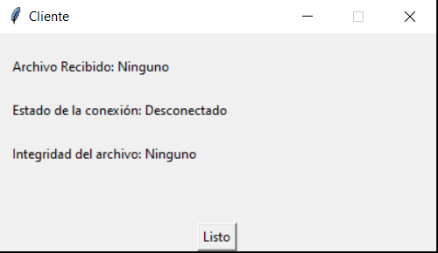
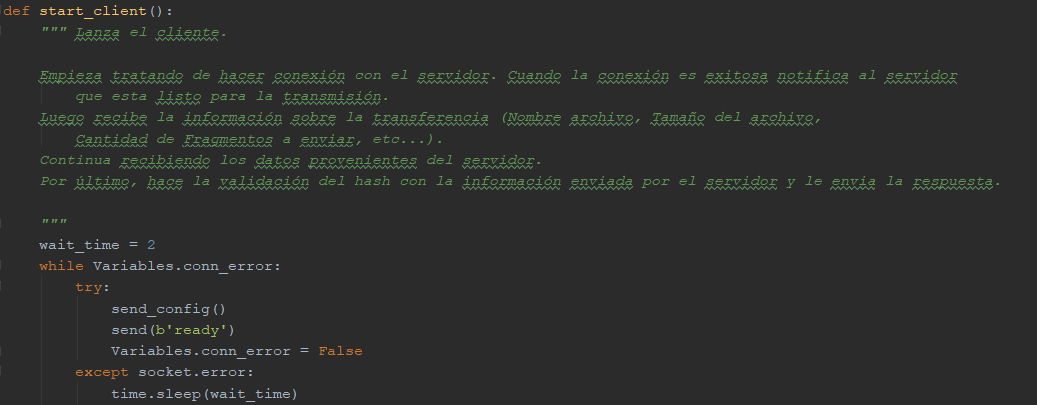
**INFORME LABORATORIO 3 – SERVIDOR UDP**

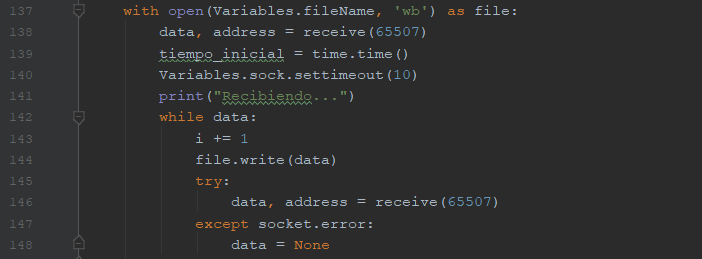
**Integrantes: Mateo Salcedo, Sergio Naranjo Puentes, Iván García Laverde.**

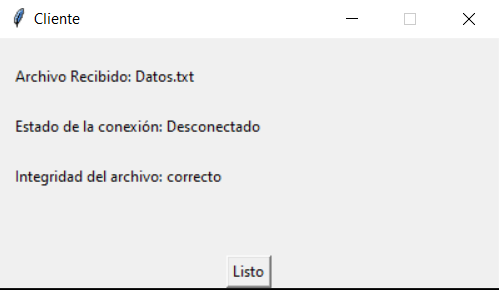
1. Creación del cliente:

* ***Conectarse al servidor TCP y mostrar que se ha realizado dicha conexión. Mostrar el estado de la conexión.***

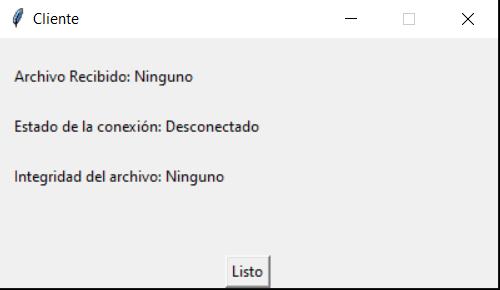
Estado de la conexión tiene como función mostrar el estado en que se encuentra el cliente con respecto al servidor. Este campo tiene 3 estados posibles: ’Desconectado’, ’Conectando...’ y ’Conectado y Recibiendo’. El estado ’Conectado y Recibiendo’ hace referencia a que el cliente está Conectado al servidor y se encuentra recibiendo el archivo que está transmitiendo este

