J. Davila, J. Lopez, and R. Roman, "Introducción de Aplicaciones UDP en Redes Privadas Virtuales", *III Jornadas de Ingeniería Telemática (JITEL01)*, pp. 397-404, 2001.
NICS Lab. Publications: https://www.nics.uma.es/publications

# Introducción de Aplicaciones UDP en Redes Privadas Virtuales

Jorge Dávila[1], Javier López[2], Rodrigo Román[2]
[1] Facultad de Informática, Univ. Politécnica de Madrid
jdavila@fi.upm.es
[2] E.T.S. Ingeniería Informática, Univ. de Málaga
{jlm, roman @lcc.uma.es}

***Abstract:*** *Virtual Private Network (VPN) solutions mainly focus on security aspects. However, when security is considered the unique problem, some collateral ones arise. VPN users suffer from restrictions in their access to the network. They are not free to use traditional Internet services such as electronic mail exchange and audio/video conference with non-VPN users, and to access Web and Ftp servers external to the organization. In this paper we present a new solution, located at the TCP/IP transport layer and oriented to UDP applications that, while maintaining strong security features, allows the open use of traditional network services. The solution does not require the addition of new hardware because it is an exclusively software solution. As a consequence, the application is totally portable.*

## 1 INTRODUCCIÓN

Las necesidades actuales de las empresas en el ámbito de las telecomunicaciones ha aumentado de forma ostensible debido a diversos factores, como la globalización de la economía, que aumenta la necesidad de intercomunicación entre sedes de una misma empresa, y a los acuerdos con otras empresas, que imponen la utilización de recursos compartidos. Por lo tanto, reducir el costo de la infraestructura de telecomunicaciones es algo prioritario.

Normalmente, y hasta hace poco, una empresa alquilaba líneas de comunicaciones para intercomunicar los ordenadores de su empresa. Un ejemplo de esto pueden ser las líneas Frame-Relay [1, 2]. Pero esta solución es inconveniente, debido a diversos factores (falta de competitividad en el sector, coste de las líneas, latencia en la comunicaciones,...).

Sin embargo, ahora se pueden utilizar *Redes Privadas Virtuales* (RPVs), mediante las que se construyen canales privados de información sobre un canal de comunicación público (por ejemplo, Internet). La forma de lograr ese canal privado es a través de la utilización de mecanismos criptográficos. Así, una RPV permite enviar información privada entre equipos informáticos usando una línea de comunicación pública y, por lo tanto, sin el elevado coste asociado al alquiler de las líneas dedicadas.

Además, una RPV posee otros beneficios, como la reducción de costes en el mantenimiento, debido a que está montada sobre una red de comunicación pública, con lo que el soporte técnico y la competencia es mayor que en una red alquilada. También es más fácilmente escalable, pues el equipamiento y el trabajo necesarios para añadir un ordenador suele ser menor que en una red dedicada.

No obstante, el uso de las RPVs tiene algunos inconvenientes, como por ejemplo, la obligación de compartir el tráfico privado con el tráfico de otros usuarios de la red pública, introduciendo así esperas indeseadas en el envío de la información. Otro gran inconveniente es la mayor probabilidad de que la información sea interceptada durante su transmisión. Además, si el proveedor que ofrece el servicio a la red pública tuviera problemas, la parte de la RPV asociada a ese proveedor quedaría inoperante.

Estos inconvenientes pueden ser solucionados mejorando los aspectos de seguridad (utilizando un sistema de seguridad bien diseñado), velocidad y fiabilidad (utilizando un acceso rápido y fiable a la red Internet). No obstante, si la seguridad es el objetivo principal de la RPV, existe otro gran inconveniente, y es que los usuarios sufren restricciones para acceder a los recursos de Internet que no pertenezcan a la RPV.

Este trabajo presenta una solución a ese problema. Hemos desarrollado una nueva solución de seguridad para RPVs localizada en el nivel de transporte y totalmente software. Al funcionar en el nivel de transporte, permite usar los mismos servicios tradicionales de Internet. Y al ser software, es una solución muy flexible. Todo ello, manteniendo los criterios de seguridad necesarios. La solución, *SEC-Insel* [3], ya ha sido desarrollada para ser utilizada por los servicios TCP dentro de la RPV. Ahora nos centramos en el desarrollo de un sistema, *SEC-Insel UDP*, que permita utilizar los servicios UDP.

El resto del trabajo está estructurado de la siguiente forma: en la sección 2 se argumenta por que es

necesario la incorporación de los servicios UDP en el ámbito de las RPVs. La sección 3 presenta los posibles mecanismos para implementar una RPV, y la solución propuesta en este trabajo. En las secciones 4 y 5 se expone la arquitectura de los dos módulos, denominados *Secsockets UDP* y *RPV-Insel UDP* sobre los que se basa la nueva solución. En la sección 6 se muestra un escenario de la solución propuesta, y la sección 7 finaliza con las conclusiones.

