Monitoreo Redes Mesh

Documentación
Proyecto Fin de Carrera

Diego Sosa
Guillermo Sosa
Ricardo Blengio

Tutor: Eduardo Cota

Mayo 2008
Facultad de Ingeniería
Universidad de la República

Agradecimientos

A nuestras familias, por todo el apoyo que nos brindaron durante este período.

A Eduardo Cota, Fiorella Haim y al grupo de trabajo del Plan Ceibal, por la ayuda constante y por la posibilidad que nos dieron al presentarnos esta propuesta.

A la organización OLPC, en particular a Jim Gettys y a Julia Reynolds, que nos facilitaron dos XO´s simplificandonos enormemente la tarea de desarrollo.

Contenido

Acrónimos y abreviaciones

AODV - Ad-hoc On-demand Distance Vector
ASP - Active Server Pages
BB - Beacon Broadcaster
BSS - Basic Service Set
BSSID - Basic Service Set Identification
CSV - Comma-separated values
DS - Distribution Service
E2E - End-to-End
EAP - Extensible Authentication Protocol
EAPAIE - EAP Authentication Information Element
EAPMIE – EAP Message Information Element
EMSA - Efficient Mesh Security Association
EMSAIE EMSA - Handshake information element
ESS - Extended Service Set
FTP - File Transport Protocol
GIL - Global Interpreter Lock
HWMP - Hybrid Wireless Mesh Protocol
HTTP - Hypertext Transfer Protocol
IEEE - Institute of Electronics and Electrical Engineering
JSP - Java Serve Pages
KCK-KD - Key confirmation key for key distribution
KDK - Key Distribution Key
KEK-KD - Key encryption key for key distribution
LQM - Link Quality Matrix
MA - Mesh Authenticator
MA-ID - Mesh Authenticator Identifier
MANET - Mobile Ad-hoc Networks
MAP - Mesh Access Point
MDA - Mesh Deterministic Access
MDAOP - Mesh Deterministic Access Opportunity
MEKIE - Mesh encrypted key information element
MKD - Mesh Key Distributor
MKD-ID - Mesh Key Distributor Identifier
MKHSIE - Mesh key holder security information element
MKDD-ID - MKD domain Identifier
MKDDIE - MKD domain information element
MP - Mesh Point
MPP - Mesh Point collocated with a mesh Portal
MSDU - MAC Service Data Unit
NAT - Network Address Translation
OLSR - Optimized Link State Routing
PHP - PHP: Hypertext Preprocessor

PMK-MA - Mesh Authenticator PMK
PMK-MKD - Mesh Key Distributor PMK
PTK-KD - Pairwise transient key for key distribution
RA-OLSR - Radio Aware Optimized Link State Routing
RERR - Route Error
RM-AODV - Radio-Metric Ad-hoc On-demand Distance Vector
RREQ - Route Request
RREP - Route Reply
SFTP: Secure File Transport Protocol
SSID - Service Set Identifier
TBC - To be confirmed
TBD - To be determined
TTL - Time to Live
UCG - Unified Channel Graph
URL - Uniform Resource Locator
WDS - Wireless Distribution Service
XML - Extensible Markup Language

Resumen

En Diciembre de 2006 el Presidente de la República hace anuncio del Proyecto Ceibal (Conectividad Educativa de Informática Básica para el Aprendizaje en Línea) que se enmarca en otro a nivel mundial conocido como OLPC (One Laprop Per Child) y se crea la comisión de trabajo para la implementación del mismo. "Nuestro objetivo estratégico es que todos los niños tengan acceso al conocimiento informático en un marco de equidad".[1]

El cometido del mismo es el de achicar la brecha digital, entregando a cada niño y maestro del País un PC portátil teniendo cubiertas para fines de 2009 todas las escuelas del territorio nacional. Hoy en día hay un grupo de gente trabajando en el LATU (Laboratorio Tecnológico del Uruguay) de forma constante para poder cumplir con este objetivo.

Es en este contexto que se nos propuso desarrollar una aplicación de monitoreo para que el equipo de trabajo del LATU cuente con una herramienta que facilite la gestión y administración de la red de computadores. Se plantearon requerimientos, casos de uso, especificaciones y luego de sucesivas reuniones se comenzó a desarrollar la aplicación.

Se cumplió con todos los objetivos pautados al inicio de la propuesta y dentro de los tiempos estipulados. Se espera que a corto plazo el sistema esté operativo en las escuelas donde el proyecto Ceibal esté en marcha.

[1] Presidente de la República Oriental del Uruguay, durante el lanzamiento de este proyecto

Capítulo 1

Introducción

En este capítulo se presentan los objetivos del proyecto, el que, el para que, la motivación y la propuesta inicial de trabajo. Se presenta a su vez la estructura completa del documento.

1. Descripción del Proyecto

1.1. Generalidades

Este proyecto implica el estudio del estándar de comunicación inalámbrica para redes mesh (IEEE 802.11s), su grado de penetración en la tecnología utilizada por los laptops del proyecto OLPC, y en este contexto implementar una solución integral para el monitoreo de este tipo de redes.

En lo que refiere al estudio del nuevo estándar, nos focalizamos en identificar cuál es el estado del arte en la implementación de redes mesh y en el estudio del borrador del estándar y papers referentes al tema, ya que aún está en proceso de estandarización por la IEEE.

A colación del punto anterior se presenta el desafío de conocer en detalle la implementación en los laptops con los que trabajamos y los posibles puntos de desencuentro con el estándar.

Una vez adquiridos estos conocimientos, se cuenta con las armas suficientes para el diseño de un sistema de registro, almacenamiento y tratamiento estadístico de una serie de parámetros que permitirán monitorear el comportamiento de las redes mesh.

1.2. El “Que” del Proyecto

Desarrollar una herramienta de monitoreo para ayudar a la detección de posibles problemas que puedan surgir en las redes mesh conformadas por los laptops del Proyecto Ceibal.

Dicha solución consistirá en una aplicación cliente en cada máquina que registrará eventos (topología de la red, calidad y estado de enlace, paquetes enviados, recibidos, configuración de interfaces, etc), y en un servidor remoto que recibirá periódicamente dichos datos. El servidor almacenará la información en una base de datos, y los desplegará esquemáticamente en una aplicación Web.

1.3. El Para qué del Proyecto

En primer lugar, la aplicación se utilizará para facilitar y agilizar la tarea de los técnicos y administradores de la red, brindando así un soporte técnico de mejor calidad a los usuarios del sistema.

En segundo lugar, brindará información de utilidad para el análisis, estudio y toma de decisiones sobre la topología de la red.

1.4. Propuesta de Trabajo

Se presenta a continuación la propuesta inicial del sistema con los distintos módulos que en principio formarán parte de la propuesta, así como una breve descripción del alcance de cada uno.

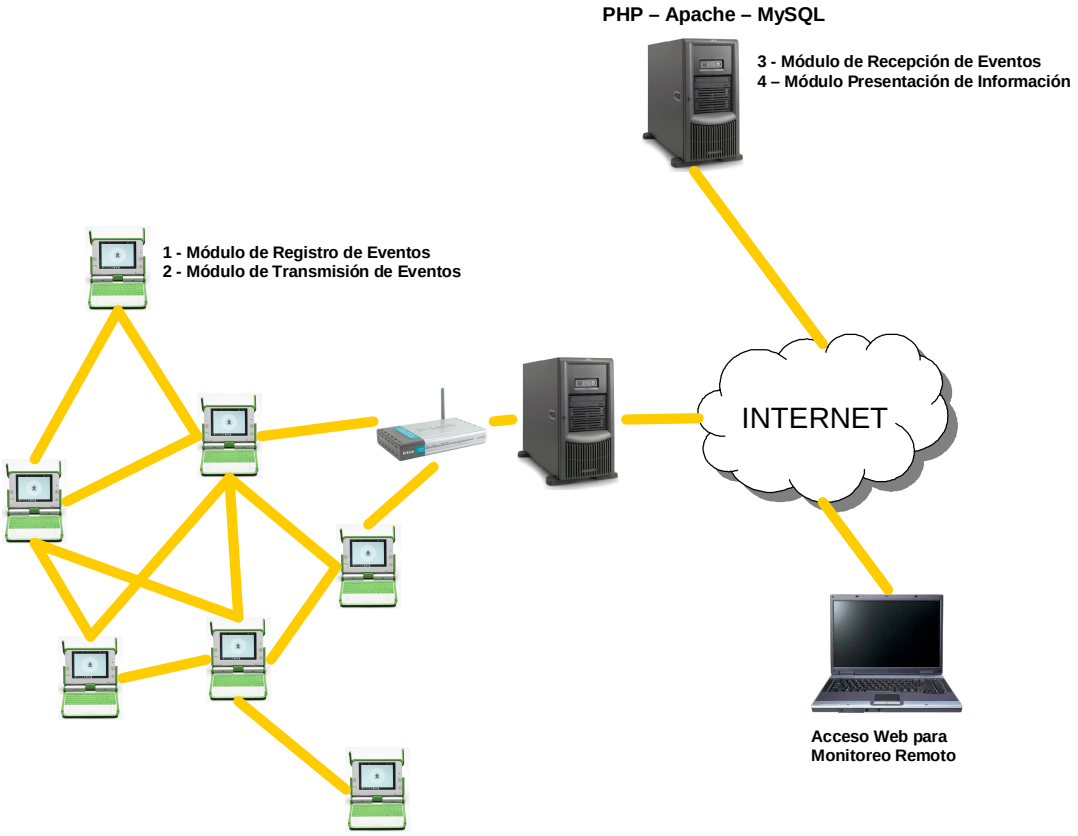


Figura 1 – Arquitectura propuesta

a) Módulo de Registro de Eventos

Este módulo se encontrará instalado en las laptops y será el encargado de registrar a nivel local todos aquellos eventos que se definan como relevantes a nivel de monitoreo. Los eventos registrados serán periódicamente enviados al servidor que los insertará en la base de datos. Existirán políticas de almacenamiento locales que definirán la capacidad máxima de registro, superados esos límites el sistema irá desechando la información más antigua.

b) Módulo de Transmisión de Eventos

La información registrada por el Módulo de Registro de Eventos será periódicamente enviada al servidor. Una vez enviada la información, este módulo depurará el registro local y establecerá las políticas para próximos registros y envíos de información las que podrán ser definidas en forma remota (por ejemplo, establecer mayor frecuencia de registro para una o un grupo de máquinas con problemas).

c) Módulo de Recepción de Eventos

Es el proceso que atenderá las solicitudes del Módulo de Transmisión de Eventos, registrará la información en una base de datos y devolverá al laptop información sobre sus políticas de registro y envío de información.

d) Módulo de Presentación de Información

Será posible consultar la información registrada por los equipos, tanto en forma particular como grupal con criterios de selección geográfica, por institución, por períodos de tiempo, etc.

1.5. Motivación

La Ingeniera Fiorella Haim (Responsable del Proyecto Ceibal), nos encomendó la grata tarea de aportar nuestro granito de arena al proyecto Ceibal, para que diseñemos e implementemos un sistema de monitoreo.

En proyectos de esta envergadura resulta indispensable contar con una herramienta de monitoreo centralizada, donde se pueda recabar la información necesaria para el mantenimiento de las numerosas redes inalámbricas que se montarán a lo largo y ancho del territorio nacional.

2. Estructura de la documentación

En el siguiente capítulo se hace una descripción del marco de trabajo y se muestran las características de las máquinas con las que se trabajó.

En el capítulo tres, se desarrolla el estudio de la propuesta del IEEE para estandarizar la normativa acerca del funcionamiento de las redes mesh (802.11s). Se hace una breve introducción a las redes inalámbricas, se compara la arquitectura clásica 802.11 con las redes mesh y se describen distintos aspectos del borrador. Dicho capítulo finaliza con una breve descripción de lo que se ha implementado en el proyecto OLPC y que diferencias presenta con el citado borrador.

El capítulo cuatro se encaran los criterios de diseño considerados a la hora de desarrollar la aplicación. Posterior a esto se describe la solución implementada, los módulos cliente y servidor, la forma de comunicación entre ellos y el modelo que se diseñó de base de datos.

En el captiulo cinco, se describe las distintas fases de prueba y puesta en producción del sistema, así como los inconvenientes y modificaciones que surgieron a lo largo de este proceso.

En el capítulo seis se hace un análisis de la gestión de tiempos, se presentan las conclusiones finales y se mencionan las posibles mejoras y trabajos a futuro.

La documentación de los programas implementados para el funcionamiento de toda la aplicación se encuentran en el Anexo I.

El manual de usuario de la aplicación Web se describe en el Anexo IV

Y finalmente en el Anexo V se describe el contenido del CD que complementa esta documentación, el cual contiene todos los manuales y software desarrollado.

Capítulo 2

Marco de Trabajo

1. Proyecto OLPC

A principios de 2005 en el Instituto tecnológico de Massachussets surgió la idea de desarrollar un laptop de bajo precio, pensando en las necesidades de los niños de pocos recursos, en regiones en vias de desarrollo del mundo. Es así que en el Estado de Delaware y con sede en Cambridge Massachusetts, EE.UU. surge la Organización sin fines de lucro OLPC, con Nicholas Negroponte director del MIT Media Lab a la cabeza siendo el principal responsable de toda la iniciativa.

La misión de esta Fundación es estimular el crecimiento de iniciativas locales comunitarias diseñadas con el fin de incrementar y hacer sustentable en el tiempo la efectividad de las laptops XO como herramientas de aprendizaje para chicos viviendo en los países menos desarrollados.

La fundación creció rápidamente incorporando personas talentosas provenientes de la enseñanza, la industria, las artes, los negocios y la comunidad open-source.
Recién en Noviembre de 2006 las primeras máquinas B1-Test (Beta 1) salen de la planta de Quanta en Shanghai siendo la primera partida en masa.

2. Características Generales de los XO

2.1. Hardware

En el diseño del XO se tomaron en cuenta varios aspectos como precio, tamaño, peso, durabilidad, resistencia y principalmente el aspecto.

La XO es aproximadamente del tamaño de un libro, liviana y flexible permitiendo al niño utilizarla como laptop estándar, como e-book, y también para juegos.

En su diseño se destacan sus puntas redondeadas, una manija pequeña diseñada para hacerle más fácil al niño el desplazamiento del equipo. El teclado es de un material plástico sintético sellado que hace imposible el filtrado de cualquier líquido o humedad.

Posee un touchpad extra-ancho y dual que puede ser usado en su modo tradicional como también al hacer presión.

El laptop cumple con todas la normas de la Directiva RoHS de la Unión Europea (Restringe el uso de materiales peligrosos en la fabricación de dispositivos eléctricos y electrónicos), no contiene materiales peligrosos y sus baterías NiMH no contienen metales pesados. Además esta pensada para obtener un mayor rendimiento de cada ciclo de recarga y tolera fuentes de recarga alternativas, tal como una batería de auto.

La experiencia muestra que los componentes de un laptop que tienen una mayor predisposición para corromperse son el disco duro y los conectores. Es por esto que los XO's vienen armados sin disco duro y con solamente dos cables de conexión internos. Para hacerlas mas robustas, se hicieron las paredes de las mismas unos milímetros mas anchas (2.0mm) de lo que establece el estándar (1.3mm).

En cuanto a la conectividad, las máquinas poseen un par de antenas con una performance mucho mas alta que la de los laptops estándar, que en su posición "cerrada" cubren los tres puertos USB y conectores de dispositivos de audio de la maquina.

La expectativa de vida de estas máquinas se estima en por lo menos cinco años, de todas formas están siendo sometidas a distintos tipos de pruebas para poder asegurar dicho tiempo.

2.2. Especificaciones técnicas

Dimensiones físicas:
Dimensiones: 242mm×228mm×32mm
Peso: 1.45KG con baterías LiFeP; 1.58KG con baterías NiMH.
Configuración: Pantalla móvil giratoria y reversible; carcasa con sistema de cierre anti-polvo y resistente a la humedad.

2.2.1. Dispositivos del núcleo

CPU: AMD Geode LX-700
Reloj de CPU: 433 MHz
Controlador gráfico: Integrado con la CPU Geode; arquitectura de memoria unificada
Memoria DRAM: 256 MiB RAM dinámica
Velocidad: Dual - DDR333 - 166 MHz
Almacenamiento: 1024 MiB SLC NAND flash, controlador flash de alta velocidad

2.2.2. Pantalla

Pantalla de cristal líquido: 19cm (7.5”) TFT modo dual
Área visible: 152.4 mm × 114.3 mm
Resolución: 1200 (H) × 900 (V) resolución (200 PPP)
Modo monocromático: Alta resolución, modo monocromático reflectivo
Modo color: Resolución estándar, sampleo quincunx, modo color transmisivo
Chip “DCON” especial, que permite el deswizzling y anti-aliasing en modo color, y permitiendo a la pantalla mantenerse activa con el procesador suspendido.

2.2.3. Periféricos integrados

Teclado: 70 teclas, recorrido de 1.2mm; ensamble de teclas con membrana de goma sellante.
Detalles de los teclados
Teclas de cursor: un área de cinco teclas de cursor; cuatro teclas de dirección más Enter
Touchpad: dual capacitancia/resistencia; soporta un modo de entrada de escritura
Audio: Dispositivos analógicos AD1888, codec de audio compatible-AC97; estereo, con dos parlantes internos; mono, con micrófono interno y usando el Analog Devices SSM2211 para amplificación de audio
Inalámbrica: Marvell Libertas 88W8388+88W8015, compatible 802.11b/g; doble antena coaxial girable y ajustable; soporte de recepción por diversidad.
Indicadores de estado: Alimentación, batería, WiFi; visible con la tapa abierta o cerrada
Cámara de video: resolución de 640×480, 30CPS (FPS)
Conectores externos:
Alimentación: entrada DC de dos contactos, 10 a 25 V, –50 a 39 V seguros, fusible descartable por excesos.
Salida audio: conector estándar estéreo de audio con 3 contactos de 3.5mm
Micrófono: conector estándar mono de micrófono con 2 contactos de 3.5mm; modo seleccionable como sensor de entrada .
Expansión: 3 conectores USB2.0 Tipo-A; ranura para tarjeta MMC/SD.
Potencia máxima: 1 A (total).

2.2.4. Batería

Tipo de empaque: 4 o 5 celdas, configuración en serie de 6V.
Cubierta “dura” totalmente sellada; removible por el usuario .
Capacidad: 22.8 Watt-horas
Tipo de celda: NiMH (o LiFeP)
Protección del empaque: Identificación de tipo de empaque integrada
Sensor térmico integrado
Limitador de corriente de fusible múltiple integrado
Ciclo de vida: Mínimo de 2,000 ciclos de carga/descarga (hasta 50% de capacidad original - IIRC)
Cargador/BIOS

2.3. Software

Las XO's tienen software libre y "open-source", permitiendo a los niños usar las máquinas en sus propios términos, brindándoles la posibilidad de modificarlas sin ningún tipo de restricciones. Junto con maestras y maestros podrán rehacer, reinventar y reutilizar software, hardware y contenidos.

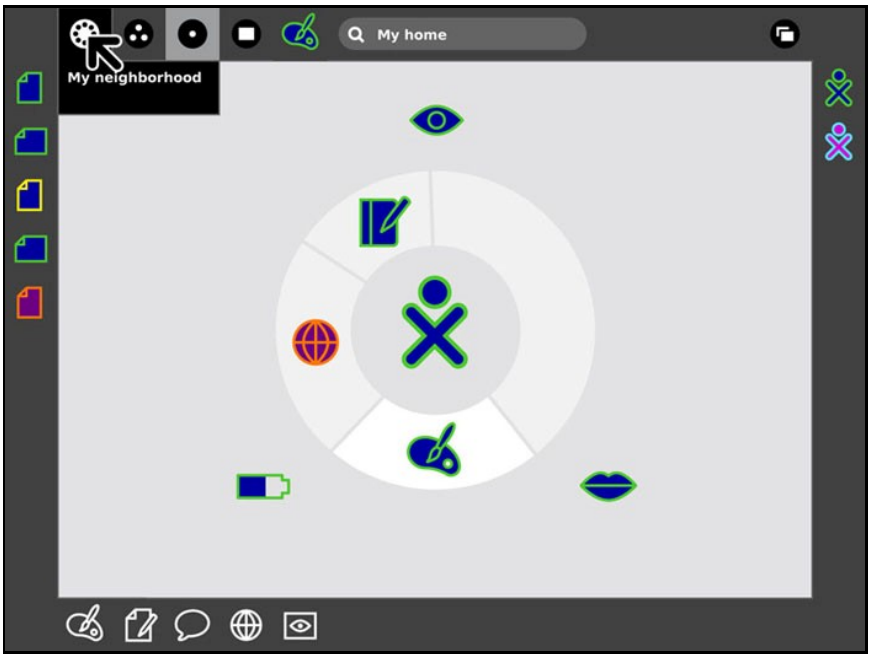
Las máquinas tienen integrado componentes del Sistema Operativo de Linux Red Hat's Fedora Core 6. Soportan distintos entornos de Programación: Python, a partir del cual se crearon las interfaces de usuario y el modelo de actividades; JavaScript, para correr scripts desde el navegador Web; Csound, entorno programable de audio y música; Squeak, una versión de Smalltalk con un ambiente muy completo para la edición y autoría usando múltiples medios; y Logo.

2.3.1. Aplicaciones

Viene instalado un explorador Web basado en Xulrunner, el entorno de ejecución del conocido explorador Firefox. Un visor de documentos basado en Evince, un procesador de texto AbiWord, un lector de RSS, un cliente de correo electrónico, un cliente de Chat, VoIP; un diario, un wiki con edición WYSIWYG; un reproductor y editor de multimedia; una herramienta de composición musical, herramientas gráficas, varios juegos, un shell y un debugger.

2.3.2. Interfaz

La interfaz de Usuario de los XO´s se diseñó acorde a las necesidades de trabajo de los niños; Aprender. Es así que, desarrolladores de Pentagram y Red Hat crearon SUGAR, una interfaz “zoom” que captura gráficamente el mundo de compañeros de estudios y maestros como colaboradores, haciendo énfasis en las conexiones dentro de la comunidad, entre las personas y sus actividades.



Capítulo 3

IEEE 802.11s

1. Redes de Área Local Inalámbricas

En los últimos años, la evolución que han tenido las redes de área local inalámbricas, ha sido increíble y muy rápida. En muy poco tiempo muchas tecnologías y estándares alrededor de las WLAN han aparecido en el mercado. Se han empezado a lanzar productos con tecnologías propietarias tratando de apoderarse del mercado de redes inalámbricas mesh, antes de que se publique la especificación final del estándar IEEE 802.11s (actualmente se está en etapa de borrador).

Las redes mesh pueden operar tanto en ambientes interiores (LAN) como exteriores, en redes tipo campus e inclusive en redes metropolitanas (MAN). Algunos fabricantes de equipos proclaman que las redes mesh podrán ofrecer en un futuro más ancho de banda a un costo más bajo comparadas con las redes celulares de tercera generación (3G).

1.1. Historia

Las redes mesh tuvieron su origen en aplicaciones militares, con el fin de brindarle a los soldados comunicaciones confiables de banda ancha en cualquier parte. Así, la confiabilidad y robustez requerida en dichos entornos se pasó a los ambientes civiles, donde las redes mesh están ocupando cada vez un lugar más importante. Uno de los requisitos principales era contar con comunicaciones de banda ancha sin tener que instalar grandes torres o antenas. Entonces, el equipo de radio de cada soldado contribuía a la formación de una red de unidades de radio que automáticamente aumentaba su cobertura y robustez conforme se unían nuevos usuarios a la misma.

1.2. ¿Qué son las redes mesh?

Una red WLAN tradicional consta de uno o más puntos de acceso (AP - Access Points) que se conectan mediante un cable UTP a un switch/hub Ethernet hacia la red cableada. De esta forma se podrían conectar más puntos de acceso para incrementar el área de cobertura de la red.

Con las redes WLAN mesh, es posible que estos puntos de acceso se puedan interconectar de forma inalámbrica, utilizando las mismas frecuencias del espectro.

Estas redes mesh son simples, todos los puntos de acceso comparten los mismos canales de frecuencia. Esto hace a los AP relativamente baratos. Los APs actúan como hubs; es decir todos los clientes contienden para acceder al mismo ancho de banda. Una desventaja de este tipo de redes es que no se puede transmitir y recibir al mismo tiempo, además se introduce un considerable retardo en cada salto. A pesar de estas desventajas, los sistemas de canal único siguen siendo populares, debido a su bajo costo. Sin embargo para lograr una mejor calidad y cobertura se necesitarán sistemas de radio con canales duales o múltiples.

La red mesh está basada en el estándar 802.11a debido a que la banda de 5 GHz está menos congestionada, habiendo menos riesgo de interferencia entre los enlaces de la mesh y los clientes. Sin embargo, el estándar 802.11 no soporta nativamente las mesh, es por esto que cada fabricante necesita implementar su propia tecnología propietaria por encima del 802.11a. El estándar 802.11s, tiene la finalidad de reemplazar estas tecnologías propietarias, tanto para sistemas de un solo canal o de varios canales de radio.

1.3. ¿Qué es el estándar IEEE 802.11s?

El estándar 802.11s es una propuesta del grupo de trabajo conocido como Wi-Mesh Alliance (www.wi-mesh.org). El borrador del estándar define la capa física y enlace de datos para redes mesh. En la figura 2 se observa esquemáticamente el impacto de la norma a nivel de capas.

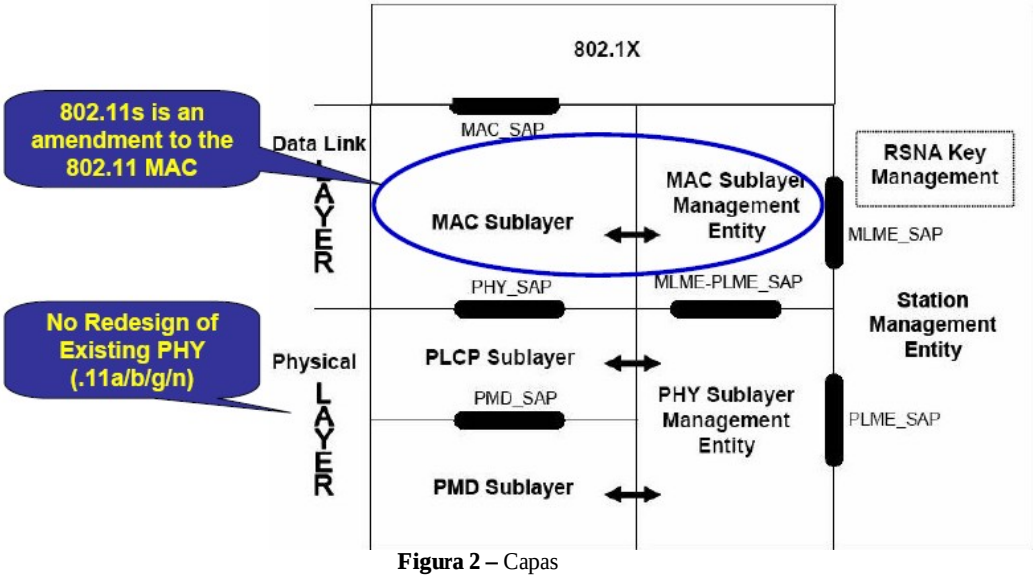


Figura 2 – Capas

Esta topología aumenta la cobertura de la red y permite estar siempre activa aún cuando uno de los AP´s falle. Para añadir capacidad, se pueden agregar usuarios y AP´s a la red. De la misma manera que la red Internet, agregar nodos la hace escalable y redundante. El estándar ofrece flexibilidad, requerida para satisfacer los requerimientos de ambientes residenciales, de oficina, campus, seguridad pública, etc.

El estándar especifica plataformas para equipos de simples y múltiples canales de radio. También se especifica en algunos ítems esquemas de priorización de calidad de servicio (802.11e), medición de recursos de radio (802.11k) y administración del espectro (802.11h). La especificación también incluye características tales como: censado adaptativo de portadora para el rehúso espacial del espectro, coordinación de canales de acceso y soluciones de administración de recursos de radio frecuencia (RF). El 802.11s también provee características de descubrimiento extendido de mallas con auto configuración automática y seguridad (802.11i).

En las redes descriptas en este documento, la selección de caminos y forwardeo de información se realiza a nivel de capa dos. Son robustas, abarcan grandes áreas, pero tienen también varias desventajas; en particular problemas de consumo de energía y temas referentes a la seguridad. No se puede asumir que a la hora de implementar una red Mesh todos los dispositivos van a funcionar con este protocolo.

