS based Emergency Management Application. Proceedings of the Workshop: Semantic Web Challenge at The 5th Int. Semantic Web Conference (ISWC06), Athens, Georgia, USA. 2006.
- [Tarasewich, 2003] Tarasewich, P. Designing Mobile Commerce Applications. Comm. of the ACM, 46(12), pp. 57-60. 2003.
- [Tentori, 2008] Tentori, M., Favela, J. Collaboration and Coordination in Hospital Work through Activity-Aware Computing. Int. J. on Cooperative Information Systems 17(4), pp. 413-442, 2008.
- [Vuolle, 2008] Vuolle, M., Aula, A., Kulju, M., Vainio, T., Wigelius, H. Identifying Usability and Productivity Dimensions for Measuring the Success of Mobile Business Services. Advances in Human-Computer Interaction, Vol. 2008. Available at URL: <http://www.hindawi.com/journals/ahci/2008/680159>. Last visit: May, 2009.
- [Wang, 2005] Wang, Y., van de Kar, E., Meijer, G. Designing mobile solutions for mobile workers: lessons learned from a case study. Proceedings of the ACM 7th International Conference on Electronic Commerce (ICEC). Xian, China. Pp.582-589. 2005.

Anexo A: Código Fuente de los Principales Componentes del Framework

A continuación se presenta el código fuente asociado a los principales componentes del framework. Con el objetivo de mantener la simplicidad y legibilidad del código asociado a estos componentes, sólo se presenta la estructura básica de cada uno. El primer componente que se muestra es *PeerNearMe*: mecanismo base (core) encargado de la detección de peers/usuarios.

```
namespace PeerNearMe
{
    //public delegate void DelegateSendOtherMsg();

    public class Controller
    {
        private static IPeersManager iPeersManager;
        private static Timer ticker;

        public Controller(IPeersManager peerManager)
        {
            iPeersManager = peerManager;
        }
        public void Start()
        {
            ticker = new Timer(Maintain, null, 1000, 1000);
        }

        public void Stop()
        {
            ticker.Dispose();
        }

        private static void Maintain(Object state)
        {
```

```

//Lista para almacenar temporalmente los elementos a eliminar
List<string> tmpRemove = new List<string>();

//Se recorren todos los peers restando 1 a su calidad de
conexión
foreach (IPeer peer in iPeersManager)
{
    peer.ConnectionQuality--;

    //si la calidad llega a 0 el peer se marca para ser
borrado
    if (peer.ConnectionQuality < 0)
        tmpRemove.Add(peer.Id);
}

//Se eliminan todos los peers marcados
foreach (string id in tmpRemove)
{
    iPeersManager.Remove(id);
}

//Se envía el mensaje de identificación del peer local a la
MANET
try
{
    iPeersManager.SendLocalInfo();
}
catch
{
}
}
}
}

```

IApplication: Este componente define la interfaz de programación de las aplicaciones compartidas.

```
namespace ISharedApplication
{
    public interface IApplication
    {
        string Name { get; }
        string Description { get; }
        string Author { get; }
        string Version { get; }
        bool IsRunning { get; }
        string SessionId { get; set; }
        string ApplicationId { get; set; }

        void Initialize();
        void Dispose();

        void ProcessIncomingMessage(byte[] message);
    }
}
```

LocalSharedApplication: Este componente corresponde a la clase mediante la cual se crean los objetos correspondientes a aplicaciones compartidas, que pueden ser ejecutadas localmente por el usuario. Es aquí donde se inicializan las aplicaciones usando la interfaz *IApplication*.

```
namespace Coordination
{
    public class LocalSharedApplication : SharedApplication
    {
        private string applicationFileName;
        private IApplication application;
    }
}
```

```

    /// <summary>
    /// Crea un nuevo objeto LocalSharedApplication
    /// </summary>
    /// <param name="fileInfo">FileInfo del descriptor de la
aplicacion</param>
    public LocalSharedApplication(FileInfo fileInfo, string
sessionId)
        : base(LocalUser.Instance().Id + "-" +
++LocalUser.Instance().SharedObjectsCount, fileInfo.Name,
fileInfo.GetHashCode(), sessionId, LocalUser.Instance().Id)
    {
        this.fileInfo = fileInfo;
        synchronized = true;
    }

    /// <summary>
    /// Crea un objeto LocalSharedApplication para el cual ya se
tenia el id, se usa para cargar archivos que ya han sido compartidos al
iniciar el sistema.
    /// </summary>
    /// <param name="id">id del SharedApplication</param>
    /// <param name="fileInfo">FileInfo del archivo</param>
    public LocalSharedApplication(string id, FileInfo fileInfo,
string sesssionId)
        : base(id, fileInfo.Name, fileInfo.GetHashCode(), sesssionId,
LocalUser.Instance().Id)
    {
        this.fileInfo = fileInfo;
        synchronized = true;
    }

    public void LaunchApplication()
    {
        if (application!=null && application.IsRunning)
            return;
        ReadApplicationFile();
        LoadApplication();
    }

