AGENTES DE SOFTWARE APLICADO A GESTIÓN DE REDES BASADA EN WEB

SOFTWARE AGENTS APPLIED TO WEB NETWORK MANAGEMENT

RESUMEN

Los dispositivos que ofrecen acceso inalámbrico a servicios de red exigen un constante seguimiento por parte de profesionales en redes para garantizar su correcta operación, en este documento se presenta el diseño y desarrollo del conjunto de componentes que facilitan las funciones de monitoreo de dispositivos de red para garantizar al usuario una adecuada experiencia de navegación desde clientes inalámbricos. El estado del dispositivo que provee servicios de navegación es monitoreado mediante agentes que registran los eventos de interés y envían la información al módulo de gestión el cual permite la interacción mediante interfaces y protocolos Web, liberando al administrador de las tareas de revisión presencial de cada uno de los dispositivos instalados.

Palabras clave: agentes, gestión de red, gestión basada en Web, OpenWrt, VPN

ABSTRACT

Devices that offer wireless access to network services require continuous monitoring by network professionals to ensure proper operation, this document presents the design and development of a set of components that facilitate monitoring features to ensure a proper browser experience to users from wireless clients. The device status that provides browsing services is monitored by agents that record events of interest and send the information to the management module which allows interaction through interfaces and web protocols, freeing administrator from tasks like personal review of each of the installed devices.

Key words: agents, network management, OpenWrt, VPN, Web based management

Rolando Rubio Rodriguez
Ingeniero de Sistemas
Universidad Distrital Francisco José de Caldas
rolando81@gmail.com
Bogotá, Colombia

Julio Barón Velandia
Ingeniero de Sistemas
Universidad Distrital Francisco José de Caldas
jbaron@udistrital.edu.co
Bogotá, Colombia

Carlos Enrique Montenegro Marin
Ingeniero de Sistemas
Universidad Distrital Francisco José de Caldas
cemontenegrom@udistrital.edu.co
Bogotá, Colombia

Tipo: Artículo reflexión

Fecha de Recepción: Octubre 22 de 2013
Fecha de Aceptación: Diciembre 22 de 2013

1. INTRODUCCIÓN

La palabra agente tiene varias connotaciones, para el desarrollo de una solución que permita la gestión remota de dispositivos de red, se asume la propuesta desde las propuestas de gestión de redes de telecomunicaciones, sin embargo se presentan las consideraciones que generalmente debe presentar un agente y la clasificación de agentes que mejor describen al modelo desarrollo para permitir la gestión de redes basada en Web sobre un ambiente embebido. Dadas las características de los dispositivos resulta fundamental la optimización de la capacidad de procesamiento y almacenamiento.

2. CONOCIMIENTOS PREVIOS Y METODOLOGÍA

2.1. Administración de redes basada en web

La administración de redes basada en Web facilita el monitoreo y administración de dispositivos red en tiempo real sin importar el lugar donde estén ubicados los dispositivos gestionables, para lo cual se utiliza Internet como medio de acceso [1], entre las principales ventajas de emplear este mecanismo se encuentran: la facilidad de uso, independencia de plataforma y facilidad de acceso.

Identificar y referenciar es fundamental en las actividades de gestión, los psicólogos sugieren que la tendencia del ser humano de colocar nombre a territorios, animales, ríos, mares e incluso partes del cuerpo es para dar pertenencia y control sobre el objeto, es decir que si se puede nombrar el objeto se puede tener control sobre él [2], en la gestión de redes, esto implica la capacidad de nombrar a cada objeto con el fin de conocer exactamente el segmento o el nodo que está presentando problemas y de esta forma corregirlo, ¿Cómo facilitar al administrador de red detectar el componente que presenta fallas, sin que este tenga que verificar de manera presencial su estado de operación?. Para lograrlo se presenta la implementación de un agente encargado de monitorear el funcionamiento del sistema, reportando un conjunto de eventos que pueden incidir en su adecuada operación.

2.2. Agente de software y sus definiciones

Los servicios proporcionados por agentes de software benefician tanto al administrador de red como al usuario final, con lo cual se mejora la eficiencia de los sistemas de comunicación de manera integral [2], razón por la cual en esta industria se invierten más de 700 millones de dólares en adquisición de software que facilite la toma de decisiones para la correcta configuración y operación del sistema, siendo un ambiente muy fértil para los agentes de software y programas con inteligencia artificial en el futuro cercano [3].