2. Descripción General del Borrador

2.1. Redes mesh de área local inalámbricas (WLAN Mesh)

En la mayoría de las redes de área local inalámbricas de hoy en día, existe una clara diferencia entre los dispositivos que determinan la infraestructura de dicha red y los dispositivos que hacen uso de los recursos de la misma. En particular podemos diferenciar entre Access Points y Stations. Los primeros brindan una gran cantidad de servicios, como por ejemplo acceso a la red, autentificación, etc. En el segundo caso, nos referimos a dispositivos clientes de la red, los cuales se asocian a los distintos APs para poder tener acceso a los recursos que ofrece.

A modo de ejemplo en la figura 3 vemos una red 802.11 estándar (no mesh), donde las distintas stations (laptops) se asocian a los APs (en Gral. routers) que se encuentran dentro de sus radios de cobertura.

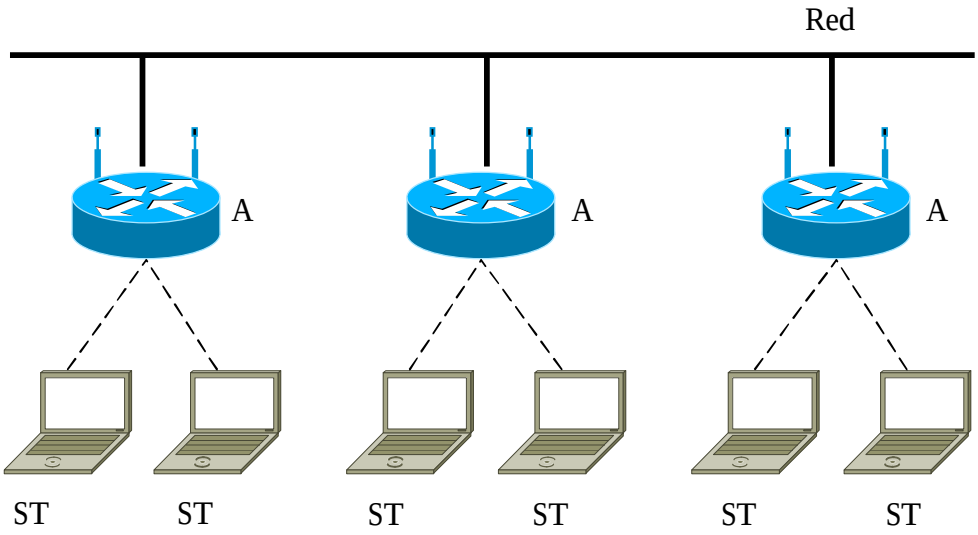


Figura 3 – Arquitectura 802.11

En cambio, en la figura 4 vemos un ejemplo de una WLAN Mesh. Aparecen los Mesh Points, entidades que tienen un papel importante en la formación y operación de la mesh. Cuando los MP tienen funcionalidades de AP, se denominan Mesh Access Point (MAP). Las stations no participan de los procesos de selección de camino, forwarding, etc.

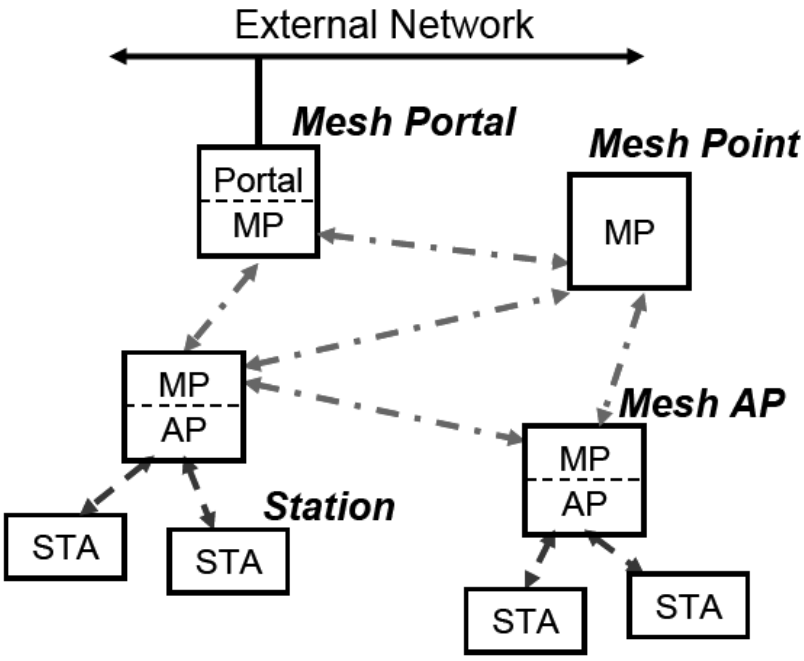


Figura 4 – WLAN Mesh

2.2. Modelo de la WLAN Mesh

Una red mesh inalámbrica de área local es equivalente desde el punto de vista de otras redes y protocolos de capas mayores a una red ethernet de broadcast. A nivel de capa de enlace todos los MP´s que forman la red se ven como si estuviesen directamente conectados.

En la figura 5 se puede ver el modelo de referencia para la interoperabilidad de la WLAN Mesh. Los Mesh Portals (MPP) son los dispositivos encargados de interconectar la red con otro tipo de redes, realizando funciones de router de borde implementando un protocolo hacia adentro y otro hacia afuera de la mesh.

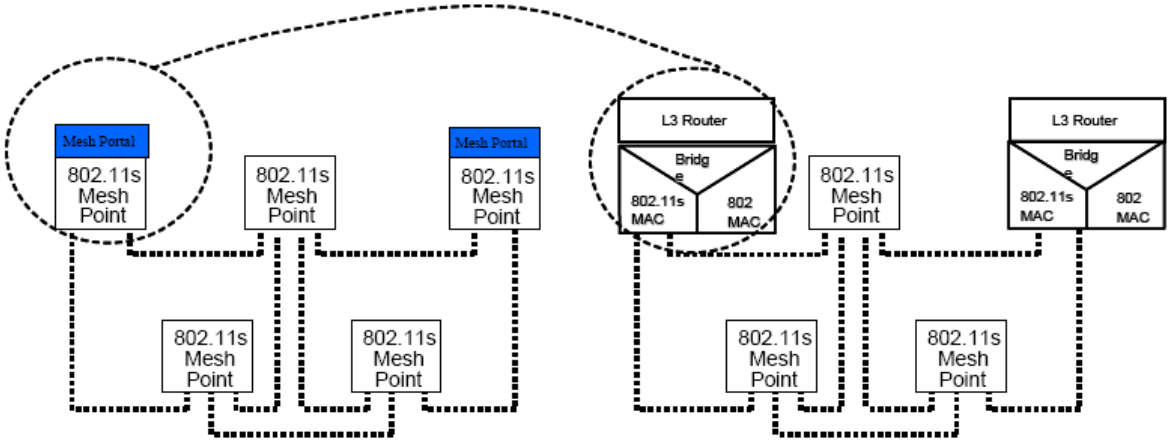


Figura 1 – Interoperabiildad

En la figura 6, el comportamiento de la WLAN Mesh a nivel de capa 2 es transparente para protocolos de capas superiores.

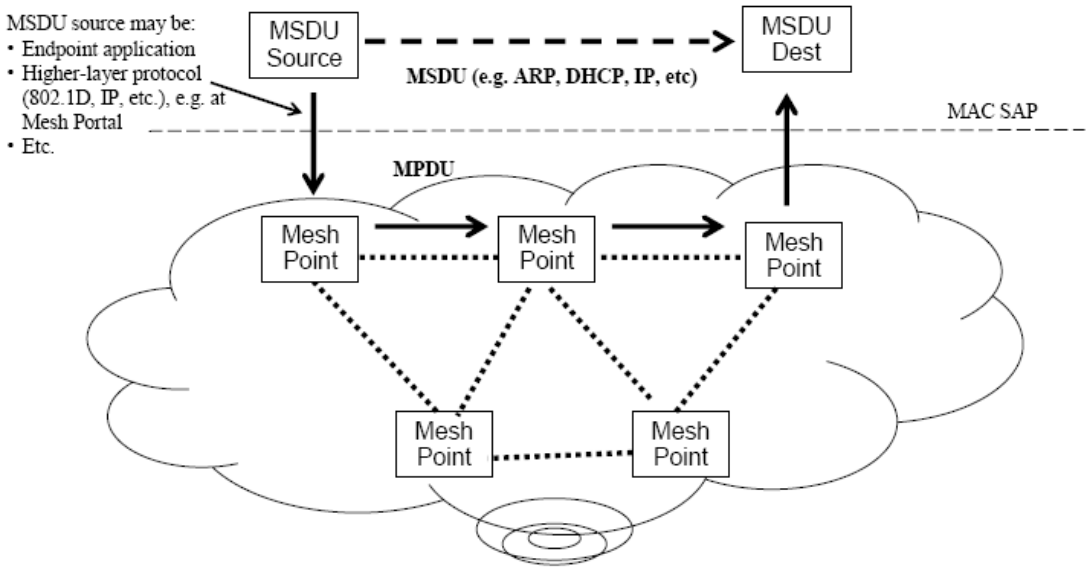


Figura 6 – Capa 2 WLAN Mesh

2.3. Descubrimiento de la Mesh y Establecimiento de Enlace entre Pares

El descubrimiento de una mesh y el establecimiento del enlace requiere que los MP tengan suficiente información acerca de sí mismos y sus potenciales vecinos. Este proceso requiere la detección de potenciales vecinos mediante el uso de balizas (beacons) o mediante un escaneo activo usando solicitudes de prueba de mesh (mesh probe requests). Se trata de un proceso continuo de monitoreo de los vecinos para detectar cambios en la conectividad.

2.3.1. Uso del Identificador de Mesh (Mesh ID)

Es usado para identificar a una red mesh establecida con propiedades conocidas y creada por una autoridad administrativa conocida. Conceptualmente, el Mesh ID tiene un propósito similar al SSID que es utilizado por las STAs para identificar APs candidatos para asociarse.

2.3.2. Perfil para Extensibilidad

Cada dispositivo de Mesh debe soportar al menos un perfil. Un perfil consiste en:

- Un Identificador de Mesh
- Un Identificador de Protocolo de Selección de Camino
- Un Identificador de Métrica de Selección de Camino

El protocolo de selección y métrica puede ser diferente para cada uno de los perfiles.

2.3.3. Descubrimiento de Vecinos

El objetivo de este proceso es descubrir los MPs vecinos y sus propiedades. Un MP tiene por definición al menos un Perfil de Mesh activo.

Un MP que no es miembro de ninguna WLAN Mesh realiza un escaneo pasivo o activo para descubrir MPs vecinos. Un dispositivo será considerado un MP vecino si y solo si:

- Un beacon es recibido desde este dispositivo
- El beacon recibido contiene un Mesh ID de al menos uno de los perfiles en el MP.
- El beacon recibido contiene un “WLAN Mesh capability element” con:
  - Un número de versión soportado.
  - Un indicador de MP activo.
  - El identificador de protocolo de selección de caminos y el identificador de métrica deben coincidir con el del perfil seleccionado.
- El beacon recibido contiene un “WLAN Mesh capability element” con disponibilidad de enlace distinta de cero.

El MP intenta descubrir todos sus vecinos y candidatos a enlace y mantiene un registro de la dirección MAC de cada uno de ellos, los parámetros de estado del enlace recientemente observados, el número de canal recibido y un estado igual a “vecino” ó “candidato a enlace”.

Si un MP no es capaz de detectar ningún MP vecino, adoptará uno de los Mesh ID de sus perfiles y se pone en estado activo.

2.3.4. Establecimiento de Enlace entre Pares de la Mesh

El propósito de este proceso es establecer al menos uno o en ocasiones varios enlaces con uno o más MPs.

Los MPs no deben transmitir ninguna trama de datos ni de control a menos aquellas que se usan para establecer el enlace, hasta que el enlace no esté establecido satisfactoriamente.

Es posible que existan más candidatos a establecer un enlace que la capacidad que tiene el dispositivo de enlaces simultáneos, en este caso deberá seleccionar los MPs con los que establecerá el enlaces en base a la calidad de la señal u otra información recibida por los candidatos.

Se presentan a continuación algunos ejemplos de cómo funciona el protocolo.

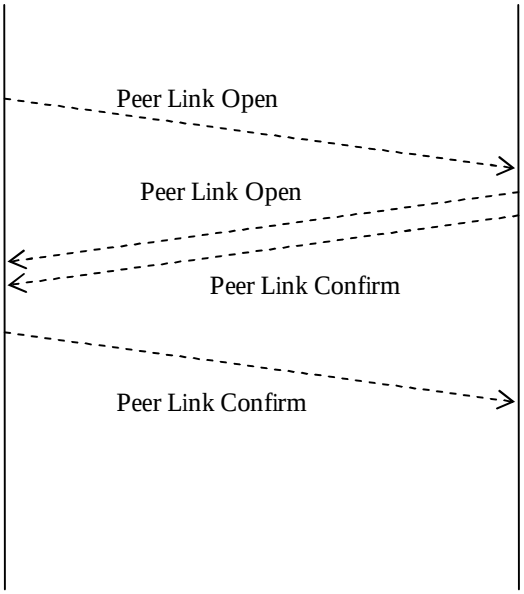


Figura 7: Enlace Satisfactorio, siempre y cuando ambos equipos reciban un Peer Link Open y un Peer Link Confirm

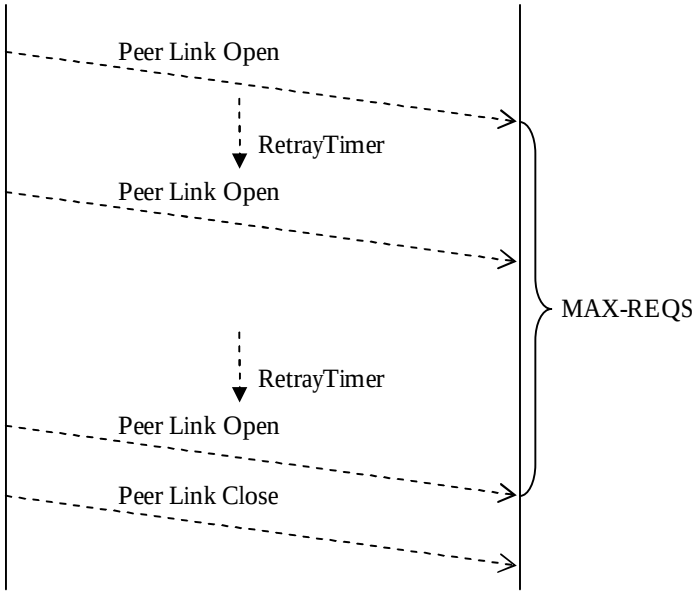


Figura 8: No se logra establecer la conexión porque el equipo que inicia el proceso nunca recibe apertura ni confirmación por parte del otro.

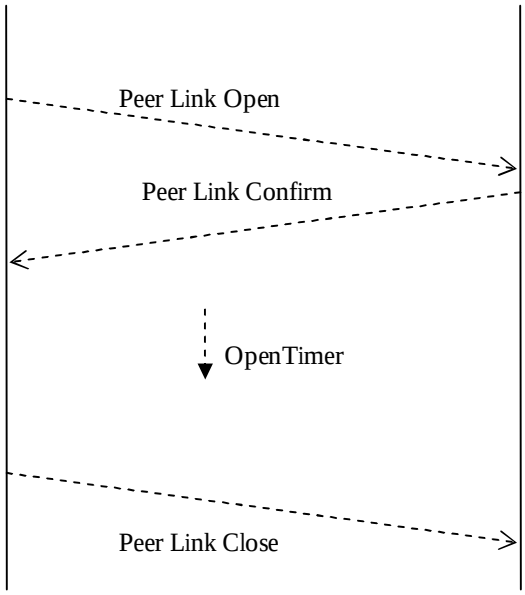


Figura 9: El equipo que inicia la comunicación recibe la confirmación pero no recibe la solicitud de apertura, por lo que después de un tiempo envía una solicitud de cerrar el enlace.

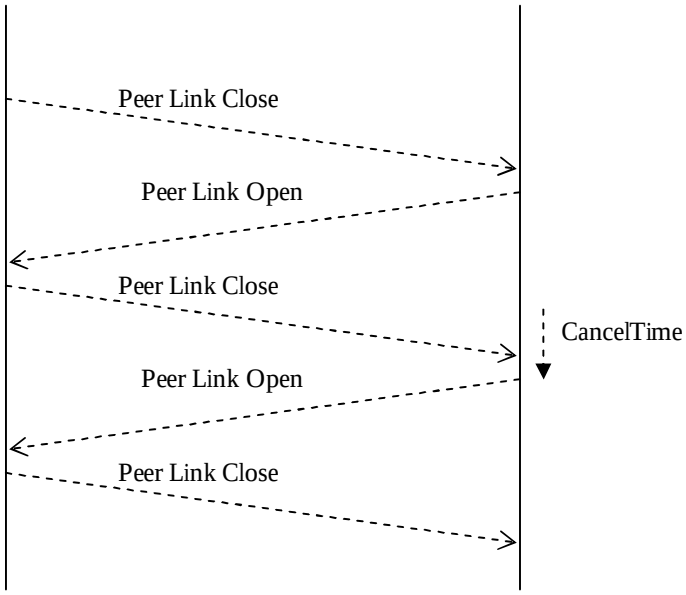


Figura 10: Cuando uno de los equipos quiere cerrar el enlace debe enviar por un tiempo determinado mensajes de cierre del enlace ante solicitudes de apertura del otro equipo.

Una vez que se establece satisfactoriamente el enlace, el MP con mayor dirección MAC es referenciado como “principal” y el MP con menor dirección MAC se denominará “subordinado” del enlace.

2.3.5. Autómata de Estado Finito

Cada enlace se maneja como una máquina de estados diferente, a continuación se describen los posibles estados, eventos y acciones posibles de la misma.

2.3.5.1. Estados

Se presentan a continuación los diferentes estados por los que se pasa en el protocolo de establecimiento del enlace. Los estados “terminales” son IDLE y ESTABLISHED, cuando el enlace se cierra ó cuando se establece satisfactoriamente.

Estado 0	IDLE	Estado donde el sistema local rehúsa cualquier intento de establecer una conexión del sistema remoto.
Estado 1	LISTEN	Estado donde el sistema local está pasivamente escuchando mensajes entrantes de Peer Link Open provenientes del equipo remoto.
Estado 2	OPEN_SENT	Estado donde el sistema local ha enviado activamente un mensaje del tipo Peer Link Open y se encuentra a la espera de los mensajes entrantes del tipo Peer Link Open y Peer Link Confirm.
Estado 3	CONFIRM_RCVD	Estado donde el sistema local recibió el mensaje Peer Link Confirm pero no recibió el Peer Link Open, por lo cual el sistema local todavía no ha enviado el Peer Link Confirm
Estado 4	CONFIRM_SENT	Estado donde el sistema local ha enviado el mensaje Peer Link Confirm luego de recibir el Peer Link Open, pero no ha recibido el Peer Link Confirm

Estado 5	ESTABLISHED	Estado donde el sistema local ha recibido los mensajes Peer Link Open y Peer Link Confirm
Estado 6	HOLDING	Estado donde el sistema local está cerrando la conexión.

2.3.5.2. Eventos y Acciones

La siguiente tabla resume los eventos y acciones presentes en el proceso de establecimiento de enlaces entre pares.

Eventos	**Acciones**
CancelPeerLink (CNCL)	Send-Open (SOP)
ActivePeerLinkOpen (ACT)	Send-Confirm (SCN)
PassivePeerLinkOpen (PAS)	Send-Close (SCL)
CloseReceived (CLR)	
OpenReceived (OPR)	
ConfirmReceived (CNR)	
Timeout (RetryTimer) (TOR)	
Timeout (OpenTimer) (TOO)	
Timeout (CancelTimer)(TOC)	

2.4. Medición de la Calidad del Enlace

Una vez que se establece el enlace con un MP vecino, es necesario establecer una medida de la calidad del enlace. Esta información es requerida por el algoritmo de selección de camino para funcionar adecuadamente.
Antes de establecerse la medida del enlace, el mismo es marcado como “Down” (caído) y recién luego de establecida la medida del enlace se marca como “Up” (levantado). Hasta que el enlace no figura como levantado, el MP no participa en el reenvío de tramas.

2.4.1. Selección del Canal

En su modo más simple, una WLAN Mesh opera sobre un solo canal.. Para una operación multi-canal, los dispositivos necesitan múltiples antenas o implementar un switcheo de canal. Los dispositivos con más de una antena sintonizan un canal en cada antena, por el contrario, los dispositivos con switcheo, switchean cada canal por un período de tiempo sobre la misma antena.

Una topología de red WLAN Mesh puede incluir MPs con una o más antenas y puede utilizar más de un canal para la comunicación entre MPs. Cuando el switcheo de canal no es soportado, cada antena en un MP opera en un canal a la vez, pero dicho canal debe cambiar a lo largo de la vida de la red mesh de acuerdo a los requerimientos de DFS2 (dynamic frequency selection)

Se muestran a continuación varios ejemplos de redes mesh indicando los enlaces por diferentes canales.

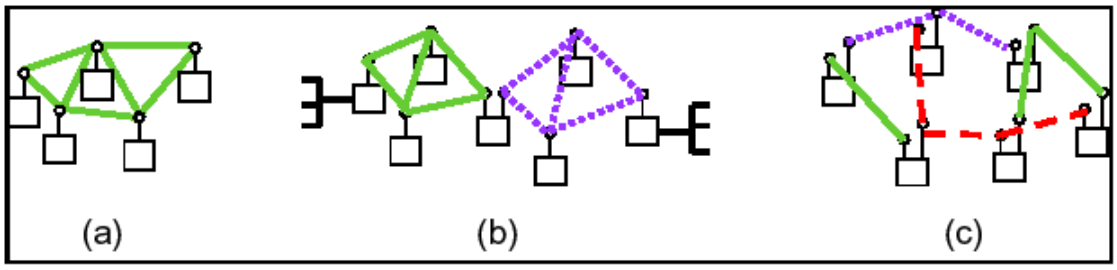


Figura 11 – Enlaces

[2] Facilidades agregadas para satisfacer determinados requerimientos en distintos dominios para la detección de radar y "spread" uniforme de canal en la banda de los 5GHz.

Un grupo de MPs que comparten el mismo canal se denomina Unified Channel Graph (UCG). Un mismo dispositivo puede formar parte de distintos UCG´s.

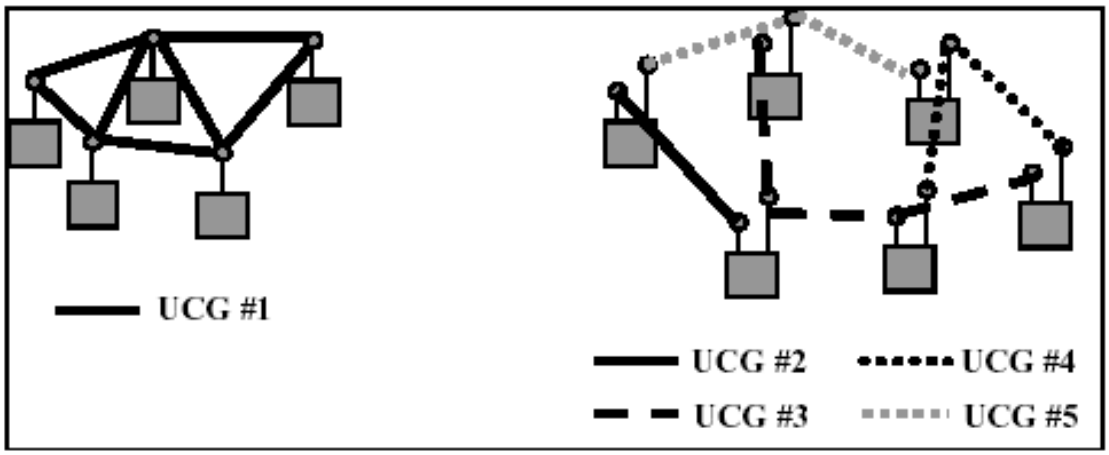


Figura 12 – Canales

2.4.1.1. Dispositivos con uno o más equipos de radio

Un dispositivo puede tener una o varias radios. Un dispositivo que puede operar en 802.11a y 802.11b, pero de forma separada se considera que tiene un único radio, aunque físicamente tenga dos. Un dispositivo de múltiples radios, utiliza una dirección MAC distinta por cada interfaz y es considerado como múltiples MPs interconectados.

2.4.1.2. Modos de selección de canal para Antenas lógicas

Un MP debe especificar el modo de selección de canal para cada antena lógica, ya sea simple o avanzado. En el modo simple, la antena lógica debe seleccionar un canal de forma controlada permitiendo la formación del UCG. La antena lógica del MP debe establecer los enlaces con los vecinos que tengan el mismo Mesh ID y perfil, y seleccionar el canal en base al mayor valor de precedencia de canal.

En el modo avanzado, la antena lógica configura el canal en base a complejas reglas de gestión (que van más allá de este trabajo).

2.4.1.3. Protocolo de Unificación de Canal Simple

Este protocolo es ejecutado en antenas lógicas de MPs configuradas en el modo simple que hacen un escaneo, pasivo o activo para descubrir MPs vecinos. En el caso de no encontrar vecinos, adopta un ID de alguno de sus perfiles, selecciona un canal para operar y un valor de precedencia para dicho canal (número en microsegundos entre booteo y un número aleatorio). En el caso de encontrar MPs vecinos, selecciona el canal del MP con mayor número de precedencia de canal. En el caso de cambiar de canal, ejecuta el protocolo de switcheo de canal.

2.4.1.4. Protocolo de Switcheo de Canal

Describe el procedimiento usado por un MP para iniciar el switcheo de un UCG a un nuevo canal, con un nuevo indicador de precedencia de canal. Dado que varios MPs en un UCG implementan este protocolo se pueden provocar conflictos; se cuenta con un timer " UCG switch wait timer" que asegura un tiempo adecuado para iniciar el proceso y evitar problemas.

El MP que determina que necesita un switcheo de UCG selecciona un UCG switch wait time y envía una trama "UCG switch announcement" a cada enlace par (activo), copiando el valor del nuevo canal candidato y el nuevo indicador de precedencia de canal y seteando el contador de switcheo de canal en el mismo valor que UCG switch wait timer.

2.4.2. Selección de caminos mesh y reenvío

Los términos "selección de caminos mesh" (mesh path selection) y reenvíos mesh (mesh forwarding) son utilizados para describir la selección de caminos de saltos simples (simple-hop) o saltos múltiples (multiple-hop) entre puntos de la mesh a nivel de la Capa de Enlace.

Los mensajes de estas redes utilizan el formato estándar 802.11 de cuatro direcciones más cierta información adicional y específica de las redes mesh.

Los servicios de selección de caminos consiste básicamente en el manejo de mensajes para descubrimiento de vecinos (neighbor discovery), medición del estado de enlace local (local link state measurement) y su mantenimiento y la identificación de un protocolo de selección de caminos activo (active path selection protocol).

2.4.3. Framework de Selección de Camino

Esta especificación incluye un marco de trabajo extensible para permitir implementaciones flexibles de protocolos y métricas de selección de caminos. Se define un protocolo y una métrica obligatorios para todas las implementaciones, para asegurar interoperabilidad entre dispositivos de diferentes fabricantes. Sin embargo, la especificación además permite a cualquier fabricante implementar cualquier protocolo o métrica para cubrir necesidades especificas.

2.4.3.1. Métricas y Protocolos de Selección de Camino

Como se comentaba en el apartado anterior, las especificaciones prevén ciertas métricas y protocolos obligatorios para asegurar la interoperabilidad entre dispositivos de distintos fabricantes, se analizarán dos protocolos de selección de caminos:

a. *Hybrid Wireless Mesh Protocol* (HWMP) (Por defecto)
b. *Radio Aware OLSR Path Selection Protocol* (Opcional)

2.4.4. Reenvío de Tramas de Datos y Gestión de la Mesh

En una WLAN mesh, una trama de datos es reenviada en los MPs intermedios en base a cuatro direcciones en el encabezado MAC. Si embargo, estas direcciones no son suficientes

para llevar la información de direccionamiento cuando se encuentran entidades que no soportan los servicios WLAN mesh, como en el caso de STAs asociadas con MAPs o estaciones externas (no 802.11) que utilizan como Proxy MPPs.

Para lograrlo, las tramas en la modalidad Dirección Extendida (Address Extensión, AE flag = 1) utilizan dos campos de dirección opcionales “Address 5” y “Address 6”.

2.4.4.1. Ordenamiento de MSDUs

En redes WLAN Mesh, las operaciones de selección de camino y reenvío están implementadas como un mecanismo de capa 2. Se presentan problemas de tramas duplicadas o desordenadas en el MP de destino.

