Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Redes

Remote-QR

Desarrollado por

- #### Manuel Ruiz Androvetto (2017074102)
- #### Kevin Segura Rojas (2017153767)

Alajuela, I-S 2020

Introducción

La idea central del presente trabajo es aprender sobre el desarrollo e implementación de redes de comunicación y el modelo de capas. Las redes de comunicación son sistemas utilizados por las personas para poder compartir información de manera eficiente y típicamente segura. Una buena red de comunicación puede ser fundamental para el desarrollo pleno de una sociedad y la utilización de la computación para mejorar la rapidez y eficiencia de las mismas ha devenido en el planeta altamente conectado en el que vivimos.

A pesar de esto, sigue existiendo desigualdad en el acceso a la información e incluso se ha podido establecer límites políticos a la información que se transmite en un determinado territorio. Es por esto que resulta de vital importancia, para un estudiante de computación, comprender la importancia de este tema y buscar soluciones que puedan garantizar otros mecanismos de comunicación perfectamente aplicables en casos de un acceso limitado a las redes más comunes.

El desarrollo de la red planteada, se basará en el modelo OSI. El modelo OSI propuesto por la Organización Internacional para la Estandarización (ISO) en 1977 y aprobado en 1984 consta de 7 capas que definen el proceso de transferencia de datos de un punto a otro. En este preciso caso, se implementarán cuatro capas.

Siendo Python uno de los lenguajes más utilizados en la actualidad, resulta oportuno utilizarlo como el lenguaje principal para el desarrollo de la red. Además, Python cuenta con una extensa comunidad de desarrolladores que contribuyen a distintas bibliotecas diariamente. No obstante, no se descarta la posibilidad de, si es necesario, hacer uso de lenguajes como Java o C y también la utilización de la línea de comandos de Ubuntu y MacOS.

Para realizar el trabajo, se tendrán cuatro partes principales:

1. Capa 1 Medio Físico: Se generará un protocolo para la transmisión de información a través de luz en forma de códigos QR. Se implementará una biblioteca libre en el lenguaje python para generar la comunicación entre el nodo y el dispositivo de transmisión con el objetivo de que el sistema pueda ser escalable y se puedan desarrollar nuevas implementaciones.
2. Capa 2 y 3 Remote-QR: red que funciona sobre TCP/IP. La idea con esta red es que los paquetes enviados sean trasladados de forma anónima. El sistema considerado tiene que poder generar la transmisión de información a dispositivos como cámaras, celulares, u otros aparatos que puedan interpretar las señales de luz, por lo que tiene que existir puertas de enlace a estos dispositivos.
3. Capa 4 Aplicación Remote-QR: Aplicación que permita el envío de mensajes a través de la red Remote-QR y también la lectura de mensajes recibidos.
4. Capa 4 Aplicación Clearnet: aplicación que permite la comunicación dentro de la red y que estará centralizada en un servidor de tipo IRC. Esta aplicación debe estar constantemente encargándose de la transmisión de nuevos mensajes. Una vez implementadas todas las partes del sistema, se podrán compartir mensajes de chat a través de remote-QR. Estos mensajes podrán ser generados a través de la aplicación de escritorio y se podrán interpretar desde otra aplicación remote-QR de escritorio o bien desde un dispositivo de transmisión, interpretando la luz emitida en una pantalla en forma de un código QR.

Ambiente de Desarrollo

Herramientas a utilizar: * Python3, Pycharm CE como ambiente de decodificación. * MacOS Catalina y Ubuntu 18.04 como sistemas operativos. * Bibliotecas de Python openCV, Socket y matplotlib * Github para el control de versiones. * Forma de Debugging: Pycharm Debugger o ipdb.