La definición general de agente de software es el de una entidad de software que automatiza algunas de las tareas que son realizadas comúnmente de forma repetitiva o que resultan bastante engorrosas para un agente humano. Los usos más frecuentes de este termino son el de Agentes móviles (MA) y Agentes Cooperativos Inteligentes (IA) [2], los agentes de software hoy en día son una de las tecnologías más importantes y se encuentran entre las que más rápido han venido evolucionando. Otro de los campos que ha ganado interés por parte de los investigadores ha sido el paradigma de los Sistemas Multi-Agente (MAS), planteando que actividades complejas, pueden ser realizadas por medio de interacciones entre entidades de software relativamente independientes llamadas agentes.

En términos de sistemas operativos un agente de software puede ser descrito como un programa en ejecución generalmente con múltiples procesos, con capacidad de tomar una decisión sin la intervención directa de un humano. Las propiedades que los agentes de software pueden tener según los autores de [2] son:

- *Habilidad Social:* comunicarse con otros para alcanzar un objetivo global o individual.
- *Autonomía:* deben operar sin intervención

de elementos externos.
- *Reactividad:* responder en un periodo corto a cambios del entorno.
- *Adaptabilidad:* definir sus propios objetivos de acuerdo a un campo de interés definido.
- *Aprendizaje:* implementar algoritmo de aprendizaje para alterar sus acciones futuras.
- *Pro-actividad:* además de responder a eventos anticiparse los eventos futuros.

Los agentes pueden clasificarse en reactivos, deliberativos e híbridos, los agentes reactivos generalmente no tienen una representación interna del entorno sin embargo algunos de ellos puede tener una representación limitada del mundo, la ventaja principal es que son los más rápidos ya que no deben realizar técnicas de razonamiento muy complejas. Por otra parte los agentes deliberativos tienen una base de conocimiento extra que les permite mayor flexibilidad y habilidad de acondicionarse a su medio ambiente. El tipo de agente hibrido sería el ideal ya que contaría con una base de conocimiento propia que le permitiría mayor proactividad y una parte reactiva que le permitiría responder de forma rápida a determinados eventos [4]. En MAS los agentes pueden ser cooperativos, trabajan juntos para la consecución de un objetivo o agentes egoístas, es decir que siempre van a trabajar por su propio interés.

Algo vital para un agente es la comunicación ya que por medio de ella se relaciona con su entorno, con otros agentes o con un servidor central, como en toda comunicación debe existir un lenguaje que es el encargado de transmitir la información de manera que el emisor y el receptor reciban el mensaje, entre los lenguajes más populares están: *Knowledge Query Manipulation Language* (KQML) [5] y la especificación creada por la *Foundation for intelligent Physical Agents* (FIPA) [6], sin embargo el lenguaje no es suficiente, los agentes deben tener una ontología común es decir un conjunto de términos y una significación con el mundo real, en este aspecto se destaca la estandarización de objetos liderada por el OMG y la del consorcio World Wide Web consortium empleando el lenguaje de marcado *Extensible Markup Language* (XML) [7].

2.3. Planteamiento del problema

HomeDATA para manejar sus Hotspots cuenta con la aplicación *HomeDATA Hotspot Management* (HHM) que traduce Administrador de Hotspots HomeDATA y se caracteriza por ser una aplicación centralizada, en ella se realizan labores como la creación de tarjetas prepago, cuentas pospago, registro de equipos para salir libremente a Internet además de información de las ventas, enlazado a sistemas de facturación populares en los sistemas de información de las cadenas hoteleras. HHM está enlazado a su parte remota llamada nodo que puede representar: un hotel, restaurante, centro comercial, etc. Cuenta con uno o más agentes encargados de asignar las direcciones de red y desplegar un portal cautivo que solicita al usuario que quiere acceder a Internet, una clave de acceso o pin, al ingresar correctamente el pin al sistema, permite al usuario navegar mientras, descontando el tiempo de navegación en cada sesión, y actualizando el tiempo restante para próximas sesiones. No siempre el usuario puede utilizar la opción salir para dejar de navegar dado que se pueden presentar algunas de las siguientes situaciones:

- Desactivación de la conexión inalámbrica.
- Bloque del sistema operativo
- Fallas en el navegador
- Desplazamiento del equipo fuera de la zona de cobertura.
- Computador encendido con una cuenta activa.