Con el par “Dirección Origen” y “Mesh E2E Sequence Number” el MP destino es capaz de detectar tramas duplicadas o desordenadas. Las tramas repetidas deben ser descartadas, mientras que las desordenadas son almacenadas temporalmente en un buffer hasta ser reordenadas. Para evitar un excesivo retraso en las tramas almacenadas, se utiliza un timer local para no esperar en forma indefinida por una trama perdida. Cabe destacar que el ordenamiento de los paquetes lo hace solo el MP destino y que por el contrario los MPs intermedios no los ordenan.

2.4.4.2. Reenvío Unicast

Se presenta a continuación una breve descripción del reenvío unicast en los siguientes casos de MP a MP, de STA a STA y fuera de la Mesh a fuera de la Mesh también.

- **En los MP fuentes**

  Para comunicaciones de **MP a MP**, el MP fuente debe utilizar tramas de cuatro direcciones (con el flag AE = 0) donde de la dirección 1 a la 4 tienen el significado usual (Address 1 para RA, Address 2 para TA, Address 3 para DA, y Address 4 para SA).

Para comunicaciones de **STA a STA**, los STAs asociados con MAPs usan tramas de tres direcciones para enviar las tramas de datos al MAP. Si el STA que envía está autenticado y el STA destino corresponde a una dirección de la tabla de reenvíos, el MAP debe reformular la trama recibida en una trama de seis direcciones (con el flag AE = 1) donde las direcciones de origen y destino son almacenadas en Address 6 y Address 5 respectivamente, reenviándolo al enlace par que figura como siguiente salto en la tabla de reenvíos.

Para comunicaciones de **fuera de la Mesh a fuera de la Mesh,** a través de la mesh, las partes externas usan sus formatos de trama, que al menos contienen una dirección origen (S) y destino (D) del tipo de tipo 802. El MPP que hace de interfase entre la red externa y la mesh pone las direcciones MAC de D y S de las tramas entrantes en Address 5 y Address 6, pone también en Address 3 el destino de la mesh para D, deducido de las tablas de ruteo y en Address 4 su propia dirección MAC.

- **En los MP intermedios y destinos**

Al recibir tramas de cuatro direcciones, el MP las descifra y chequea su autenticación, si no es de una fuente autenticada debe descartarse. Paso seguido, chequea si la dirección de destino en Address 3 es conocida, si es desconocida puede descartarlo o iniciar el procedimiento de descubrimiento de rutas dependiendo de que protocolo de selección de camino está activo en la mesh. Si el destino en la mesh no coincide con la propia dirección del MP que recibe la trama, pero tiene una dirección conocida en la tabla de reenvíos, se decrementa el TTL, si alcanza el cero es descartado, de lo contrario es reenviado al siguiente salto que surja de la tabla de reenvíos.
Si la dirección de destino coincide con la propia dirección del MP, entonces debe chequear el flag AE, si el AE = 0 se trata de un destino final y el marco es enviado a la capa superior, si AE = 1 significa que el MP es un MAP asociado con STAs, por lo tanto debe buscar en Address 5 la dirección de la STA destino, si se corresponde con una STA asociada, la trama es convertida a tres direcciones y encolada para ser enviada a su destino final.

2.4.4.3. Reenvío Broadcast

Al recibir una trama de cuatro direcciones con la Address 1 seteada en “todos unos”, el MP la descifra y chequea la autenticación. Si no es de un MP par es descartada. Los valores de la dirección de origen y el número de secuencia Mesh E2E de la cabecera de la trama pueden ser utilizados como una firma única del mensaje. El MP chequea que el mensaje no haya sido recibido previamente por este nodo, si lo fue, deberá descartarlo, sino será retenido para su uso. El TTL es decrementado en 1, si llega a cero se descarta, de lo contrario es encolado para ser reenviado como una trama de cuatro direcciones a todos sus MPs vecinos. Si el nodo es un AP, crea la trama de tres direcciones y la reenvía a todos los STAs asociados con él.

2.5. Framework de Interconexión

Esta sección define el comportamiento de un MPP para permitir el puenteo desde una red WLAN mesh de capa 2 con otros segmentos LAN 802 de manera compatible con 802.1D.

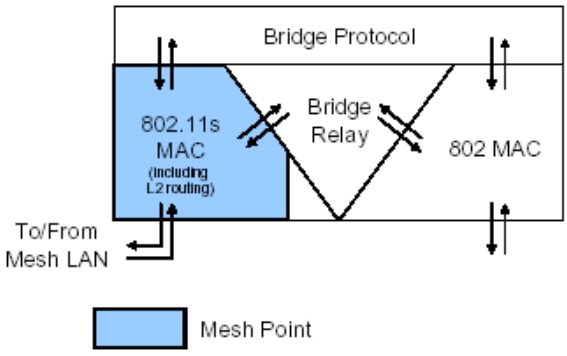


Figura 13 – Framework

La funcionalidad de puenteo se efectúa en los MPPs, que son MP con funcionalidades de puenteo basado en 802.1D. Las funcionalidades de puenteo definen el comportamiento externo (relativo a la red mesh) del MPP. Los MPPs utilizan el protocolo de aviso para informar a otros MPs de su presencia.

2.5.1. Protocolo de Aviso

- **Funcionamiento**

El elemento de información de aviso de portal “Portal Announcement IE” (PANN) es utilizado para anunciar la presencia de un MP configurado como MPP (tiene conexión con la red externa). Cuando un MP recibe un PANN no tiene porqué leer su contenido, simplemente lo retransmite como se explicará a continuación.

- **Condiciones para Generar y Enviar un PANN**

    - **Caso A (Transmisión Original)**
        - Cuando un MP es configurado como MPP
        - Periódicamente como parte del protocolo

    - **Caso B (Retransmisiones)**
        - Un MP ha recibido un PANN
        - PORTAL_PROPAGATION_DELAY ha expirado
        - El TTL decrementado del PANN es mayor o igual a uno

- **Procesamiento de un PANN**

Una vez recibido un PANN IE es sujeto a ciertos criterios de aceptación, su procesamiento depende de su contenido y la información disponible en el MP que lo recibe.

- **Criterios de Aceptación**

El PANN es descartado si el DSN (Destination Sequence Number) de dicho PANN es menor al DSN del PANN que haya sido recibido previamente de ese portal.

- **Recepción de PANN**

Sobre un PANN que no haya sido descartado el MP efectúa las siguientes acciones:

a. Debe inicializar el PANN_PROPAGATION_DELAY
b. Debe retransmitir el PANN

2.5.2. Comportamiento del MP

Cuando un MP recibe un mensaje PANN, procesa el mensaje y almacena la dirección MAC y la métrica de la ruta hasta ese MPP junto al resto de los MPPs que existan en la mesh. Si un MP tiene mensajes para enviar, primero intentará enviarlo dentro de la mesh, si no tiene éxito asume que la dirección de destino está fuera de la red y lo envía a todos los MPPs activos.

2.5.3. Comportamiento del MPP

Los MPPs deben participar de un puenteo transparente a nivel de capa 2, permitiendo a los usuarios construir redes que incluyan WLAN mesh combinadas con otras redes capa 2.
Los MPPs aprenden las direcciones de los MPs y STAs asociados a estos en la mesh.

2.5.3.1. Manejo de Mensajes de Salida

Un paquete enviado por un MP en la WLAN mesh tiene las siguientes posibilidades de destino:

- Un nodo o dispositivo adjunto que el MPP sabe que está dentro de la red En ese caso el MPP envía el paquete al MP destino.
- Un nodo que el MPP sabe que está fuera de la red. En ese caso el MPP envía el paquete a la red externa.
- Un nodo desconocido para el MPP. En este caso el MPP lo envía hacia la mesh y hacia la red externa

2.5.3.2. Manejo de Mensajes de Entrada

Un paquete recibido por un MPP de la red externa tiene dos posibilidades:

- Un nodo o dispositivo adjunto que el MPP sabe que está dentro de la red. En ese caso el MPP envía el paquete al MP destino.
- Un nodo desconocido para el MPP. En este caso el MPP tiene dos posibilidades:
    - Intenta establecer una ruta al destino
    - Hace un broadcast del paquete dentro de la mesh.

2.5.4. Procedimiento de cálculo de la métrica de enlace

El siguiente cálculo representa una medida del camino por defecto que debe ser utilizada por los protocolos de selección de camino para escoger el camino óptimo. El costo del enlace representa la cantidad del canal consumido al transmitir una trama sobre un enlace particular. Esta medida es aproximada y está diseñada para la puesta en práctica de la interoperabilidad.

$$c_a = \left[O_{ca} + O_p + \frac{B_t}{r} \right] \frac{1}{1 - e_{pt}}$$

Donde O_{ca}, O_p y B_t son constantes:

Parámetro	Valor (802.11a)	Valor(802.11b)	Descripción
O_{ca}	$75ms$	$335ms$	Overhead de acceso al canal
O_p	$110ms$	$364ms$	Overhead de protocolo
B_t	8224	8224	Nº de Bits en la trama de Test

r = tasa de bits Mbit/s y e_{pt} = tasa de error de tramas para un tamaño de trama B_t.
Los parámetros r y e_{pt} se determinan durante la fase de Descubrimiento del Estado del Enlace Local.

Se muestra a continuación un ejemplo de la función de costo unicast basado en la medición de la calidad del enlace.

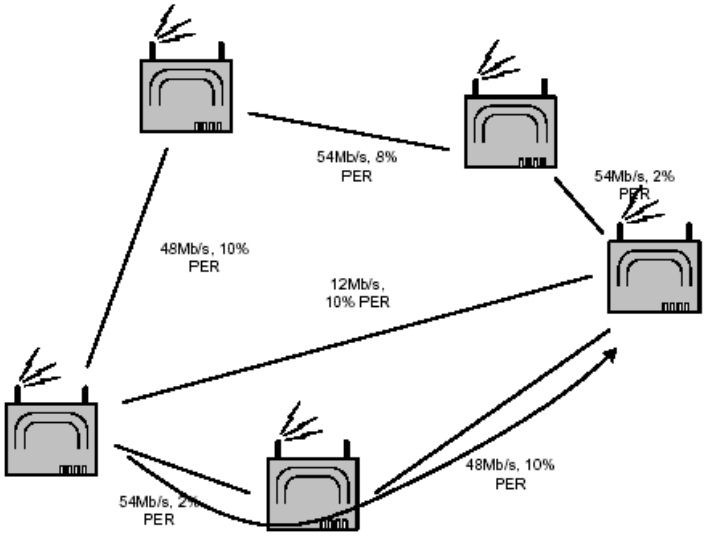


Figura 14 – Funcion de costo

2.5.5. Descubrimiento de Estado de Enlace Local

El propósito de este procedimiento es cargar los campos r y e_{pt} utilizado por la métrica por defecto para cada MP par en la tabla de vecinos.

Un enlace entre pares es considerado caído (DOWN) cuando no tiene valores asignados de r y e_{pt}; una vez que se completa el descubrimiento del enlace local y estos valores son asignados es que se considera que el enlace está activo (UP).

2.5.6. Mantenimiento del Estado del Enlace

Dados un par de MP's con un enlace establecido, el nodo principal determina la calidad del enlace y envía la información en una trama de aviso del estado del enlace local y si es transmitido con éxito, actualiza su tabla de vecinos. El nodo subordinado actualizará la información del enlace al recibir dicha trama.

2.6. Protocolo HWMP - Hybrid Wireless Mesh Protocol

HWMP es un protocolo de selección de caminos que:

- Combina la flexibilidad del ruteo "On Demand" con el ruteo proactivo a un Mesh Portal, lo que permite una eficiente y óptima selección de caminos. El ruteo "On Demand" se basa en el protocolo de ruteo AODV (Ad Hoc On Demand Distance Vector – RFC 3561).

- Soporta cualquier métrica, en particular "airtime" que se utiliza por defecto. En el campo 'metric' se propaga dicha información.

- Utiliza un mecanismo de número de secuencia para garantizar conectividad libre de loops.

- Dependiendo de como se configure se pueden tener dos modos de operación (no exclusivos):
    - Modo On Demand : Los MPs se comunican a través de rutas peer-to-peer. Se utiliza en general cuando no hay un MP configurado como root.
    - Modo Proactive Tree Building: En este modo se pueden utilizan cualquiera de los métodos RREQ o RANN.

- Sus modos de operación tienen reglas de procesamiento y mensajes en común; los mensajes de control utilizados son Route Request (RREQ), Route Reply (RREP), Route Error (RERR) y Root Announcement (RANN).

2.6.1. Modo On Demand

Para encontrar una ruta al destino, el MP origen hace un Broadcast de un RREQ con el campo 'metric' en cero y con la dirección MAC del MP destino. Cuando un MP recibe dicho RREQ se fija si ya tiene una ruta a ese MP origen, en caso de no tenerla la genera (se

van creando las reverse routes) y en caso de tenerla compara el número de secuencia del RREQ. Si el número de secuencia es mayor u ofrece una mejor métrica e igual número de secuencia, actualiza la ruta. Luego, actualiza la métrica y forwardea el RREQ haciendo otro broadcast. Cuando se crea o actualiza el camino entre MP origen y destino, el último hace un Unicast con un RREP al origen (vía las reverse routes). Cuando los MPs intermedios reciben el RREP crean una ruta al destino (forward routes) y lo envían al origen.

Los MPs intermedios mandan RREPs solamente cuando el campo "Destination Only" (DO) está seteado en cero. Por defecto DO = 1, por lo que únicamente el MP destino es el que va a contestar con un RREP. En caso de tener una ruta válida al destino y recibir un RREQ con DO = 0, el MP intermedio manda el RREP al origen informándole la ruta y además si el campo "Reply and Forward" está en uno, hace un Broadcast de un RREQ con el DO = 1 para evitar que otros MP's intermediarios sigan mandando RREP.

En otras palabras, los campos DO y RF se utilizan principalmente para que el MP origen pueda establecer lo antes posible la mejor ruta con el MP destino utilizando el RREP de un MP intermediario.

2.6.2. Modo Proactive Tree Building

En este modo hay dos métodos para inundar la información de ruteo de forma proactiva y así poder llegar al MP raíz:

- El primero utiliza los mensajes proactivos RREQ creando rutas entre todos los MPs de la mesh y el MP raíz de forma proactiva.

- El segundo se basa en los mensajes RANN en los cuales se lleva la información de como llegar al MP raíz pero a diferencia del anterior brinda la posibilidad de crear dichas rutas On demand.

Veamos en más detalle cada uno de estos métodos para entender mejor su funcionamiento:

- **Método Proactivo utilizando mensajes RREQ**

Este proceso comienza cuando el MP raíz envía un RREQ proactivo con los campos 'destination address' en uno (dirección de broadcast), DO en uno y RF en uno . Este mensaje incluye también la métrica (seteada en cero) y un numero de secuencia. Este mensaje es enviado por el MP raíz de forma periódica, aumentando en cada envío el número de secuencia.

Cuando un MP de la mesh recibe este mensaje, crea o actualiza su información de forwarding al MP raíz y actualiza también la información de métrica y saltos del RREQ reenviándolo. En el caso de que le lleguen varios RREQ, el MP va a actualizar su información sólo si el número de secuencia del RREQ es mayor al del último que le llegó, o si es el mismo pero ofrece una mejor métrica. Luego el RREQ proactivo se sigue procesando como en el modo On Demand.

En el caso que el RREQ proactivo se mande con el campo "Proactive RREP" seteado en cero, el MP que reciba dicho mensaje, en caso de ser necesario podría llegar a mandar un "gratuitous RREP" (por ejemplo, si tiene que enviarle datos a la raíz y necesita establecer una ruta bidireccional con esta). Por otro lado, si el RREQ proactivo se manda con el campo "Proactive RREP" en uno, el MP que lo reciba deberá mandar un "gratuitous RREP" ya que es el responsable de establecer la ruta desde la raíz a dicho MP. Cuando la ruta de un MP a la raíz cambia y la raíz había mandado el RREQ con el campo "proactive RREP" en uno, debería mandar un "gratuitous RREP" a la raíz con las direcciones de los MP's que han establecido una ruta a la raíz a través de si mismo.

- **Método Proactivo utilizando mensajes RANN**

La raíz inunda la red de forma periódica con mensajes RANN. La información en estos mensajes es la que se utiliza para diseminar la información de distancias a la raíz.

Cuando un MP recibe un RANN, crea o actualiza su ruta a la raíz y manda un RREQ (unicast) a la raíz a través del MP del cual le llego el RANN. Este RREQ se procesa de la misma forma que en el modo On Demand.

La raíz envía un RREP en respuesta a cada RREQ, de esta forma el RREQ determina la ruta inversa de la raíz al MP en cuestión y el RREP determina la ruta del MP hacia la raíz.

Cuando la ruta de un MP a la raíz cambia, dicho MP podría mandar un RREP con las direcciones de los MP's que han establecido una ruta a la raíz a través de este.

2.6.3. Ejemplos

A modo de ejemplo y para dejar en claro los principales conceptos del protocolo HWMP veamos algunos ejemplos de Aplicación:

2.6.3.1. Ejemplo 1

Supongamos una WLAN mesh sin MP raíz y el destino de un pedido es dentro de la mesh.

Caso: MP 4 desea hablar con el MP 9.

a. El MP 4 consulta su tabla de forwarding por algún destino al MP 9
b. De no existir un camino activo al MP 9, hace un broadcast de un RREQ para encontrar el mejor camino al MP 9
c. El MP 9 responde el pedido haciendo un unicast de una trama RREP al MP 4 estableciendo un camino bidireccional
d. MP 4 comienza a intercambiar datos con el MP 9

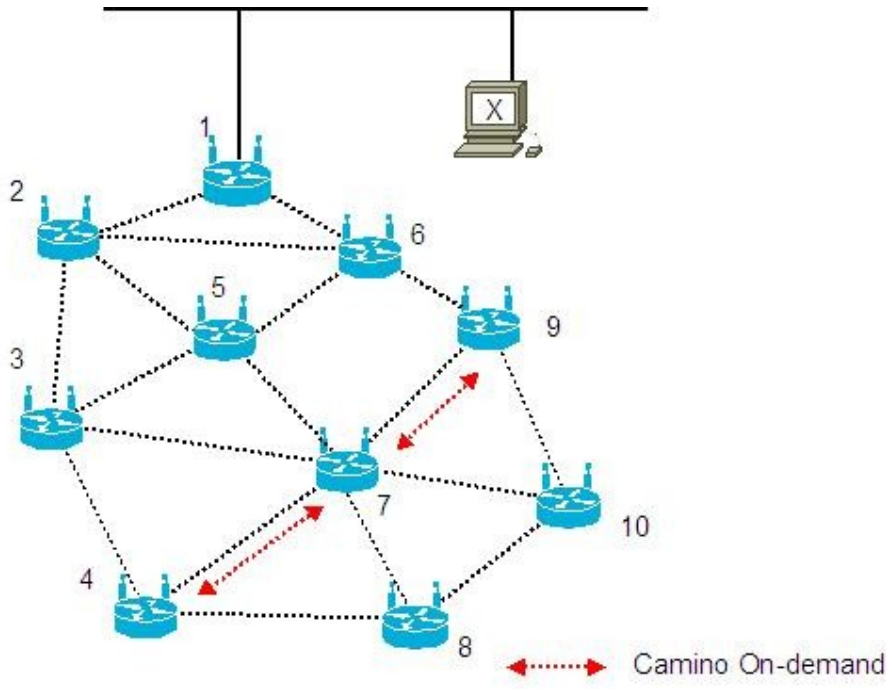


Figura 15 – Ejemplo 1

2.6.3.2. Ejemplo 2

Supongamos una WLAN mesh sin MP raíz y el destino de un pedido es fuera de la mesh.

Caso: MP 4 desea hablar con X.

a. El MP 4 consulta su tabla de forwarding por algún destino a X.
b. De no existir un camino activo a X, hace un broadcast de un RREQ para encontrar el mejor camino a X.
c. Al no obtener respuesta el MP 4 asume que el destino se encuentra fuera de la mesh y envía pedidos al portal (MP 1) para llegar a X.
d. MP 1 forwardea mensajes a otros segmentos de la LAN de acuerdo al tipo de interconexión localmente implementado.

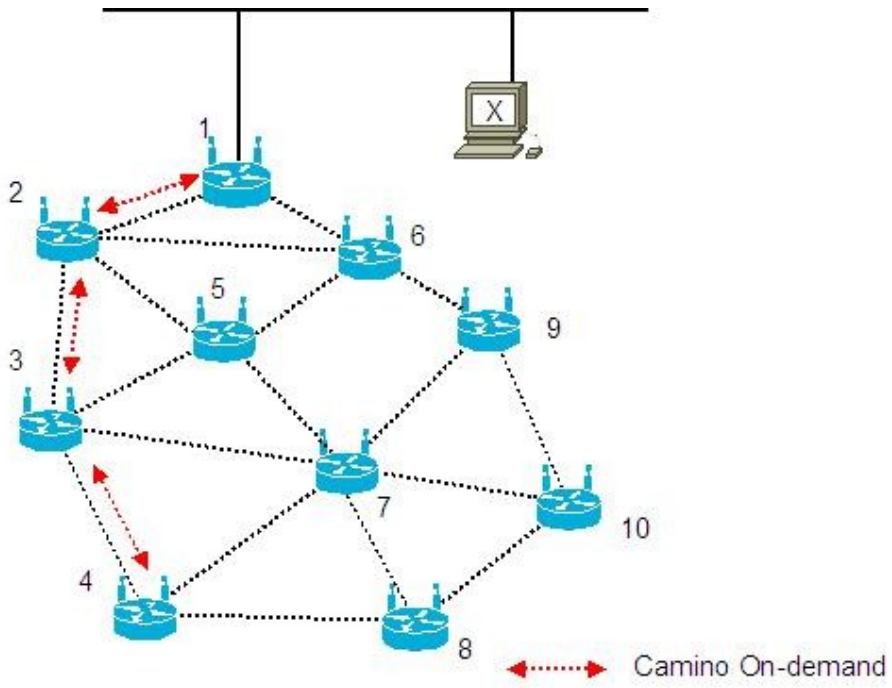


Figura 16 – Ejemplo 2

2.6.3.3. Ejemplo 3

Supongamos una WLAN mesh con un MP portal raíz y el destino para un pedido es fuera de la mesh.

Caso: MP 4 desea hablar con X.

a. El MP 4 consulta su tabla de forwarding por algún destino a X.
b. De no haber un camino activo a X, el MP 4 inmediatamente envía los mensajes a través del camino proactivo al MP raíz (1).
c. Cuando el MP 1 recibe el mensaje, si no tiene un destino a X asume que el destino es fuera de la mesh y lo forwardea a otro segmento de la LAN.

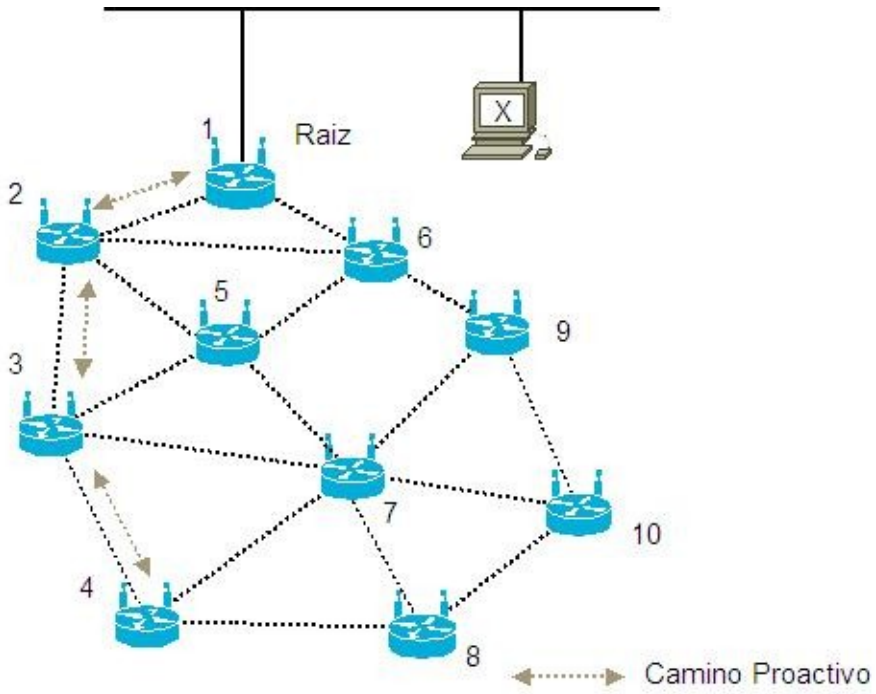


Figura 17 – Ejemplo 3

Nota: Para destinos fuera de la mesh no es necesario hacer Broadcast.

2.6.3.4. Ejemplo 4

Supongamos una WLAN mesh con un MP portal raíz y el destino para un pedido es dentro de la mesh

Caso: MP 4 desea hablar con MP 9.

a. El MP 4 consulta su tabla de forwarding por algún destino al MP 9.
b. De no haber un camino activo al MP 9, el MP 4 inmediatamente envía los mensajes a través del camino proactivo al MP raíz (1).
c. Cuando el MP 1 recibe el mensaje, lo marca como “intra - mesh” y lo forwardea al MP 9 a través del camino proactivo.
d. Cuando el MP 9 recibe el mensaje, podría llegar a hacer un RREQ on-demand al MP 4 para establecer el mejor camino MP-to-MP intra-mesh para forwardear futuros mensajes.

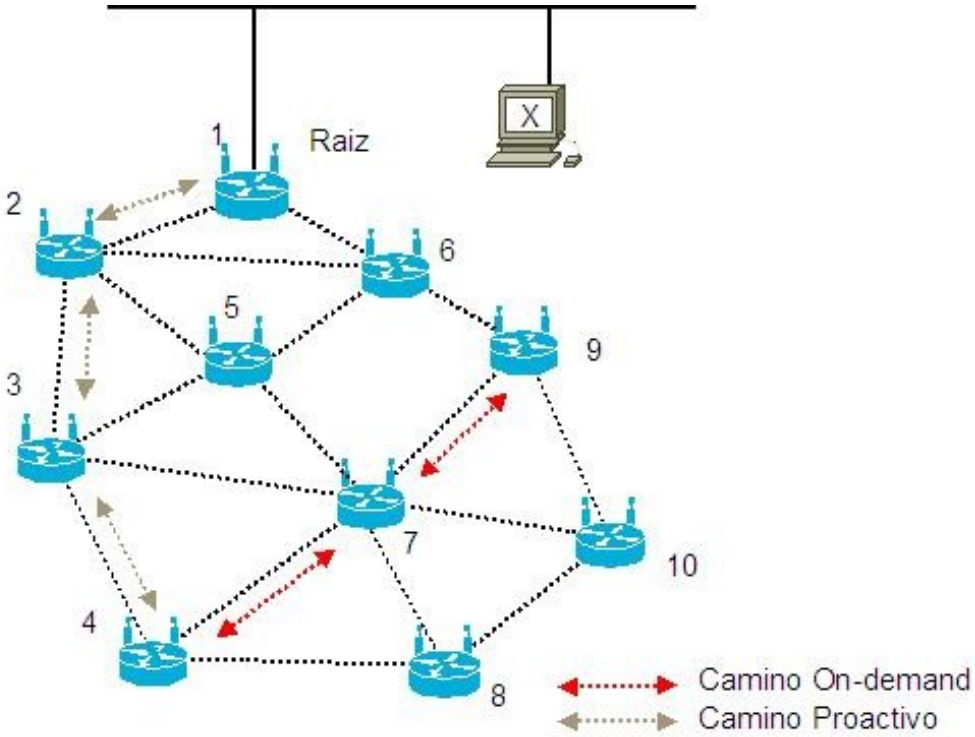


Figura 18 – Ejemplo 4

2.7. Control de Congestión

El control de acceso al medio en el protocolo 802.11 original y sus sucesores fueron diseñados para redes inalámbricas de un salto, una de la diferencias principales de la redes mesh la naturaleza de reenvío de información con múltiples saltos. Cada MP compite por el canal en forma independiente, sin cuidado de lo que esté sucediendo hacia arriba o hacia abajo.

La congestión en un MP se produce cuando éste recibe más paquetes de los que puede reenviar. El resultado de una congestión es que el buffer local se desborda y se produce una pérdida de información por paquetes eliminados.

La congestión existe tanto en redes cableadas como en redes inalámbricas, pero la degradación producida resulta mucho peor para las inalámbricas, por la naturaleza propia del canal.

En las redes cableadas el control de flujo “end to end” proporcionado por TCP (capa cuatro) resulta muy útil para controlar el efecto de la congestión, pero está demostrado que en redes mesh inalámbricas no es suficiente con el control implementado por TCP.