Flujo en el sistema de control de versiones:

1. Creación del repositorio en github
2. Compartir link al repositorio
3. Creación de un branch para el desarrollo de cada feature
4. Realizar pruebas respectivas al feature del branch dado
5. Rebase o merge del branch a master
6. Repetir pasos 3, 4 y 5 hasta finalizar con todos los features solicitados por el profesor

Estructura de datos usadas y funciones

1. remote-QR:
    - Dispositivo de Transmisión: el dispositivo de transmisión se encarga de tomar un mensaje o un archivo indicado y transmitirlo a otro punto a través de códigos QR. Para esto, se genera un encabezado de trama (levemente) dinámico y se toma una parte del mensaje o bien del archivo convertido a arreglo de hexadecimales, se junta con el encabezado y luego se genera un código QR (en formato .png) y además se representa la imagen en pantalla.
        - file_a_qr(): abre un archivo como un arreglo de bytes en formato hexadecimal y hace el llamado a texto_a_qr para procesar la información
        - texto_a_qr(): toma la version, direccion y el payload en formato hexadecimal y genera multiples tramas que luego son convertidas en códigos QR.
        - crear_qr(): toma la trama en formato de texto string y lo convierte en un código qr dando como resultado un archivo de tipo .png

        - generar_nombre_archivo(): genera el nombre del archivo donde se almacenará el código qr

    - Dispositivo Luz Adaptador: El dispositivo luz adaptador, tiene dos funcionameintos distintos. El principal es el original y el secundario fue generado para testeo. Por un lado, el funcionamiento principal consiste en utilizar la cámara web de la computadora para poder leer códigos QR mostrados a través del medio que fuera necesario. El segundo funcionamiento se da cuando se desea probar un archivo de gran tamaño para el que se tienen todos los códigos QR ya generados. En este caso, se toma el directorio con los archivos, se ordenan alfabéticamente y se ingresa en un ciclo que va extrayendo la información y generando un archivo o un mensaje a partir de eso.

        - leer_imagenes(): Lee los archivos .png a partir de una ruta a un directorio
        - leer_con_camara(): Lee los QR a partir de luz interpretada por la cámara web de la computadora
        - interpretar_data(): Interpreta cada arreglo de bytes de manera que se separen los datos contenidos en cada trama
        - crear_archivo(): Genera un archivo llamado tempfile donde se va a contener la informacion transmitida a través de archivos QR.
        - verificar_direccion(): Verifica que la direccion del nodo donde es transmitido el paquete sea el mismo de la maquina actual
        - mostrar_direccion(): Muentra en pantalla la direccion asignada a la computadora actual

    - Generador Trama: clase con metodos que sirven como ayuda para la creacion de una trama a partir del payload. Sus métodos ayudan en la construcción de una trama en formato de string que luego será utilizado para crear el código qr.

        - crear_trama(): Método que le da la estructura a la creación de la trama
        - armar_header(): Método que le da la estructura la header concatenando los valores hexadecimales de los parámetros separados por un '|'
        - calcular_checksum(): Método que calcula el checksum de cada trama, retornando su valor numérico (int)
        - armar_payload(): Método que arma el payload concatenando el checksum con el payload utilizado. Retorna la trama lista y el payload restante

  En ambos mecanismos, para cada trama que se lee, se obtiene una serie de datos como la versión del protocolo utilizado, el checksum y el id del código QR. Estas propiedades son utilizadas para asegurarse de que la lectura haya sido realizada en el lugar adecuado, que cada trama llegue completa y se lea en el orden que es debido.

2. Red Mesh:

    - Entre las estructuras creadas por nosotros tenemos:
        - Paquete Mesh: Que contiene la clase "Paquete", estructura que vendría siendo el paquete que se envia a traves de la red mesh, cuenta con los campos: los datos del destino siguiente y el mensaje. Donde el mensaje puede ser otro paquete, representando así el sistema de "carta dentro de otra carta" del sistema anonimo ejemplificado en clase.
        - Manejo nodos: Que cuenta con dos clases, "CrearNodo" que representa a cada cliente y almacenas los datos del mismo(ip, puerto y un número de identificación único), y "ListaNodos" que almacenas a todos los clientes disponibles y administra esta lista.
    - Mesch principal: Funciona como un servidor de registros a la red mesh, se encarga se agregar a los que clientes nuevos y habilitar el envio de mensajes a este nuevo cliente desde el resto de clientes. Sus funciones principales son:
        - crear_nodo_registro(): Que genera todos los datos requeridos para que el cliente se integre a la red, y los almacena en la estructura de datos "CrearNodo".
        - agregar_cliente: función Que agrega el nuevo cliente a la mesh.
        - registro: Función que esta constantemente a la escucha de la solicitud de ingreso o salida de un cliente.
    - Nodo Mesh: Esta clase vendría a representar cada cliente de la red mesh, se conecta con el servido de registro para su inscripción, se encarga de generar los paquetes y enviarlos. Entre sus funciones principales están:
        - solicitudes(): Que esta en constante escucha de mensajes que recibió el cliente.
        - reenviar_paquete(): Si el mensaje recibido no es para el cliente, o se desea mandar un mensaje propio, esta función se encarga de quitar una capa del paquete(de ser requerido) y reenviarlo a su siguiente destino.
        - crear_ruta(): cuenta con varias funciones auxiliares, y en conjunto con estas, genera una ruta aleatoria para llegar al destino y generar el paquete que se enviará por esta ruta.
        - enviar_mensaje(): Función que actua como terminal del usuario, esta constantemente esperando que el cliente escriba, y reacionar de acuerdo a la solicitud del cliente.