# 2 LA IMPORTANCIA DE UDP

En el nivel de transporte de la pila de protocolos TCP/IP, TCP proporciona un servicio confiable orientado a la conexión, es decir, los paquetes llegan sin error y en el orden en el que se envían. Por otro lado, UDP es un protocolo que proporciona un servicio orientado a datagramas, no asegurando que los paquetes lleguen a su destino, y si llegaran, no garantizando su orden.

UDP es un protocolo más simple que TCP, y mucho menos fiable, aunque más rápido. Es útil para aplicaciones que sean simples, que no necesiten de una transmisión fiable de datos, o incluso que necesiten que sus datos sean transmitidos lo más rápidamente posible. Un ejemplo de aplicaciones que utilizan UDP son aquellas aplicaciones que realizan tareas simples, TIME [4], que sincronizan y monitorizan redes usando SNTP [5] o SNMP [6], o que realizan transmisión de audio/video, usando RTP [7] y RTSP [8]. Estas últimas merecen una especial atención.

Los protocolos de audio y video están adquiriendo una gran importancia en la actualidad, debido a la gran ayuda que pueden prestar en el ámbito académico y empresarial. Así, mediante videoconferencia y audioconferencia, los miembros de diversas sucursales de una empresa pueden comunicarse, y grupos de investigación de varios países pueden compartir sus opiniones y resultados de trabajos utilizando para ello la red pública Internet, con el consiguiente abaratamiento en los costes.

Además, utilizando el mecanismo de *streaming* (envío de flujo de información por demanda de un servidor a muchos clientes), se pueden escuchar o visualizar contenidos previamente grabados o que están siendo filmados en tiempo real. Un ejemplo de esto son las Webcams.

Las aplicaciones de audioconferencia y videoconferencia utilizan UDP porque no necesitan de los mecanismos de comunicación fiable que TCP ofrece. Estos mecanismos minimizarían el tamaño de ventana, aumentarían el número de paquetes a enviar durante la comunicación, y no se adecuarían al envío de información en tiempo real. Es decir, TCP perdería el tiempo tratando de retransmitir paquetes que no llegasen a su destino.

Además de utilizar UDP, estas aplicaciones necesitan de un protocolo que, utilizando los servicios que UDP nos ofrece, gestione todo el proceso de comunicación, ya que la transmisión de audio y video no consiste sólo en enviar imagen y sonido. También es necesario controlar una serie de parámetros, tales como la calidad de la información enviada, los codecs empleados, un nº de secuencia, y otros aspectos de la comunicación. El protocolo más utilizado para este fin es RTP, que utiliza los servicios de UDP para proporcionar funciones que construyan aplicaciones multimedia.

Por lo tanto, UDP es un protocolo que está adquiriendo cada vez mayor importancia en el campo de las tecnologías multimedia sobre Internet. Por ello, en este trabajo se plantea la necesidad de añadir seguridad a servicios que utilicen este protocolo en el entorno de una RPV.

# 3 COMUNICACIÓN PRIVADA PARA SERVICIOS UDP

## 3.1 OTRAS SOLUCIONES

El objetivo que nuestra solución persigue es dotar a las aplicaciones UDP de la RPV de mecanismos de seguridad, pretendiendo que ésta sea lo suficientemente flexible como para acceder a equipos externos a ella, manteniendo, por ejemplo, una videoconferencia privada y una pública al mismo tiempo. Además, ha de ser fácil de mantener y con bajo coste.

Para lograr tal objetivo podríamos basarnos en algún esquema existente. Pero realmente no se ha diseñado ningún mecanismo estándar específico para lograr un entorno seguro para aplicaciones UDP. Tan sólo existen protocolos propietarios de compañías de software, o "parches" en los programas. Sin ser específicos para UDP, existen dos mecanismos que proporcionan un entorno seguro: son los sistemas hardware del nivel de enlace y el protocolo IPSEC [9] del nivel de red.

Más concretamente, una de las soluciones existentes son los dispositvos hardware, denominados *cifradores en linea*. Son unos dispositivos físicos que disponen de dos puertos, ambos de entrada/salida. Cada uno de ellos tiene una función especifica: uno cifra la información, y otro la descifra. De esta forma, cuando la información entra a través de un puerto, es modificada (cifrada o descifrada) y enviada al otro puerto.

Su funcionamiento es el siguiente: Cuando los mensajes salen del sistema, entran en el dispositivo, el cual los cifra y los envía a la red. Cuando los mensajes llegan a otro sistema, ya sea un terminal o una pasarela, deben atravesar el dispositivo, el cual los descifra. De esta forma, los mensajes viajan siempre cifrados a través de la red.