Para obtener un buen control de la congestión se recomienda implementar un control nodo por nodo. Este mecanismo se debe implementar en cada MP e incluye tres elementos básicos:

- Monitoreo de la Congestión
- Señalización de Control de Congestión
- Control de Tasa Local

La idea básica es que cada nodo monitorea la condición de utilización del canal local y detectar la congestión cuando suceda. Para esto se hace uso de tres nuevas tramas:

- Congestion Control Request
- Congestion Control Response
- Neighborhood Congestion Announcement

2.7.1. Monitoreo de la Congestión

Con la finalidad de controlar o evitar el congestionamiento, cada MP debe monitorear la congestión local o de vecinos y cuando lo detecta enviar mensajes de "Congestion Control Request" en forma de broadcast y/o "Neighborhood Congestion Announcement" en modo unicast.

La implementación del control de la congestión local no lo define la norma, por lo que cada fabricante podría implementarlo según sus necesidades, a continuación se presentas dos modalidades recomendadas:

2.7.1.1. Regulación de las Tasas

Se monitorea la tasa de envío y recepción y se hace el control de la diferencia entre ambas, procurando que esté siempre en el entorno de cero.

2.7.1.2. Tamaño de la Cola

Se controla el tamaño de la cola de datos, procurando que este se encuentre entre un mínimo y máximo determinado.

2.8. Administración de la energía

En los equipos portátiles de la red, que poseen baterías propias y pequeñas, es necesario administrar inteligentemente la energía (no así en los MAP fijos). Sin embargo, esta administración es especificada como opcional en el Standard, y por tanto puede haber

pares que no la soporten, especificándose esto en la negociación de los enlaces (que incluso podrán ser rehusados por este motivo). Por lo tanto todos los pares conectados a un MP en modo de ahorro de energía deberán soportar dicha funcionalidad, y a su vez el modo de ahorro de energía está coordinado con el descubrimiento de vecinos y los descubrimientos de rutas.

Se evitará transmitir innecesariamente, por ejemplo utilizando buffers para enviar más eficientemente, y se realizará una disminución en la frecuencia de envío de los Beacon, y se indicará en los mismos que se están usando las funciones de ahorro de energía.
También se definen mecanismos para que los pares estén "despiertos" sólo el tiempo necesario para la recepción, mediante bits que indican si a los enviados le siguen más datos o no.

2.9. Seguridad

Es necesario implementar funciones que permitan el transporte seguro de datos; es decir: evitar que dispositivos no autorizados envíen o reciban datos desde o hacia la mesh, tanto en tráfico unicast como en tráfico broadcast. La primera aproximación a este objetivo es re-utilizar la ya existente y muy expandida 802.11i. Esta implementación provee una base para la distribución segura de claves (PMK).

Se agrega una negociación inicial de roles en las comunicaciones entre pares, para determinar los parámetros de seguridad.

2.9.1. Introducción a EMSA

Los servicios EMSA (Efficient mesh security association) se utilizan para permitir el establecimiento de enlaces de seguridad entre dos MP en una red mesh, y soporta tanto entornos centralizados como distribuidos. Se especifica una jerarquía de claves derivadas que es establecida mediante el uso de PSK o cuando un MP realiza autentificación IEEE802.1x

La operación de EMSA se basa en *key holders*, que son funciones implementadas en los MP de la mesh. Existen dos tipos de *key holder:* MA (Mesh Authenticators) y MKD (Mesh Key Distributors). Un MP puede implementar uno, otro, o ambos tipos.

Si un MP implementa MA pero no implementa MKD, EMSA provee mecanismos para la comunicación segura entre *key holders*. Están definidos también protocolos de transporte seguro de claves y un protocolo EAP de transporte de mensajes.

Los MAs y los MKDs, administran la jerarquía de claves de la red realizando derivación y distribución segura de claves. Un *dominio MKD* queda definido por la presencia de un solo MKD; en el dominio pueden existir múltiples MAs y cada uno mantiene una asociación de seguridad y una ruta al MKD del dominio. El MKD "deriva" las claves para crear la jerarquía, y distribuye las claves a los Mas.

La jerarquía de claves permite a un MP crear asociaciones seguras con pares MP sin la necesidad de realizar cada vez una autentificación IEEE 802.1X. Un MA recibe claves derivadas desde el MKD, y deriva claves adicionales para asegurar enlaces con un MP que lo solicite.

El diseño de EMSA asume que existe un canal confiable entre AS y MKD, que puede ser usado para intercambiar claves criptográficas sin exposición a intermediarios.

La jerarquía de claves en la mesh consiste de dos ramas: seguridad de los enlaces (en tres niveles), y seguridad de la distribución de claves (permitiendo a un MP convertirse en MA, y asegurando comunicaciones entre MA y MKD).

3. Implementación en laptops de OLPC

Los laptops del proyecto OLPC implementan solo una parte de los drafts de 802.11s, veremos en esta sección los puntos más importantes y las principales diferencias entre ambos.

3.1. Selección de caminos

En el proyecto OLPC, el mecanismo implementado para la selección de caminos se basa en una versión simplificada de HWMP. Es más, el modo Proactivo no es soportado por los XO´s. Los caminos son creados a partir del modo 'On Demand', ampliamente basado en el AODV (Ad-Hoc On Demand Distance Vector).

3.2. Comportamiento frente a caída de rutas

Si una trama no puede ser enviada al próximo salto, la ruta por la que fue enviada se marca como inválida. Si dicha ruta tiene predecesores, se mandan mensajes RERR a la fuente de la ruta. Esto mejora el tiempo de recuperación luego de que un nodo de la red queda fuera del área de cobertura.

3.3. Broadcast Limitado

El método de RREQ/RREP funciona solamente con tráfico unicast. Para hacer Broadcast, se hace inundación de mensajes pero de forma limitada. Cada trama de datos enviada tiene un único número de secuencia que va de punta a punta y es seteado en el origen. Los nodos intermediarios tienen tablas donde las tramas recientemente enviadas están indexadas con este número de secuencia y la dirección del nodo que las envió. Estas tablas sirven para evitar retransmisiones innecesarias.

3.4. Algoritmos de Asociación en la Mesh

Bajo el protocolo HWMP, un MP debería hacer un escaneo activo o pasivo para descubrir a sus vecinos y establecer dichos enlaces. En el proyecto OLPC esto no es así, los vecinos son descubiertos vía el ciclo RREQ/RREP.

Al encender un laptop XO, dicho aparato se va a asociar a una red mesh de la siguiente forma:

- Hace un pedido DHCP seguido de un RREQ a una dirección ANYCAST de un Mesh Portal y espera por RREP´s, todo esto en los tres canales propuestos para el OLPC (1,6 y 11)
- Si cualquiera de los canales ofrece una mínima cantidad de saltos al Mesh Portal es seleccionado, a menos que la potencia de la señal sea muy baja.
- Si todos los canales ofrecen la misma cantidad de saltos a un Mesh Portal, se elije uno de ellos de forma aleatoria.

3.5. Seguridad

Como no se hace ningún tipo de autentificación, la mesh no está protegida contra los distintos tipos de ataques. No está bien definido todavía en el proyecto OLPC si dicha autentificación puede ser viable de implementar.

Capítulo 4

Desarrollo del Sistema

En el presente capítulo se presenta el detalle completo de las etapas de análisis, diseño e implementación del sistema, abarcando desde la aplicación cliente que se ejecuta en cada laptop, los procesos del servidor que reciben la información y la procesan para ser cargadas en una base de datos y la interfaz gráfica de usuario.

1. Arquitectura

La premisa fundamental para el diseño del sistema fue la de permitir un monitoreo remoto de las redes y la posibilidad de mantener un histórico de la información registrada. Para esto se hizo imprescindible contar con una aplicación cliente que implemente los diferentes procesos de recolección de información en forma periódica y los transmita a un servidor remoto. Del lado del servidor se implementaron los procesos que reciben la información y la cargan en una base de datos relacional. Por último una interfaz web permitirá a los usuarios parametrizar el comportamiento del cliente, configurar los servidores, mantener las principales entidades del sistema y consultar la información recolectada en forma estadística o detallada.

1.1.Requerimientos

Luego del estudio del funcionamiento de las redes mesh y de las prestaciones de los laptops se consideraron en una primera instancia determinados parámetros a registrar. Posteriormente fueron presentados al cliente donde fueron discutidos y se plantearon nuevas alternativas. A partir de las distintas reuniones y pruebas de los sistemas se llegó a una lista definitiva de parámetros:

- Redes Disponibles: MAC de AP´s, ESSID, frecuencia, Modo (Mangaed, Ad-Hoc, etc), calidad de señal, SNR y encripatción.
- Pares de un laptop en la red; MAC, saltos al vecino y SNR
- Valores de distintos parámetros de las rutas que tiene cada máquina a determinados destinos.
- Datos de la configuración a nivel de capa tres; dirección IP de cada interfaz, dirección de máscara de Subred, dirección de Broadcast, etc.
- Información que pasa a través de las máquinas; paquetes y bytes enviados , recibidos y forwardeados, así como también calidad de señal, SNR, número de reintentos, etc. Hay que agregar que se distingue si los paquetes son TCP, ICMP o UDP.
- Presencia de Portales; Dirección MAC de MP´s que se encuentran configurados como Portales y se encuentran al alcance de un determinado MP.
- Parámetros del Sistema; Versión del Kernel de Linux, versión de Firmware, versión de Sugar instalada, procesos corriendo, activos, memoria disponible, nivel de batería, etc.

Más adelante en el documento se describen cada uno de los parámetros.

1.2. Generalidades

En la figura 19 se presenta un diagrama de la solución completa, donde se pueden apreciar claramente los procesos que corren en forma permanente en los laptops y aquellos que sirven desde un servidor remoto, tanto a los procesos locales como a la interfaz de usuario.

El monitoreo en los laptops es continuo, incluso si estos no tienen comunicación con el servidor. Esto se logra gracias a un proceso periódico, independiente del proceso de envío, que registra la información en archivos planos en forma local. Luego el proceso de envío mediante el protocolo FTP transfiere los archivos al servidor e invoca al proceso remoto que carga la información a la base de datos. En paralelo a estos dos procesos, corre el proceso de actualización de parámetros, el que periódicamente invoca a un proceso del servidor que le actualiza la configuración para los procesos de registro y envío. En esta parametrización se “prenden” ó “apagan” las diferentes consultas implementadas en el proceso de registro, se establece el tiempo entre muestras, se parametrizan los detalles del servidor contra el cual se enviará la información y el tiempo entre envíos.

Como se mencionó anteriormente, del lado del servidor existen dos rutinas que dan respuesta a los procesos locales. Una de ellas recibe la orden de procesar los archivos enviados por FTP, descomprime los archivos identificados con la dirección MAC del laptop que lo invoca y procesa cada uno de los archivos, realizando las actualizaciones en la base de datos. La segunda de las rutinas, al ser invocada por un laptop, chequea si se realizaron modificaciones en su configuración y de ser así la envía para actualizar su configuración local.

Ningún sistema de monitoreo cumpliría su cometido sin una buena Interfaz Gráfica de Usuario (GUI), es por ello que se implementaron una serie de páginas web dinámicas donde se podrá consultar los datos del monitoreo, hacer el mantenimiento de las principales entidades (escuelas, responsables técnicos, laptops, etc.) y configurar el comportamiento de los procesos locales de cada una de las laptops.

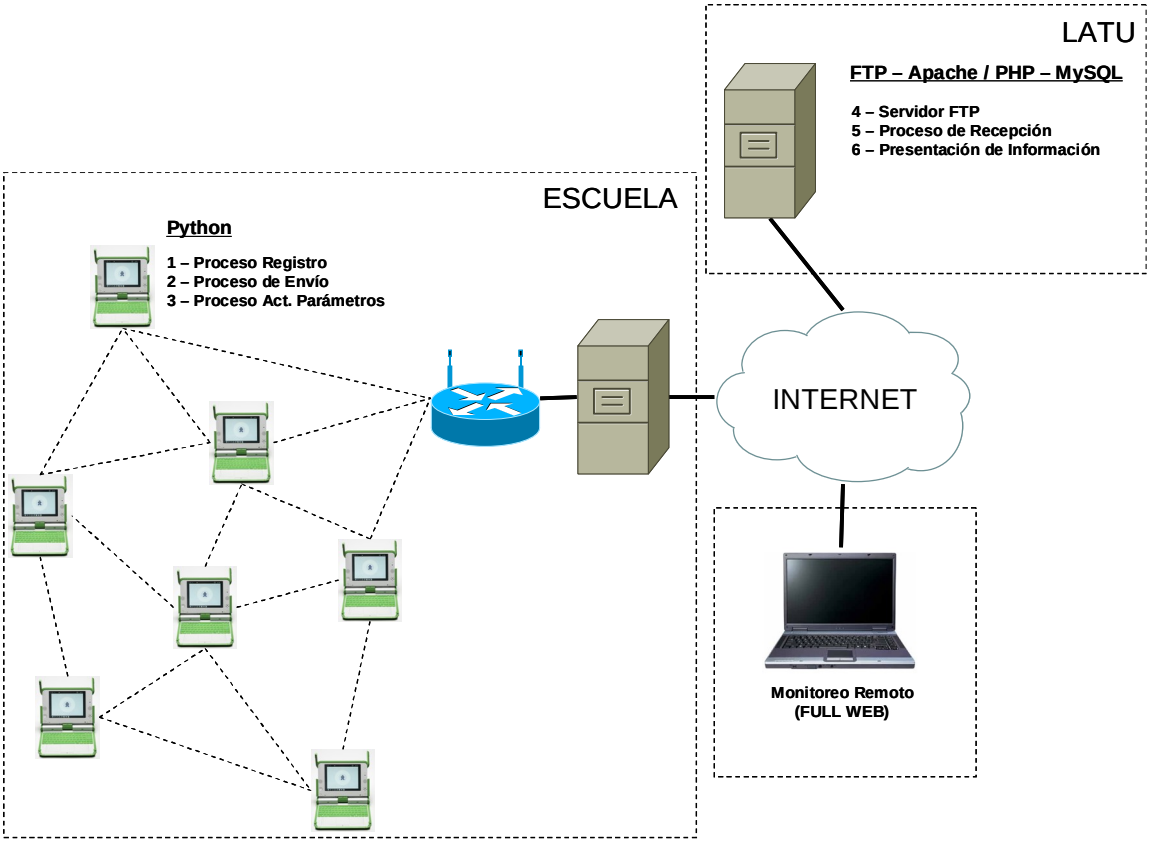


Figura 19 – Arquitectura completa

2. Criterios de Diseño

En esta sección se presenta el análisis previo al desarrollo de los distintos módulos de la aplicación en lo que refiere a distintos aspectos de lenguajes de programación y protocolos. Evaluando características como performances de lenguajes de programación disponibles, reutlización de software, nivel de programación, facilidad de entendimiento y consumo de recursos.

2.1. Modulo Cliente

2.1.1. Lenguajes de Programación

Desde el punto de vista del módulo cliente se analizaron las aplicaciones ya instaladas en los XO's resultando en que los lenguajes de programación utilizados son principalmente dos: Python y C.

Si bien programando en C se pueden obtener aplicaciones de bajo nivel, que consumen pocos recursos, cuyo desempeño en tiempo de ejecución es óptimo, no es un lenguaje fácil de utilizar por ende el tiempo de aprendizaje del mismo demandaría un tiempo extra no contemplado en el cronograma.

Por el otro lado está Python, un lenguaje algo más intuitivo que "C" pero no tan eficiente en lo que refiere a recursos. No es un lenguaje de tan bajo nivel, pero brinda la posibilidad de integrarle rutinas de "C" pudiendo desarrollar aplicaciones de más bajo nivel. Este lenguaje es interpretado, característica que facilita casos de pruebas intermedias. Este lenguaje es interpretado, característica que además de facilitar las pruebas intermedias, brinda la posibilidad de desarrollar fácilmente programas multiplataforma.

Tomando en cuenta todos los aspectos anteriores, y que la mayoría de las aplicaciones de las máquinas del proyecto están desarrolladas en Python, se consideró la segunda opción como la más viable, cubriendo las necesidades técnicas del caso.

2.1.2. Threads de Ejecución

Los threads son un concepto similar a los procesos: también se trata de código en ejecución. Sin embargo los threads se ejecutan dentro de un proceso, y los threads del proceso comparten recursos entre si, como la memoria por ejemplo. A veces se les llama "procesos ligeros" o "contextos de ejecución". Típicamente, cada thread controla un único aspecto dentro del programa, lo cual se realiza mediante el uso del "time-sharing", apareciendo como simultáneos hacia el usuario. El sistema operativo necesita menos recursos para crear y gestionar los threads y además, dado que comparten el mismo espacio de memoria global, es sencillo intercambiar información entre ellos: cualquier variable global que tenga el programa es vista por todos los threads.

El objetivo de su uso es optimizar el uso del procesador y los tiempos de ejecución, y es de gran utilidad cuando se presentan mezcladas actividades de distinta índole, que insumen tiempos netamente distintos. Por ejemplo, el acceso a disco suele ser varios órdenes más lento que el procesamiento de datos; o las consultas web de cualquier tipo suelen ser relativamente lentas. Por lo tanto, no es deseable que la ejecución del programa se bloquee esperando estos eventos. Actualmente, la mayoría de los desarrollos incluyen esta característica, desde servidores Web a simples programas desktop con interfaz gráfica.

La ejecución de los threads en Python está controlada por el GIL (Global Interpreter Lock) de forma que sólo un thread puede ejecutarse a la vez, independientemente del número de procesadores con el que cuente la máquina. Esto posibilita que el escribir extensiones en C para Python sea mucho más sencillo, pero tiene la desventaja de limitar mucho el rendimiento, por lo que a pesar de todo, en Python, en ocasiones puede interesar más utilizar procesos que threads, que no sufren de esta limitación.

Ciertos cuidados deben ser tomados al acceder a recursos compartidos desde threads distintos, ya sean accesos a discos o a memoria, para no generar "colisiones"; a tales efectos se encuentran en los lenguajes que soportan threading procedimientos de "bloqueo".

Teniendo todo esto en cuenta, se desarrolló el cliente Python utilizando threads para las distintas tareas, además de threads del tipo "timer" para la programación de las tareas

periódicas. El lenguaje provee ciertas bibliotecas a tales efectos, basadas en el sistema de threading de C, pero notoriamente más fácil de utilizar. Sin embargo, posee ciertas limitaciones al respecto, por ejemplo: instrucciones que sean ejecutadas como una subrutina C, no podrán ser "sub-divididas" y por lo tanto el programa aguardará siempre hasta la finalización de la misma antes de ceder el uso del procesador a otro thread.

2.1.3. Almacenamiento de Datos

Dado que las máquinas no dispondrán siempre de una conexión remota se hace imprescindible guardar la información registrada en cada uno de los laptops. Tomando en cuenta esto y la capacidad de almacenamiento de los XO's se vieron las distintas alternativas a la hora de crear los archivos de registros y en el caso de existir varios cómo almacenarlos.

Se presentan un par de alternativas para ésta tarea, las más clásicas: Archivos XML (Extensible Markup Language) y CSV (Comma separated values).

2.1.3.1. Extensible Markup Language

XML es un lenguaje muy robusto diseñado principalmente para almacenar y describir datos brindando la posibilidad de posteriormente leerlos desde casi cualquier otra plataforma. Es estructurado, auto contenido y portable. Además ya existen implementadas una gran cantidad de herramientas de parseo de este tipo de archivos.

Por otro lado, las etiquetas que forman parte de su estructura, como son los nombres y descripciones de parámetros generan un gran "overhead" lo que implica que a la hora de parsear una gran cantidad de datos el tiempo que demora en procesar dicha información en el contexto en el que estamos trabajando comienza a ser considerable.

Un ejemplo de archivo de registro de la aplicación utilizando XML seria el siguiente:

```
<!--- Ejemplo de Registro en XML --->

<REGISTROS FECHA X>

     <XO AA:BB:CC:DD:EE:FF>

          <REDES DISPONIBLES>
               <ESSID> HOME </ESSID>
               <NIVEL> 75% </NIVEL>
               <CANAL> 11 </CANAL>
               <ENCRIPTADA> SI </ENCRIPTADA>
          </REDES DISPONIBLES >

          <PORTALES>
               <PORTAL1>
               <MAC>Dirección MAC del Portal 1</MAC>
               <SALTOS>Saltos al Portal 1</SALTOS>
               </PORTAL1>
               <PORTAL2>
               <MAC> Dirección MAC del Portal 2</MAC>
               <SALTOS>Saltos al Portal 2</SALTOS>
               </PORTAL2>
          < /PORTALES >
.
.
     </XO AA:BB:CC:DD:EE:FF>

 </REGISTROS FECHA X>
```

En el ejemplo se observa claramente que para transmitir relativamente poca información "útil", es necesario enviar una cantidad mucho mayor de caracteres en etiquetas y descripciones

2.1.3.2. Comma Separated Values

Este es un formato de archivo simple utilizado en todo tipo de implementaciones. A la hora de almacenar datos tienen un tamaño óptimo donde cada parámetro está diferenciado de otro con sólo un separador; el más común es la coma, pero puede ser utilizado cualquier otro. Cabe agregar que el tiempo de procesamiento es óptimo para este tipo de archivos.

Sin embargo, a diferencia de XML el parsing es necesario implementarlo y además al no estar completamente estandarizado puede llegar a generar ambigüedades.

2.1.4. Decisión

En base a estas opciones, y considerando la capacidad de las máquinas tanto de almacenamiento de datos como de procesamiento y sin olvidar tampoco que los datos serán enviados por la red, el formato más óptimo en este contexto es CSV. Además se tomó en cuenta que la aplicación está desarrollada en Python y el soporte para XML en este lenguaje se encuentra en sus primeras etapas.

Con esta decisión, asumimos la desventaja de tener que implementar el parsing para este caso en particular, haciendo más "rígido" el sistema y perdiendo reusabilidad, a cambio de ganar en tiempo de procesamiento y menor overhead en los datos guardados y enviados por la red.

2.2. Módulo Servidor

2.2.1. Lenguajes en el servidor

Se debió buscar un lenguaje apropiado para utilizar en el servidor para la interacción del módulo cliente y de los usuarios con el mismo.

Los lenguajes script son aquellos que no necesitan compilarse para su ejecución. En el desarrollo web se realizan decenas de pruebas al día de manera que sería sumamente engorroso tener que compilar el código previamente, por lo cual la utilización de lenguajes script acelera notoriamente el proceso de desarrollo.

Existen diversas opciones para esta tarea, destacándose: Perl, ASP, JSP, Python y PHP. A continuación se presentan las ventajas y desventajas de los mismos.

2.2.1.1. Perl

Perl fue el lenguaje por excelencia para desarrollo de aplicaciones web desde su creación hace más de 10 años, y se ha mantenido su desarrollo en este tiempo de manera que aún sigue siendo una muy buena opción. El problema principal que presentaba era la sobrecarga de procesamiento en el servidor por su forma de implementar los procesos. Posteriormente se desarrollo Apache-Perl, que lo fusiona con el popular servidor web, levantando el problema del alto número de inicios de procesos de Perl.

Es uno de los lenguajes más versátiles y poderosos, con una enorme cantidad de librerías y una vasta documentación. Sin embargo, es un lenguaje muy complejo, con una sintaxis fea y poco intuitiva, lo que enlentece notoriamente su aprendizaje.

2.2.1.2. ASP

Solución de Microsoft basada en Visual Basic, pensada para ser implementada en servidores Windows, implicando un alto costo, poca flexibilidad y escasa seguridad.
No presenta ventajas técnicas que lo hagan resaltar sobre el resto.

2.2.1.3. JSP

Solución de Sun basada en Java, que ha logrado una gran penetración en el mercado. Provee soluciones versátiles, pero no presenta ventajas notorias frente a una necesidad concreta, y juegan en su contra la complejidad del desarrollo y el hecho de ser una tecnología propietaria.

2.2.1.4. Python

Lenguaje completamente libre, orientado a objetos, rápido, intuitivo, y sencillo de aprender con una sintaxis simple y bonita. Si bien promete pisar fuerte en el ámbito de los lenguajes web en un futuro cercano, actualmente está muy poco difundido, haciendo difícil la búsqueda de ayudas o ejemplos de desarrollos web.

2.2.1.5. PHP

Lenguaje orientado a objetos, simple, libre, rápido, relativamente sencillo e intuitivo, que ha tenido un gran auge en los últimos tiempos, y cuya alianza con MySQL lo llevaron a convertirse en un standard en la red, el más popular de los lenguajes web.
Si bien su estabilidad ha sido cuestionada, la realidad es que aumentando los recursos del servidor es un lenguaje sumamente escalable.

2.2.2. Decisión

ASP y JSP fueron descartadas desde un comienzo por ser soluciones propietarias y no ofrecer notorias ventajas. Python promete ser una buena opción a futuro, pero actualmente no parece ser una opción lo suficientemente madura. Perl y PHP, empatados técnicamente, presentan una diferencia sustancial que hace inclinar la balanza rápidamente: mientras Perl es conocido por su dificultad de aprendizaje, PHP no solo es sencillo sino que es dominado muy bien por uno de los integrantes de nuestro grupo de trabajo; por lo tanto PHP fue el camino a seguir.

2.2.3. Procesamiento de datos

Para el procesamiento de los archivos recibidos, podría procederse de dos maneras básicas:

1 – procesar cada archivo inmediatamente luego de subido.
2 – escanear periódicamente el directorio de subida en busca de archivos nuevos, y procesarlos.

El procesamiento de los archivos recibidos será realizado en el lenguaje PHP como ya quedó establecido. Realizar la supervisión de un directorio en busca de cambios, no es fácil en ningún lenguaje, y menos aún en PHP.

Por lo tanto, se optó por una opción de procesamiento “on demand”, donde el cliente envía una “orden de procesado” (petición HTTP) luego de subir cada archivo. De esta

manera nos evitamos la supervisión del directorio de subida, además de asegurarnos que el procesamiento será iniciado inmediatamente luego de subir los datos.

Como desventaja, debemos asumir el overhead de esta petición http, que con otra implementación podría ser evitada.

2.2.4. Almacenamiento de Datos

Al momento de pensar en almacenar grandes cantidades de información para luego ser consultadas desde una página web dinámica, no cabe duda que se debe pensar en una base de datos relacional. Este modo de almacenamiento de información se encuentra altamente extendido y existen soluciones desde las más caras y difundidas a nivel empresarial (como Oracle), hasta soluciones de uso libre con muy buen rendimiento, entre las que se encuentra MySQL, que además de obtener buenos resultados en sí misma se complementa muy bien con PHP, lenguaje de programación elegido para el servidor.

2.3. Intercambio de información Cliente – Servidor

De algún modo la información que se encuentra en el cliente debe viajar en forma segura al servidor, ya sea información recolectada como pedidos del cliente. Con este objetivo, fueron analizadas distintas alternativas con el fin de encontrar la mejor solución posible.

2.3.1. Compresión de Archivos

Desde el punto de vista de la red, el estar enviando información cada determinada cantidad de tiempo impacta directamente en los recursos de la misma. Pensando en que la información registrada esta en archivos de texto plano, se consideró comprimirlos previo al envío. Es sabido que para archivos de texto plano se logran muy buenos niveles de compresión (del orden del 50%). Sin embargo, el proceso de compresión trae aparejado un

mayor requerimiento de procesamiento, tanto en el cliente como en el servidor. Visto que el efecto en la XO es inapreciable desde este punto de vista, y que la compresión reduce a la mitad aproximadamente el tráfico en la red, se optó por sacrificar tiempo de procesamiento a cambio de menor tráfico.

2.3.2. Instante de Compresión

Se presentó la decisión de cómo y en que momentos comprimir. Se analizaron las opciones de hacerlo archivo por archivo o acumulando múltiples registros.

Con la primera opción, se reduce el tiempo de procesamiento respecto a la segunda. Sin embargo, se aumenta el overhead en el envío, y se aumenta el espacio ocupado en disco debido a la estructura de clusters de los mismos.

Se evaluó también en qué momento de la ejecución realizar la compresión, si al enviar, al registrar o como un proceso independiente. La opción de realizarlo asíncronicamente en un proceso independiente, la descartamos porque no presenta ventajas apreciables y genera complicaciones de implementación, como ser el acceso exclusivo a los archivos.

La opción de realizarlo al enviar, si bien minimiza también el volumen de información, no optimiza el uso de disco duro, y además genera una mayor carga de procesador en un momento determinado, en vez de distribuir dicha carga en el tiempo como lo hace la segunda opción. Por lo antedicho, la opción elegida fue la de comprimir al registrar.