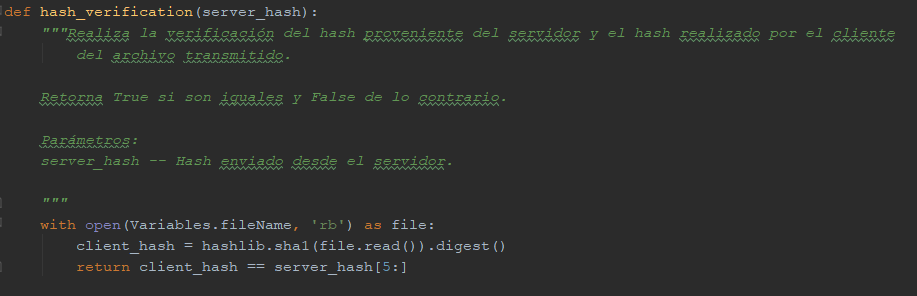
* ***Enviar notificación de preparado para recibir datos de parte del servidor.***La anterior imagen muestra la forma en que el cliente trata de conectarse con el servidor. Cuando este logra hacer la conexión envía un mensaje de notificación (’ready’) que indica al servidor que ese cliente está listo.
* ***Recibir un archivo del servidor por medio de una comunicación a través de sockets UDP.***

La imagen muestra la forma en que el cliente recibe los paquetes enviados por el servidor (Línea 144). Este lee del buffer una cantidad máxima de datos de 65507, que es la cantidad máxima de datos que se pueden leer debido a los 20 bytes usados para la transferencia por la capa de red y 8 usados en el header del dato por el protocolo UDP.

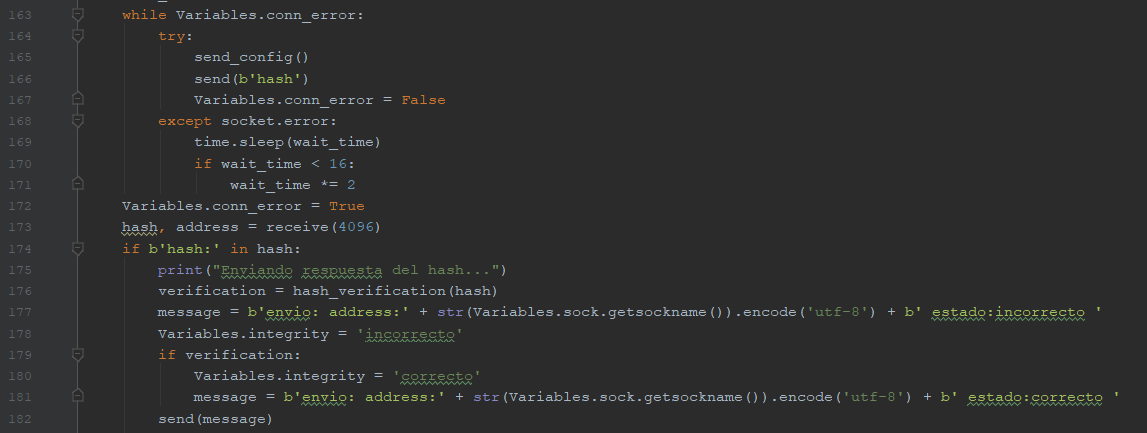
Cuando el archivo es recibido en el campo Archivo Recibido muestra el nombre del archivo en cuestión. Además, se muestra la integridad del archivo en el campo Integridad del archivo.

* ***Verificar la integridad del archivo con respeto a la información entregada por el servidor.***

Integridad del archivo tiene 3 estados posibles: ’Ninguno’, ’Correcto’ y ’Incorrecto’. Este campo tiene como propósito mostrar el resultado de la verificación de la función hash siempre y cuando se haya recibido un archivo, en su defecto el estado es ’Ninguno’.

