**Diseño de un stack de protocolos que permita crear un sistema de comunicación entre dos hosts**

Kevin Acuña Hernández, Bosco Jiménez Córdoba,

Gustavo Vargas Quesada

**Resumen**

Se diseñó un stack de protocolos que permite crear un sistema de comunicación. Este stack de protocolos se definió basado en el modelo OSI, aunque algunas de las capas no son necesarias para su implementación del proyecto. Se renombran las capas con un nombre más significativo, acorde a las funcionalidades que desempeñan.

Se requirió transmitir un mensaje de un punto A hasta un punto B utilizando el Internet como medio de transmisión, por ende, el protocolo diseñado debe contar con un software que demuestre su funcionamiento, el mensaje que se envía debe usar un protocolo de comunicación no conocido.

Cuando un usuario desee enviar un mensaje a un destino utiliza una interfaz diferente y novedosa, es decir no utiliza la tradicional escritura empleando un alfabeto ya sea en castellano o inglés, si no por medio de algún otro sistema, en este caso se emplea el lenguaje de símbolos.

**Introducción**

Se diseñó un sistema que cuenta con un stack de protocolos basados en el modelo OSI, este sistema se desarrolla bajo el lenguaje Python 3 y creada mediante la herramienta o IDE de Visual Studio community, para la parte gráfica se utilizó la librería TKinter que proporciona una interfaz gráfica amigable y fácil de implementar.

Cada implementación de las capas contiene los métodos o variables necesarias para el funcionamiento de estos.

El stack de protocolos desarrollado no incluye la capa física, esto debido a que no se cuenta con la capacidad de crear su propia NIC (Network Interface Controller), por este motivo se utiliza el stack de Internet.

Se requería transmitir un mensaje de un punto A hasta un punto B utilizando el Internet como medio de transmisión, por ende, el protocolo diseñado debía contar con un software que demostrara su funcionamiento, el mensaje que se envía debe usar un protocolo de comunicación no conocido. Además, la capa de red no se implementó muy técnicamente, ya que lo único que se requiere de esta es que contenga la información tanto del host de origen, como el host del destino al que se quiere transmitir el mensaje.

También se debía desarrollar un sistema similar al del cliente pero que contuviera el lado del servidor, por consiguiente, se implementa las mismas capas, pero con la variante de que estas se ejecutan inversamente a comparación a la del cliente

**Desarrollo**

1. **Capa de Aplicación**

Esta capa sirve como ventana a los usuarios y los procesos de las aplicaciones para acceder a servicios de red, ella contendrá los métodos del la GUI que permiten al usuario interactuar con el software en sí. Esta capa controla los eventos realizados por el usuario mediante la interfaz gráfica, (Ver figura 1) la cual contiene los métodos desarrollados en esta capa.

Captura de pantalla de un celular con letras

Descripción generada automáticamente

Figura 1 Capa de aplicación OSI. (Elaboración propia).

1. **Capa de presentación**

El objetivo de esta capa es encargarse de la representación de la información, de manera que, aunque distintos equipos puedan tener diferentes representaciones internas de signos los datos lleguen de manera reconocible. Esta capa contiene dos métodos desarrollados, estos métodos son el de codificar y el decodificar, además definen el protocolo de encriptación de invención propia y el que cual se efectúa de forma exitosa para seguridad o cifrado de los datos transmitidos a través de la red. Estos métodos cuentan con una serie de sentencias **if** que lo que hace es comparar la letra enviada y luego asignar a una cadena de texto el símbolo equivalente a esta (Ver figura 3), esto para el caso de la codificación ( Ver figura 2), pero en el proceso de decodificación lo que hace al igual que el método anterior mencionado es recibir el símbolo y colocar la letra equivalente a esta en una cadena de texto para luego ser manipuladas en las demás capas del stack (Ver figura 3).

Imagen que contiene texto

Descripción generada automáticamente

Figura 2 Método de codificar. (Elaboración propia).

Imagen que contiene computadora

Descripción generada automáticamente

Figura 3 Método de decodificación. (Elaboración propia).

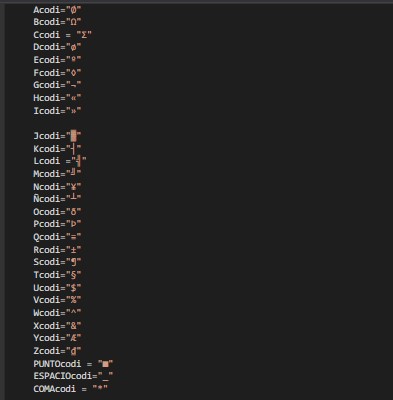


Figura 4 Lenguaje de encriptación empleado. (Elaboración propia).