2.3.3. Conexión directa con Base de Datos

Esta opción implica pasar la mayoría del procesamiento al cliente, haciendo necesario abrir una conexión con la Base de Datos en el laptop descartando la posibilidad de comprimir la información. Luego de tener los datos, habría que enviar también las sentencias SQL a ejecutar en la base generando un gran overhead y además una sobrecarga en la red.

Por otro lado, tiene la ventaja de que no es necesario tener una instancia de procesamiento de la información en el servidor. Hay que agregar también que al no estar centralizado el

procesamiento de los registros, la información quedaría más expuesta a problemas de seguridad.

Dada la ineficiencia que implica este tipo de solución fue en una primera instancia descartada como una opción de diseño.

2.3.4. HTTP - Posteo de Datos

Se evaluó la posibilidad de enviar por HTTP cada registro en un string para ser procesado en el servidor y cargado en la Base de Datos. Al igual que la opción del punto anterior, presenta un gran overhead y una sobrecarga importante de la red.

2.3.5. FTP (File Transfer Protocol)

Este protocolo es de los más eficientes desde todo punto de vista para envío de grandes volúmenes de información. Es seguro y soporta sin problema múltiples conexiones simultáneas y además el cliente se encuentra implementado en bibliotecas de lenguajes de programación y sistemas operativos. Por otro lado, implica tener corriendo un servicio FTP del lado del servidor.

2.3.6. Decisión

Después de analizar las distintas alternativas de comunicación entre los dos módulos, se decidió a la hora de hacer pedidos y actualizar parámetros utilizar el protocolo HTTP. Al momento de enviar no un pedido, sino un volumen de información se consideró más viable el uso del protocolo FTP.

3. Módulo Cliente

Se presentan a continuación los detalles de la implementación del cliente. Se describen los distintos procesos a ejecutar, su modo de funcionamiento así como una descripción detalla de la información recolectada.

3.1. Visión General

En el diagrama de la figura 20 se presenta de forma esquematizada el comportamiento del cliente. En el programa a instalar en cada uno de los laptops, se ejecutan tres hilos independientes con las interacciones que se detallan a continuación.

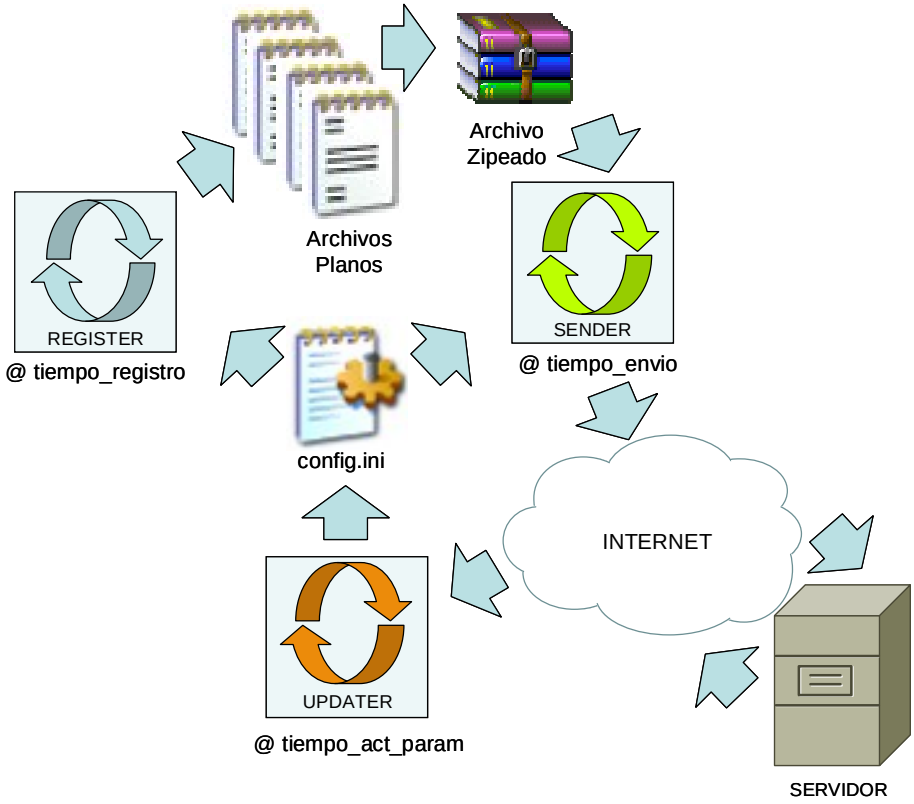


Figura 20 – Cliente

3.2. Proceso de Registro (REGISTER)

El proceso de registro es el encargado de recolectar la información del monitoreo, interactuando directamente con el sistema operativo o con algunas implementaciones adicionales que se detallarán mas adelante. En la figura 21 se presenta un diagrama de capas para ubicar el nivel de nuestras implementaciones.

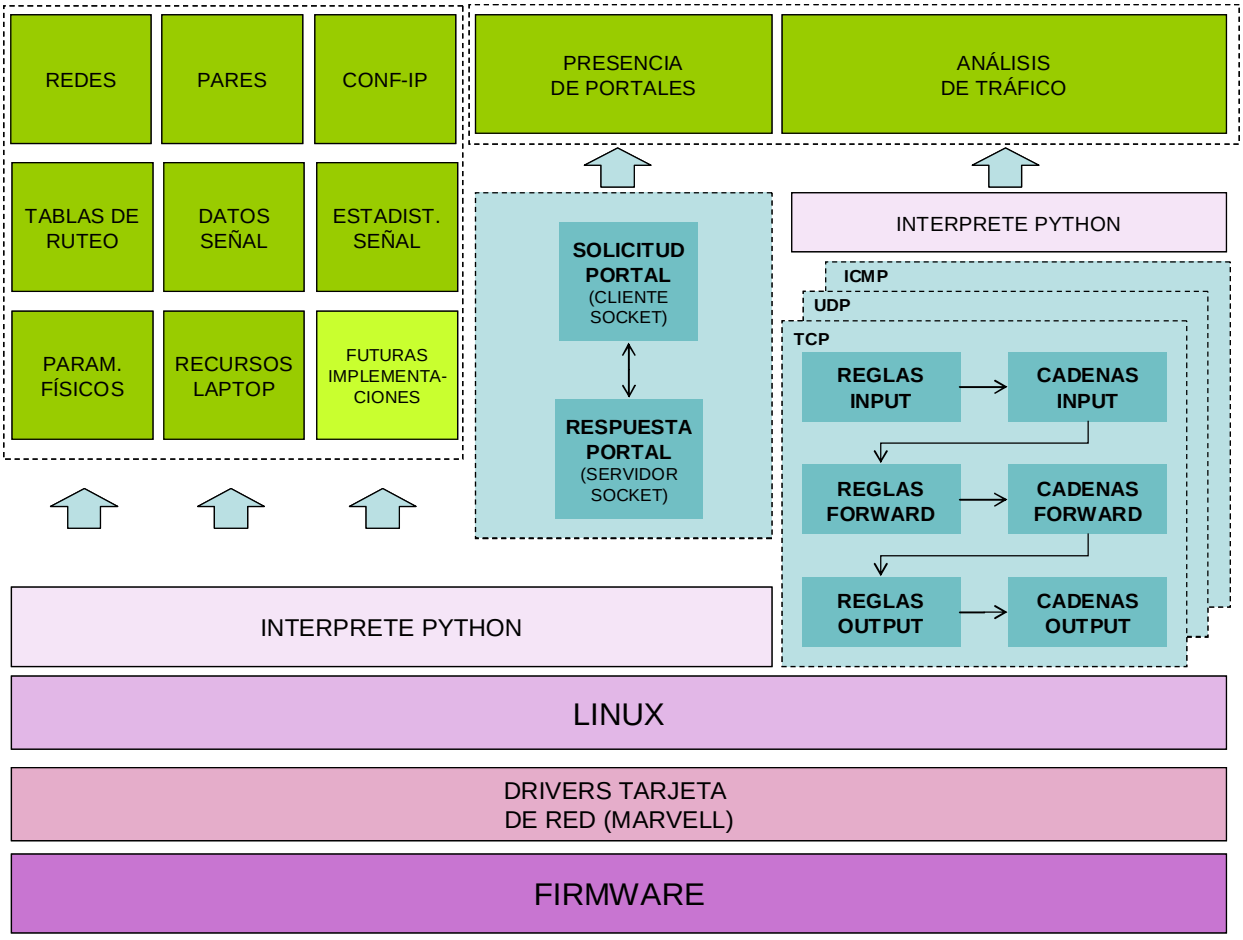


Figura 21 – Capas implementadas

3.2.1. Formato de Archivos

Previo a describir cuales son los distintos registros, se muestra a continuación el formato de los archivos generados en este módulo.

Son generados dos tipos distintos de archivos:

- archivos de texto plano (DAT) conteniendo los datos registrados
- archivos comprimidos (ZIP) conteniendo varios DAT

3.2.1.1. Archivos DAT

Cada registro de información genera un único archivo independiente, donde cada línea representa un tipo de información diferente y está encabezada por un identificador de seis caracteres. Para separar cada valor de una línea se utiliza un tabulador.

Para nombrar los archivos se utiliza la siguiente nomenclatura: "YYYYMMDDHHMMSS.dat". Por ejemplo, un archivo generado el 15 de febrero de 2008 a las 14:35:05 tendrá el siguiente nombre: "20080215143505.dat"

La fecha/hora usada corresponde a la del momento de creación del archivo. Se asume, y se comprobó en la práctica, que el proceso de registro de datos y escritura del archivo no demora menos de un segundo (lo cual podría acarrear sobreescritura de datos).

Con este formato, se asegura:

- Ordenación alfabética que proporciona a su vez una ordenación temporal. Esto será de gran utilidad para simplificar los procesos que requieran realizarse en orden cronológico.
- Unicidad de nombres, de manera de que nunca se crearán dos archivos con el mismo nombre.

3.2.1.2. Archivos ZIP

Los archivos ZIP contienen en su interior varios archivos de datos; la cantidad de archivos es variable.

Para los archivos ZIP se optó por un formato de nombres de archivo que permita identificar los mismos con la XO en el servidor.

El formato elegido se forma a partir de la dirección MAC de la XO en cuestión, y la fecha de registro. Por ejemplo, para la MAC: “00:17.C4:05:23:97”, con fecha de registro 15 de febrero de 2008, los archivos ZIP tendrán el nombre: “0017C4052397[20080215143505].zip”.

El cliente al generar un nuevo archivo se encargará de escribir correctamente la fecha del registro; de esta manera, no habrá nunca problemas de sobreescritura en el cliente, y nos aseguramos que localmente el mayor índice corresponda al último archivo.

Los archivos se enviarán y se procesarán en el orden creciente del índice, lo que siempre corresponderá a un orden creciente temporal también.

3.2.2. Parámetros Registrados

En este apartado se describen de forma detallada todos los parámetros que se registran periódicamente en el módulo cliente.

3.2.2.1. Lista de Redes Disponibles

Uno de los principales indicadores para el monitoreo de redes mesh es el de determinar la presencia de APs que permitan la interconexión de la mesh con otras redes. La presencia de estos routers permitirá a los laptops más cercanos conectarse a ellos y eventualmente configurarse como portales (MPPs) con las ventajas de extensión del área de cobertura que esto conlleva.

Mediante la utilización de *iwlist*, una de las extensiones de Linux perteneciente a las Wireless Tools que permite realizar consultas a la interfaz de red inalámbrica, se determinó el listado de las redes disponibles. Esta es una herramienta muy potente ya que permite obtener una gran cantidad de parámetros de las redes que se encuentran al alcance de un laptop.

Los atributos recogidos fueron: el valor de ESSID (nombre con que se identifica a la red), el modo de funcionamiento (Ad-Hoc, Managed, Master o Monitor), la frecuencia del espectro a la que opera, la dirección MAC del AP y otros parámetros como son la calidad del enlace, el nivel de señal, el nivel de ruido y el nivel de seguridad implementado.

3.2.2.2. Lista de Pares

Otro dato relevante en la conformación de una red mesh resulta ser la lista de laptops que se encuentran asociados directamente con un laptop específico. A esta lista se la conoce como pares de un MP.

En esta ocasión la información requerida se obtiene de otra de las extensiones de Linux para el manejo de interfases wireless conocida como *iwpriv*, esta tiene la particularidad de interfasear directamente con los drivers de la tarjeta de red.

Los principales datos relevados sobre esta lista de pares fueron: la dirección MAC del par, el modo de funcionamiento (on/off), y la relación señal a ruido de cada enlace.

3.2.2.3. Tablas de Ruteo

Las tablas de ruteo aportan un valor relevante en el análisis del comportamiento de la mesh, siendo información de vital importancia en este tipo de redes donde la topología es muy dinámica. Para obtener esta información la extensión *iwpriv* es la que se utiliza para registrar todo los parámetros de las tablas de ruteo.

A continuación se presentan los parámetros registrados:

Nombre Atributo	Descripción
Da	Dirección MAC del destino
Ra	Dirección MAC del Host del primer salto
Valid	Indica validez de ruta (valido-1, no válido-0)
Metric	Métrica o Costo de la ruta
Dir	Sentido de la ruta, directa o inversa
Rate	Tasa de envío al host del primer salto
Dsn	Numero de Secuencia del destino
Ssn	Numero de Secuencia del origen
Hop count	Cantidad de Saltos
Ttl	Tiempo de Vida
Expiration	Tiempo de Expiración
Sleepmode	Sleepmode del RA
Snr	Relación señal a ruido en el enlace del primer salto
Precursor	Predecesor en rutas directas

3.2.2.4. Presencia de Portales

En lo referente a la topología de la red, la presencia de portales tiene un papel determinante a la hora de hacer un análisis de conectividad de la mesh ya que los portales permiten la interconexión con otras redes como es Internet.

Se analizaron las secuencias de eventos previos que se suceden a que un MP comience a rutear tráfico entre dos redes distintas.

Como una primera alternativa se estudió el intercambio DHCP que ocurre cuando un laptop se enciende, generando un paquete DHCP DISCOVER y posteriormente sniffeando la red para obtener los DHCP OFFER de los MP que están como Portales. Si bien esta solución es efectiva, implica terminar el demonio DHCP cliente que corre en las máquinas por lo que no fue considerada viable. A continuación se muestra una salida del Programa WireShark donde se sniffeó la red para ver la secuencia de paquetes generados al enviar un DHCP DISCOVER.

No.	Time	Source	Destination	Protocol	Info
14	25.702465	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0xefac34f
15	25.752245	QuantaMi_05:23:97	Broadcast	ARP	Who has 169.254.255.254? Tell 169.254.103.202
16	26.155760	169.254.103.202	169.254.255.254	DHCP	DHCP Offer - Transaction ID 0xefac34f
17	26.156603	0.0.0.0	255.255.255.255	DHCP	DHCP Request - Transaction ID 0xefac34f
18	26.182888	169.254.103.202	169.254.255.254	DHCP	DHCP ACK - Transaction ID 0xefac34f
19	26.752283	QuantaMi_05:23:97	Broadcast	ARP	Who has 169.254.255.254? Tell 169.254.103.202
20	27.063364	QuantaMi_05:2a:a1	Broadcast	ARP	[Malformed Packet]

Figura 22 – Análisis de paquetes

Como alternativa a lo recién mencionado se implementaron dos procesos, uno cliente y otro servidor dónde el primero cada un determinado tiempo envía un paquete UDP a la dirección de Broadcast de la mesh y el segundo escucha en un puerto libre. Cuando un MP recibe uno de estos paquetes consulta el parámetro interno ANYCASTMASK[3] y responde mediante un paquete UDP vía Unicast al MP que hizo la consulta. Del paquete respuesta, el cliente obtiene la dirección mac, el valor de la máscara de anycast y la cantidad de saltos al destino (obtenido mediante el TTL). En función de estos valores se hace posible obtener un grafo de las máquinas que se encuentran al alcance del cliente, la cantidad de saltos y si están configuradas como portales (máscara = 0x2) o no (máscara = 0x0).

3.2.2.5. Análisis del Tráfico

Contar con valores de tráfico en una red de computadores brinda una infinidad de información, como son valores de “throughput” de los terminales, picos de trafico, de inactividad, mensajes de control, permitiendo realizar estadísticas y hacer mejores dimensionados. Cabe agregar que en este tipo de redes, los nodos poseen más inteligencia

[3] Los valores del parámetro ANYCASTMASK son seteados internamente en las máquinas por el servicio NetworkManager, herramienta de gestión de redes en linux.

debido a que todos son “pequeños routers”, por lo que el hecho de poder saber que entra y sale de cada uno permite hacer una fácil y mejor gestión de la red.

Se utilizaron las IpTables de Linux para obtener valores de los paquetes que circulan por la red. Esta herramienta es por definición un filtro IP que opera principalmente a nivel de capa de red del stack TCP/IP, aunque también lo hace a nivel de capa de Enlace.

Los principales conceptos en lo que refiere a esta herramienta son:

Regla – Una regla agrupa un conjunto de condiciones al que se le asigna un destino o target en la mayoría de los filtros IP. Existen algunas implementaciones que permiten asignar más de un target a una regla.

Target – A cada regla se le asigna en general un target, que especifica que hacer con el paquete (Por ejemplo, descartarlo, forwardearlo, aceptarlo, etc o pasarlo a otra cadena).

Cadena – Una cadena está compuesta por un conjunto de reglas que son aplicadas a un paquete que la atraviesa. Tienen un determinado propósito, así como área de aplicación según el tipo de tabla al que pertenecen. Cada cadena puede tener asociada ninguna o varias reglas, si un paquete entra a una cadena sin reglas, se le aplica una regla por defecto que depende de la cadena.

Tabla – Una tabla tiene asociado un propósito, en IpTables existen cuatro tablas; Filter, Nat, Mangle y Raw. La primera por ejemplo se utiliza para filtrar paquetes, la segunda para hacer NAT, la tercera para modificar campos de los paquetes como puede ser el TOS, etc.

Se focalizó el estudio en la tabla *filter* ya que el objetivo de la aplicación es filtrar paquetes y no modificarlos. Esta tabla cuenta con tres cadenas principales; INPUT, FORWARD y OUTPUT. Cuando un paquete entra a una cadena se lo compara con todas las condiciones que tienen las reglas de esa cadena. En caso de que un paquete cumpla las condiciones de una regla, se le aplicará el target asociado a esa regla y no se continúan evaluando las demás. Esto implica que si un paquete matchea la primer regla de una cadena, las siguientes condiciones (en el caso de que existan) no serán procesadas. La arquitectura de las tablas es la siguiente:

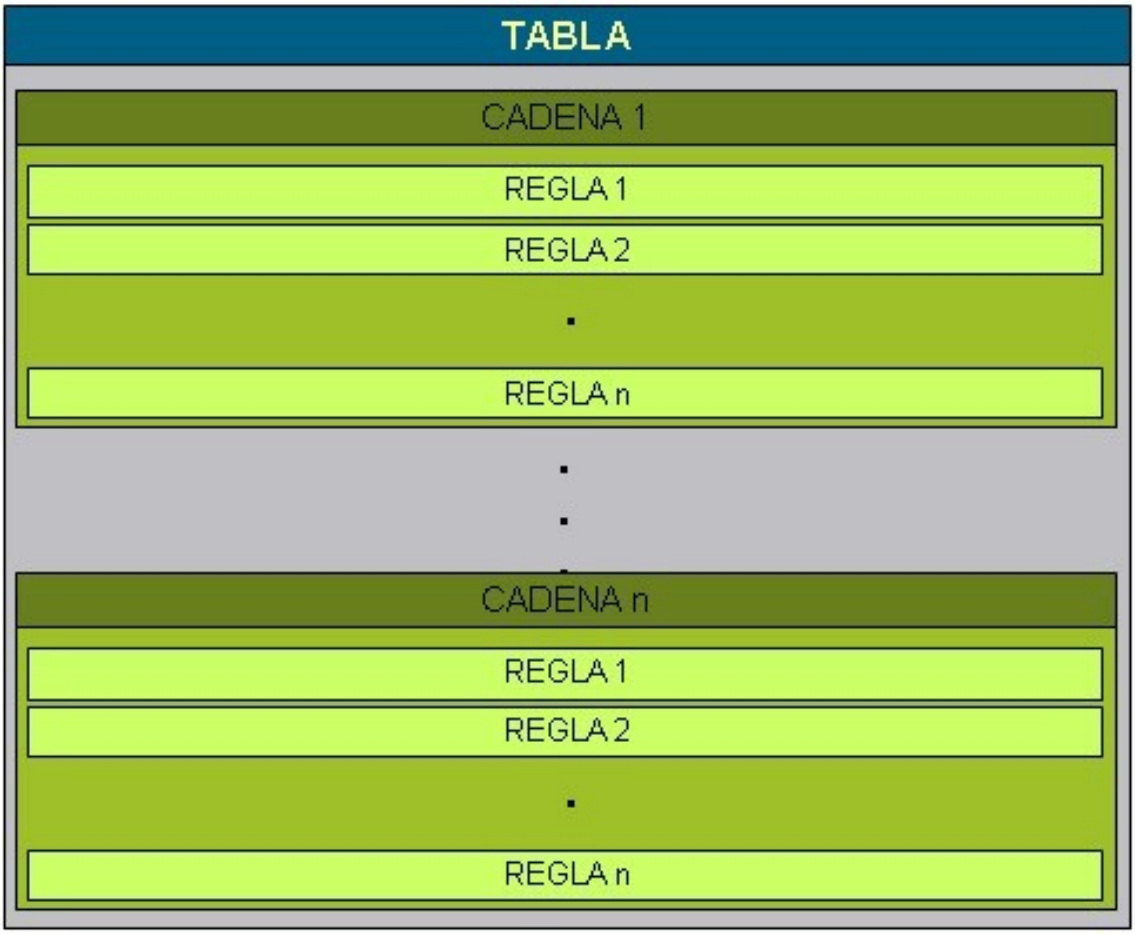


Figura 23 – IPtables

Con el objetivo de ser transparente a las cadenas y reglas ya cargadas en las máquinas se crearon cadenas según los protocolos de filtrado (tcp, icmp y udp), sin reglas y al principio de las cadenas ya existentes se insertaron reglas con el target “jump” (salto) hacia otra cadena. Así los paquetes no son modificados por nuestras reglas, y las reglas y cadenas posteriores a las creadas se continúan evaluando normalmente.

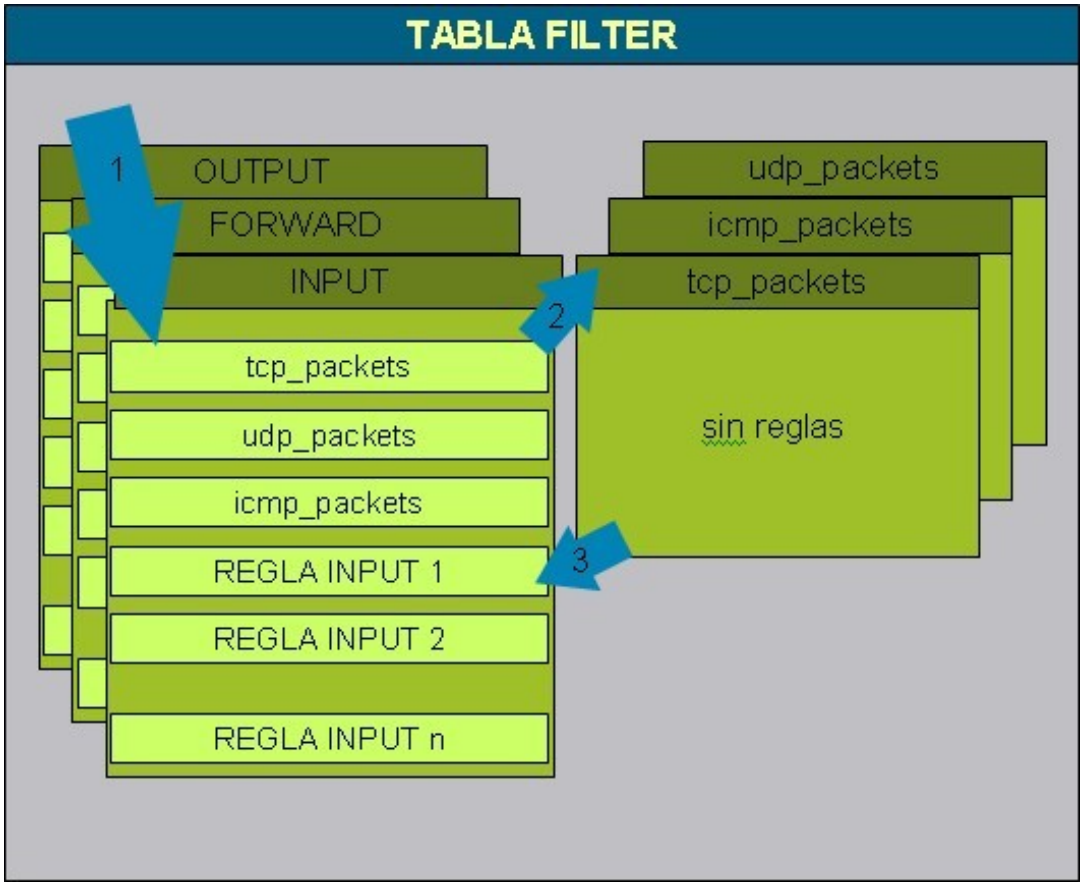


Figura 24 – Iptables 2

A modo de ejemplo un paquete tcp con destino el host local entraría a la tabla INPUT (1) y al atravesar la regla tcp_packets seria redirigido a la cadena tcp_packet (2) donde al no haber reglas terminaría siendo redireccionado a la cadena a la cual entró (3) (INPUT). De esta forma en la regla tcp_packets queda registrado el cardinal de paquetes y el tamaño en bytes de información que lo atraviesa.

3.2.2.6. Datos de Señal

En las redes inalámbricas existen diversos parámetros para tomar en cuenta en lo referente a la calidad de las señales; componentes externos a las redes pueden llegar a interferir o deteriorar la calidad de las mismas. Es por esto que se decidió obtener para cada una de sus interfaces de

red, diversos descriptores de las señales inalámbricas como son, la potencia de señal, la calidad de enlace, nivel de ruido y número de retransmisiones.

3.2.2.7. Estadísticas de la Señal

Como complemento a los datos de la señal se obtuvieron también valores de estadísticas de información; en lo que refiere a paquetes y bytes con errores, perdidos, enviados y recibidos de cada host. Esta información es muy útil en caso de querer determinar el "throughput" de las máquinas en determinados entornos. Se puede pensar que en entornos densos como el aula de un colegio este "throuhput" tiende a ser bajo y en otros, como en las cercanías de los hogares y en presencia de un AP este valor seria considerablemente más alto.

3.2.2.8. Configuración IP

A pesar de que el ruteo en las redes mesh es a nivel de la capa MAC, cada uno de los MPPs presentes en la mesh implementa un servidor DHCP asignando una dirección IP a cada laptop de la mesh. En este tipo de redes, contar con información a nivel de capa tres para todas las interfaces del equipo (msh0 y eth0) brinda a los administradores de red una ayuda extra a la hora de determinar la topología de la red. Contar con una dirección IP en la mesh permite por ejemplo, pensando en el protocolo ICMP determinar la presencia o el tiempo de ida y vuelta a un host.

Mediante la herramienta NetworkManager de Linux, se obtuvo no sólo la información referente a Dirección IP sino también a los valores de Máscara de Subred, Dirección de Broadcast de la red, dirección IP del Gateway (en caso de que haber uno), dirección de DNS Primario, DNS Secundario y valor del ESSID de conexión.

3.2.2.9. Utilización de Recursos

La particularidad de las redes mesh en que cada uno los MPs participa de las actividades de reenvío de paquetes, hace que la capacidad de procesamiento de estos paquetes se vea afectada por la actividad de los procesos que corran en cada uno de los laptops. Es por ello que el monitoreo aporta información referida a la utilización de los recursos en cada uno de ellos.

Mediante una sencilla instrucción de "top" al Linux, se obtiene información acerca de: Tiempo de Encendido, Total de Tareas en proceso, porcentaje de CPU Utilizado, porcentaje de CPU Libre, porcentaje de CPU utilizado por el Sistema, Memoria Utilizada y Memoria Libre

3.2.2.10. Parámetros Físicos y de Sistema

La dinámica evolución del software de los equipos ha provocado (en ciertas ocasiones) dificultades a nivel de las comunicaciones en la mesh. Por tal motivo, se incorporó al monitoreo información referente a versiones de hardware y software de cada uno de los laptops.

Una serie de consultas al sistema operativo aportan información sobre: Versión del Firmware, Versión Sistema Operativo, Versión del SUGAR, Nombre que identifica al equipo y el estado y nivel de la batería.