Esta solución no es adecuada para nuestros objetivos, ya que implica un alto coste pues es necesario tener un cifrador para cada equipo que acceda al exterior de la LAN. Además, la información se traslada siempre cifrada, impidiendo acceder a equipos externos a nuestra red, con lo que la flexibilidad es prácticamente nula.

Otra solución existente que se puede aplicar para UDP es IPSEC [9], extensión que se encarga de añadir seguridad a las tramas enviadas a través de IP. Combinando IPSEC con routers o cortafuegos, podemos conseguir todo lo necesario para que una RPV funcione con capacidad de autenticación, cifrado e integridad de los datos, y generación y gestión automática de claves.

IPSEC utiliza un mecanismo denominado *asociación de seguridad* (SA), por el cual los equipos deben acordar una serie de parámetros de seguridad para todas sus comunicaciones (claves, métodos criptográficos,...). Una vez acordados estos parámetros, IPSEC proporciona diversos mecanismos para añadir seguridad a IP: autentificación, creando una "huella digital" de los mensajes, y privacidad, mediante el cifrado de los paquetes. Esto se logra con dos cabeceras especiales que se añaden a IP: AH (Authentication Header) y ESP (Encapsulation Security Payload).

Sin embargo IPSEC tiene ciertas desventajas que le impiden ser viable para realizar una RPV flexible. Una de ellas es que no puede crear una asociacion de seguridad para procesos o usuarios concretos (debido a que se trabaja a nivel de red), cuando son esos procesos o usuarios los que queremos intercomunicar. Otra desventaja consiste en que el sistema aplica la protección de forma automática según las asociaciones de seguridad, con lo que no permite a los usuarios tomar la decisión sobre cuándo aplicar los mecanismos de seguridad. La consecuencia de lo anterior es que los usuarios no pueden acceder al exterior de la RPV, objetivo que deseamos alcanzar.

En el nivel de transporte no existe ninguna solución estándar para UDP. Para TCP sí existe (protocolos TLS [17] y SSL [16]), pero estos protocolos no pueden ser utilizados con servicios UDP. La solución que buscamos podría utilizarse como un complemento a estos protocolos, por lo que tendríamos completamente asegurada la capa de transporte (TCP y UDP)

Si se diseñara un mecanismo de seguridad para UDP a este nivel, se añadirían ventajas a IPSEC, como la identificación de distintos procesos o usuarios (característica del nivel de transporte), y aplicar el cifrado cuando fuera conveniente. Por supuesto, esta solución debería incorporar también mecanismos para lograr las ventajas de IPSEC (autentificación, manejo automático de claves...).

Respecto a las posibles soluciones en el nivel de aplicación, sería cada aplicación la que habría de proporcionar los mecanismos de seguridad. Esto sería sencillo de cara al usuario final, pero tendría ciertas desventajas pues dejaría de ser una solución homogénea y cada aplicación tendría sus propios mecanismos de cifrado y control de claves. Además, sería necesario controlar que cada uno de los usuarios actualizara sus aplicaciones con aquellas que proporcionan seguridad, hecho éste difícil de conseguir. Más aún, si fuera necesario modificar algún aspecto de la RPV deberíamos cambiar cada uno de los programas que forman parte de la RPV.

## 3.2 NUEVA SOLUCIÓN

El análisis de estas posibles soluciones, existentes o por diseñar, para ser utilizadas en dotar de seguridad al protocolo UDP, plantea la pregunta de cuál se ajusta más a nuestras necesidades.

El diseño que este trabajo presenta es una solución realizada en el nivel de transporte, totalmente software, que adopta las ventajas del IPSEC, sin arrastrar ninguna de sus desventajas. Este diseño permite a los usuarios decidir cuando acceder a la RPV, y comunicarse (de forma totalmente segura) tanto con miembros de la RPV como con miembros externos a ésta. Es, por lo tanto, muy flexible y fácil de mantener (existe un control centralizado), con bajo coste por basarse en software y sin necesidad de adquirir ningún equipamiento adicional. Este mismo diseño, aunque con ciertas diferencias dependientes del protocolo, fue utilizado para usar servicios TCP, con resultados óptimos [3].

El escenario genérico en que nos hemos basado para el desarrollo de la aplicación se representa en la figura 1. Existe una *entidad principal*, que centraliza el control de la empresa o entidad donde se va a implantar la RPV, y que puede representar la oficina principal de la misma. También hay varias *entidades secundarias*, cada una con su red de área local.

El diseño se divide en dos subniveles, denominados *Secsockets UDP* y *RPV-Insel UDP*. La interfaz Secsockets es el subnivel inferior. Este interfaz trabaja por encima del interfaz sockets tradicional que proporciona acceso a TCP-UDP/IP.