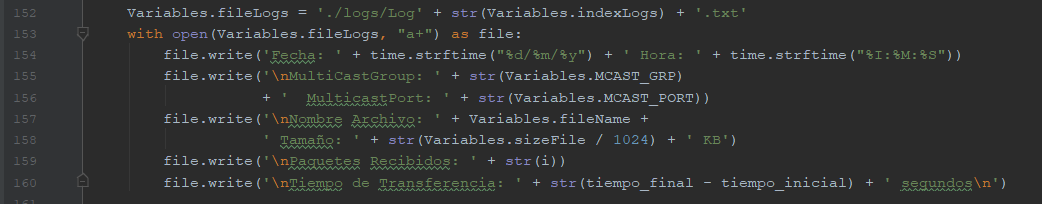
Función Utilizada para la verificación del hash.

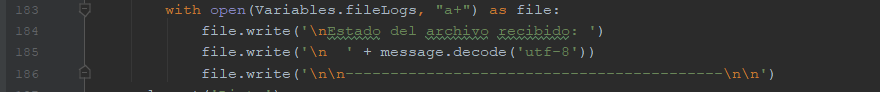
* ***Enviar notificación de recepción del archivo al servidor y si este está correcto o no (resultado de la comparación del hash).***



La anterior imagen muestra como el cliente, luego de recibir el archivo, trata de conectarse con el servidor. Cuando este logra conectarse le notifica el servidor que le envié el hash (Línea 166), luego el cliente valida con la información entregada por el servidor y le retorna la respuesta de esa validación (Línea 102).

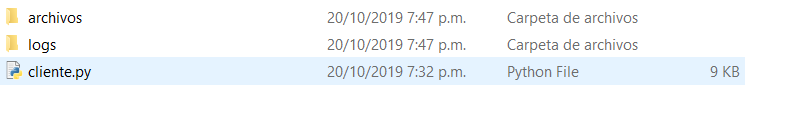
* ***La aplicación debe permitir medir el tiempo de transferencia de un archivo en segundos. Este tiempo debe calcularse desde el momento en que se envía el primer paquete con datos del archivo en el servidor hasta el momento en el que se recibe el último paquete del archivo en el cliente. Al final de cada transferencia la aplicación debe reportar si el archivo está completo y correcto y el tiempo total de transferencia. Para esto, genere un log para cada intercambio de datos entre cliente y servidor.***





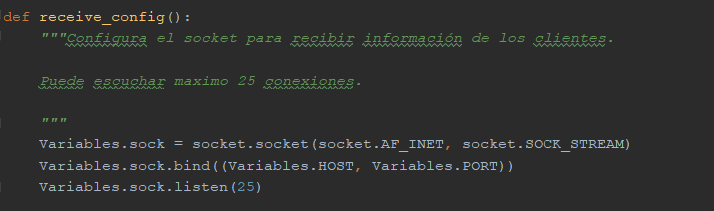
Las anteriores imágenes muestran cómo se generan los logs de cada transferencia para un cliente. El tiempo de transferencia se calcula realizando una resta entre una variable llamada ‘tiempo\_final’ y la variable ‘tiempo\_inicial’. La variable ‘tiempo\_inicial’ se calcula al momento en que el servidor envía el primer paquete y la variable ‘tiempo\_final’ se calcula al momento en el que llega el último paquete.

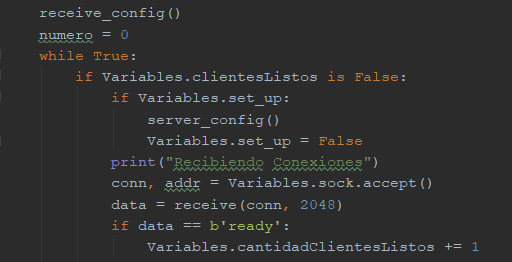
* ***Disponer un repositorio de los archivos recibidos y logs.***



1. Creación del servidor:

* ***Recibir conexiones UDP. La aplicación debe soportar 25 conexiones en simultáneo.***

La imagen muestra cómo se configura el socket que va a aceptar las peticiones de los clientes. Se puede notar que sólo se pueden recibir máximo 25 clientes.

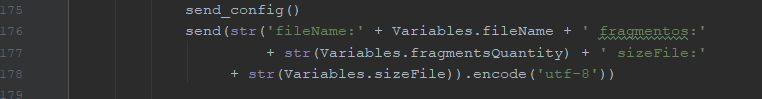
La imagen muestra cómo se hace la recepción de cada cliente con su respectivo mensaje de ’ready’ el cúal informa al servidor que el cliente está listo.