3. IRC:

    - Utilidades IRC: Contiene la clase "IRC", clase que se encarga de las posibles acciones en la comunicación con el servido IRC, acciones representadas con las funciones:
        - conexion_servidor(): Conecta con el servidor solicitado.
        - enviar(): Enviar un mensaje.
        - respuesta(): Obtener los mensajes recibidos del servidor.
    - Bot IRC: Representa al bot IRC, se encarga de escuchar las solicitudes del cliente y hacerselas saber al servidor IRC. Las principales funciones con las que cuenta son:
        - buzon(): Función que está constantemente esperando respuestas del servidor para mostrarselas al cliente.
        - terminal_usuario(): Función que recibe una solicitud de envio del cliente y se la hace llegar al servidor IRC.
        - activar(): Función que "enciende" al bot y todas sus funcionalidades.

Instrucciones para ejecutar el programa

Para ejecutar el programa, simplemente es necesario abrir una terminal de comandos y dirigirse hasta el directorio donde está almacenado el archivo ejecutable `remote-qr`. Una vez en este directorio, solo hace falta ingresar el comando `./remote-qr` y el programa iniciará. Una vez el programa esté corriendo, las instrucciones aparecerán en pantalla.

Si en algún momento es necesario salir del programa de manera forzada, puede estripar las teclas `control + z` y el programa se cerrará.

Actividades realizadas por el estudiante

Manuel Ruiz

- Fecha 8 de junio. Duración: menos de una hora. Actividad: Creación de repositorio. Generar un repositorio en la plataforma Github para el manejo de versiones del programa. El siguiente es el link al repositorio utilizado. https://github.com/mruiz75/remote-qr
- Capa 1:
    - Fecha 20 de junio. Duración: 3 horas. Investigación. Investigación en internet sobre código QR, procesamiento de códigos QR, bibliotecas de Python para le manejo de códigos QR.
    - Fecha 22 de junio - 2 de julio: Duración 10 horas. Creación del protocolo: análisis de elementos indispensables para el funcionamiento de la red como fue solicitada. El protocolo hace uso de tramas de 128 bytes compuestas por un header y le payload. Ambas partes tienen un tamaño dinámico dependiendo del tamaño de las propiedades agregadas al header.
    - Fecha 30 de junio - 10 de julio: Duración: 8 horas: Dispositivo de Transmisión. Generación de un dispositivo que se encarga de convertir un mensaje o

un archivo a una serie de códigos QR
    - Fecha 30 de junio - 10 de julio: Duración 8 horas: Dispositivo Luz Adaptador. Generación de un dispositov que se encarga de convertir una serie de imágenes de tipo 1 .png conteniendo los códigos QR en un mensaje o un archivo final.
- Fecha 11 de julio. Duración 5 horas. Actividad: Menú principal. Desarrollo del menú principal del programa para facilitar la interacción con las distintas partes del proyecto. Testeo del funcionamiento de cada una de las partes y corrección de errores.
- Fecha 13 de julio. Duración 2 horas. Actividad: RFC: creación del RFC que contiene la información detallada sobre el protocolo utilizado.
- Horas totales trabajadas: 36 horas