En cualquiera de las situaciones presentadas se produce una desconexión forzada y si se intenta ingresar nuevamente al sistema con el mismo pin el sistema no lo permite, indicando que el pin de navegación ya está siendo utilizado. Estas situaciones impiden que se pueda determinar el número de usuarios que realmente están navegando por medio del agente, apareciendo en el sistema muchos usuarios conectados desde hace varios días, meses o años a estos usuarios se les ha denominado coloquialmente como “usuarios pegados”.

Una solución a este problema sería poder desconectar al usuario para permitir el acceso a quien requiere el servicio, pero no ha sido implementada, la única solución en cada uno de los nodos ha sido entregar al usuario un nuevo pin, esto produce problemas en facturación, incremento de costos asociados, además en algunos casos llamadas a la sección de soporte de HomeDATA. Desde el punto que se mire son pérdidas económicas tanto para cada nodo como para la empresa y una reducción del nivel de satisfacción del cliente.

Así como no existe una forma de saber qué usuarios están realmente conectados al sistema, se desconoce también los agentes activos en cada nodo y si están funcionando correctamente, tampoco es posible ingresar a algunos de los agentes en caso de requerir labores de mantenimiento utilizando un shell remoto seguro *Security Shell* (ssh), debido a que se ubican en puntos donde el acceso es restringido, tal es el caso de agentes detrás de un firewall, un router, o que no cuentan con una dirección pública válida como en el caso del método *Network Address Translation* (NAT), aun en el caso de llegar el agente a tener dirección IP pública, nada garantiza que ésta la pueda cambiar ya sea por motivos prácticos del proveedor de servicios de Internet (ISP) o el cambio del mismo. Desde la actual aplicación resulta difícil obtener la dirección IP de un agente ya que se emplea un archivo de texto plano que se actualiza manualmente. Se puede observar entonces que los costos de personal de soporte son elevados y crecen a medida que la empresa captura más clientes en otras ciudades y en otros países debido a los costos de transporte, estadía y alimentación en estos lugares.

Existe también el IP spoofing que consiste en falsificar la dirección de un determinado servidor por el cual el sistema actual es completamente vulnerable, bastaría únicamente conocer la IP de un agente valido para que el servidor central lo admitiera sin problemas, podría ocurrir incluso en este momento que varios agentes estuvieran utilizando este método o en los nodos hacer una copia completa del sistema del agente en otro computador, y salir por un router con dos o más agentes sin el sistema percatarse del cambio, se tendrían entonces agentes consumiendo recursos sin pagar la mensualidad por el servicio y congestionando el servidor.

Una variable a tener en cuenta es que los dispositivos en los que será instalado el agente HomeDATA cuenta con una gran diversidad de clientes cada uno con sus requerimientos especiales y diferentes escenarios a ser considerados, por tanto en algunos lugares el agente se instala sobre un computador de arquitectura x86 con el sistema operativo Linux, en otros en cambio el agente es instalado sobre un router dotado con una versión mínima de Linux, generalmente el router empleado es un Linksys WRT54G version 4, el cual tiene un procesador mipsel de 200 MHz, una memoria flash de 4 MB y 16 MB de memoria RAM. Este ambiente con recursos computacionales reducidos, representa un reto para el nuevo agente el cual tendrá que ser instalado en este tipo de routers, por tanto se debe buscar que el software a desarrollar sea lo suficientemente ligero en cuanto a requerimientos de procesador y de espacio de almacenamiento para ejecutarse de forma adecuada en este tipo de dispositivos. Teniendo en cuenta lo anteriormente enunciado se debe buscar un método que permita ubicar fácilmente al agente sin importar su ubicación en la red, conocer su estado y poder autenticar el agente de manera única evitando suplantaciones de identidad, empleando la nueva plataforma para la administración de Hotspots HHM 2.0 la cual ha sido desarrollada en el lenguaje de programación Java™ y utiliza para comunicarse entre los diferentes procesos el lenguaje de marcado XML.