3.3. Proceso de Envío (SENDER)

Como ya se ha explicado, los archivos generados en el cliente por el módulo de registro de datos, serán comprimidos y almacenados localmente hasta su posterior envío. Periódicamente según el tiempo establecido en el archivo de configuración, el cliente intentará comunicarse con el servidor FTP configurado. De no poder establecerse la comunicación, ya sea porque el cliente no está apropiadamente conectado a Internet (o a la red local del servidor FTP), o

porque el servidor FTP no esté respondiendo, se dejará registro de la acción en el archivo log y se reiniciará el ciclo de espera.

Una vez establecida la conexión con el servidor FTP, se realizará un upload de los archivos ZIP, desde el más viejo al más nuevo, uno a uno. En caso de que el envío de alguno de los archivos genere un error se logueará el mismo y se intentará con el siguiente archivo. Si el envío se realiza correctamente se logueará el evento como satisfactorio y se eliminará el archivo ZIP local.

Por último, luego de que se hayan subido todos los archivos, se le debe indicar al servidor encargado de insertar los mismos a la base de datos de que tiene nuevos registros disponibles. Esto se realiza mediante una consulta HTTP a un script determinado en el servidor, indicándole en los parámetros de la misma, la dirección MAC de la máquina de origen y la hora local. Este último dato se proporciona a los efectos de poder realizar una corrección de la hora local al momento de ingresar los datos al servidor. La dirección remota hacia la cual se debe dirigir la consulta en cuestión también es configurable desde el archivo de configuración.

3.3.1. Sincronización temporal

Visto que el sistema se encuentra distribuido en distintos equipos, por problemas de sincronización natural o porque cada usuario puede alterar la configuración del XO, los euipos no tendrán necesariamente todos la misma hora. No solo podrían presentarse pequeñas desviaciones de algunos segundos, sino que también podrían recibirse fechas completamente “incoherentes”, que distorsionarían el análisis de los resultados.

Por lo tanto, es necesario utilizar algún mecanismo de corrección de la hora, de manera de poder comparar datos temporalmente en forma coherente.

El mecanismo utilizado, no pretende lograr una precisión absoluta. Es decir, lograr una “sincronización” exacta de la hora en Internet no es una tarea fácil, debido a retardos en las transmisiones, colas de esperas, etc. Existen protocolos y herramientas para lograr una mayor precisión, pero para nuestra aplicación en particular no es tan crucial.

El mecanismo implementado, consiste en:

- El servidor mantiene su hora local, y registra los eventos en la base de datos según la misma.

- El cliente tiene también su hora local, y registra los eventos conjuntamente con dicha hora.
- Al realizar la orden de procesado desde el cliente hacia el servidor (consulta HTTP) se envía adicionalmente como parámetro de dicha consulta, la hora local del cliente.

- En base a la información disponible en el servidor (hora del servidor al momento de la orden de procesado, hora del cliente en ese momento) el mismo calculará el desfasaje horario y lo corregirá en cada uno de los registros que hayan disponibles para ingresar a la base de datos.

Este tipo de corrección, además de no ser de gran precisión como se aclaró anteriormente, no puede ni pretende solucionar problemas transitorios que se pueden presentar al actualizar la hora en algún cliente entre dos envíos sucesivos de datos. Está pensado para mantener sincronizada la hora, bajo la hipótesis de que la hora en el cliente no es cambiada arbitrariamente.

3.4. Proceso de Actualización de Parámetros (UPDATER)

El archivo de configuración local (descrito en el manual de instalación y uso), almacena localmente las opciones de configuración del cliente en el formato establecido. Si bien este archivo es un archivo de texto plano que puede ser modificado perfectamente en forma local, el objetivo es que bajo un funcionamiento normal del sistema sea el servidor quien mantenga la información de configuración de todas las laptops, y de ser necesario las mismas los actualicen.
A tales efectos, se desarrollaron rutinas en el servidor y en el cliente, así como también se crearon tablas para almacenar los datos en la base.

El procedimiento será el siguiente:

- Periódicamente (cada un tiempo configurable) el cliente realizará una consulta al servidor.
- El servidor consultará la base de datos para detectar si tiene una configuración actualizada para la XO en cuestión.
- Si hay datos actualizados, serán enviados al cliente como respuesta a la consulta generada, en un formato de mensaje preestablecido.

- El cliente actualizará el archivo de configuración local, generando un archivo de respaldo de la configuración anterior.

Dada la implementación, si los parámetros son actualizados localmente, no se reestablecerán a la configuración seteada por la base de datos hasta que haya algún cambio en la misma. Esto se debe a que para minimizar el tráfico de datos, no se envía innecesariamente la configuración en ninguno de los dos sentidos, por lo que el único indicador de "sincronización" de configuraciones en una bandera indicadora en la Base de Datos.

El tiempo entre sucesivas actualizaciones claramente es configurable. De encontrarse algún tipo de error en el proceso, ya sea en la comunicación con el servidor o en los parámetros recibidos, se reiniciará el ciclo de espera sin cambios en la configuración.

3.5. Sistema de logging

Con el objetivo de mantener un registro de las acciones realizadas por el programa, se implementó un sistema de logueo tanto en el cliente como en el servidor. El formato es el clásico de los logs Unix, en archivos de texto plano separados por tabuladores, indicando: fecha y hora del evento, tipo de evento (warning, error, info, etc), el originador del evento, y detalles del mismo.

En el cliente se utilizaron para la implementación librerías de Python a tales efectos. Dichas librerías permiten generar archivos de log, y están desarrolladas de tal modo que son "thread-safe", es decir, permiten escribir en el mismo log desde distintos threads de ejecución sin tener que realizar ningún control extra.

Se optó por registrar de modo "rotativo", es decir: se genera un archivo log hasta llegar a un determinado tamaño máximo; al llegar a ese tamaño máximo, se realiza un backup del mismo, y se empieza un archivo log nuevo. La cantidad de backups a guardar es configurable.

Desde el archivo de configuración se leerán las opciones necesarias, es decir: nombre del archivo log, tamaño máximo, cantidad de backups, nivel de registro.

3.6. Configuración del programa

Nos propusimos que el cliente fuera lo más parametrizable posible, de forma sencilla, efectiva, eficiente, y entendible.

La solución usada de forma estándar para este objetivo es un archivo con un formato conocido, separado en secciones y parámetros (comúnmente ".ini" en Windows). Decidimos ceñirnos a ésta solución conocida, utilizando para la configuración del cliente un archivo que contiene todos los parámetros necesarios. El formato es el siguiente:

```
[seccion1]

parametro1 = valor1
parametro2 = valor2
parametroN = valorN


[seccionM]

parametroK = valorK
…
```

Este archivo de configuración es mantenido y actualizado remotamente en forma automática mediante la interfaz web del sistema. Una descripción de sus contenidos se encuentra en el Manual de Usuario del cliente.

3.6.1. Backup de configuración

Para que el proceso de configuración del cliente y actualización de parámetros sea robusto, se implementó en caso de modificaciones un respaldo del archivo de configuración preexistente. Si falla la lectura del archivo de configuración, se intentará continuar con el archivo de respaldo.

Este respaldo es realizado automáticamente al actualizar los parámetros, y la restauración del archivo de backup sobre el viejo es realizado en caso de que falle la lectura del archivo original

y sea exitosa la lectura del archivo de backup. La implementación actual intenta restaurar el archivo de configuración respaldado solo si presentan errores en la lectura del mismo y no en caso de errores de parámetros (por ejemplo, error de comunicación con el servidor).

Sin embargo, es prácticamente imposible implementar un sistema que sea altamente configurable y a la vez extremadamente robusto. Es decir, no podemos ser inmunes a todo tipo de intervenciones externas sobre los archivos de configuración. Por ejemplo, si ambos archivos (el de configuración, y el de backup) son eliminados, o modificados externamente en forma maliciosa, el programa no podrá iniciar correctamente.

4. Módulo Servidor

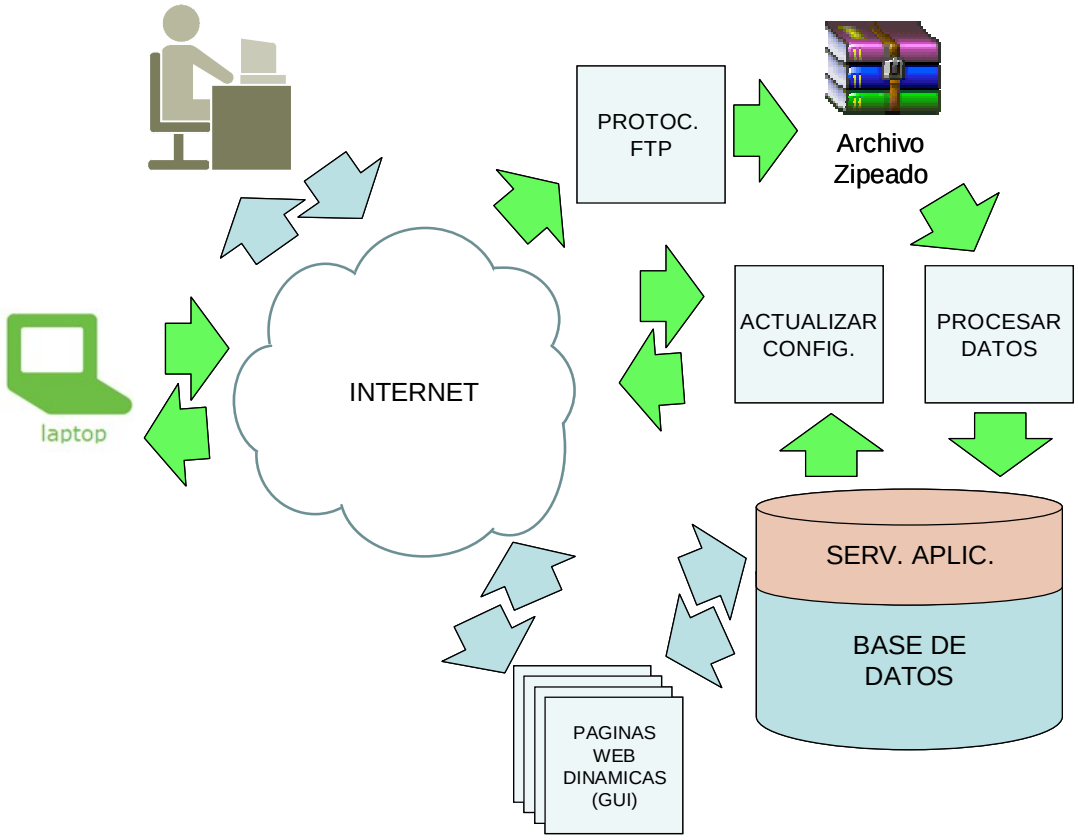


Figura 25 – Servidor

4.1. Servidor FTP

Como se comentó en los criterios de diseño, resultaba mucho más eficiente la transmisión de los archivos vía FTP, por lo que la implementación de la solución requerirá levantar un servidor FTP en una carpeta determinada, desde la cual se leerán los archivos para ser procesados.

Sobra la marcha del desarrollo, se nos solicitó por parte del cliente la posibilidad de realizar el envío de datos mediante el protocolo SFTP ya que los servidores con los que se cuenta en el proyecto Ceibal ya están utilizando el mismo. Por lo tanto, se contempló soportar dicho protocolo de forma de dejar a elección el protocolo a utilizar (FTP / SFTP).

Dado que Python no posee soporte nativo para el protocolo SFTP, fue necesario resolver la transferencia utilizando llamadas al programa "sftp" sistema operativo. Para esta tarea de interacción con un programa que presenta distintas respuestas según el caso de uso, nos fue necesario capturar y reconocer la salida del mismo, y enviarle una respuesta acorde. Para esto último, resultó de suma utilidad.

4.2. Recepción de Datos

Luego de enviar los archivos mediante FTP, el proceso en los laptops invoca a un URL en el servidor mediante el protocolo HTTP, el cual implementa el procesado de los datos. Esta rutina recibe por POST la dirección MAC del laptop que lo invoca y procesa los archivos comprimidos que se encuentren el la carpeta compartida mediante FTP. Cada uno de los archivos comprimidos que procesa puede contener más de un archivo de datos, por lo que descomprime el contenido en una carpeta temporal y procesa cada uno de ellos levantando la información en la base de datos. Algunas de las principales características de este proceso son:

- No es necesario que el laptop esté registrado en la base de datos, ya que si no se encuentra, el sistema lo da de alta.
- Si por error en el proceso de envío se enviara el mismo archivo más de una vez, no se registrarán datos anteriormente enviados.
- Si el archivo comprimido está corrupto lo elimina y continúa con el siguiente.

- Al finalizar el proceso se eliminan los archivos temporales de datos y los archivos comprimidos de la carpeta del FTP.
- Toda la actividad de este proceso queda logueada.

La funcionalidad de registro de información detallada anteriormente se complementa con una actividad de depuración de información que se ejecuta en la misma rutina. La misma consiste en consultar un parámetro del sistema que especifica (a nivel de cada laptop) la cantidad de días de información a mantener en la base de datos y aquellos registros del laptop que se está procesando que estén por fuera de esa ventana de tiempo, serán eliminados.

4.3. Actualización de Parámetros

De forma periódica el servidor recibe pedidos de los laptops, para detectar cambios en su configuración por parte del administrador del sistema. Como para cada laptop se mantiene en el servidor un registro de su configuración, al actualizar administrativamente desde la interfaz gráfica dichos parámetros se prende una bandera indicando que debe actualizarse la información local. El proceso que atiende los pedidos de los laptos mediante protocolo HTTP, recibe por POST la dirección MAC de la maquina que solicita y chequea si hay necesidad de actualizar los parámetros locales, si es así, envía al laptop la información necesaria para componer un nuevo archivo de configuración.

4.4. Interfaz Gráfica de Usuario

Además de la calidad y utilidad de la información recolectada, se puso mucho énfasis en dotar al sistema de una interfaz gráfica de usuario amigable para la administración y la realización de consultas. El carácter full web de esta interfaz permitirá darles a los administradores del sistema acceso remoto desde cualquier sitio.

Se detallan a continuación las principales funcionalidades que ofrece la aplicación, las que se expondrán más detalladamente en el manual de usuario que se anexa a esta documentación.

4.4.1. Consultas

En el módulo de consultas se expone toda la información que es registrada por los laptops. Para hacer más ordenada la presentación de la información, el sistema permite asociar cada laptop a una escuela, la que se encuentra en un determinado departamento y una determinada localidad. A su vez cada escuela tiene asociado un responsable técnico. Se presentan a continuación los casos de uso asociados a la consulta de información en el sistema.

4.4.1.1. Buscar Laptop (Consulta General)

El caso de uso Buscar Laptop permite buscar en el sistema un laptop por Departamento, Localidad, Escuela, Responsable Técnico, Dirección MAC y/o Serial.

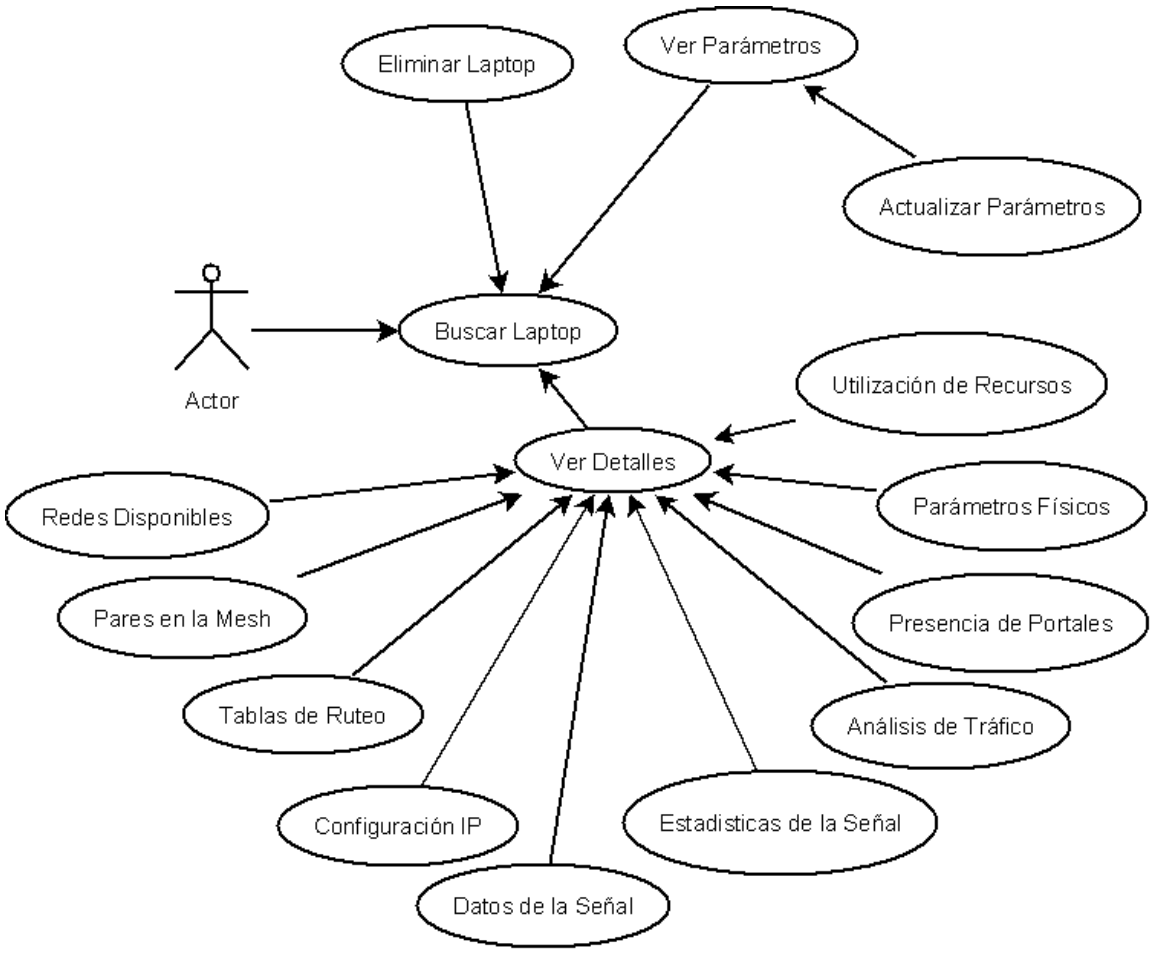


Figura 25 – Buscar Laptop

Una vez ubicado se podrá consultar un resumen de toda la información con que se cuenta para este laptop en "Ver Detalles" y desde allí acceder a los registros de las diferentes consultas con que cuenta el sistema. Desde la misma página de Ver Detalles es posible acceder a actualizar los parámetros e inclusive eliminar el laptop y todo registro asociado al mismo.

4.4.1.2. Últimos Registros

Consiste en una consulta que muestra los últimos cincuenta registros que se encuentran en el sistema; esto permite una vista rápida para la verificación de su correcto funcionamiento. Es posible acceder desde esta página al detalle de los registros de cada laptop.

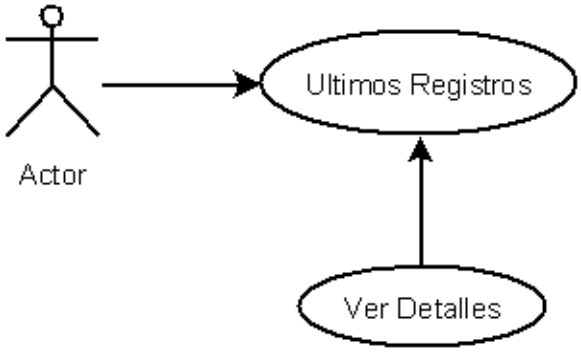


Figura 26– Ultimos registros

4.4.1.3. Laptops Sin Transmitir

Esta consulta permite identificar laptops que se encuentren registradas en el sistema y que lleven más de un cierto tiempo sin transmitir datos al servidor (1, 2, 5, 10, 15 o 30 días.). Como en otras consultas del sistema permite filtrar por Departamento, Localidad, Escuela y Responsable Técnico. Es posible además acceder desde esta página al detalle de los registros de cada laptop.

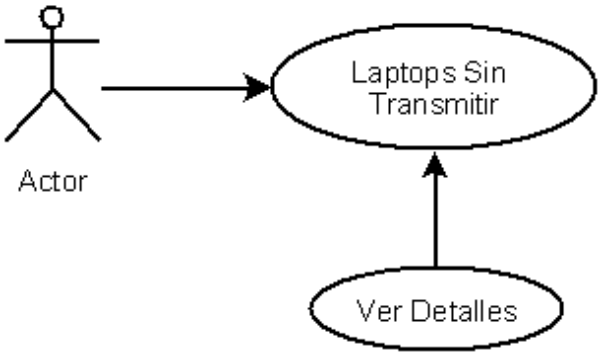


Figura 27 – Laptops sin transmitir

4.4.1.4. Estadisticas de Escuelas

En esta consulta es posible obtener una serie de indicadores de diversa índole, que permiten tener una visión global de la situación de la escuela en un determinado rango de tiempo.

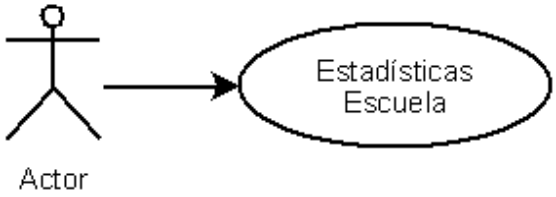


Figura 28 – Estadísticas de escuelas

4.4.2. Parametrización

Como ya fue comentado es posible parametrizar desde la interfaz gráfica muchas de las características de funcionamiento del monitoreo local y el servidor a donde se envíe la información. Estos parámetros de configuración se agrupan en:

- **Registro**: Se especifica el tiempo entre registros y se prenden o apagan los indicadores de datos a registrar.

- **Local**: Se definen carpetas de almacenamiento local de información y comportamiento del logueo local.

- **Zipeado**: Se define el tamaño máximo de respaldo de información local.

- **Servidores**: Se configura todo lo referente al tiempo de envío de la información y la localización del servidor al cual se comunicarán.

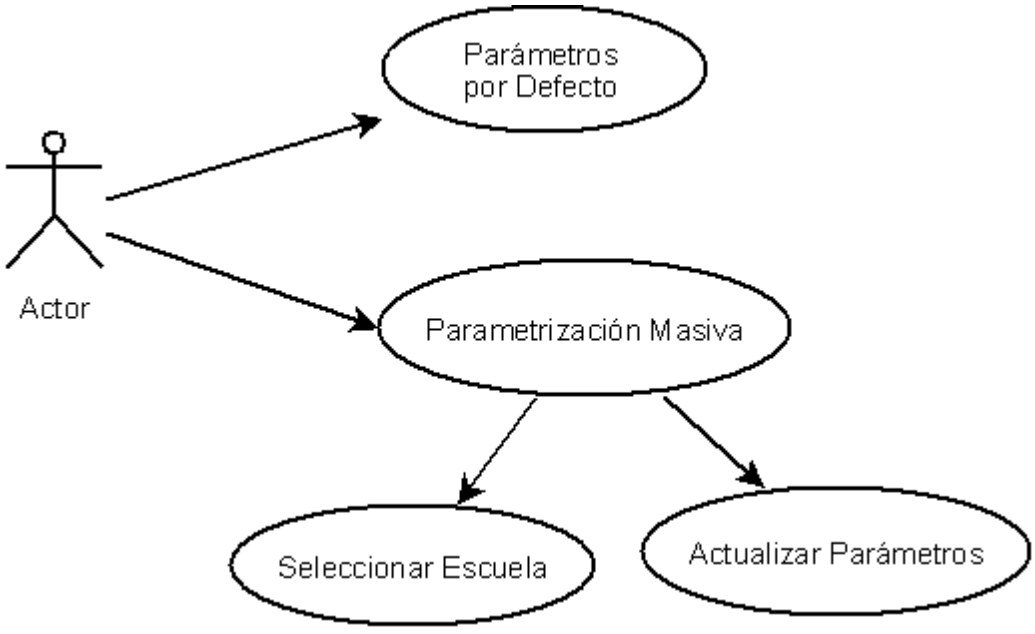


Figura 29 – Parametrización

Cada laptop registrado en el sistema tiene su propia parametrización; los que envíen información por primera vez, tomaran la información del archivo de configuración local y al momento de registrarse en el sistema tomarán los parámetros configurados por defecto en la Base de Datos.

En “Parámetros por Defecto” es donde se definen los datos de configuración que tomarán las laptos la primera vez que transmitan, o si se cargan en cualquiera de las dos opciones de carga masiva que se detallarán mas adelante.

La “Parametrización Masiva” consiste en cambiar los parámetros de varias laptops al mismo tiempo. El nivel mas bajo de actualización es una escuela, pero se puede hacer a nivel de todas las escuelas a cargo de un responsable técnico, a nivel de toda una localidad, a nivel de todo un departamento o para todo el sistema.

4.4.3. Mantenimiento

Se desarrollaron una serie de páginas que permite el mantenimiento de las diferentes entidades del sistema. Las diferentes opciones de mantenimiento son:

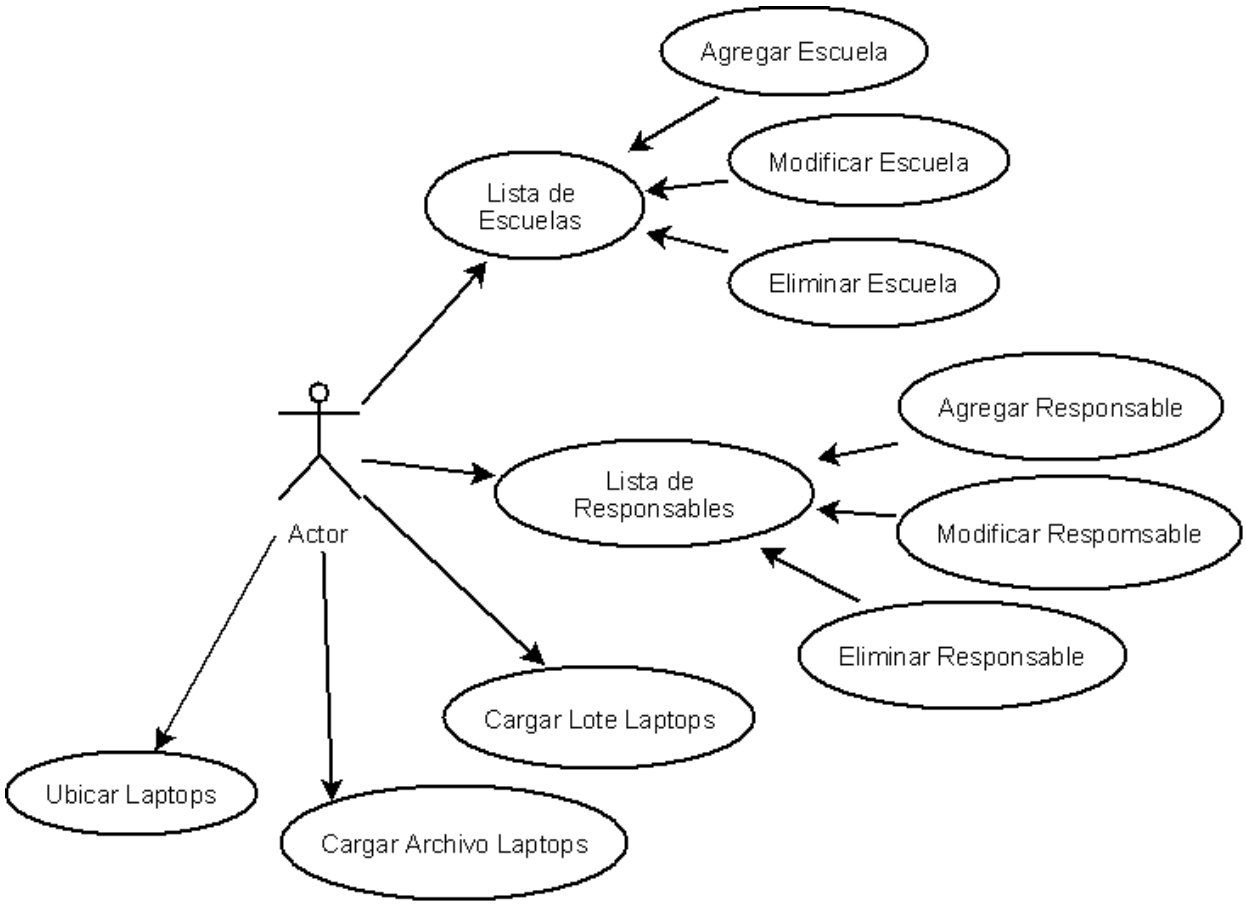


Figura 30 – Mantenimiento

- **Mantenimiento de Escuelas**: Permite consultar la lista de escuelas, agregar nuevas y modificar o eliminar las existentes.