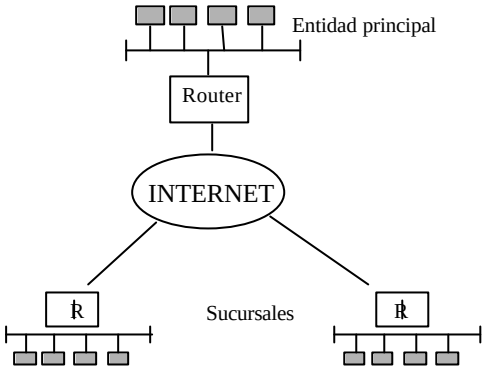


*Figura 1. Escenario para la RPV*

El servicio ofrecido por la interfaz Secsockets es la transmisión de datos de un proceso origen a un proceso destino a través de un canal seguro y autenticado. El módulo RPV-Insel trabaja sobre Secsockets y, haciendo uso de los servicios que este subnivel proporciona, da soporte de seguridad a las aplicaciones proveyéndolas de los servicios necesarios.

# 4 EL INTERFAZ SECSOCKETS UDP

## 4.1 INTRODUCCIÓN

El interfaz Secsockets es una extensión del interfaz socket tradicional, al cual le añade las funciones de seguridad que necesitamos para establecer nuestra RPV: autenticación (la información proviene de alguien que pertenece a la RPV), privacidad (quien no pertenezca a la RPV no puede acceder a los datos) e integridad (la información del mensaje no puede ser alterada).

El objetivo de esta interfaz es proporcionar un servicio de envío de datos seguro utilizando paquetes UDP, siendo fácil de utilizar de cara al usuario. Esta interfaz se basa en un modelo cliente-servidor, y su funcionamiento conlleva dos fases, una de conexión, en la que se ponen en marcha las bases para la comunicación segura, y otra de comunicación, donde los usuarios pueden ya enviar datos seguros de un extremo a otro mediante UDP.

## 4.2 FASE DE CONEXIÓN

La fase de conexión se realiza de forma automática, y en ella se produce la clave secreta necesaria para realizar una comunicación segura. Debido a que ambas partes (cliente y servidor) no disponen de esa clave hasta el final de la negociación, es necesaria la utilización de certificados digitales de clave pública para lograr la autenticación mutua y el intercambio seguro de los parámetros de seguridad.

Esta fase de conexión se realiza utilizando TCP, y no UDP. Esto es debido a que necesitamos un servicio de entrega fiable para el intercambio de los parámetros de comunicación. Al finalizar esta fase, se devolverá un socket UDP, a través del cual procederemos a realizar la comunicación UDP segura.

Los pasos que se siguen son los siguientes:

1. El cliente obtiene el certificado del servidor y le manda un primer mensaje con el algoritmo de clave pública a utilizar y su dirección electrónica.

2. El servidor recibe el paquete, solicita el certificado del cliente, y le envía a éste un mensaje de aceptación o rechazo. En este momento, tanto el cliente como el servidor poseen los certificados digitales del otro. De esta forma, ambos pueden enviarse mensajes cifrados utilizando el mecanismo de clave pública – clave privada.

3. El cliente envía cifrado con la clave pública del servidor una petición de conexión incluyendo los parámetros de seguridad. Estos parámetros son el algoritmo de cifrado simétrico a utilizar para cifrar la futura comunicación (DES [10], IDEA [11], Blowfish [12] o RC4 [13]), la función hash a utilizar para verificar si un mensaje es íntegro (MD5 [14] o SHA [15]), la compresión a utilizar (ninguna o GZIP) y un valor aleatorio que se usa para el cálculo posterior de la clave secreta.

4. Tras recibir y descifrar el mensaje de petición, el servidor envía al cliente un mensaje de aceptación o rechazo, cifrado con la clave pública del cliente. Además, si la negociación es aceptada, incluye otro valor aleatorio que también se usará para el calculo de la clave y el puerto UDP en el que el servidor escuchará las peticiones.

5. Ambos extremos calculan la clave secreta usando la función hash *H* y los valores intercambiados:

*H( H(aleatorio_Cliente) Å H(aleatorio_Servidor) )*

Además, el servidor cierra la conexión TCP con el cliente y abre la conexión UDP esperando los mensajes del éste.

Cabe reseñar que el algoritmo de cifrado simétrico más aconsejable para Secsockets UDP es el RC4. La causa es que este algoritmo es unas 10 veces más rápido que algoritmos como el DES, y esto es algo vital teniendo en cuenta que pretendemos trabajar con servicios que necesitan de recepción de datos en tiempo real.