3. DISEÑO DEL AGENTE

Como solución se diseñó un agente con identificador único, es de tipo reactivo por lo tanto no cuenta con una base de conocimiento de todo el entorno ni la posibilidad de aprender. Las características implementadas en este agente son: capacidad social pero limitada, únicamente existe comunicación entre el agente y el servidor sin tener en cuenta la existencia de otros

agentes a su alrededor, autonomía, es decir al iniciarse el sistema se ejecuta y puede actuar de acuerdo a los eventos ocurridos. El agente está diseñado de tal forma que permite cierta proactividad al momento de producirse una falla en el sistema, y busca la forma de corregirla. Se empleó XML como el lenguaje para la comunicación entre el agente y el servidor.

3.1. Meta

Permitir a los usuarios legítimos del sistema acceder al servicio de Internet brindándole una experiencia adecuada de navegación y permitir al administrador del hotspot conocer los usuarios conectados proporcionando la posibilidad de desconectar usuarios no deseados.

3.2. Ambiente

El ambiente es un router inalámbrico que ha sido modificado para permitir la instalación del sistema operativo OpenWrt [7] el cual requiere aproximadamente 2.2 MB quedando disponible únicamente 1.8 MB para instalar el software restante para el agente y las librerías necesarias.

3.3. Percepción

El agente puede recibir desde el exterior, comandos en formato XML para ejecutarlos en el sistema operativo y tiene la capacidad de responder a ellos, de igual modo puede validar la petición recibida empleando un esquema XML.

3.4. Estado inicial

Al cargar el sistema operativo OpenWrt se inicia la ejecución del agente el cual deberá vigilar cada uno de sus componentes y en caso de terminarse algún proceso del agente, el mismo tendrá la capacidad de reiniciar los servicios o procesos perdidos, cuenta con los componentes de transporte, control de acceso, configuración y gestión remota.

3.4.1. Componente transporte

En este componente se encuentra la aplicación Openvpn [8], encargada de comunicarse con el otro extremo en el servidor, creando un canal seguro empleando TLS que garantiza la autenticidad y unicidad del cliente por medio de un certificado digital exclusivo para cada cliente. Todos los certificados deben ser firmados por una entidad certificadora lo que garantiza que solo estos certificados serán validos, como punto adicional el certificado creado para el servidor incluirá una propiedad única, por lo que teniendo un certificado emitido para un cliente no puede ser posible convertirlo en un certificado de servidor. Como medida adicional se configuró el servidor de tal manera que cada cliente pueda ver únicamente al servidor y no a los demás clientes conectados. En la carpeta /etc/openvpn el cliente deberá contar con dos archivos como se presenta en la figura 1, el primero corresponde al identificador único de cada agente en este caso es el 000022 este archivo es creado como referencia y no como identificación, el archivo id contiene el nombre de identificación único, el archivo ca.crt es el certificado digital encargado de verificar la firma, tanto del cliente como del servidor, el archivo client.crt es la llave pública encargada de identificar de forma única con el servidor, usando esta llave el servidor cifra toda la comunicación hacia el cliente.

```
root@OpenWrt:~# cd /etc/openvpn/
root@OpenWrt:/etc/openvpn# ls

000022       client.conf  client.key
ca.crt       client.crt   id
```

Figura 1. Listado archivos carpeta openvpn

Parte del certificado se presenta en la figura 2, client.key representa la llave privada que se utiliza para descifrar los mensajes que le sean enviados desde el servidor. Los archivos id, ca.crt, client.key, client.crt son nombres genéricos de modo que puedan ser fácilmente invocados, de lo contrario se deberían definir archivos de configuración personalizados, haciendo complejas las labores de mantenimiento.

La dirección IP 10.8.0.1 es utilizada por el servidor y cada cliente tiene la dirección de acuerdo a su identificador en el caso del agente