1. **Capa de Sesión**

Esta capa es la que se encarga de mantener y controlar el enlace establecido entre dos ordenadores que están transmitiendo datos. En esta capa se encuentra generado un método que establece comunicación con el servidor, el cual consiste en verificar si el servidor con el cual el sistema se va a estar comunicando se encuentra en línea y listo para recibir los datos a enviar, para esto lo que hace es enviar un mensaje(bandera) al servidor y si se recibe respuesta por parte de este es que se encuentra activo y funcional (Ver figura 5). Por lo tanto, la prestación suministrada por esta capa es la capacidad de asegurar que, dada una sesión establecida entre dos máquinas, la misma se pueda consumar para las operaciones establecidas de principio a fin.

Captura de pantalla de un celular

Descripción generada automáticamente

Figura 5 Capa de sesión OSI. (Elaboración propia).

1. **Capa de transporte**

El objetivo de esta capa es recibir los datos del nivel superior, dividirlos (segmentación y reensamblado) si fuera necesario y enviarlos al nivel de red para que los transmita, asegurando que todos los trozos lleguen de manera correcta y eficiente al otro extremo (ordenación de paquetes). Para ello se implementó un método llamado multiplexar (Ver figura 6), el cual hace que la cadena codificada sea partida en segmentos para poder ser enviado a través de la red de forma segura, en el lado del servidor cuenta con un método similar para reensamblar los segmentos recibidos y asegurar que esta cadena recibida sea la correcta mediante un método que realiza la comprobación.

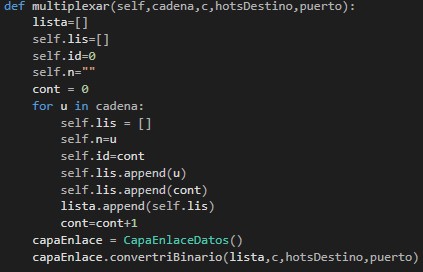


Figura 6 Método de segmentar el código. (Elaboración propia).

1. **Capa de red**

La capa de red suministra los medios necesarios para la transferencia de información entre sistemas finales a través de algún tipo de red de datos. De esta manera, las capas superiores no tienen que conocer la técnica de conmutación empleada o el camino que seguirá el mensaje hacia el destino. Esta capa en el proyecto va a contener la información necesaria para la conexión de los sockets, esta información será los datos de la máquina de origen y el host de destino (Ver figura 7).

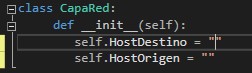


Figura 7 Capa de red OSI. (Elaboración propia).

1. **Capa de enlace**

La capa de enlace de datos tiene como objetivo brindar a los niveles superiores un vínculo libre de errores, suministrando mecanismos para el control y la localización de errores. Además, ofrece medios para activar, mantener y desactivar este enlace. En esta capa se elaboró cuatro métodos (Ver figura 8), el de escuchar servidor lo que hace es recibir información por parte del servidor y realizar ciertas acciones preestablecidas, luego está el de enviar datos lo que hace esta método es hacer el envío de la cadena generada en las capas anteriores, por otra parte se encuentra el método de convertirBinario, lo que hace este método es tomar los segmentos de símbolos generados en las capas superiores y convertirlos en segmentos binario, por consiguiente el método convertirOriginal toma los segmentos recibidos en binario y convertirlos al lenguaje de símbolos desarrollados.

Captura de pantalla de un celular

Descripción generada automáticamente

Figura 8 Capa de enlace OSI. (Elaboración propia).

**Resultados**

A la hora de ejecutar el programa desarrollado obtuvimos los siguientes datos:

Como podemos ver en la figura 9 hay una ventana inicial en el lado del cliente la cual permite escribir un mensaje para luego por medio del botón con rotulación de enviar, lo cual poder transmitir el mensaje al servidor mediante los sockets empleados (ver Figura 10).



Figura 9 Ventana inicial del lado del cliente. (Elaboración propia).

