Redes de Computadores I
Profesor Agustín González V.
Departamento de Electrónica UTFSM

Comunicación Cliente – Servidor
Protocolo Pydlock

Proyecto final, primer semestre de 2014

Ariel Gamboa - Nicolás Hermosilla - Oscar Lizama – Mario Marín

Índice

Consideraciones preliminares

Antes de proceder a detallar el grueso de la temática abordada, conviene retomar ciertos conceptos que han sido tratados tanto en clase como en el principio mismo del desarrollo del presente trabajo.

Se ha llevado a cabo una investigación exhaustiva por parte del equipo para encontrar opciones sobre lenguajes de programación que facilitaran la labor de crear desde cero una aplicación cliente y otra de tipo servidor, para lo cual se ha revisado documentación de C, C++, Java, C#, Python, Ruby y una serie de otras opciones, aunque estas son las más potentes y simples de todas ellas. Por otro lado, se investigó sobre numerosos algoritmos de hashing, entre los que se cuentan MD5 (algoritmo ya roto), SHA1 (otro algoritmo roto, aunque para lograrlo se requiere mucho más trabajo y potencia de cálculo), SHA256, RSA, y opciones más sencillas, aunque mucho menos seguras y más susceptibles a colisiones.

Se evaluó todo esto en el supuesto de contar con al menos un servidor, un pc o tablet cliente y uno más nodos que recibieran instrucciones, estos últimos pudiendo tratarse de mini computadoras Raspberry Pi o microcontroladores Arduino.

Finalmente, el grupo también incursionó en el tema del desarrollo de plataformas web, pues en un principio se pensó en desarrollar un cliente con acceso a través del navegador, pero esto fue descartado al considerar el limitado plazo con que se contaba para desarrollar una aplicación funcional y confiable.

Ideas básicas, objetivos y resumen del proyecto

El proyecto tiene como objetivo permitir acceder a un grado de control semejante al que se tendría estando físicamente al lado de una puerta, ventana, o cualquier otro aparato, pero a través de la red. Es necesario, por lo tanto, contar con una comunicación eficiente y efectiva entre aparatos, además de realizarla de manera segura. Idealmente se espera poder establecer canales cifrados para otorgar un grado de eficacia en lo anterior muy similar al que se tendría con una aplicación comercial, no obstante que es importante notar la naturaleza meramente académica de esta experiencia, por lo tanto se deja este como un objetivo secundario.

Contamos con los siguientes entes cuyo funcionamiento se detallará a continuación: el usuario, que cuenta con una aplicación cliente; el servidor, que mantiene al corriente los datos de los usuarios; y los terminales, que se basan en la micro computadora Raspberry Pi, e interactúan con los objetos cuyo control se busca tener.
La razón de elección de estos componentes y no otros, se detalla más adelante en el presente texto.

Para mantener el diseño relativamente escalable, optamos por un modelo similar al de Napster, vale decir, un híbrido entre p2p y conexiones centralizadas, que nos permite mantener cierto control sobre el sistema, sin sacrificar rendimiento de manera perceptible.

Los desafíos y las soluciones propuestas

Existen varios problemas que surgen de los objetivos mencionados anteriormente, y de los mismos se destacan algunos de importancia gravitante en el desarrollo del proyecto:

1. La seguridad

Pese a ser un diseño experimental y para nada un producto destinado a su implementación masiva, no deja de ser importante rellenar este espacio con algunas metodologías que permitan garantizar (al menos parcialmente) que el mecanismo entero tiene sentido. Se proponen en primer término las soluciones más usadas (y más complejas) del estilo de SSL/TLS, además de enmascarar todo el tráfico en una VPN. No obstante, incluso de no utilizarse aquello, se considera necesario proteger los datos de los usuarios tanto como sea posible, por lo que se propone utilizar en lugar de nombres de usuario y contraseñas en el servidor y los nodos, los valores de algún hash obtenido a partir de esos datos.

2. Presencia de NAT

En el entendido de que el servidor estará en una red ajena a la red privada donde operarán los nodos terminales, es menester contar con algún método para poder traspasar el filtro de la NAT. Se decide experimentar con dos soluciones básicas: UPnP (Universal Plug and Play) y VPN, con objeto de verificar cuál de los dos es más efectivo y simple para el hipotético usuario (además de ser viable para la demostración).

3. Eficiencia general del sistema

Este tema fue uno de los tópicos fundamentales a la hora de desarrollar todas las partes tanto del protocolo como del software que elaboramos. Se optó por evitar toda transmisión innecesaria de datos, para lo cual se propuso establecer mecanismos de comprobación de validez de la base de datos, además de limitar los mensajes posibles a sólo unos pocos, los cuales teóricamente deberían reducir las interacciones a entre 5 y 10 por cada sesión.