## 4.3 FASE DE COMUNICACIÓN

Los pasos que se realizan durante la fase de comunicación (o de transmisión de datos) son, para cada mensaje: 1) compresión de la tramas, si así se

ha decidido durante la negociación inicial; 2) cálculo del valor hash de cada trama; 3) firma y cifrado de los datos; 4) envío de los datos.

La figura 2 muestra la composición de cada una de las tramas UDP. El tamaño máximo del mensaje depende de la longitud de los paquetes que acepte la red, por lo que es un parámetro configurable. Tras la recepción del mensaje, el receptor debe deshacer las operaciones que se han realizado sobre la trama recibida: descifrar (utilizando la recién conseguida clave simétrica), calcular y comparar su valor hash (para ver si ha sido modificado), descomprimirse (si así fue indicado en la negociación) y finalmente pasar los datos al nivel superior.

### 4.4 FUNCIONES DE LA INTERFAZ SECSOCKETS UDP

Ya se ha visto anteriormente que el servicio que ofrece la interfaz Secsockets es enviar datos de un proceso origen a un proceso destino a través de un canal seguro y autenticado. Este servicio se ofrece a través de un conjunto de funciones.

Tales funciones se comportan, en su interfaz, como un protocolo orientado a la conexión siguiendo el paradigma cliente/servidor. Esto es así debido a que las aplicaciones seguras que utilizan UDP necesitan un interfaz que les apoye a crear sistemas cliente/servidor paralelos. Además, de esta forma se puede elegir si utilizar para la comunicación TCP o UDP utilizando un simple byte de parámetro.

Las funciones de inicialización del interfaz Secsockets son las siguientes:

- *sec_init( )*: Crea un punto final de conexión en el extremo del servidor. Lo inicializa asignándole un puerto local de comunicación, y deja al servidor en espera de posibles peticiones de conexión de clientes a través de ese puerto.

- *sec_accept ( )*: El servidor llama a esta función para reconocer una conexión entrante, tras la cual empieza una negociación bajo TCP. Luego se devuelve un socket UDP con su dirección correspondiente, además de los parametros de seguridad.

- *sec_connect( )*: El cliente llama a esta función para crear un punto final de comunicación para el cliente. Tras una negociación con TCP (autentificando e intercambiando los parámetros de seguridad), se devuelve un socket UDP y su dirección, además de los parámetros de seguridad.

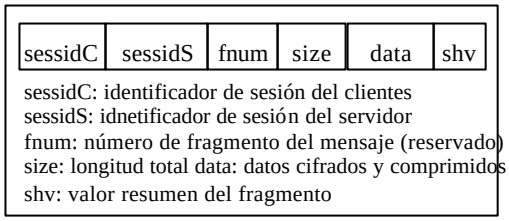


*Figura 2. Formato tramas UDP*

Con posterioridad a estas llamadas, y utilizando los sockets devueltos y los parámetros de seguridad, es posible establecer una comunicación segura. El intercambio de información y el cierre del canal se realizaría con las siguientes funciones:

- *sec_recv( )*: Esta función permite la recepción de datos a través de una conexión socket segura UDP, los descifra, chequea su autenticidad y los descomprime si es necesario.

- *sec_send( )*: Esta función es simétrica a la anterior. Comprime los datos si así se negoció previamente, calcula el valor hash, cifra los datos y, finalmente, los envía a través del socket UDP. Estas operaciones se realizan de acuerdo a la negociación de parámetros inicial entre cliente y servidor.

- *sec_close( )*: Esta función cierra el socket seguro UDP.

# 5 RPV-INSEL UDP

## 5.1 ESQUEMA RPV-INSEL

El módulo RPV-Insel utiliza los servicios proporcionados por la interfaz Secsockets para proporcionar seguridad a las aplicaciones UDP de la RPV. Además, se encarga de gestionarla, ya que inicializa todo lo que ésta necesita de forma automática, gestiona a los usuarios que forman parte de ella, y permite la instalación de servicios UDP exclusivos para sus miembros.

La RPV se basa en la existencia de 4 tipos de procesos: el *servidor principal* (S1), los *servidores secundarios* (S2), los *servidores auxiliares* (SAux), y los *clientes*, como se puede observar en la fig. 3.

Los clientes (C1). Utilizan los servicios que nos ofrece la RPV. Estos equipos forman parte de una LAN (C1).

El servidor principal de la RPV se encarga de: controlar a los demás servidores, mantener una base de datos con la información relativa a éstos (p. ej. sus direcciones IP), y servir información referente a la RPV.