000022 la IP es 10.0.8.22, cuando el identificador es mayor de 254, por ejemplo el identificador 000255, se pasa al siguiente conjunto de direcciones por tanto la dirección queda de la forma 10.0.9.1.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 22 (0x16)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=CO, ST=Colombia, L=Bogota, O=Homedata
LTDA, OU=I+D, CN=Homedata-
CA/emailAddress=homedata@homedata.com.co
        Validity
            Not Before: Dec 19 21:02:46 2013 GMT
            Not After : Dec 17 21:02:46 2023 GMT
        Subject: C=CO, ST=Colombia, O=Homedata LTDA,
OU=I+D, CN=000022/emailAddress=homedata@homedata.com.co
```

Figura 2. Parte contenido del archivo client.crt

3.4.2. Componente control de acceso

El componente de control de acceso es el que realiza las labores de bloqueo de tráfico de los usuarios de la red local (LAN) desplegando un portal cautivo y solicitando una contraseña para permitirle al usuario navegar por Internet. Para controlar el acceso a los usuarios de una red de área local (LAN), se utilizó el componente de código abierto Chillispot [9], el cual permite autenticar usuarios de una red inalámbrica, soporta el inicio de sesión basado en Web, y permite la Autenticación, Autorización y Tasación de Servicio utilizando un servidor RADIUS. En esta parte se realizaron mejoras en cuanto a la configuración del aplicativo para corregir los problemas que se habían encontrado como: la falta de control sobre los usuarios, opciones para permitir la desconexión deshabilitadas, falta de fiabilidad del sistema para garantizar que los usuarios que son reportados en el servidor RADIUS son los que realmente están.

Se habilitó el envío de un paquete al servidor RADIUS llamado Iterim-Interval el cual envía cada 300 segundos información de accounting de cada cliente. También se realizaron algunos cambios en el archivo de configuración /etc/chilli.conf, el parámetro Lease encargado de la renovación de la dirección ip se dejó en 150 segundos, de este modo el administrador observa en tiempo real los usuarios conectados.

Los parámetros de configuración incluidos en el archivo se presentan en la figura 3. El servidor RADIUS queda conectado directamente a la VPN para incrementar la seguridad y permitir que se pueda abrir cualquier puerto de forma segura, el parámetro coaport 3799 permite realizar remotamente la petición para la desconexión de usuarios.

```
domain Homedata
radiusserver1 10.8.0.1
radiusserver2 10.8.0.1
radiusauthport 1812
radiusacctport 1813
dhcpif br0
coaport 3799
coanoipcheck
uamport 3990

radiussecret secret123
uamserver
http://IP_SERVIDOR_RADIUS/html/autenticar/portalAutenti
cacion.php?agente=000022&
net 192.168.18.0/24
lease 150
uamallowed www.google.com
```

Figura 3. Contenido archivo /etc/chilli.conf

Otra aplicación necesaria dentro del componente de control de acceso es una aplicación que permita libremente salir a Internet en caso que el servidor central presente algún fallo o no puedan llegar al servidor por algún motivo, el programa a utilizar es dnsmasq [10], un servidor liviano de DNS y DHCP ideal para los limitados recursos del OpenWrt. El inicio de este programa se realiza ejecutando un shell script con los parámetros establecidos por el componente de configuración.

3.4.3. Componente configuración