Criterios de decisión e implementación básica del protocolo

Luego de sopesar todo lo anterior, se determinó lo siguiente:

- Se abandona la idea de usar encriptación SSL/TLS, pues no se cuenta con el tiempo para conseguir hacerlo de manera aceptable.

- Dada la complejidad de implementar tanto técnicas de NAT traversal (a través de UPnP) como VPN, se opta por prescindir de esa parte del proyecto, principalmente porque para la presentación no es en absoluto necesario contar con ello.

- Se opta por el lenguaje de programación Python 2.7, que cuenta con bibliotecas de red de bajo nivel suficientes para lo que se busca conseguir, y además permite aprovechar los conocimientos que todos los miembros del equipo poseen del lenguaje, dada su presencia en la malla curricular universitaria.

- Se decide utilizar el algoritmo de hash SHA1, que está presente en la biblioteca hashlib de Python 2.7, el cual es suficientemente seguro, sencillo de utilizar y entrega lo que se necesita y nada más.

Situación final del proyecto: protocolo estable y listo para el uso

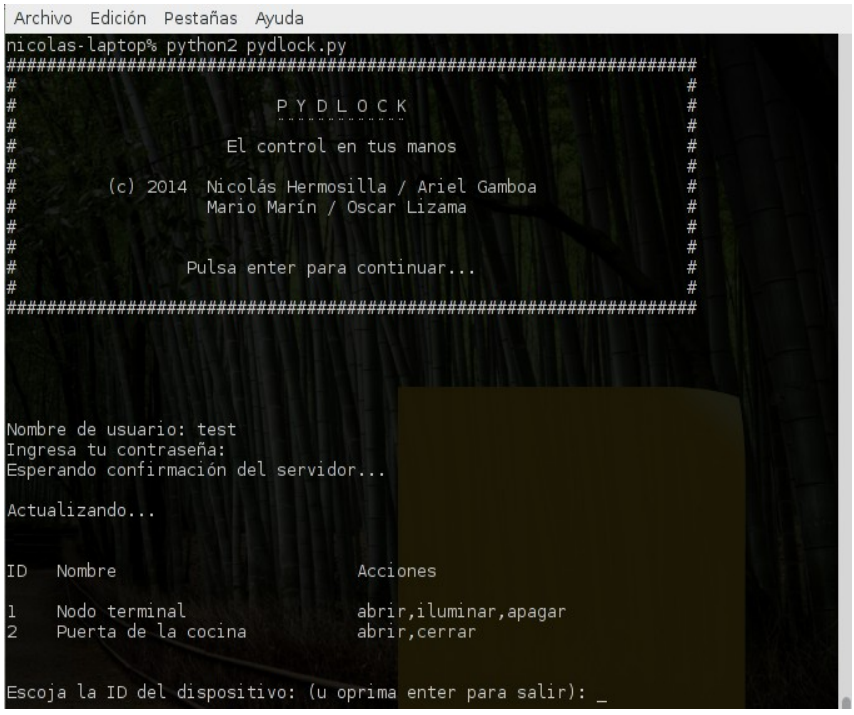
El protocolo final y la implementación del proyecto queda dividido en las siguientes partes:

1. El cliente de usuario

Partimos de la base que el usuario no tiene por qué conocer la forma en que interactúan los nodos de la red, por lo tanto la aplicación cliente no hace nada más que mostrarle lo que puede hacer y hacerlo. Para esto, realiza una serie de peticiones al servidor, que responderá convenientemente en cada caso:

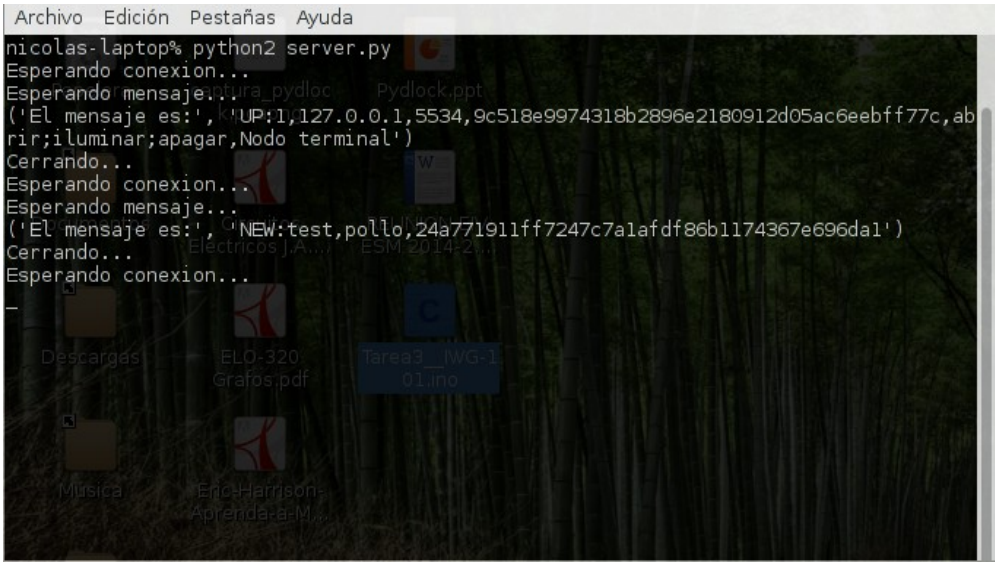
- GET:usuario,contraseña : si no se tiene conocimiento de base de datos alguna en el lado del usuario, se solicita a través de esta petición, cuya respuesta puede ser el archivo, línea por línea, con los datos necesarios (ver “Sobre la base de datos”), o simplemente “NOT-FOUND”, en cuyo caso se entenderá que el par usuario-contraseña no existen en el servidor.