* ***Definir el tamaño de los mensajes en que se van a fragmentar los archivos. Nota: Recuerde que los fragmentos UDP pueden ser hasta de 64 KiB.***



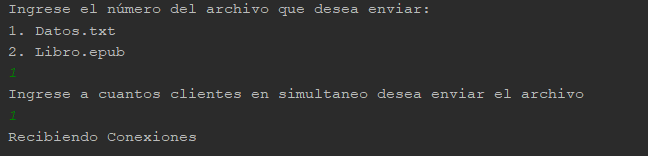
La anterior imagen muestra cómo se calcula la cantidad de fragmentos que se van a enviar para transmitir el archivo (Línea 107). Esta toma el tamaño del archivo y lo divide entre 64000 y hace función techo para poder enviar la totalidad del archivo.

* ***Indicar a cada cliente el número de fragmentos a enviar. Además, se debe entregar el valor hash calculado para cada archivo. Este dato se usará para que los clientes validen la integridad del mismo.***



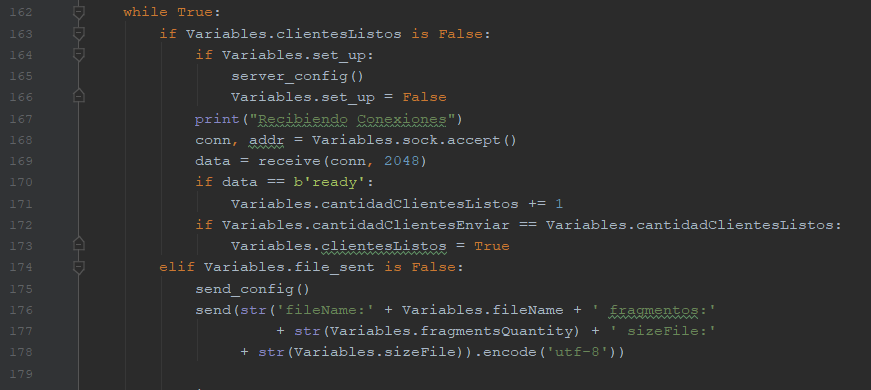
La anterior imagen muestra la información que es enviada a todos los clientes pertenecientes al grupo multicast. Se envía el nombre del archivo, la cantidad de fragmentos y el tamaño del ar chivo.

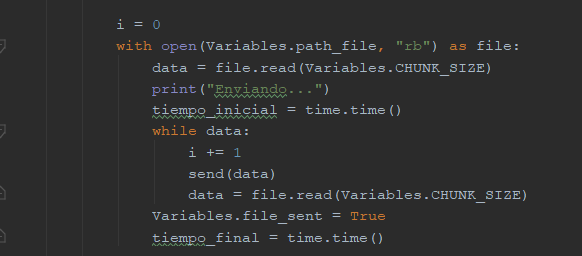
* ***La aplicación debe permitir seleccionar qué archivo desea enviarse a los clientes conectados y a cuántos clientes en simultáneo. A todos se les envía el mismo archivo durante una transmisión.***



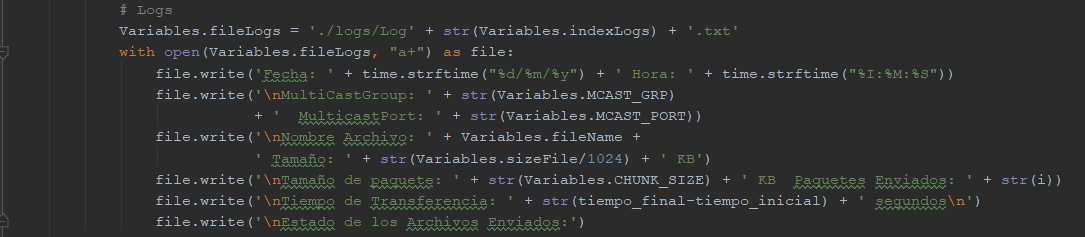
La imagen muestra un ejemplo de cómo se debe configurar el archivo que se desea enviar y la cantidad de clientes en simultaneo que debe haber para poder empezar la transmisión del archivo. Esta transmisión se hace a un grupo multicast y a un puerto en específico, los cuales fueron configurados anteriormente.