Kevin Segura

- Fecha: 22 de Junio. Duración: 3 horas. Actividad: Investigación básica sobre redes, sockets, puertos, redes mesh y como hacer uso de estos elementos en Python3. Además se creo una pequeña funsión que genera los identificadores únicos requeridos en ciertas partes del proyecto.
- Fecha: Del 24 al 28 de Junio. Duración: 8 horas. Actividad: Implementación de todas las clases y funcionalidades que el grupo vio necesarias para poder crear una red mesh que cumpla con las características requeridas del proyecto. Además de hacer pruebas de su correcto funcionamiento.
- Fecha: 2 de Julio. Duración: 1 hora y 30 minutos. Actividad: Se agrego una funcionalidad faltante de la red mesh, se limpio el código y se hizo la documentación interna.
- Fecha: 8 de Julio. Duración: 3 horas y media. Actividad: Investigación sobre IRC, el como poder utilizar un servidor IRC público y crear un bot IRC.
- Fecha: 9 de Julio. Duración: 4 horas. Actividad: Se implemeto al completo las clases para comunicarse con un servidor IRC público y un bot simple que maneje las comunicaciones con este. Pruebas del servicio y documentación interna.
- Fecha: 12 de Julio. Duración: 1 hora. Actividad: Solución de pequeño bug en la red mesh al unir todos los elementos del proyecto.
- Fecha: 13 de Julio. Duración: 1 hora. Actividad: Aporte en la documentación y RFC.
- Horas totales trabajadas: 22 horas.

Ambos (Reuniones)

- Fecha: 20 de Junio. Duración: 2 horas. Actividad: Se habló sobre nuestros conocimientos y lo que entendiamos del proyecto, se hizo el diseño inicial a grandes rasgos y se hizo la separación de tareas.
- Fecha: 29 de Junio. Duración: 1 hora. Actividad: Se informo sobre las tareas realizadas en ese tiempo y se hicieron pruebas al servicio mesh.
- Fecha: 10 de Julio. Duración: 1 hora y 30 minutos. Actividad: Reunión de seguimiento del proyectos, demostración de la transmisión y recomposición de los QR, pruebas al servicio IRC junto con el bot implementado.
- Fecha: 12 de julio. Duración: 1 hora. Actividad: Plática sobre nuestros pensamientos sobre el proyecto, prueba del producto final y definición de los aspectos importantes a plasmar en el RFC.
- Fecha: 13 de Julio. Duración: 1 hora. Actividad: Revisión de todos los entregables y subida de los mismos.

Comentarios finales

Manuel Ruiz

La experiencia de trabajar con redes es altamente educativa desde mi punto de vista. Las redes, como hemos aprendido, no solo están conformadas por medios físicos convencionales como cables de metal, sino que se pueden generar a través de cualquier tipo de método físico. Este proyecto es un claro ejemplo de cómo se pueden desarrollar grandes ideas si tan solo se sale un poco de la línea de pensamiento típica de las personas. Remote-QR nos presenta un pequeños protocolo de comunicación que no pareciera resolver ningún problema actual de comunicación, en especial si se compara con el internet convencional. No obstante, la transmisión por luz puede verse como una herramienta útil para cuando se corta la comunicación vía internet (e.g. algunos de los países durante la primavera árabe del 2011).

Más allá de la idea de redes, este proyecto planteó un reto fuerte para los estudiantes al exponerlos a un tipo de desarrollo que requiere una fuerte planificación y estructuración para lograr el objetivo. Este proyecto representa un acercamiento fuerte a la solución de problemas cada vez más complejos y al mismo tiempo explorando la creatividad y el ingenio de los estudiantes.

El proyecto está planteado de manera escalable y se planea liberar el código por si alguien en algún momento desea mejorarlo o aportar a su desarrollo.

Kevin Segura

El proyecto me parecio muy interesante y educativo, esto porque se exploro como hacer redes de forma diferentes a la vez que dejamos nuestra marca personal en estas. Con esta experiencia se pudo vivir más de cerca cómo es que una red se comunica, que si bien no es la misma que se usa a nivel mundial, sí ayuda a dar un mejor entendimiento de esta.