Los servidores secundarios de la RPV. Existe un S2 por cada LAN de la RPV, y se encarga de controlar las peticiones de comunicación de los equipos de la LAN, además de controlar qué equipos de la LAN están conectados a la RPV. Cada S2 posee un

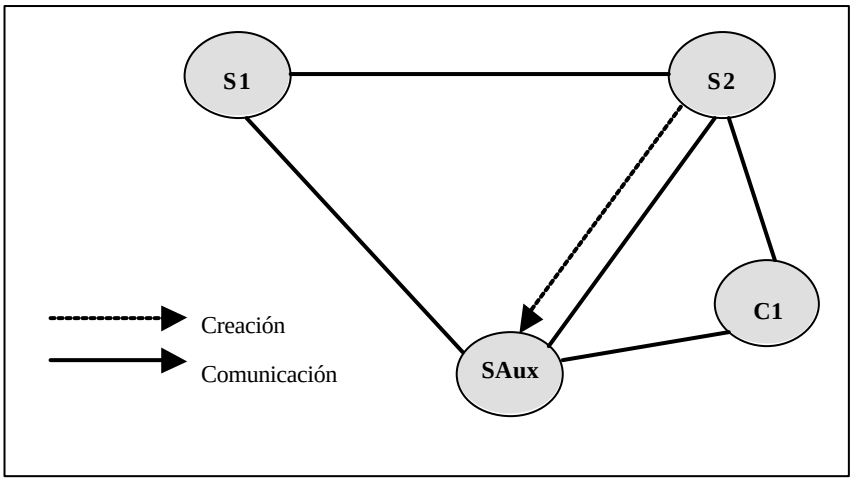


*Figura 3. Esquema RPV-Insel UDP*

certificado que se utilizará para la negociación de los canales seguros con Secsockets.

Los servidores auxiliares de la RPV son creados por los servidores secundarios, y su función consiste en llevar a cabo las comunicaciones seguras que han sido pedidas por los usuarios de la RPV. Existe un Saux por cada petición de conexión de un cliente.

De esta forma, cuando un cliente desea comunicarse con otro equipo de la RPV, contacta con su servidor secundario, el cual crea un servidor auxiliar que servirá de "puente" entre el cliente y el otro equipo de la RPV. Así, todo el tráfico cifrado pasará a través de los servidores auxiliares cifrado, mientras el cliente cree que el tráfico está siendo enviado como siempre.

## 5.2 PROBLEMAS DE DISEÑO AL UTILIZAR UDP

Durante la realización de este diseño y su puesta en práctica bajo UDP, han surgido diversos problemas, todos debidos a las características intrínsecas del protocolo UDP. A continuación se describen los problemas y las soluciones que se han adoptado.

Como las aplicaciones UDP basadas en transmisión de audio y video se basan en una arquitectura cliente-cliente (nadie es un servidor predefinido), existe el problema de controlar qué servidor secundario se encarga de actuar como servidor de una petición de comunicación. Por usar un símil, podríamos pensar que siempre hay alguien que llama por teléfono y hay alguien que recibe la llamada y responde, pero cualquiera de los dos puede manejar el rol de emisor/receptor de la llamada.

Para solucionar este problema, hay que pensar que siempre existe un cliente que realiza primero la comunicación, y un servidor que la recibe. Por lo tanto, el esquema es el siguiente: Ambos servidores secundarios están preparados para actuar como servidores (esperan una llamada), pero en el momento que uno mande una petición, se convierte en cliente, y convierte al otro en servidor.

Otro problema consiste en que se debe realizar una arquitectura cliente/servidor paralela (los servidores secundarios crean los servidores auxiliares), y el protocolo UDP no está preparado de antemano para lograr esta tarea, ya que está orientado a una arquitectura petición/respuesta.

La solución consiste en “simular” el protocolo TCP, en el aspecto de proporcionar un nuevo canal de comunicación cliente/servidor. De esta forma, se reserva un puerto temporal UDP en la máquina servidor, se comunica ese puerto al cliente, y se hace que toda la comunicación que el cliente vaya a enviarnos a partir de ese momento fluya por ese puerto. Una vez se deje de utilizar, ese puerto se cierra para permitir su reutilización.

# 6 ESCENARIO DE PRUEBA

La base de todo el sistema RPV-Insel es que las aplicaciones UDP conectan con los servidores auxiliares (en vez de con el host destino) y a través de ellos envían/reciben las tramas. Estos servidores se encargan de “puentearse” las tramas entre sí utilizando canales de comunicación seguros, de tal forma que la información viaja por la red sin ninguna posibilidad de ser alterada.