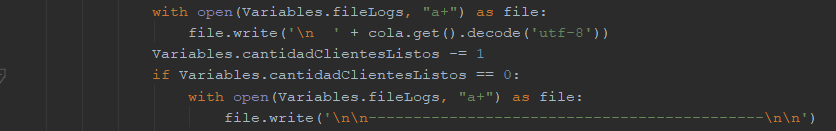
* ***Realizar la transferencia de archivos a los clientes definidos en la prueba. El envío debe realizarse solo cuando el número de clientes indicados estén conectados y su estado sea listo para recibir.***

Cuando la cantidad de clientes listos es la configurada al iniciar el server (Línea 172) comienza el envío del archivo (Imagen Siguiente).



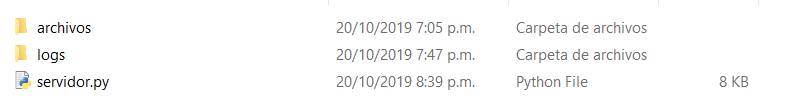
* ***La aplicación debe permitir medir el tiempo de transferencia de un archivo en segundos. Este tiempo debe calcularse desde el momento en que se envía el primer paquete con datos del archivo en el servidor hasta el momento en el que se recibe el último paquete del archivo en el cliente. Al final de cada transferencia la aplicación debe reportar si el archivo está completo y correcto y el tiempo total de transferencia. Para esto, genere un log para cada intercambio de datos entre cliente y servidor.***





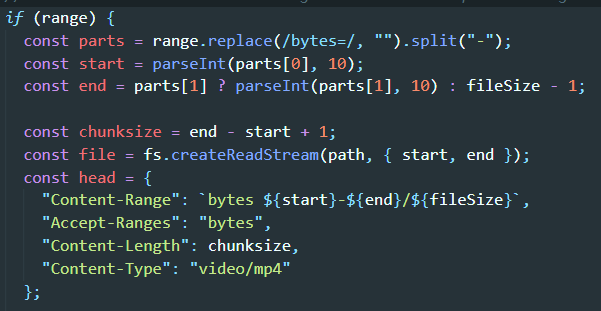
La anteriores imágenes muestran la forma que en que se generan los logs para cada transferencia hecha. El tiempo de transferencia se mide desde que se envía el primer paquete hasta que llega el último paquete al cliente.

* ***Disponer un repositorio de los archivos recibidos y logs para cada una de las pruebas.***



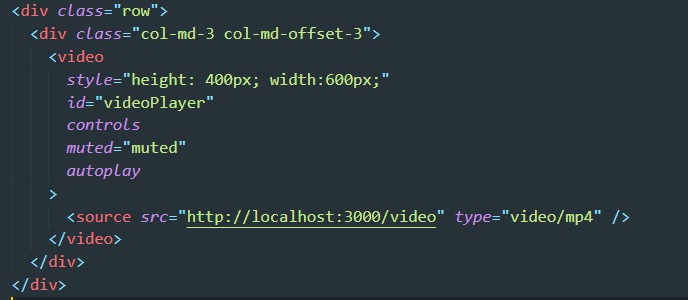
1. Streaming via UDP:

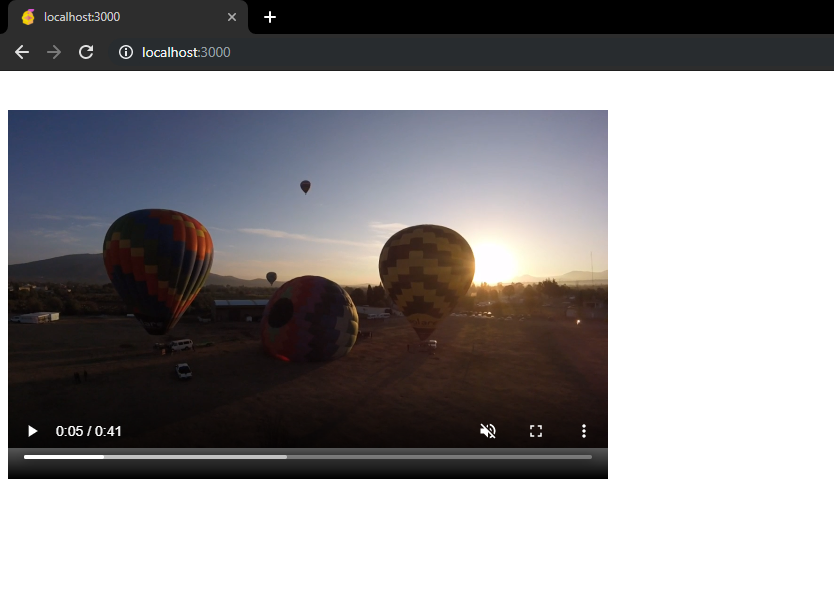
Se realizó en javascript un servidor que recibe los segmentos de datos enviados por el protocolo udp desde el servidor udp. Acá se toman los segmentos para enviarlos a la vista