```

```

        RunApplication();
    }

    public void ReadApplicationFile()
    {
        string psaFilePath = fileInfo.FullName;

        FileStream psaFile = new FileStream(fileInfo.FullName,
FileMode.Open);
        XmlTextReader reader = new XmlTextReader(psaFile);
        reader.ReadStartElement("PasirSharedApplication");
        reader.ReadStartElement("Header");
        this.applicationName = reader.ReadElementString("Name");
        this.description = reader.ReadElementString("Description");
        this.author = reader.ReadElementString("Author");
        this.version = reader.ReadElementString("Version");
        reader.ReadEndElement();

        bool canExecute = true;
        reader.ReadStartElement("Application");

        this.applicationFileName = fileInfo.DirectoryName +
"\\\\"+reader.ReadElementString("ApplicationFileName");

        while (true)
        {
            try
            {
                if
(!CheckDllExists(reader.ReadElementString("DllFile")))
                    canExecute = false;
            }
            catch
            {
                break;
            }
        }
        reader.ReadEndElement();
    }

```

```

        reader.ReadEndElement();
        reader.Close();
        if (!canExecute)
        {
            throw new Exception("Falta dll definida en archivo");
        }
    }

    private bool CheckDllExists(string dllFile)
    {
        FileInfo file = new FileInfo(fileInfo.DirectoryName + "\\\" +
dllFile);
        return file.Exists;
    }

    private void LoadApplication()
    {
        Assembly applicationAssembly =
Assembly.LoadFrom(applicationFileName);

        //Tipos
        foreach (Type applicationType in
applicationAssembly.GetTypes())
        {
            try
            {
                if (applicationType.IsPublic &&
!applicationType.IsAbstract)
                {
                    Type typeInterface = (typeof(IApplication));

                    if (typeInterface != null)
                    {
                        application =
(IApplication)Activator.CreateInstance(applicationAssembly.GetType(applic
ationType.ToString()));

```

```

                break;
            }
        }
    }
    catch
    {
    }
}

private void RunApplication()
{
    application.Initialize();
    application.SessionId = this.sessionId;
}

public IApplication Application
{
    get { return application; }
    set { application = value; }
}
}
}

```

WebServiceBridge: Este componente redirige la mensajería proveniente del servidor web, a la aplicación correspondiente.

```

namespace Coordination
{
    public class WebServiceBridge
    {
        /// <summary>
        /// Escucha mensajes del servicio web
        /// </summary>
    }
}

```



```

        IpcTcpListener webServiceListener;

        public WebServiceBridge()
        {
            webServiceListener = new IpcTcpListener(6968);
            webServiceListener.MessageArrived += new
MessageArrivedEventHandler(webServiceListener_MessageArrived);
webServiceListener.StartListen();
        }

        void webServiceListener_MessageArrived(object source, object
message, string service)
        {
            throw new NotImplementedException();
        }

        void webServiceListener_MessageArrived(object source, byte[]
message)
        {
            try
            {
                MemoryStream ms = new MemoryStream(message);
                XmlTextReader reader = new XmlTextReader(ms);

                reader.ReadStartElement("IpcTcpMessage");
                string sessionId = reader.ReadElementString("SessionId");
                string applicationId =
reader.ReadElementString("ApplicationId");
                reader.Close();
                FowardMessageToApplication(sessionId, applicationId,
message);
            }
            catch(Exception e)
            {
                string ex = e.Message;
            }
        }
    }