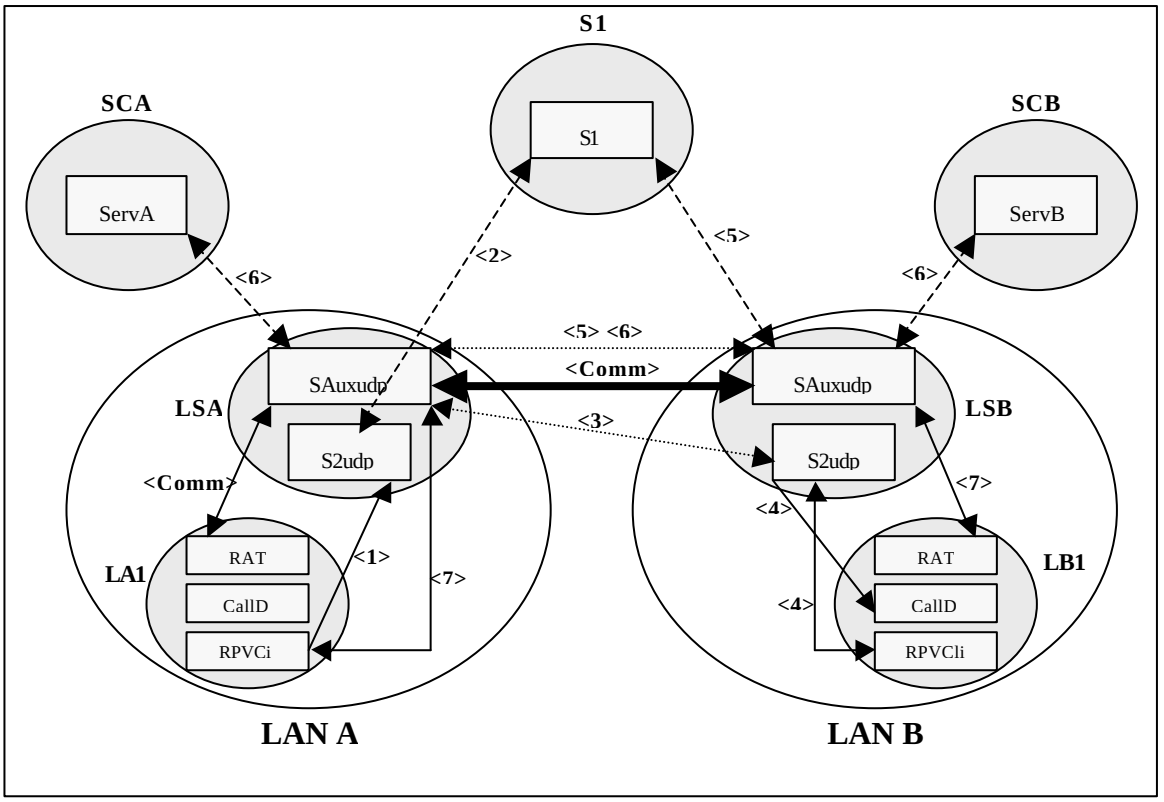


*Figura 4: Escenario de prueba*

Para la explicación de este sistema de vamos a proceder a la exposición de un escenario (Figura 4), el cual fue testeado bajo máquinas Linux. En él tenemos dos máquinas (LA1 y LB1) que quieren comunicarse entre sí usando el programa RAT (audioconferencia bajo RTP [18]), además de dos servidores secundarios (LSA y LSB, uno para cada LAN), dos máquinas servidoras de certificados (SCA y SCB, una para cada LAN), y un servidor principal (S1).

Al iniciarse la comunicación por petición del usuario, LA1 avisa a su servidor secundario, LSA, que quiere empezar una comunicación con LB1. A partir de ese momento la siguiente secuencia de eventos toma lugar:

1. LSA comprueba que la petición es para realizar una llamada y procede de un usuario de la RPV, por lo que pasa a modo servidor.

2. El servidor secundario envía una petición a S1 para que le diga en que LAN se encuentra LB1 y cual es el S2 de esa LAN. S1 responde a la comunicación (devuelve la dirección IP de LSB).

3. LSA crea un servidor auxiliar, que se encargará de ahora en delante de todo el proceso. Este servidor envía una petición a LSB (mediante un canal TCP), y espera su respuesta.

4. LSB recibe la petición, guarda el puerto del canal TCP, y avisa a LB1 que esta siendo llamado. A continuación LB1 responde a LSB, y LSB (tras comprobar que LB1 pertenece a la RPV) pasa a modo cliente.

5º) LSB crea un servidor auxiliar para que se encargue de la comunicación. Este, tras localizar a LSB vía S1, responde al auxiliar de LSA mediante el puerto TCP anteriormente guardado, comenzando la negociación.

En este momento LSA y LSB saben la dirección del otro, y mantienen una comunicación abierta.

6º) LSA y LSB realizan la negociación de dos canales seguros UDP (RTP necesita de dos canales UDP para funcionar). Es en este momento cuando se accede a las máquinas SCA y SCB.

7º) LSA y LSB comunican a sus respectivos clientes que la comunicación segura puede empezar.