```
ip_base=192.168.18
lan_ipfija=$ip_base.1
lan_netmask=255.255.255.0
#parametro de la red en el chilli
lan_net=$ip_base.0/24
#Rango de direcciones dhcp
#dhcp inicial
lan_dhcp_i=$ip_base.2
#dhcp final
lan_dhcp_f=$ip_base.250
#tiempo en que el cliente solicita nuevamente una
dirreccion ip
lease=150
# Esta ip se activa al momento de iniciar el chilli
no debe ser
# conocida ni facil de obtener debe ser secreta
lan_ipoculta=18.12.31.121
# Es la dirreccion ip del servidor remoto
#servidor de diveo
servidor_remoto=200.31.69.50
#serial que identifica a cada linksys de forma unica
serial=`cat /etc/openvpn/id`
```

Figura 4. Contenido archivo /etc/config.sh

Contiene los parámetros que son comunes entre los componentes de transporte, control de acceso y gestión remota, en la figura 4. Se observa el contenido del fichero /etc/config.sh , se han incluido los parámetros para configurar la asignación dinámica de direcciones (DHCP), la IP del servidor remoto, y se incluye la variable id, con el identificador único del agente, este archivo se invoca desde cada uno de los componentes para evitar problemas como introducir nuevamente algún parámetro y propagarlo en el sistema.

3.4.4. Componente gestión remota

El componente de gestión remota se encarga de recibir las peticiones realizadas al agente, y de enviar la información acerca de su estado utilizando XML, para realizar la comunicación se debe contar con un esquema común entre el servidor y el cliente, este fichero ha sido llamado heraldo.xsd, las principales partes de este esquema es el RequestSet y el ResponseSet que contienen el conjunto de peticiones y respuestas a ser enviadas o recibidas, las peticiones son AgentUpdate y HostExecute.

AgentUpdate envía la información del agente acerca del tiempo que se encuentra encendido el sistema, la carga del procesador y un secreto entre el cliente y el servidor como medida extra de seguridad.

HostExecute es enviado desde el servidor para solicitar la ejecución de un comando de forma remota. Se debe especificar el usuario que ejecutará el comando, el directorio desde donde se ejecutará la aplicación, las variables de entorno y el comando a ser ejecutado, como respuesta se envía la salida del comando ejecutado y los errores en caso de existir.

3.5. Estado intermedio

Después de iniciar el sistema operativo y la ejecución del agente, su primera labor es iniciar la VPN y esperar a que este disponible el canal cifrado utilizando el componente de transporte, luego se envía el objeto AgentUpdate empleando el componente de gestión remota, si se envía un secreto valido entre el cliente y el servidor, el agente iniciará Chillispot para el control de los usuario que ingresen al sistema, en cualquier momento el agente puede recibir una petición de ejecución de un comando y deberá devolver la respuesta si contiene el secreto compartido. La interacción descrita se presenta en el diagrama de secuencia [11] de la figura 5.

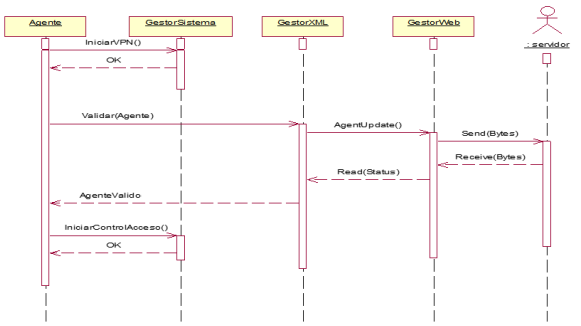

Figura 5. Diagrama de secuencia

3.6. Estado final

El agente debe estar en ejecución de manera continua a menos que el sistema operativo sea reiniciado o se presente un corte de energía.

3.7. Reglas del agente

El agente cuenta con reglas para los casos en que al tratar de autenticarse en el servidor reciba una respuesta de error o no le sea posible conectarse al servidor central, en este caso ingresa por medio del componente de control de acceso en modo abierto, si se presentan problemas de conexión intenta reiniciar el componente de transporte.

3.8. Diagrama de clases

En la figura 6 se observan el esquema de clases [11] necesarias para la implementación del agente, se observa la clase Agente que es la encargada de servir de unión entre las otras tres clases, se puede considerar la clase principal ya que desde ella se realiza la ejecución. La clase GestorSistema contiene las funciones para ejecutar los comandos del sistema operativo, GestorXML es la encargada de construir las peticiones XML a ser enviadas con la ayuda de

las clases RequestSet y ResponseSet, escuchar las peticiones remotas empleando el protocolo Web y enviar las respuestas al servidor como una cadena de caracteres son funciones que realiza GestorWeb.

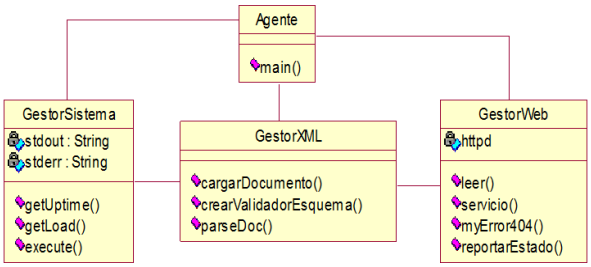

Figura 6. Diagrama de clases

4. IMPLEMENTACIÓN DEL AGENTE

Para la implementación del agente se empleó el lenguaje ANSI C estándar, debido a que tan solo emplear el lenguaje C++ implicaría instalar la librería libstdc++ la cual requiere 500 KB de memoria de almacenamiento, de igual modo cualquier otro leguaje requiere software adicional lo que implica espacio de almacenamiento con el que no se cuenta. A pesar de que el lenguaje C no es orientado a objetos (OO) se realizó una estandarización en las variables y funciones de tal modo que permitiera simular de manera apropiada un lenguaje OO.

Los pasos que se siguieron para la generación de los módulos fue crear un archivo .c y otro .h con el nombre de cada módulo por ejemplo los archivos GestorSistema.c y GestorSistema.h, los atributos del objeto son creados en una estructura de la siguiente forma:

```
struct sistema_t {
        char *stdout;
        char *stderr;
};
```

A todas las funciones dependiendo del nombre de la clase, se les agregó como prefijo un nombre representativo de la clase, en el caso de GestorSistema por ejemplo:

```
unsigned char *sistemagetUptime();
char *sistemaGetIPv4(const char *);
unsigned char *sistemaGetFile(char *);
```

Se llamó a la variable que debía hacer las veces del apuntador this en un lenguaje OO de este mismo modo, se observa como en la función sistemaExecute el primer parametro es la estructura GestorSistema_t con el nombre *this:

```
int sistemaExecute(struct GestorSistema_t *this,unsigned char *p_command,unsigned char *p_arguments,unsigned char *p_environment,unsigned char *p_user,unsigned char *p_directory)
{
        ……
}
```

De este modo se consigue la persistencia del objeto a lo largo del programa, del mismo modo se implementó la función que hiciera lo mismo que un constructor en los lenguajes OO. Por ejemplo el siguiente es una parte del constructor de GestorXML:

```
int xmlNew(struct xml_t *this,struct options_t *options)
{
        this->requestDoc=NULL;
        this->responseDoc=NULL;
        this->validschema=NULL;
        /*cargar opciones del agente*/
        this->schemaFilePath=options->schemaFilePath;
        this->secreto=options->secreto;
        this->interface=options->interface;
        this->id=options->id;
        this->validschema = xmlAgente_crearValidadorDeEsquema (this, this->schemaFilePath);
        if (this->validschema = = NULL)
{
fatal("Error inesperado al crear validador.");
}
        this->valido=0;
        return 1;
}
```

Empleando la misma metodología se creó la totalidad del programa que fue compilado utilizando el programa buildroot [7], posteriormente instalado en el Linksys OpenWrt.

5. CONCLUSIONES

La clave para el desarrollo de sistemas embebidos está en buscar las herramientas adecuadas teniendo en cuenta el espacio que ocupa en disco, los requerimientos de memoria, las librerías de las que depende el programa y preferiblemente que esté escrito en ANSI C ya que aunque no es lenguaje orientado a Objetos, se puede por medio de una programación metódica y ordenada obtener resultados similares a los obtenidos por un lenguaje orientado a objetos, adicionalmente facilita su compilación en otras arquitecturas como el caso del Linksys con su procesador MIPSEL.

Los agentes de software son una herramienta efectiva en la construcción un sistema robusto para la gestión de redes basada en Web, para su diseño se deben establecer de manera explícita y detallada cada una de las características y comportamientos de sus diferentes estados, realizar el motor de inferencia en un lenguaje intermedio como lo es el seudo código facilita la comprensión del problema.

Referencias Bibliográficas

[1] C. Harler, Web-based network management beyond the browser, Libro, New York: Wiley, 1999.

[2] J. Bighan, . L. . G. Cuthbert, A. Hayzelden y M. Woolridge, Software Agents for future communication systems, Libro, London, Springer, 1999.

[3] L. Lewis, AI and Intelligent Networks in the 1990 and the 21st Century, Worldwide Intelligent System, IOS Press, pp 109-124, 1995.

[4] M. Woolridge y M. Jennings, Intelligent Agents: Theory and Practice, Libro, London, Springer, pp. 115-152, 1995.

[5] UMBC Agent Web. [En línea], consultado en Enero 10 de 2013, disponible en: http://www.cs.umbc.edu/kqml/.

[6] Foundation for Intelligent Physical Agents. [En línea], consultado en enero 20 de 2013, disponible en: http://www.fipa.org/repository/aclspecs.html.

[7] OpenWrt Wireless Freedom. [En línea], consultado en abril 25 de 2013, disponible en: http://www.openwrt.org.

[8] OpenVPN Technologies, Inc. [En línea], consultado en enero 10 de 2013, disponible en: http://openvpn.net/index.php/open-source.html.

[9] ChilliSpot. [En línea]. consultado en febrero 1de 2013, dicponible en: http://www.chillispot.org.

[10] www.thekelleys.org.uk. [En línea]. consultado en diciembre 20 de 2012, disponible en: http://www.thekelleys.org.uk/dnsmasq/doc.html .

[11] OMG. [En línea], consultado en enero 18 de 2013, disponible en: http://www.omg.org/gettingstarted/what_is_uml.htm.