En cuestion del estado del proyecto, estoy complacido porque se logro en gran medida completar lo que se solicito. Mientras que cuestión de problemas y limitaciones se junto el que yo no he llevado sistemas operativos, por lo que cosas que podrían parecer obvias como sockets y puertos no tenía ni la más mínima noción de qué eran, por lo que tuve que redoblar esfuerzos en esa parte para al menos tener un entendimiento antes de empezar a programar. Otro problema que surgió fue que se tuvo hacer un estudio profundo de los comando que utilizan los IRC públicos por definición, pero por dicha este no fue mayor inconveniente al estar muy bien documentado.

Conclusiones y recomendaciones

La comunicación es una amplia área de estudio pero también es un derecho que todos tenemos. La combinación de métodos prácticos con métodos técnicos para garantizar la mejor transmisión de la información, nos lleva al desarrollo de redes de comunicación con estrictos protocolos que hoy en día mantienen al mundo entero conectado y que simultáneamente democratiza la comunicación a corta y larga distancia.

Para pensar en redes, es fundamental entender que las redes van más allá de cables de metal. Las redes pueden desarrollarse de infinitas maneras y con cualquier tipo de medio que se desee, la importancia está en el establecimiento de canales de comunicación que sigan protocolos para el envío y recibo eficiente y eficaz de información.

Por otra parte, la organización es fundamental para el desarrollo de proyectos programados grandes. Independientmente del nivel de complejidad de un proyecto, para su éxito, siempre va a ser necesario un alto grado de organización. Esta organización va desde la interacción entre compañeros de equipo y el manejo de las responsabilidades de cada uno. Establecer sólidos canales de comunicación y comprender el compromiso son elementos básicos para una buena organización. Para el caso específico de desarrollo de software, utilizar software para el manejo de version (e.g. Git) resulta de muchísima ayuda.

También, como parte de las tecnologías necesarias, se escogió Python como lenguaje designado para todo el proyecto. Python presenta la facilidad de su sencilla y muy legible sintáxis. Aun así, se tuvieron múltiples problemas para tratar el manejo a bajo nivel de la memoria para el caso de las tramas de Remote-QR. Como se sabe, Python se encarga de manejar la memoria por su cuenta, normalmente, al instanciar una variable, Python asigna un espacio de memoria más grande del de la

variable con el fin de ahorrar tiempo si la variable debe ser extendida. Esta automatización de Python hace que al tratar de precisar cada trama como menor de 128 bytes, se obtenían resultados ambiguos y a veces poco concluyentes.

Finalmente, se recomienda el involucramiento de cualquier persona interesada en expandir una red como remote-qr y aprovechar sus beneficios. Al liberar el código, los miembros de este grupo nos comprometemos a la ayuda de cualquier personas que quiera seguir avanzando con el proyecto.

Bibliografía

- Cultiracion.(s.f). "¿Qué es un puerto y para qué se utiliza?". Disponible en: https://culturacion.com/que-es-un-puerto-y-para-que-se-utiliza/
- SPEEDCHECK.(s.f). "Socket". Disponible en: https://www.speedcheck.org/es/wiki/socket/
- DORKY.(2017). "Cuál es la diferencia entre un puerto y un socket?". Disponible en: https://www.dokry.com/7182#:~:text=Un%20puerto%20puede%20describirse%20como,procesos%2C%20en%20Internet%20o%20localmente.
- Unipython.(s.f). "Programación de Rede en Python: Sockets". Disponible en: https://unipython.com/programacion-de-redes-en-python-sockets/
- Marc_intosh.(2018). "Qué son las redes Mesh". Disponible en: https://www.macnificos.com/blog/que-son-las-redes-mesh
- Son Link.(2020). "[Python] Programar Un Bot Para IRC. [online] Desde Linux". Disponible en: https://blog.desdelinux.net/python-programar-un-bot-para-irc/
- Galache.(2002). "Comando de IRC". Disponible en: https://www.geeknetic.es/Guia/7/Comandos-de-IRC.html
- mIRC Co. (2020). "IRC Networks and Servers". Disponible en: https://www.mirc.com/servers.html