Tras recibir este aviso, los clientes llaman al programa RAT. Estos se comunicarán con los respectivos servidores auxiliares, ya que se se habrá indicado a éstos como destino de las comunicaciones. Se encargarán de enviar los datos a través de los sockets seguros, y de esta forma la información viajará por el exterior de la LAN de forma totalmente segura.

Una vez que las comunicaciones se cortan, se liberan los sockets y los recursos que estaban funcionando, y los servidores secundarios terminan.

# 7 CONCLUSIONES

El presente trabajo propone una nueva solución para implementar una RPV que cubre las necesidades de seguridad que implican las comunicaciones internas de una organización distribuida. La propuesta que realizamos se basa en dos subniveles: la interfaz Secsockets y RPV-Insel, que se localizan sobre el nivel de transporte de la torre de protocolos TCP/IP y por debajo de las aplicaciones.

Esta solución presenta algunos inconvenientes respecto a las soluciones existentes. Un inconveniente es que no ofrece servicio de no repudio de origen, debido a la sobrecarga que sufriría el sistema. Otro inconveniente consiste en que, al permitir que la comunicación entre distintas LANs este cifrada o no lo esté, estamos creando un posible punto de ataque al sistema (es el precio de la flexibilidad, obliga al administrador a tomar un mayor cuidado al establecer la topología de nuestro sistema).

No obstante, las ventajas merecen ser tenidas en cuenta. La primera ventaja que ofrece este diseño es que es una solución exclusivamente software por lo que su instalación no conlleva la modificación del hardware existente ni la adquisición de dispositivos nuevos, y además, es portable.

La segunda ventaja es la alta seguridad (ya que las comunicaciones internas de una organización se autentican y aseguran), con lo que se consigue la misma funcionalidad que en una red realmente privada. Además, se logra una gran flexibilidad ya que se sigue permitiendo el acceso genérico a cualquier punto de Internet.

Por último, se mantiene el uso de los mismos servicios, por lo que no hay que modificar el software UDP existente, siendo compatible con cualquier software que cumpla los estándares correspondientes.

De esta forma, obtenemos una RPV segura y flexible que, aunque nos exiga un poco más de control al crear nuestra red, nos permite ciertas funcionalidades muy necesarias en estos tiempos (acceso al exterior, flexibilidad,...)

# Referencias


[1] U. Black, “Frame-Relay: Specifications and Implementations”. McGraw-Hill, 1994

[2] R. Harbison, “Frame-Relay: Technology for our Time”. LAN Technology, Diciembre 1992

[3] J. Dávila, J. López, R Peralta, “Implementation of Virtual Private Networks at the Transport Layer”, Information Security Workshop, LNCS 1729, Springer 1999.

[4] J. Postel, K. Harrenstien, “Time Protocol”. RFC 868, May 1983.

[5] D. Mills, “Simple Network Time Protocol Version 4 for Ipv4, Ipv6 and OSI”, RFC 2030, Oct. 1996.

[6] SNMP Working Group, SNMP Documents, RFCs 1905, 1906, 1907, 2576, 2578, 2579, 2580.

[7] H. Schulzrinne, S. L. Casner, R. Frederick, V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications”, RFC 1889, Jan. 1996.

[8] H. Schulzrinne, A. Rao, R. Lanphier, “Real Time Streaming Protocol (RTSP)”, RFC 2326, April 1998.

[9] R. Atkinson, “Security Architecture for the Internet Protocol”. RFC 2401, Nov. 1998.

[10] National Bureau of Standards, “Data Encryption Standard”. U.S. Department of Commerce, FIPS pub. 46, Jan. 1977.

[11] X. Lai, J. Massey, “Hash Functions Based on Block Ciphers”. Advances in Cryptology – EUROCRYPT '92, Springer-Verlag, 1992.

[12] B. Schneier, “Description of a New Variable-Lenght Key, 64-Bit Block Cipher (Blowfish)”, Fast Security Workshop Proceedings, Springer, 1994.

[13] R. Rivest, “The RC4 Encryption Algorithm”, RSA Data Security, Mar 1992.

[14] R. Rivest, “The MD5 Message Digest Algorithm”. RFC 1321, April 1992.

[15] National Institute of Standards and Technology, NIST FIPS PUB 180. “Secure Hash Standard”. U.S. Department of Commerce, May 1993.

[16] Netscape Communications, “SSL 3.0 Specification”. *http://home.netscape.com/eng/ssl3/*

[17] T. Dierks, C. Allern, “The TLS Protocol version 1.0”. Internet Draft, November 1998.

[18] Varios, "RAT: Robust Audio Tool" *http://www-ice.cs.ucl.ac.uk/multimedia/software/rat*