- NEW:usuario,contraseña,hash : si ya se tiene una copia de la base de datos, entonces se envía al servidor un valor de hash de la misma, para ver si existen cambios. Si los datos no han variado, entonces el servidor responde con “NOT-MODIFIED”, tras lo cual la aplicación cliente seguirá utilizando los datos antiguos. Si ha cambiado, se procede a transmitir los datos de la misma forma que con el procedimiento GET.



2. El servidor

Esta parte es fundamental en el diseño de nuestra aplicación, y permite que se conecten correctamente los nodos cliente y terminal. El servidor recibe peticiones en el puerto 5533 tanto de parte de los clientes como de parte de los nodos terminales. Lo único que varía entre ambos tipos de petición es el primer campo de la misma, que identifica unívocamente al emisor. Los mensajes de tipo GET y NEW son únicos para el cliente de usuario, y el mensaje UP es enviado solamente por el terminal.



3. Los nodos terminales

Estos elementos están en permanente contacto con aquellas cosas que se busca controlar. Poseen un archivo en el cual se detalla el hash del usuario-contraseña dueño del aparato, así como su dirección IP, el puerto al que deben dirigirse las peticiones, y un identificador único o ID. Reciben peticiones únicamente de parte del usuario, quien envía los comandos según los datos que posee en su archivo propio, descargado previamente del servidor. El formato básico de petición que reciben es:

usuario:contraseña:orden

Además de recibir peticiones, envían constantemente peticiones de actualización de datos al servidor. Para esto, utilizan el siguiente método:

UP:id,IP,puerto,”hash de usuario-contraseña”,acción1;acción2;etc,nombre

4. La base de datos

Está presente principalmente en el servidor, pero es copiada a cada nodo cliente, donde se almacena y actualiza si es necesario. Posee los datos correspondientes a la ID de cada nodo terminal que el usuario está autorizado a acceder, así como los comandos válidos para cada uno, las direcciones ip y los puertos a los cuales es necesario dirigirse. Obtiene su nombre del hash SHA1 que arroja el usuario y su contraseña al unirlos, con la extensión dat. El archivo posee líneas con la siguiente estructura:

ID,”nombre del objeto”,acción1:acción2:acción3...,IP,puerto

Conclusión de la experiencia

A través del no menor trabajo de diseñar e implementar un protocolo propio, pudimos notar que esta labor dista mucho de ser trivial, pero eso no impide que sea posible para cualquier equipo que se proponga innovar. La enorme cantidad de documentación y la implementación desde una perspectiva abierta y en capas de Internet, hace posible que cualquiera pueda montar un servicio que aproveche las capas inferiores y se centre en aquello que desea modificar, sin comprometer por ello el funcionamiento del resto de aplicaciones.

Lo que estudiamos durante el curso fue fundamental para comprender mucho de aquello con lo que nos encontramos a la hora de buscar alternativas para poder hacer realidad lo que partió como una idea que se veía compleja, difícil y ardua, pero que finalmente probó valer la pena y dejarnos con la satisfacción de saber que los proyectos de esta clase no son para nada imposibles de poner en práctica, y que los recursos que se requieren para hacerlo son en gran medida tiempo y la voluntad de llegar a buen puerto.

Como curioso apunte, también comprobamos que lo estudiado en Programación I resultó ser bastante útil al momento de poner las manos en la masa, pues sin ese conocimiento previo habríamos precisado de mucho más tiempo para poder empezar a escribir el código de nuestra aplicación. Así también confiamos en que este mismo proyecto sentará las bases de nuestro propio desarrollo en otras iniciativas orientadas a la innovación tecnológica, en las cuales, hemos visto, siempre hay mucho por hacer.