```

```

        private void ForwardMessageToApplication(string sessionId, string
applicationId, byte[] message)
        {
            try
            {
                foreach (ISharedObject sharedObject in
CoordinationManager.Instance().Sessions[sessionId][LocalUser.Instance().I
d].SharedObjects.Values)
                {
                    if
(sharedObject.GetType().Equals(typeof(LocalSharedApplication))
&&
((LocalSharedApplication)sharedObject).Application.IsRunning
&&
((LocalSharedApplication)sharedObject).Application.ApplicationId.Equals(a
pplicationId))
                    {
                        ((LocalSharedApplication)sharedObject).Application.ProcessIncomingMessage
(message);
                    }
                }
            }
            catch
            {
            }
        }

        public void ListenerStop()
        {
            this.webServiceListener.StopListen();
        }
    }
}

```

Synchronization: Esta es la clase encargada de dirigir la sincronización de dos documentos XML, usando XMLSync.

```
namespace XMLSync
{
    static public class Synchronization
    {
        public static void Synchronize(ArrayList rutasEntrada, string
rutaSalida)
        {
            try
            {
                XmlDocument docA = new XmlDocument();
                XmlDocument docB = new XmlDocument();
                //sincronizo 2 documentos y el resultado de ellos lo
sincronizo con un próximo documento.
                String rutaDocA = (String)rutasEntrada[0];
                docA.Load(rutaDocA);
                //archivosOK revisará que todos los documentos estén bien
formados.

                bool archivosOK = true;
                for (int i = 1; i < rutasEntrada.Count; i++)
                {
                    String rutaDocB = (String)rutasEntrada[i];
                    docB = new XmlDocument();
                    docB.Load(rutaDocB);
                    DiffTree diffT = new DiffTree(docA, docB);
                    if (!diffT.checkDocuments(docA, docB))
                    {
                        archivosOK = false;
                        break;
                    }
                    XmlDocument diff = new XmlDocument();
                    diff = diffT.getDiffTree();
                    Reconcile r = new Reconcile();
                    docA = new XmlDocument();
```



```

{
    if (Notify != null)
    {
        Notify(this, e);
    }
}

#region INotificator Members

public void AddUserToIgnoreList(string userId)
{
    if (!usersIgnored.Contains(userId))
        usersIgnored.Add(userId);
}

public void AddSessionToIgnoreList(string sessionId)
{
    if (!sessionsIgnored.Contains(sessionId))
        sessionsIgnored.Add(sessionId);
}

public void AddObjectToIgnoreList(string sharedObjectId)
{
    if (!sharedObjectsIgnored.Contains(sharedObjectId))
        sharedObjectsIgnored.Add(sharedObjectId);
}

public void RemoveUserFromIgnoreList(string userId)
{
    if (usersIgnored.Contains(userId))
        usersIgnored.Remove(userId);
}

public void RemoveSessionFromIgnoreList(string sessionId)
{
    if (sessionsIgnored.Contains(sessionId))
        sessionsIgnored.Remove(sessionId);
}

```

```
    }

    public void RemoveObjectFromIgnoreList(string sharedObjectId)
    {
        if (sharedObjectsIgnored.Contains(sharedObjectId))
            sharedObjectsIgnored.Remove(sharedObjectId);
    }

    #endregion

    #region INotificator Members

    public event EventHandler<NotificationEventArgs> Notify;

    #endregion
}
}
```