- **Mantenimiento de Responsables Técnicos**: Permite consultar la lista de responsables técnicos, agregar nuevos y modificar o eliminar los existentes.

- **Cargar Lote Laptops**: Permite cargar un lote de laptops vinculándolos a una escuela siempre que se trate de un lote continuo en número de serie. En otras palabras, se ingresa el número de serie inicial y final, se selecciona la escuela a la que se asignarán y el sistema genera en forma automática el lote completo de laptops.

- **Cargar Archivo Laptops**: Otra opción de carga de laptops al sistema es mediante el procesamiento de un archivo plano con los números de serie a ingresar. El sistema levanta el archivo seleccionado por el usuario y realiza la carga en la base de datos.

- **Ubicar Laptop**: Si no se realiza la carga masiva de laptops al sistema y el registro se hace al momento que éstos comienzan a transmitir, los laptops aparecerán vinculados a una escuela indefinida y será necesario ubicarlos en la escuela correspondiente. Además, si se hace alguna reubicación de laptops de una escuela a otra, resulta indispensable contar con una funcionalidad del sistema que lo soporte. Para cubrir ambas funcionalidades, se cuenta con una interfaz en la que se puede seleccionar escuela origen y destino (incluye la escuela Indeterminada) y hacer los movimientos correspondientes.

4.4.4. Administración

La interfaz gráfica cuenta con un mecanismo de autenticación de usuarios propio, con lo cual se debe mantener desde el sistema la lista de usuarios autorizados para su uso. Se cuenta con una interfaz desde la cual el administrador del sistema podrá realizar esta tarea.

Figura 31 – Administración

4.5. Modelo de la Base de Datos

Se presenta a continuación un diagrama de la base de datos donde se muestran cuatro grupos de tablas:

- **Entidades**: Tablas donde se cargan Departamentos, Localidades, Escuelas, Responsables Técnicos y las propias Laptops.

- **Parametrización**: Tablas donde se mantienen los parámetros de registro, sistema, servidor y criterios de compresión.

- **Registro**: Tablas que contienen los datos registrados por el sistema, una por cada tipo de monitoreo implementado.

- **Autenticación**: Tablas vinculadas al registro de los usuarios del sistema.

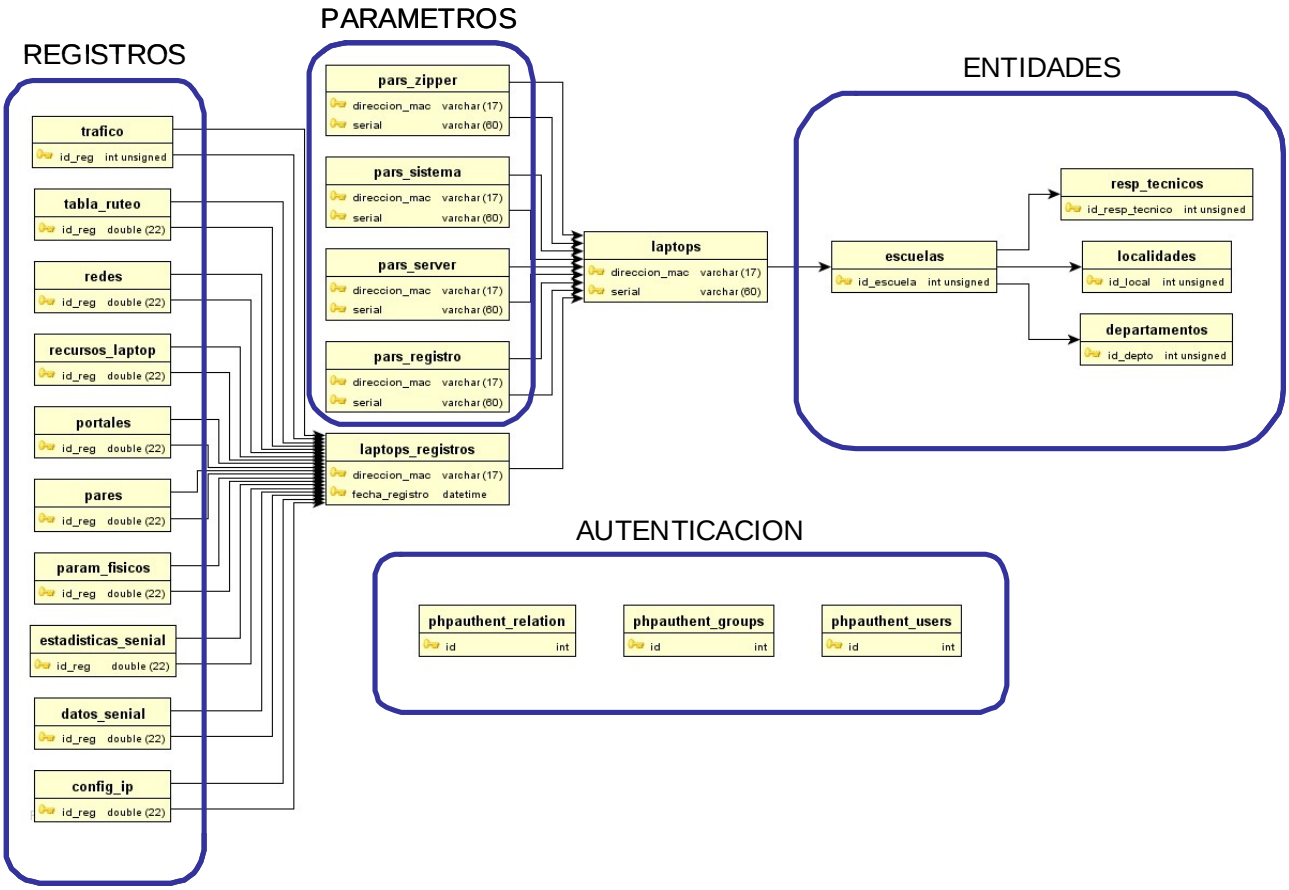


Figura 32 – Modelo base de datos

Capítulo 5

Puesta en Producción

Luego de finalizada la etapa de desarrollo se pasó a la realización de las pruebas; en una primera instancia se realizaron en los dos XO's con los que contamos durante todo el proyecto (enviados por la organización OLPC) y posteriormente se realizaron al menos tres visitas al LATU para pruebas con mas equipos y deploy en el servidor que nos fue asignado para nuestro sistema.

1. Pruebas con dos Laptops – Desarrollo

Existieron pruebas continuas del cliente desarrollado en los dos equipos con los que contábamos, esto nos fue de mucha utilidad para minimizar los tiempos de desarrollo. De todas formas, con tan solo dos equipos a disposición no era fácil replicar muchos de los escenarios que se presentan en las redes mesh, sino que tan solo nos permitió probar los siguientes casos:

i. Ambas XO's conectadas al AP.
ii. Ambas XO's conectadas a la MESH.
iii. Una XO conectada al AP y la otra XO conectada a la MESH.

Estas pruebas nos fueron de utilidad para la depuración general del sistema, de todos modos, en las pruebas realizadas en el LATU surgieron algunas otras incidencias propias del cambio de contexto.

2. Deploy Windows en Servidor de Desarrollo

Todo el desarrollo del servidor fue realizado en ambiente Microsoft Windows utilizando WAMP (Windows Apache MySQL PHP) una solución de rápida configuración para levantar en Windows una base de datos MySQL y un servidor Apache con interprete PHP. Por ende,

las pruebas iniciales también se realizaron contra este servidor, permitiendo hacer todas las depuraciones necesarias en el proceso de recepción de información, actualización de parámetros e interfaz gráfica de usuario.

Dada la tecnología seleccionada para el servidor, nos aseguramos que la decisión de utilizar este software en particular, no afectaría la puesta en producción en el ambiente Linux ya que su portabilidad es ampliamente conocida.

Las mismas apreciaciones aplican para el servidor FTP/SFTP utilizado para las prueba, ya que son protocolos ampliamente difundidos y sencillos con diversas implementaciones libres para cualquier plataforma.

3. Deploy Linux en Servidor de Desarrollo

En vista de que el cliente nos confirmó que el servidor del LATU iba a ser una distribución Debian de Linux, fue necesario realizar la instalación del servidor en una máquina de las mismas características.

En el proceso de cambio de plataforma hubo algunas dificultades con la definición de los permisos sobre algunas carpetas de la aplicación, y yendo específicamente al sistema detectamos algunos inconvenientes con el manejo de paths, que por su modo de implementación no eran soportados por Linux. Una vez subsanados estos inconvenientes, el sistema quedó completamente operativo en la plataforma de producción.

4. Pruebas con varias XO's en el LATU

La primera prueba realizada en el LATU estuvo plagada de inconvenientes, en una primera instancia por problemas de conectividad con el servidor a causa de configuraciones erróneas en el mismo que afectaban a las XO del LATU y no habían presentado problemas con nuestras XO de prueba. Paralelamente se presentaron problemas de conectividad de las mismas XO a los AP presentes, generados por el propio sistema operativo y su configuración. Con ayuda del equipo técnico presente se logró finalmente detectar los problemas, corregirlos, y poner en marcha la fase de depuración de nuestro sistema. Como es lógico en una primera prueba, se

presentaron imprevistos en nuestro sistema que insumieron parte del escaso tiempo disponible para las mismas.
De todas formas, algo no menor a destacar de esta visita fue la interacción con el equipo de trabajo del Plan Ceibal, y los aportes brindados por los mismos, ya sea en materia de detalles adquiridos en la práctica diaria, así como ultimar detalles de los requerimientos del sistema.

En la segunda prueba, también se presentaron algunos problemas: el servidor que llevamos nosotros (WAMP preinstalado y preconfigurado) poseía un firewall que si bien nunca había generado inconvenientes con nuestras de XO de prueba, sí lo hizo con las XO suministradas por el LATU.
Yendo a las pruebas del cliente, se detectó un comportamiento extraño en el proceso de detección de portales, el cual está implementado mediante un socket en cada máquina que escucha y responde solicitudes de las XO contiguas. A causas que solo pudimos atribuir a las diferentes versiones de software, nuestras XO's se contestaban entre sí y las XO's del LATU también se contestaban ente sí, pero no funcionaba el socket entre las nuestras y las del LATU. (consultados los técnicos del LATU nos comentaron que comportamientos como esos son comunes al trabajar con grandes diferencias de versiones).

La decisión de mantener la versión durante todo el proyecto fue en común acuerdo con el tutor para evitar complicaciones en la etapa de desarrollo, pero en esta instancia de pruebas se hizo inminente la necesidad de actualizar la versión de software y firmware en nuestras XO´s.

Más allá de los inconvenientes antes mencionados se logró avances significativos con respecto a la primer prueba en el LATU, pudiéndose instalar correctamente el cliente en 5 XO y probar distintos escenarios de conectividad, registrando estos datos en la base.

La siguiente prueba en el LATU tuvo como objetivo primordial el deploy del sistema en el servidor Linux de producción, lo cual se logró de forma satisfactoria una vez vencidos pequeños detalles de conectividad y permisos de usuario. A su vez, se realizó la instalación del software cliente en 3 XO propiedad del LATU, y una instrucción general al personal técnico sobre el uso e instalación del sistema. Finalmente se dejó el sistema completo funcionando en el mencionado servidor y en las 3 XO, de manera de poder realizar pruebas en los días siguientes por el personal ténico.

5. Pendientes

Quedó pendiente a la fecha del presente documento una prueba más extensa, al menos en una escuela, para lo cual el LATU dispone del servidor instalado y el cliente listo para distribuirlo a las XO's. Este tipo de pruebas consideramos que pueden llegar a ser más representativas; una red densa con mucha actividad de tráfico puede generar situaciones que no hayan aparecido en las pruebas realizadas.

Capítulo 6

Conclusiones

A continuación se presenta un análisis del cumplimiento del cronograma, los criterios de éxito y las conclusiones finales que se obtuvieron una vez finalizado el proyecto.

1. Gestión de Tiempos

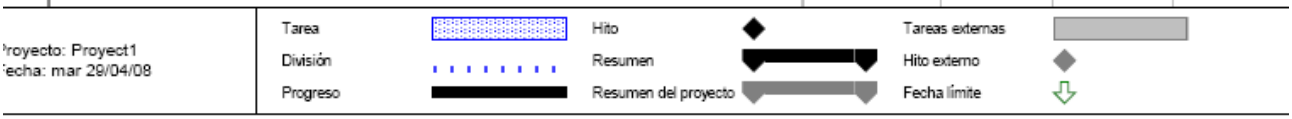
Podemos decir en términos generales que el cronograma fue ampliamente respetado, cumpliendo con la fecha final de entrega así como también con todas las fechas pautadas para entregables intermedios.

Si bien hubo momentos dónde existió un desfazaje con el cronograma, estos fueron gestionados gracias a buffers de tiempo planificados. En este aspecto, se hace referencia al entrar en contacto con el software de las máquinas, que de no contar con los equipos que nos proporcionaron, hubiese sido mucho más extenso de lo estimado.

En lo que refiere a las pruebas finales principalmente en escuelas, coincidió con que el proyecto Ceibal estaba distibuyendo los laptops, haciendo imposible acordar la fecha de los mismos.

A continuación se muestra el diagrama de GANTT junto con el Cronograma del proyecto

Id	Nombre de tarea	Trabajo	Duración	Comienzo	Fin
1	**Proyecto Fin de Carrera**	**2.012,63 horas**	**253 días**	**mar 15/05/07**	**mié 30/04/08**
2	**Objetivo 1 (Entrega de Monografía IEEE 802s)**	**468 horas**	**89 días**	**mar 15/05/07**	**sáb 15/09/07**
3	Búsqueda bibliográfica – 802.11, 802.11s, OLPC	72 horas	10 días	mar 15/05/07	lun 28/05/07
4	Estudio de la misma	180 horas	50 días	mar 29/05/07	lun 06/08/07
5	Realización de una monografía	180 horas	50 días	mar 29/05/07	lun 06/08/07
6	Elaborar la presentación del proyecto	36 horas	5 días	mar 07/08/07	lun 13/08/07
7	Realización de una breve presentación pública (LATU)	0 horas	0 días	vie 07/09/07	vie 07/09/07
8	Presentación de los trabajos realizados.	0 horas	0 días	sáb 15/09/07	sáb 15/09/07
9	**Objetivo 2 (Entrega de un Software Básico)**	**290,4 horas**	**44 días**	**mar 14/08/07**	**vie 12/10/07**
10	**Estudio de las distintas alternativas posibles en los lenguajes a utilizar**	**28,8 horas**	**3 días**	**mar 14/08/07**	**jue 16/08/07**
15	**Interiorización en el software de los equipos (Linux Fedora)**	**93,6 horas**	**13 días**	**vie 17/08/07**	**mar 04/09/07**
19	**Desarrollo de un software básico**	**117,6 horas**	**13 días**	**mié 05/09/07**	**jue 20/09/07**
24	**Pruebas de la aplicación**	**50,4 horas**	**7 días**	**vie 21/09/07**	**lun 01/10/07**
28	Entrega de software desarrollado	0 horas	0 días	vie 12/10/07	vie 12/10/07
29	**Objetivo 3 (Entrega del Software Funcionando)**	**708 horas**	**99 días**	**lun 01/10/07**	**vie 15/02/08**
30	**Estudiar la viabilidad de reutilización de software existente**	**36 horas**	**5 días**	**lun 01/10/07**	**vie 05/10/07**
33	**Definir exactamente el alcance y las funcionalidades del software a implementar**	**132 horas**	**15 días**	**lun 08/10/07**	**vie 26/10/07**
39	**Codificación del sistema.**	**396 horas**	**55 días**	**lun 29/10/07**	**vie 11/01/08**
44	**Pruebas del sistema**	**144 horas**	**10 días**	**lun 28/01/08**	**vie 08/02/08**
48	Entrega del software desarrollado	0 horas	0 días	vie 15/02/08	vie 15/02/08
49	**Objetivo 4 (Entrega del Proyecto Terminado - Defensa)**	**424,8 horas**	**53 días**	**vie 15/02/08**	**mié 30/04/08**
50	**Realización de los cambios sugeridos en la presentación del prototipo**	**100,8 horas**	**14 días**	**vie 15/02/08**	**mié 05/03/08**
54	**Documentación del proyecto**	**252 horas**	**25 días**	**jue 06/03/08**	**mié 09/04/08**
60	**Exposición del proyecto**	**72 horas**	**14 días**	**jue 10/04/08**	**mié 30/04/08**
64	**Documentación**	**121,43 horas**	**253 días**	**mar 15/05/07**	**mié 30/04/08**
65	Recopilación de Información	121,43 horas	253 días	mar 15/05/07	mié 30/04/08



Página 1

Figura 33 – Cronograma

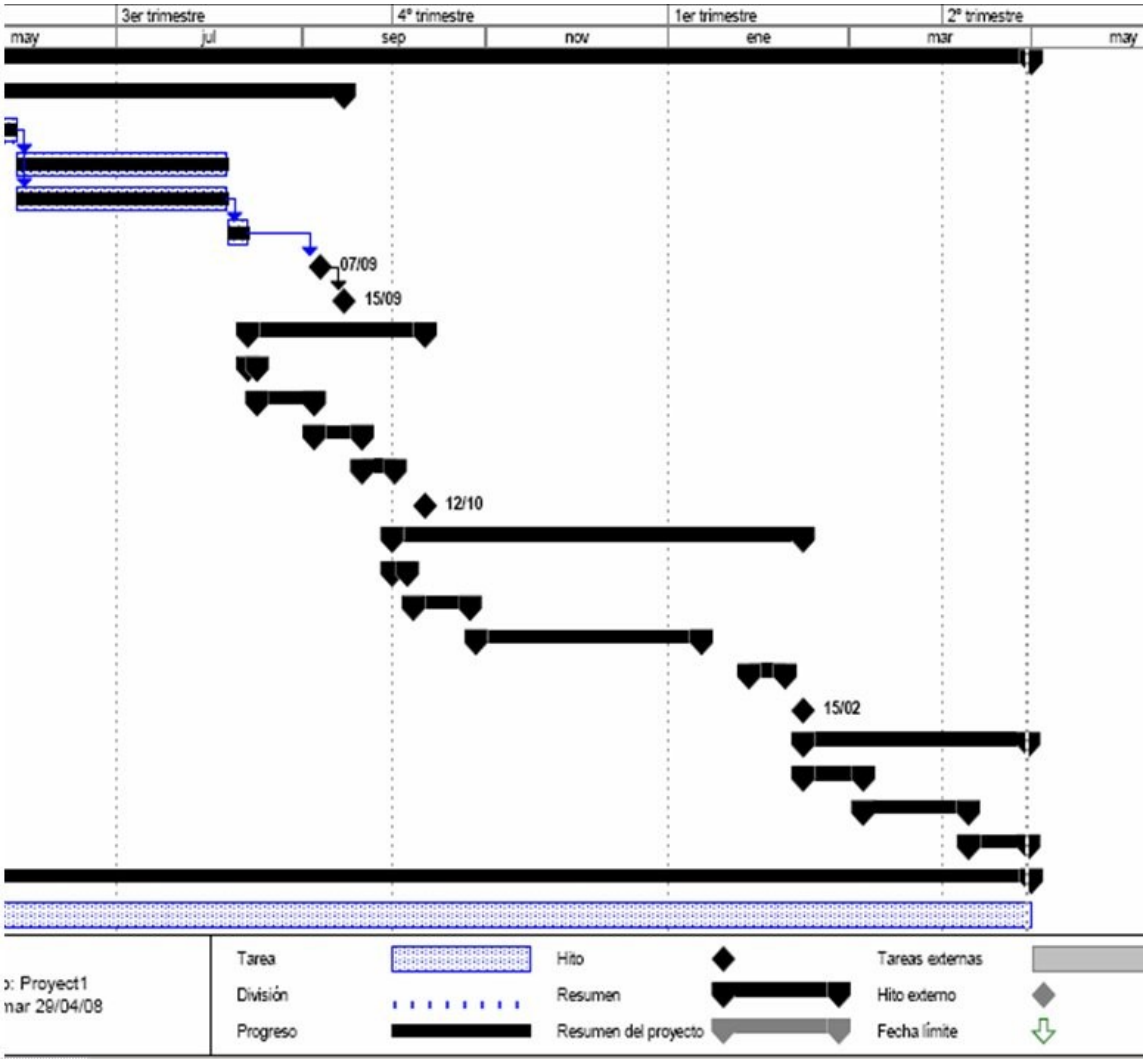


Figura 34 – Diagrama de Gantt

2. Criterios de Éxito

En este apartado se evalúan los criterios de éxito que se plantearon en la planificación.

2.1. Impacto del sistema

Como ya se ha remarcado en la presente documentación, se tomó como premisa durante todo el desarrollo la optimización de los recursos en las laptops y las redes, habiendo sido un criterio decisivo para protocolos, períodos, formatos, lenguajes, etc. Dada la versatilidad del sistema, el medir este impacto es muy relativo.

El impacto en el servidor variará según las características del mismo, estando ampliamente documentado y probado que un servidor Web con las características del nuestro es muy escalable. Además, está pensado que el sistema se ponga en producción en distintos servidores distribuidos, y no centralizando todos los procesos relativos a todas las XO del país en un solo servidor, lo cual claramente suaviza los requerimientos de recursos de los mismos.

Desde el lado de las XO, el impacto en el rendimiento de las mismas, así como el efecto en la performance de la red, depende completamente de parámetros que son configurables por el usuario, como ser la cantidad de datos a recolectar, la frecuencia de recolección, tamaños máximos de uso en disco, etc. A su vez, los recursos de los laptops están variando constantemente, habiendo aumentado más de una vez incluso durante el transcurso de nuestro proyecto.

Sin embargo, en escenarios de pruebas que exceden en exigencia de recursos a lo que se espera sea común en la práctica, no se perciben diferencias de rendimiento en las XO, y los valores estadísticos del uso de disco, procesador y memoria del sistema son los usuales. El programa cliente se mantuvo en ejecución varios días en laptops utilizadas por el personal técnico, no habiéndonos manifestado los mismos ningún comentario negativo en cuanto al funcionamiento. Consideramos que la conformidad del cliente es uno de los principales indicadores a tomar en cuenta a la hora de evaluar el rendimiento del sistema.

2.2. Prueba de volumen

Se planteó como criterio de éxito hacer una prueba controlada en una red de entre 10 y 100 máquinas donde técnicos especializados evaluarían el funcionamiento total del sistema. Se manejó a lo largo del proyecto la posibilidad de implantar el sistema en una escuela, pero por un tema de indisponibilidad de tiempo de ambas partes fue imposible.

El proyecto Ceibal está recién implantándose y es lógico pensar que existen otras prioridades, en particular, hacer la distribución de todas las máquinas a los niños de las escuelas del país. Se destaca la predisposición del grupo que trabaja en el LATU para facilitarnos más máquinas, pero el total de máquina con que dispusimos en la pruebas efectuadas no alcanzaban para complicar cabalmente con este criterio.

Es así que tuvimos cambiar el alcance del criterio, realizando una prueba con un volumen menor, donde se evaluó el funcionamiento del sistema en una red de al menos 5 laptops. En este contexto el sistema no presentó problemas significativos, mas allá de los que fueron expuestos en el capítulo de puesta en producción y que fueron solventados para la versión final del sistema.

2.3. Documentación y Manuales

En lo que refiere a los productos entregados, se tuvo siempre presente el criterio de desarrollar una interfaz amigable e intuitiva, acompañada de un manual de usuario accesible desde la propia aplicación. Esto se complementa con una guía para la instalación del servidor, detallando las instrucciones necesarias para la creación de la base de datos y el deploy del sistema.

De cara a la instalación de la aplicación en las XO's, se desarrolló un programa instalador, que permite la configuración, instalación y desinstalación del sistema en forma automatizada y configurable, permitiendo por ejemplo la ejecución del cliente una vez que las máquinas se enciendan; todo esto acompañado con la guía de instalación correspondiente.

A lo largo de la documentación del proyecto se explicó en forma más que detallada la arquitectura del sistema, el modelo de base de datos y se anexó documentación de los programas, cumpliendo con todo lo estipulado en este criterio de éxito.

3. Conclusiones

Como primer y más importante conclusión se cumplió en tiempo y forma con los objetivos propuestos al inicio del proyecto. Afortunadamente, el haber planteado un cronograma definido con fechas de entregas intermedias facilitó la tarea de desarrollo.

Este es un proyecto que contiene muchos elementos de software, por lo que fue necesario adentrarse poco a poco en los distintos lenguajes de programación. Desarrollar una aplicación intermedia, simple, de menor magnitud que la propuesta, fue de gran ayuda para conocer más el funcionamiento de dichos lenguajes y para tomar como base para la implementación definitiva.

En lo que refiere a redes inalámbricas, las redes mesh están recién comenzando a difundirse, y es por esto que el acceder a información respecto al tema no fue una tarea fácil. Papers, presentaciones, y el Draft incompleto de la norma fue con lo que se contó como material de referencia para conocer más en profundidad la temática de estas redes, pensando además que mucha de la información que contenía podría en un futuro cambiar. De todas formas se focalizó el estudio de los protocolos de ruteo y descubrimiento de rutas que son de los conceptos mejor desarrollados y cuyo funcionamiento es muy poco probable que varíe al momento de estandarizarse la norma.

Se suma a lo antedicho que los laptops del proyecto OLPC no implementan exactamente lo que se plantea en el borrador, sino que son soluciones propietarias que toman muchas veces sólo algunos conceptos. El estudio del funcionamiento de los XO´s demandó un tiempo considerable pero que de todas formas estaba previsto en el cronograma.

La ayuda que nos brindó el proyecto OLPC facilitando dos máquinas, fue beneficiosa para todo el desarrollo de nuestro proyecto. De no contar con los XO´s hubiese sido necesario trabajar con emuladores de las máquinas, instalando imágenes del sistema operativo para poder hacer pruebas. Inclusive consideramos que la viabilidad del proyecto hubiese estado comprometida al no contar con dichos XOs, ya que reproducir los distintos escenarios de testeo y/o coordinar numerosas pruebas en el LATU con nuestros compromisos personales y las restricciones de horario del mismo hubiese sido una tarea imposible.

Consideramos, y es la impresión que recogimos de las muestras parciales del sistema al personal técnico del Plan Ceibal y del tutor del proyecto, que se logró realizar un sistema

suficientemente maduro y a la altura de los requerimientos planteados, y que podrá ser de una utilidad real en el futuro.

Como conclusión final destacamos la buena coordinación que hubo entre los compañeros de trabajo durante todo el desarrollo del proyecto para poder obtener un sistema completo, de gran magnitud y en funcionamiento en los tiempos estipulados.

4. Trabajos a Futuro

Como fue mencionado repetidas veces a lo largo del documento estas redes no han llegado aún a su apogeo, por lo que este proyecto en particular abre un abanico de opciones a trabajos futuros.

En cuanto a la aplicación, existen mejoras y módulos a agregar. Por ejemplo, el contar con una herramienta que dado los datos recolectados genere una imagen con el grafo de la red sería de gran utilidad. También el tema de reportes no fue muy ahondado en el proyecto, un módulo de generación de reportes y gráficos hubiese sido muy útil. Dentro de todas las opciones, otra a considerar es el tema de las alarmas, por ejemplo, sería factible el envío de correos electrónicos a los responsables técnicos al momento de detectase algún valor con rangos inaceptables de los valores registrados.

Se desarrolló la aplicación de tal forma que el agregar funcionalidades extra, más en particular funcionalidades de recolección de datos de la red, implica agregar dichas funciones en el programa sin tener que realizar grandes modificaciones en el sistema, haciéndola una herramienta medianamente extensible.

Referencias

[1] *A Scheme for Dynamic Monitoring and Logging of Topology Information in Wireless Mesh Networks*, Cristian Popi, Olivier Festor, MADYNES - INRIA Nancy Grand Est, popicris/festor@loria.fr.

[2] *Advanced Python Topics*, Dave Kuhlman, http://www.rexx.com/~dkuhlman/python_201/python_201.html.

[3] *CSV Reader*, http://www.csvreader.com/csv_format.php

[4] *Dev Shed*, http://www.devshed.com/c/b/Python/

[5] *Dynaim Drive*, http://www.dynamicdrive.com/.

[6] *IEEE P802.11s™/D1.00,Draft Amendment to Standard for InformationTechnology - Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: Amendment: ESS Mesh Networking*, 802.11 Working Group of the LAN/MAN Committee, Institute of Electrical and Electronics Engineers.