La vista es muy sencilla, es una página html básica con un reproductor multimedia el cual reproduce el video que es enviado por el api endpoint del servidor que está en el puerto 3000.





## Análisis de resultados.

### 4.1 Pruebas de desempeño

Para efectuar las pruebas de desempeño se usó la siguiente metodología:

1. Se desplego el servidor UDP Python en una máquina virtual Ubuntu server en AWS, de la misma manera que se hizo con el servidor TCP.
2. Se desplego el servidor en la máquina y se desplegaron n clientes de manera local para probar el servicio de streaming.
3. Una vez solicitada la transmisión, el servidor envía iterativamente el archivo seleccionado para los clientes y al final retorna el numero de paquetes enviados y el hash del archivo.
4. Los clientes consumen los datos de la trasferencia y al final devuelven el numero de paquetes y el hash del archivo recibido.
5. Ponderando el número de paquetes recibidos sobre enviados se calcula el porcentaje de error de transferencia del servidor UDP y se compara respecto a lo esperado de un servidor TCP. Además, se tiene en cuenta la comparación del hash para determinar si el archivo tuvo algún daño de integridad.

Escenarios de prueba: en todos los escenarios de prueba se usó el archivo ***vid\_prub.mp4* (61693 Kb)**

1. Número de clientes: 1, 3 y 5.

Tamaño/Paquete: 1024 Bits (1Kb).

1. Número de clientes: 1,3 y 5.

Tamaño/Paquete: 4096 Bits (4Kb).

1. Número de clientes: 1, 3 y 5.

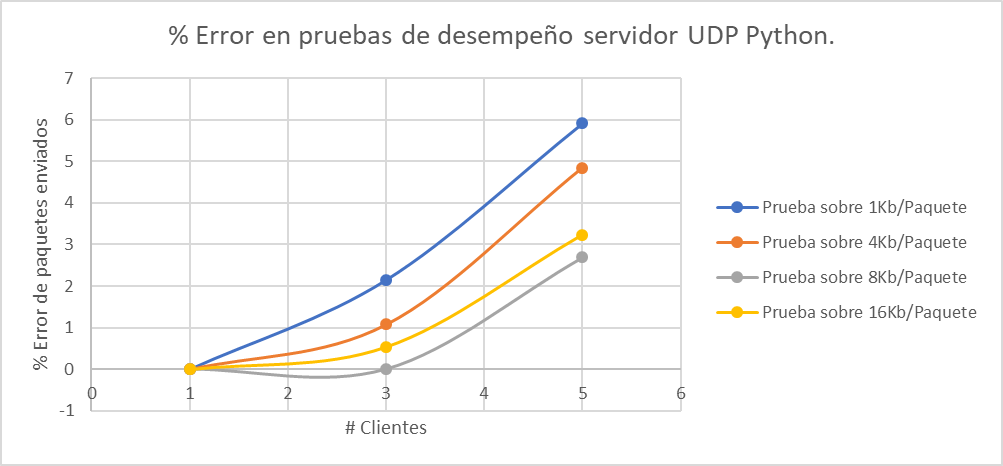
Tamaño/Paquete: 8096 Bits (8Kb).

1. Número de clientes: 1, 3 y 5.

Tamaño/Paquete: 16384 Bits (16Kb).