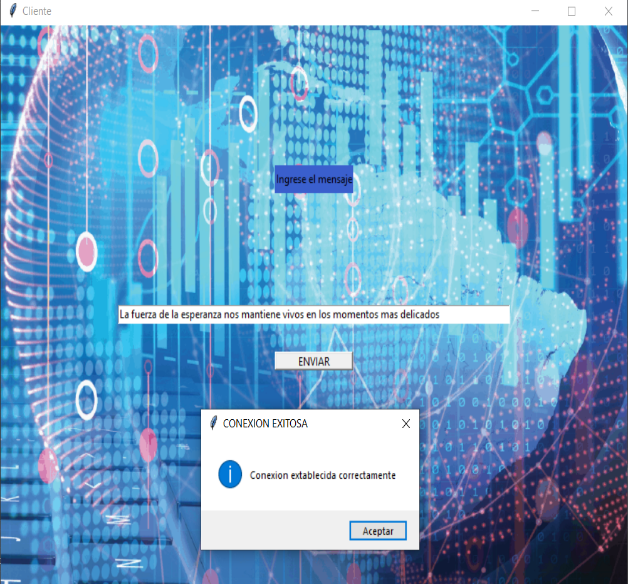


Figura 10 Ventana después de haber enviado el mensaje. (Elaboración propia).

En la figura 11 podemos observar la ventana inicial del servidor el cual se encuentra inactivo ya que no se ha accionado mediante el botón de iniciar.

En la figura 12 podemos observar la ventana del servidor el cual se encuentra activo y en espera del mensaje enviado por parte del cliente.

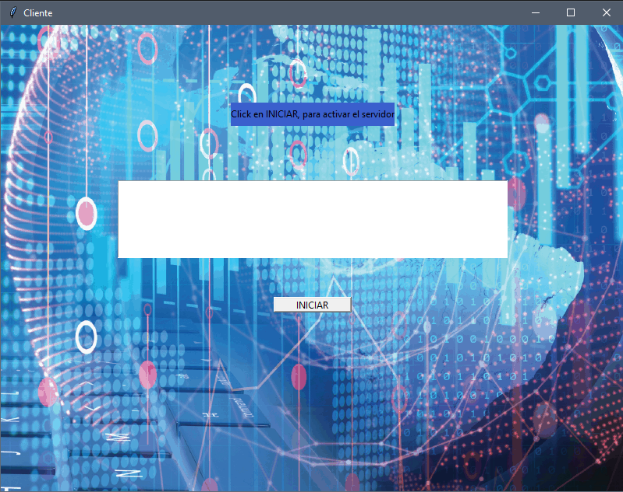


Figura 11 Ventana inicial del lado del servidor. (Elaboración propia).

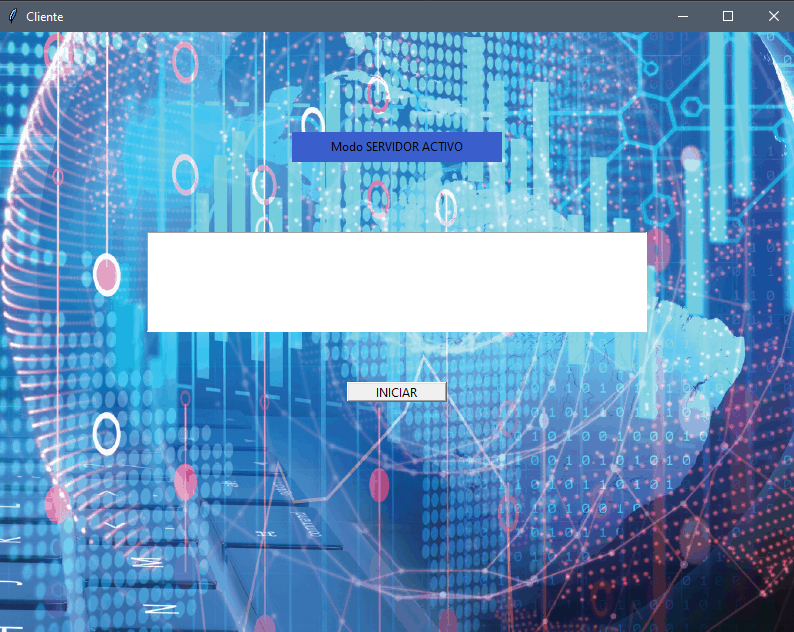


Figura 12 Ventana del servidor en espera del mensaje. (Elaboración propia).

Una vez detectado que hay error en la cadena enviada se le notifica al servidor que hay un error (Ver Figura 13) y que este va a ser corregido mediante el método empleado para recuperación de segmentos perdidos. Una vez corregido se muestra la cadena correcta recibida por parte del cliente (Ver figura 14).