Anexo B: Código Fuente de Aplicación MobileChat

En forma similar a lo presentado en el anexo anterior, aquí se muestra la estructura básica de los componentes que forman parte de la aplicación *MobileChat*. El primer componente que se describe es *ChatGui*, el cual implementa la lógica del formulario de la aplicación de chat:

```
namespace ChatApplication
{
    delegate void SetTextCallback(string userId, string text);

    public partial class ChatGui : Form
    {
        private ChatApplication application;

        public ChatGui(ChatApplication application)
        {
            InitializeComponent();
            this.application = application;
            this.application.IsRunning = true;
        }

        private void Gui_Closed(object sender, EventArgs e)
        {
            this.application.IsRunning = false;
        }

        public void SetTextChat(string userId, string text)
        {
            if (this.textBoxChat.InvokeRequired)
            {
                SetTextCallback d = new SetTextCallback(SetTextChat);
                this.Invoke(d, new object[] { userId, text });
            }
        }
    }
}
```

```

else
{
    //ponemos un try por si el mensaje a sido enviado por un
usuario
    // que por algun motivo no se encuentra en el UserManager
string nombre = "Desconocido";
    try
    {
        nombre=
CoordinationManager.Instance().Users[userId].Name;
    }
    catch
    {
    }
    this.textBoxChat.Text += "["+nombre+"]: "+text + "\r\n";
    }
}

private void buttonEnviar_Click(object sender, EventArgs e)
{
    MobileChat.MobileChatService proxy = new
MobileChat.MobileChatService();
    foreach (UserSession userSession in
CoordinationManager.Instance().Sessions[application.SessionId].UserSessio
ns)
    {
        proxy.Timeout = 5000;
        if (userSession.IUser.Ip == null)
        {
            proxy.Url =
"http://localhost/MobileChatService.asmx";
        }
        else
            proxy.Url = "http://" + userSession.IUser.Ip +
"/MobileChatService.asmx";
        try
        {

```



```

public void Dispose()
{
    try
    {
        gui.Dispose();
    }
    catch //gui ya se habia cerrado
    {
    }
}

public void ProcessIncomingMessage(byte[] message)
{
    ChatMessage chatMessage = new ChatMessage();
    chatMessage.Process(message);
    gui.SetTextChat(chatMessage.UserId, chatMessage.Message);
}
...

#endregion
}
}

```

MobileChat (WebService): Este componente implementa el servicio web usado por la aplicación MobileChat.

```

namespace MobileChatService
{
    public class MobileChat
    {
        public MobileChat()
        {

```

```
    }

    [WebMethod(MessageName = "PutMessage")]
    public void PutMessage(string sessionId, string applicationId,
string userId, string message)
    {
        try
        {
            ChatMessage chatMessage = new ChatMessage();
            IpcTcpClient client = new IpcTcpClient(6968);
            client.SendMessage(chatMessage.Make(sessionId,
applicationId, userId, message));
        }
        catch (Exception e)
        {








        }


    }
}
}
```

Anexo C: Iconografía de la Interfaz de Usuario del Manejador de Sesiones









A continuación se muestran los diferentes íconos usados en las interfaces de usuario correspondientes al manejador de sesiones y al detector de usuarios del sistema:

Usuarios:

Icono	Descripción
	Usuario local del sistema. Icono de usuario color celeste.
	Usuario conectado a la sesión seleccionada (usuario distinto al usuario local) Icono de usuario color verde.
	Usuario conectado a la sesión seleccionada (usuario distinto al usuario local) Baja calidad de conexión se representan con tonalidades de verde más suave.
	Usuario desconectado de la sesión seleccionada pero conectado a la MANET. Icono de usuario color gris. No depende de si el usuario está suscrito o no.
	Usuario invitado a la sesión seleccionada. Icono de usuario color gris con un certificado representando una invitación.
	Usuario solicitando invitación a la sesión seleccionada. Icono de usuario color gris con un signo de interrogación verde.
	Grupo de usuarios suscritos a la sesión seleccionada(conectados o no a esta) Icono de usuario color azul

	Grupo de usuarios no suscritos a la sesión seleccionada. Icono de usuario color gris
---	---

Sesiones:

Icono	Descripción
	Sesión abierta a la que el usuario local se encuentra conectado. Icono de grupo de usuarios en colores.
	Sesión abierta a la que el usuario local no está conectado. Icono de grupo de usuarios gris.
	Sesión cerrada (con o sin roles) a la que el usuario local se encuentra conectado. Icono de grupo de usuarios en colores con un candado cerrado.
	Sesión cerrada (con o sin roles) a la que el usuario local no está conectado. Icono de grupo de usuarios gris con un candado cerrado.
	Sesión cerrada a la que el usuario ha sido invitado (pero aún no acepta). Icono de grupo de usuarios gris con un candado cerrado y un certificado.
	Sesión cerrada a la que el usuario ha pedido ser invitado. Icono de grupo de usuarios gris con un candado cerrado y un signo de pregunta.
	Grupo de sesiones a las que el usuario está suscrito. Icono de un lápiz firmando un papel en colores.
	Grupo de sesiones a las que el usuario no está suscrito. Icono de un lápiz firmando un papel en gris.