**Gráfica:**

*Para mayor detalle acerca de la gráfica puede revisar la tabla que se encuentra en el proyecto también.*

****

**Análisis de resultados:**

Se puede ver que a mayor tamaño del paquete en los envíos se logra disminuir algunos decimales en el porcentaje de error, desde un rango del 6% hasta un rango de 2% cercano a lo “estándar” (en condiciones más saturadas de un envío TCP) variando de manera adecuada el buffer de envío.

Es importante tener en cuenta el resultado del último test, dado que rompe con la tendencia de “más es mejor” lo que sugiere que entre más grande sea el ***Throughput*** no necesariamente la tasa de perdida será menor.

### 4.2 Comparación: teoría vs práctica.

La práctica del laboratorio confirma la teoría vista y expresada tanto en la clase como en los libros referenciados para la materia, dado que en general se concibe una tasa de perdida mucho mayor que la tasa del protocolo TCP. No obstante, se descubrió un efecto que puede ayudar a disminuir dicha tasa de error el cual es mediar el Throughput del servidor con respecto a la saturación de peticiones.

Esto sugiere que para efectos de la escalabilidad de servicios, UDP tiene características que podrían favorecer o desfavorecer el servicio de streaming a usuarios dependiendo de como se maneje el problema. En adición cabe resaltar que los resultados responden al sentido lógico del protocolo, dado que entre menos paquetes envíe, menos probabilidad habrá que se pierda 1 entre el total de paquetes; sin embargo, si envío paquetes muy grandes, se van a demorar mucho en llegar y por convención propia del cliente se puede establecer un ***timeout*** que establezca el paquete como perdido.

### 4.3 ¿Cómo implementaría un servicio streaming usando TCP?

Existen múltiples opciones cuando hablamos de protocolos de la capa de aplicación que ayudan a manejar el servicio de streaming sobre TCP. Algunas de estas opciones pueden ser:

* **RTSPT: (Real Time Streaming Protocol TCP)** es la implementación de RTSP usando TCP para el envío de datos de control acerca del flujo, transferencia de los datos circundantes en la conexión y archivos de audio y video. Inicialmente RTSP usa UDP para la transferencia de datos multimedia y TCP para los datos de control comunicación, no obstante, conscientes de la falta de confiabilidad de UDP, se implementó una versión con TCP.
* **HLS: (HTTP Live Streaming)** es un protocolo creado por Apple, que esta diseñado para reconocer el ancho de banda y la capacidad de computo del usuario y enviar contenido multimedia ajustado a dichos parámetros. Este protocolo es uno de los más populares actualmente dada su versatilidad de ser ajustable a cualquier dispositivo. Una de las ventajas de encapsular el flujo de su información en segmentos HTTP es que puede transportarse incluso a través de firewalls o proxys, a diferencia del streaming sobre UDP que no tiene esta característica.

* **MPEG – DASH o simplemente DASH: (Dynamic Adaptive Streaming over HTTP)** Es un protocolo muy parecido a HSL, dado que también usa técnicas de *adaptive bitrate streaming* que ayudan a reconocer los parámetros del dispositivo host de destino. Sus mayores diferencias con HSL radican en el formato y la configuración de su codificación para la transferencia de archivos.

***Para un análisis más preciso vea este*** [***link***](https://www.internetvideoarchive.com/documentation/video-api/progressive-download-vs-adaptive-bitrate/)

### 4.4 FTP sobre UDP ¿Confiable?

Existe el protocolo **FSP (File Service Protocol)** el cual cumple con las mismas funciones que FTP. No obstante, a pesar de que su implementación lo hace más lento en condiciones normales de la red, no necesita ningún tipo de conexión adicional (precisamente porque trabaja sobre UDP) y sigue cumpliendo con la función de listas en directorios que FTP soporta.

# Fuente: [FSP Official Documentation](http://fsp.sourceforge.net/faq/index.html#COMPAREFTP) Dicho esto cabe resaltar que, de acuerdo con [Building Internet Firewalls](http://web.deu.edu.tr/doc/oreily/networking/firewall/ch08_02.htm) no es recomendable ni confiable habilitar el paso de cualquier transferencia de datos sobre UDP a través del firewall, dado que sujetos malintencionados usan dichos métodos para atentar contra la integridad del software en páginas y servidores.