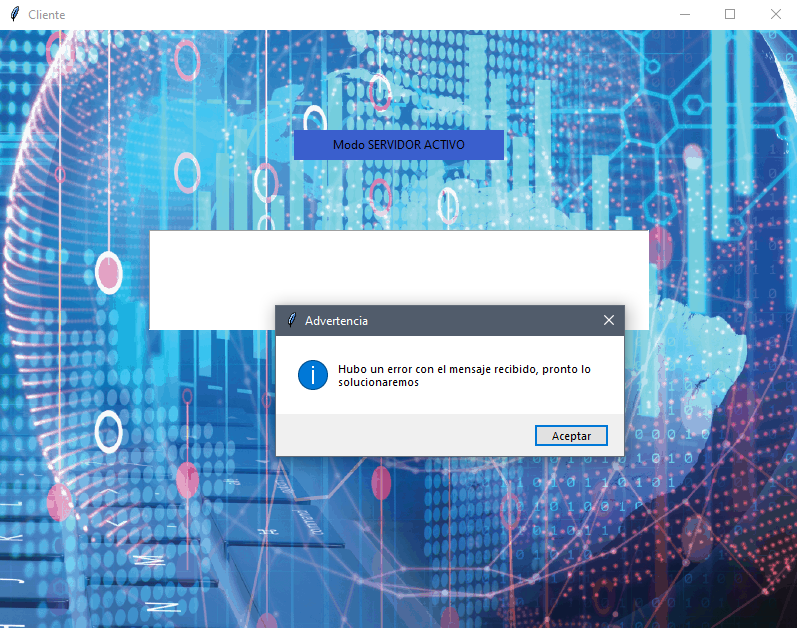


Figura 13 Ventana en el lado del servidor cuando ingresa mensaje con error. (Elaboración propia).



Figura 14 Ventana del servidor con el mensaje corregido. (Elaboración propia).

**Conclusiones**

En el transcurso de la elaboración de este trabajo tuvimos un problema con la conversión a binario, ya que el método empleado para realizar la conversión a esta y viceversa en números, lo que pasaba es que no aceptaba números de más de dos dígitos, por lo que se procedió a elaborar una solución que corrigiera esta problemática sin tener que cambiar todo el método desarrollado.

A la hora de codificar la palabra se generó otro problema ya que no se identifica algunos símbolos por que estos no se encontraban dentro del código ASCII. Por lo que se procedió a cambiarlos por otros que si se pudieran detectar.

Al encender los servidores, estos generaban que las demás cosas en la parte grafica no se pudieran utilizar ya que se generaba como un bucle, por lo que se tuvo que cada vez que se encendiera se tenia que cerrar la conexión para que esto no suceda.

Al iniciar con la elaboración del trabajo no sabíamos a que se tenia que llegar o lo que había que desarrollar en cada capa del modelo OSI, por lo que tuvimos que consultar e investigar para por lo menos tener una idea de lo que se debía que hacer.

**Recomendaciones**

Se recomienda para futuros trabajos en el desarrollo del protocolo OSI diseñarlo en un lenguaje que ofrezca mejores herramientas como lo es el lenguaje Java. Pero si se va a desarrollar en el lenguaje Python se debe tener el conocimiento base sobre este.

Además, también se recomienda tener el conocimiento base sobre lo que es el modelo OSI y todo lo que conlleva en este, ya que es un factor importante para su posterior desarrollo.

Por otra parte, en la implementación de los sockets se recomienda que, en lugar de pasar cadenas, buscar otra forma de enviar datos, ya sea como es el envío de objetos, debido a que si se procesa una cadena conlleva más trabajo en su manipulación, en cambio si es un objeto es más fácil para manipular.

# **Bibliografía**

Castillo, J. (2018). *Modelo OSI: que es y para que se utiliza*. Obtenido de https://www.profesionalreview.com/2018/11/22/modelo-osi/

Leopedrini. (2011). *Las 7 Capas del Modelo OSI*. Obtenido de http://www.telecomhall.com/ES/las-7-capas-del-modelo-osi-.aspx

Networking World. (2017). *Cómo entender (y recordar) el modelo de red de 7 capas*. Obtenido de https://www.networkworld.es/networking/como-entender-y-recordar-el-modelo-de-red-de-7

Solano, J. (s.f.). *El modelo OSI*. Obtenido de http://dis.um.es/~lopezquesada/documentos/IES\_1213/LMSGI/curso/xhtml/xhtml22/documentos/index5.html