[7] *IEEE 802.11s ESS Mesh Networking*, Prof. Young-Bae Ko (youngko@ajou.ac.kr), Ubiquitous Networked Systems (UbiNeS) Lab (http://uns.ajou.ac.kr).

[8] *IEEE 802.11s Tutorial Overview of the Amendment for Wireless Local Area Mesh Networking, W. Steven Conner*, Intel Corp. Jan Kruys, Cisco Systems Kyeongsoo (Joseph) Kim, STMicroelectronics Juan Carlos Zuniga, InterDigital Comm. Corp, IEEE 802 Plenary, Dallas, Noviembre 13, 2006.

[9] *Iptables Tutorial 1.2.2, Oskar Andreasson*, http://iptables-tutorial.frozentux.net/iptables-tutorial.html

[10] *Lsmeshd - Meshpotato*, Javier Cardona, http://www.cozybit.com/projects/lsmesh

[11] *Manuales PHP*, http://www.php.net/docs.php, http://www.desarrolloweb.com/php, http://www.webtaller.com/construccion/lenguajes/index/php/.

[12] *Mesh Networks for Digital Inclusion - Testing OLPC's XO Mesh Implementation*,
Ricardo Campanha Carrano, Michail Bletsas, Luiz Claudio Schara Magalhães, Departamento de Ingenieria de Telecomunicaciones, Universidade Federal Fluminense (UFF), Rio de Janeiro, Brazil.

[13] *Mesh Portal Utils*, Javier Cardona, http://www.cozybit.com/projects/mpp-utils

[14] *Mesh Technology enabling Ubiquitous Wireless Networks*, Guido R. Hiertz, Sebastian Max, Erik Weiß, Lars Berlemann, Dee Denteneer, Stefan Mangold, Chair of Communication Networks, Faculty 6, RWTH Aachen University.

[15]*OLPC Wiki*, http://wiki.laptop.org/

[16] *Pexpect*, Noah Spurrier, http://pexpect.sourceforge.net/
[17] *Proposed Routing for IEEE 802.11s WLAN Mesh Networks*, Michael Bahr, Siemens Corporate Technology, Information & Communications, bahr@siemens.com
[18] *Proyecto Ceibal*, http://www.ceibal.edu.uy.
[19] *Python Programming Language*, http://www.python.org.
[20] *Redes Wi-Fi en malla*, Evelio Martínez Martínez, José Antonio García Macias, Revista RED, Septiembre 2006.
[21] *The Standard Python Library*, http://effbot.org/librarybook/
[22] *Wireless Mesh Networks*, Prasant Mohapatra, Department of Computer Science, University of California.
[23] *Xml*, http://www.xml.com

ANEXO I

Documentación de Programas

1. Módulo Servidor

Se presenta a continuación una breve descripción de los programas implementados del lado del servidor, tanto para atender los procesos de carga de información y actualización de parámetros, así como las páginas dinámicas de la interfaz gráfica de usuario.

1.1. Carga de Información y Actualización de Parámetros.

Existen dos procesos invocados por los laptops al comunicarse con el servidor: un proceso de actualización de parámetros (http://servidor/olpc/procesos/parametros.php), donde se consulta si hubo cambios en la parametrización del archivo de configuración y de haberlos actualiza el archivo config.ini local. El otro proceso es el de envío de la información (http://servidor/olpc/procesos/procesar.php) que es invocado por los laptops una vez realizado el envío vía FTP/SFTP del archivo al servidor.

Las clases implementadas para dichos procesos son:

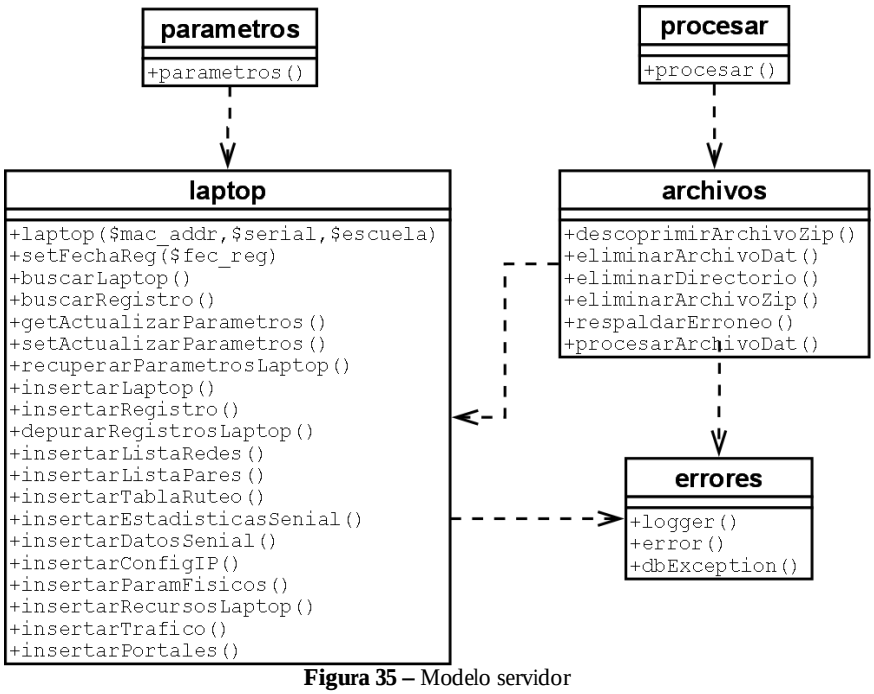


Figura 35 – Modelo servidor

laptop - Clase que implementa todos los métodos vinculados con el alta, baja y modificación de laptops y registros de información enviada desde la aplicación cliente.

archivos - Implementación de todas las funciones referentes al manejo de los archivos.

errores - Implementación de todas las funciones referentes al logueo de errores y manejo de excepciones de base de datos.

procesar - Proceso encargado del procesamiento de los archivos enviados por los laptops. Este proceso recibe por POST la dirección MAC de la máquina que lo invoca y busca en el directorio en el que está configurado el servidor FTP/SFTP, los archivos identificados con dicha MAC. A continuación procesa cada uno de estos, y los descomprime en una carpeta temporal. Como resultado de la descompresión se obtienen los archivos de datos, los que se procesan uno a uno, generando una instancia de la clase *laptop* para cargar de la información en la base de datos. Por último se realiza la eliminación de los archivos de datos, archivos comprimidos y carpetas temporales.

Conjuntamente con la dirección MAC, el proceso recibe el número de serie del laptop, con lo cual, si no encuentra en la base de datos un laptop registrado con esa MAC y número de serie lo inserta, previo al ingreso de la información registrada.

El tercer parámetro pasado por POST a este proceso es la fecha de envío, la que se utiliza para el ajuste de la fecha de registro de la información.

parámetros - Proceso que atiende las peticiones de los laptops en busca de modificaciones en los parámetros de configuración. Si por la interfaz gráfica de usuario se efectuó alguna modificación de los parámetros, se prende una bandera indicando que dicho laptop necesita actualizar el archivo local. Este proceso recibe por POST la dirección MAC y número de serie, con lo cual verifica si en necesario enviar la información actualizada; de ser así, envía un string predefinido al laptop con los nuevos datos para su procesamiento.

1.2. Interfaz Gráfica de Usuario

No se hizo un diseño orientado a objetos para la Interfaz Gráfica de Usuario, sino que se implementaron una serie de páginas dinámicas en PHP.

Se presenta a continuación un inventario de las páginas implementadas para cada una de las opciones del sistema agrupadas por módulo.

a. Consultas

Páginas para la búsqueda de laptops y consulta de la información recolectada.

SUB-MODULO	PÁGINA	DESCRIPCIÓN
Buscar Laptop	buscar_laptops.php	Página principal de búsqueda de laptops mediante la aplicación de diferentes filtros.
Registros Laptop	registros_laptop.php	Página de resumen de información disponible para un laptop y una página por cada tipo de dato registrado con la lista de fechas en que se tomaron las muestras.
	ver_registro_confip.php	
	ver_registro_datossen.php	
	ver_registro_estsen.php	
	ver_registro_paramf.php	
	ver_registro_pares.php	
	ver_registro_portales.php	
	ver_registro_recursos.php	
	ver_registro_redes.php	
	ver_registro_ruteo.php	
	ver_registro_trafico.php	
Detalle Registros Laptop	ver_detalle_confip.php	Detalle de la información registrada para un laptop en
	ver_detalle_datossen.php	

SUB-MODULO	PÁGINA	DESCRIPCIÓN
	ver_detalle_estsen.php	cada instante de tiempo.
	ver_detalle_paramf.php	
	ver_detalle_pares.php	
	ver_detalle_portales.php	
	ver_detalle_recursos.php	
	ver_detalle_redes.php	
	ver_detalle_ruteo.php	
	ver_detalle_trafico.php	
Últimos Registros	cons_ultimos_reg.php	Página que recupera los últimos 50 envíos al servidor ordenados por fecha en forma descendente.
Laptops sin Enviar	cons_laptops_sin_enviar.php	Página que permite consultar los laptops que no han enviado información al servidor en plazo de días determinado.
Estadísticas Escuela	perfiles_lista_escuelas.php perfil_escuela.php fechas_perfil.js	Páginas para la búsqueda de escuela y posterior consulta de estadísticas.

b. Mantenimiento

Páginas para el mantenimiento de las principales entidades del sistema.

SUB-MODULO	PÁGINA	DESCRIPCIÓN
Escuelas	lista_escuelas.php alta_escuelas.php modificar_escuelas.php baja_escuela.php	Páginas vinculadas al alta, baja y modificación de escuelas.
Responsables Técnicos	lista_responsables.php modificar_responsables.php baja_responsable.php	Páginas vinculadas al alta, baja y modificación de responsables técnicos.
Carga Lote Laptops	cargar_laptops.php insertar_lote.php	Páginas para la carga de laptops mediante el ingreso de números de serie inicial y final del lote.
Carga Archivo Laptops	cargar_laptops2.php subir_archivo.php	Páginas para la carga de laptops mediante el ingreso de un archivo con la lista de números

		de serie.
Ubicar Escuelas	laptops_por_escuela.php modif_escuela_laptop.php	Páginas para la ubicación y reubicación de laptos en las escuelas.

c. Parametrización

Páginas para la parametrización del archivo de configuración local.

SUB-MODULO	PÁGINA	DESCRIPCIÓN
Parámetros por Defecto	parametros_generales.php	Página para setear los parámetros que por defecto tomarán las laptops que envíen información sin estar registradas.
Parametrización Masiva	act_masiva_sel_escuela.php act_masiva_sel_parametros.php	Páginas para setear los parámetros de un grupo de laptops.
Parametrización Individual	parametros_laptop.php	Páginas para setear los parámetros de un laptop específico.

2. Módulo Cliente

Se presenta a continuación un esquema y descripción de las clases definidas en el programa cliente. Se describe también en esta sección el funcionamiento de las distintas partes de software de este módulo.

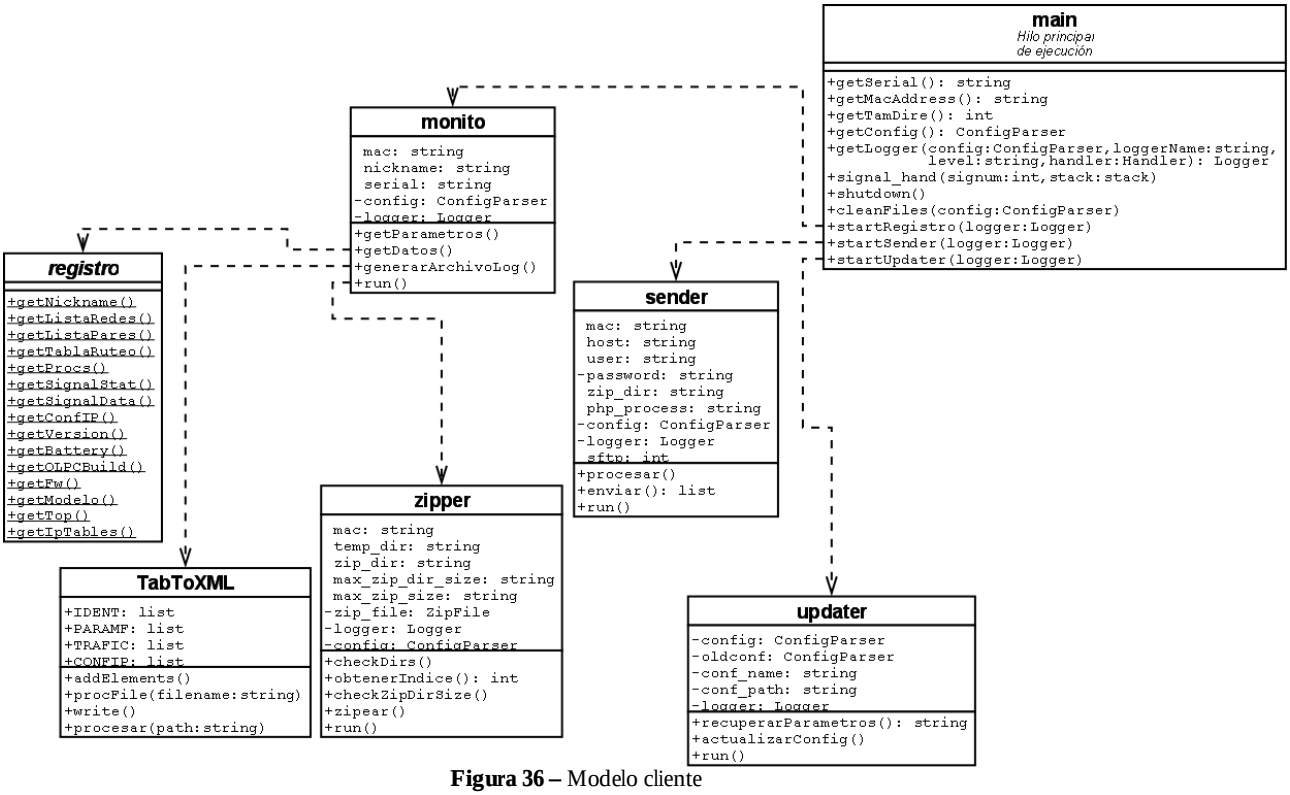


Figura 36 – Modelo cliente

2.1. Descripción

El programa consiste de un hilo principal de ejecución que será el encargado de setear las variables globales, inicializar temporizadores, crear los threads de ejecución para cada tarea, realizar tareas de finalización, etc. Contiene a su vez funciones comunes y útiles para todas las otras clases (getLogger, getConfig, getMacAddress, getSerial, getTamDire).

startRegistro, startSender, startUpdater - Son invocadas al comienzo del programa, y luego de cada ejecución de la tarea en cuestión, iniciando un thread del tipo Timer de manera de poder controlar el tiempo hasta la próxima ejecución de la tarea, la cual a su vez se iniciará en un thread aparte. Es decir, las tareas de “sender”, “updater” y “registro” se realizarán en threads independientes.

shutdown - Realiza tareas de limpieza y finalización al detectarse una finalización del programa mediante interrupciones desde el teclado o mediante una señal del sistema (TERM – HUP signal).

2.2. Clase Sender

Las instancias de esta clase serán las encargadas de realizar el envío de datos al servidor, ya sea mediante FTP o SFTP, y además finalmente ejecutar la orden de procesado de los datos. Los archivos que sean correctamente enviados al servidor, serán eliminados del directorio local.

2.3. Clase Updater

Las instancias de esta clase serán las encargadas de realizar la actualización de parámetros locales de configuración, consultando al servidor configurado, y realizando el backup del archivo de configuración en caso de una recepción exitosa de parámetros.

2.4. Clase Monito

Las instancias de esta clase serán las encargadas de realizar la recolección de datos, la generación del archivo de datos, la creación (opcional) del archivo XML local, y la compresión de los archivos.

Para la recolección de datos utiliza la clase auxiliar “*registro*” que contiene métodos estáticos (se ejecutan sin necesidad de existir una instancia de la clase registro). Se optó por realizar esta tarea de recolección de datos propiamente dicha en una clase aparte, para una mayor portabilidad y reusabilidad.

A su vez, para la generación del archivo XML local se utiliza una también una clase aparte (*TabToXml*), que recibirá los parámetros correspondientes. El código de esta clase está fuertemente ligado con la estructura de los archivos DAT, por lo cual deberá ser modificada en forma acorde al realizarse modificaciones en el formato de los archivos de datos.

La compresión la realiza la clase "*zipper*", la cual comprimirá los archivos generados y respetará las limitaciones de tamaño del archivo y del directorio impuestas en la configuración. Si bien en la implementación actual este proceso de compresión se realiza cada vez que se genera un archivo de datos para optimizar espacio en disco, la clase está diseñada para comprimir múltiples archivos en una única ejecución, pudiendo por ejemplo ser utilizada para comprimir solo al momento de enviar.

Nota: En la implementación de la transferencia de archivos al servidor vía SFTP se utilizaron las bibliotecas PEXPECT de Python (disponibles en la red bajo licencia gratuita y de código libre) la cual resultó de gran utilidad para interactuar con el servidor, capturando las respuestas del mismo y respondiendo en función de ellas.

ANEXO II

Manual de Instalación

En este apartado se describen los distintos pasos que se deben seguir para instalar, configurar y desinstalar las distintas partes de la aplicación así como las consideraciones que deben ser tomadas en cuenta a la hora de poner en producción los módulos.

1. Módulo Cliente

El cliente se proporciona en un único archivo ZIP. Para su instalación, se deberán seguir los siguientes pasos:

1. Extraer el archivo ZIP a una carpeta temporal
2. De ser necesario, modificar el archivo config.ini
3. Ejecutar el instalador escrito en Python llamado “setup.py”, con la opción correspondiente

Las opciones del instalador son las siguientes:

“`install`”:

- Preguntará por la carpeta de instalación
- Copiará a la carpeta elegida los siguientes archivos:
    - monito.py
    - port_cliente.py
    - port_servidor.py
    - pexepect.py
    - config.ini
- Modificará el archivo “/etc/rc.local” de manera de ejecutar el programa en cada inicio del sistema.

- Si la llamada se realiza con el parámetro adicional "`des`" (es decir, "`python setup.py install des`"), la instalación se ejecutará en modo desatendido, instalando el cliente en la carpeta por defecto (especificada en el código de setup.py) y no pedirá confirmaciones.

"`no-init`":

- Se quitará la información agregada en "/etc/rc.local" de manera que el cliente no se ejecute automáticamente al inicio del sistema.

"`uninstall`":

- Se quitará la información agregada en "/etc/rc.local" de manera que el cliente no se ejecute automáticamente al inicio del sistema.
- Se eliminarán todos los archivos relacionados con el programa, incluyendo archivos de log, datos, configuración, etc.
- Si la llamada se realiza con el parámetro adicional "des" (es decir, "`python setup.py uninstall des`"), la desinstalación se ejecutará en modo desatendido, terminando la ejecución, borrando los archivos instalados y quitando también del inicio del sistema sin pedir confirmaciones.

Luego de terminada la instalación, se recomienda borrar el directorio temporal, siempre y cuando no se haya elegido el mismo como carpeta de instalación (no recomendable)

El "config.ini" copiado es indispensable para la correcta ejecución del programa, y debe contar con direcciones válidas de los servidores asignados para actualización de parámetros y recolección de datos. Estos cambios deberán ser hechos por los responsables técnicos antes de comenzar la ejecución.

1.1.Ejecución

El cliente podrá ser ejecutado desde línea de comandos como cualquier programa, en caso de que no se desee ejecutarlo al inicio.

En la primera ejecución, el cliente generará los siguientes archivos:

- *log.log* – Archivo de log conteniendo todos los eventos relevantes generados por el cliente. Su tamaño está limitado, y la cantidad de "roll-outs" que se le realizarán también. Ambas configuraciones, además del propio nombre del archivo log, se encuentran en el "config.ini". El nivel de detalle de este logueo depende de una variable definida en el código principal monito.py.
- *lista.txt* – Archivo auxiliar utilizado para guardar temporalmente una lista de vecinos y es generado por el script "port_cliente.py".
- Se generarán los directorios temporales para los archivos comprimidos a enviar, y para los archivos de datos a generar, siendo los nombres de los mismos configurables mediante el config.ini
- Se generarán los archivos PID correspondientes a cada uno de los 3 scripts python en ejecución, en la carpeta estándar (/var/run), que serán borrados al terminar la ejecución.

Nota importante: Para el correcto funcionamiento del programa, es indispensable que los parámetros tanto en el servidor como en el cliente estén correctamente seteados al momento de la primera actualización. Es decir, si al momento de enviar datos existiera una dirección obsoleta / no accesible del campo server en el archivo de configuración, luego de actualizar sus parámetros el cliente quedaría "aislado" de su contraparte remota hasta una intervención administrativa. Así como también, si el servidor envía al cliente información no válida se generarían problemas, dejando aislado nuevamente a su otra parte.

Al ejecutarse en una máquina por primera vez, y conectarse a un servidor de actualización (válido) el mismo agregará el XO a su lista de XOs conocidos, pero no le enviará parámetros nuevos hasta no ser cambiado alguno de los mismos desde la interfaz web.

1.2. Archivo de configuración

A continuación se ofrece una breve descripción de las opciones disponibles en el archivo config.ini, por ejemplo en caso de modificación local. Sin embargo, el sistema cuenta con una interfaz web para la actualización de los mismos, que es el método recomendado para realizar estos cambios, y resulta más cómodo e intuitivo.

1.2.1. Sección “Server” (parámetros de actualización / envío)

- addr_act_param – URL válida para el servidor de actualización de parámetros. La dirección debe apuntar directamente al script “parametros.php” ubicado en el servidor.
- php_process – URL válida para el servidor de datos, mediante la cual se le indicará al servidor que hay disponibles datos de la XO en cuestión para actualizar la base remota. La dirección debe apuntar directamente al script “procesar.php” ubicado en el servidor.
- host – URL del servidor FTP utilizado para el envío de los datos. Debe incluir o bien una dirección IPv4 válida o una dirección DNS. No debe incluir el prefijo “ftp://” ni ningún otro tipo de información y se usará el puerto por defecto (21).
- user – Usuario de identificación en el servidor FTP
  valor por defecto: mona
- pass – Password de identificación en el servidor FTP
  valor por defecto: mona
- send_delay – Parámetro que controla el tiempo (en segundos) entre sucesivos intentos de envío de datos al servidor por parte del cliente. No es recomendable setear este parámetro en valores bajos (menores a 5 minutos), pero no es controlado explícitamente por el programa.
  valor por defecto: 600
- upd_par_time – Parámetro que controla el tiempo (en segundos) entre sucesivos intentos de actualización de parámetros por parte del cliente.
  valor por defecto: 600
- secureftp – Parámetro que indica el protocolo utilizado para la transferencia de los archivos al servidor. (0 = FTP, 1 = SFTP)
  valor por defecto: 1

- cleandelay – Especifica el tiempo (en días) de permanencia de los datos en el servidor para la XO en cuestión. Este parámetro es mantenido y utilizado puramente por el servidor, y en el archivo de configuración es de "solo lectura"

1.2.2. Sección "portales" (configuración de "port_cliente" y "port_servidor")

- sample_time: parámetro que controla el tiempo en segundos entre sucesivas actualizaciones de la lista de vecinos alcanzables
  valor por defecto: 300 (seg)

- pack_timeout: tiempo en segundos en que el socket levantado permanecerá escuchando una respuesta
  valor por defecto: 10 (seg)

1.2.3. Sección "registro" (configuración de los datos a registrar)

- tiempo_registro – Indica el tiempo (en segundos) entre sucesivos registros de datos. Debido al tiempo que insumen algunas tareas del registro de datos, y a los efectos de no sobrecargar las XO, no es recomendable setear este parámetro en valores menores a 20 seg.
  valor por defecto: 300

- xml – Indica si se generará un archivo XML local con la información de registro
  valor por defecto: 0

- xmlname – Indica el nombre del archivo XML generado
  valor por defecto: salida

- xmloverwrite – Indica si se sobreescribirán los archivos XML anteriores, o si se generará uno cumulativo.
  valor por defecto: 1

- Los siguientes parámetros pueden tomar los valores: 0=no registrar, otro=registrar

    - ind_reg_paramf
    - ind_reg_pares
    - ind_reg_top
    - ind_reg_car_senial
    - ind_reg_redes
    - ind_reg_trafico
    - ind_reg_ruteo
    - ind_reg_est_senial
    - ind_reg_confip
    - ind_reg_portal

    valor por defecto de todos los anteriores: 1

1.2.4. Sección “sistema” (configuraciones generales del cliente)

- log_name – nombre del archivo de logueo a escribir en disco
  valor por defecto: log.log
- log_max_size – limita el tamaño del archivo de log (Kb)
  valor por defecto: 1000
- log_count_backup – cantidad de “roll-outs” a realizar del archivo de log. Es decir, se generarán a lo sumo esta cantidad de archivos de log, del tamaño especificado por el parámtro “log_max_size”. Al superarse esta cantidad de backups, se borrará el más viejo de los mismos.
  valor por defecto: 2
- zip_dir – nombre del directorio utilizado para guardar los archivos comprimidos previo a su envío al servidor, es relativo al directorio de instalación del programa
  valor por defecto: zips
- temp_dir – nombre del directorio utilizado para guardar los archivos de datos previo a ser comprimidos, es relativo al directorio de instalación del programa
  valor por defecto: temp

1.2.5. Sección “zipper” (configuración de la compresión)

- zip_max_size – tamaño máximo permitido para un solo archivo comprimido (Kb)
  valor por defecto: 200
- zip_dir_max_size - tamaño máximo permitido en el disco para archivos comprimidos (Kb)
  valor por defecto: 1000

2. Módulo Servidor

Para poner en producción el servidor del sistema es necesario contar con los siguientes elementos:

- Servidor Apache 2.0 o superior
- Interprete PHP 5 o superior
- Motor de Base de Datos MySQL 5.0 o superior
- Servidor SFTP / FTP

2.1. Creación de Base de Datos

Se debe levantar el esquema de base de datos “olpc” que se creará con la ejecución del archivo “olpc.sql” que se adjunta con la distribución del sistema. Si fuese necesario se puede modificar el nombre del esquema una vez creado.

```
shell> mysql -h host -u user -p < olpc.sql
Enter password: ********
```

A continuación se debe definir un nuevo usuario de base de datos con todos los privilegios sobre el nuevo esquema.

Por último será necesario reiniciar el motor de base de datos, para que los cambios tengan efecto.

```
/etc/init.d/mysql  stop
/etc/init.d/mysql  start
```

2.2. Deploy del Sitio Web

Copiar la carpeta “olpc” que se adjunta con la distribución del sistema en la carpeta configurada en el Apache a tales efectos.

```
/var/www/apache2-default/
```

Luego se debe modificar el archivo /olpc/Connections.php e ingresar el nombre del esquema, usuario y password para la conexión a la base de datos, según lo configurado en el apartado anterior.

```
<?php
# FileName="Connection_php_mysql.htm"
# Type="MYSQL"
# HTTP="true"
$hostname_olpc = "localhost";
$database_olpc = "olpc";
$username_olpc = "root";
$password_olpc = "root";
$olpc = mysql_pconnect($hostname_olpc, $username_olpc, $password_olpc) or
trigger_error(mysql_error(),E_USER_ERROR);
?>
```

Luego se debe reiniciar Apache para que los cambios tengan efecto.

```
/etc/init.d/apache2  stop
/etc/init.d/apache2  start
```

A continuación se debe crear el usuario con el cual se hará la transferencia FTP/SFTP al servidor, con permiso de escritura sobre la carpeta “../olpc/temp/FTP” y que pertenezca al grupo de usuarios “www-data” a los efectos de que los archivos transferidos puedan ser procesados por los procesos PHP.

Nota: Verificar en la configuración del servidor Web Apache que el módulo php_zip necesario para trabajar con archivos comprimidos.

ANEXO III

Manual de Usuario

Se realizó un manual de usuario de la interfaz Web, conteniendo una explicación detallada e ilustrada de cada una de las opciones y menúes disponibles. El mismo se encuentra disponible conjuntamente con el software entregado, en el CD correspondiente.

ANEXO IV

Contenido del CD

En el CD se incluyen los scripts en Python que corren en las máquinas con sus respectivas documentaciones. Están incluidos también los códigos PHP que corren en el servidor, documentados también.

Se adjunta también el presente documento en versión digital con ambos manuales, usuario y desarrollador.

Path	**Descripción**
/	Presente Documento
/Cliente/	Scripts del módulo cliente
/WebServidor/	Scripts del Módulo Servidor
/BaseDatosServidor/	Create de la Base de Datos
/DocSoftware/	Documentación de Programas
/Manuales/	Manuales de Instalación y de Usuario