**TRABAJO PRÀCTICO Nº2**

Administraciòn y Sistemas de Redes

**Integrantes:**

* Luciano Yedro
* Lautaro Aquino

**Protocolo a Investigar e implementar:**

* BitTorrent

**¿Qué es BitTorrent?**

Con la expansión de Internet, el acceso a la información y a los contenidos se ha hecho masivo, permitiendo que un archivo pueda viajar alrededor del mundo en cuestión de minutos y ser accedido por millones de personas. Este avance conlleva desafíos tecnológicos significativos, especialmente en la compartición de archivos multimedia. El paradigma clásico de cliente/servidor se ve limitado debido a la sobrecarga del servidor y la consecuente degradación del servicio a medida que aumenta la popularidad del contenido. Las redes peer-to-peer (p2p) aprovechan el ancho de banda de los usuarios que ya poseen el contenido, ayudando en su difusión y mejorando la calidad del servicio al incrementarse las fuentes disponibles. BitTorrent, un protocolo de compartición de archivos p2p, aborda los desafíos de las redes p2p puras mediante un sistema híbrido que utiliza redes independientes para cada archivo y un servidor central (tracker) que gestiona el estado de la red sin necesidad de enviar el contenido.

**Introducción:**

BitTorrent es un protocolo de compartición de archivos p2p. Está diseñado especialmente para archivos de gran tamaño y utiliza un servidor central (tracker) que maneja el estado de la red sin enviar el contenido, lo que se deja a los usuarios. Cuando un usuario quiere compartir un archivo, crea un archivo .torrent y lo publica en un tracker. Un cliente descarga este archivo para acceder a la red y comenzar la descarga. Cada archivo se divide en pequeñas porciones que se comparten bajo la premisa de "la porción menos disponible primero". Los usuarios que poseen alguna parte del archivo pueden ayudar en su difusión, aumentando la redundancia de datos y aliviando la congestión del usuario original.

**Problemas y enfoques de solución a sistemas de distribución:**

Los sistemas basados en el paradigma cliente/servidor presentan dos desventajas fundamentales:

1. La popularidad de un archivo aumenta el tráfico en el servidor, obligando a aumentar su capacidad o limitar el tráfico, lo que deteriora el servicio.

2. La disponibilidad del contenido depende del servidor, y si este falla, el contenido deja de estar disponible.

Las redes p2p mejoran el primer problema al aprovechar el ancho de banda de los usuarios para ofrecer el contenido, permitiendo que múltiples usuarios actúen como servidores y aumentando la robustez del servicio. Sin embargo, las redes p2p descentralizadas presentan desafíos como el desperdicio de ancho de banda en la búsqueda de contenido y la limitación en la escalabilidad de la red.

BitTorrent mejora estos problemas con un sistema híbrido basado en redes dedicadas por cada archivo. Un servidor central contiene información de los usuarios conectados, evitando la congestión del servidor y de los clientes, ya que se tienen múltiples fuentes para obtener el contenido.

**¿Cómo funciona?(paso a paso)**

La red BitTorrent opera sobre un principio fundamental: descentralización. Permite a los clientes obtener un archivo desde múltiples fuentes simultáneamente, optimizando así la velocidad de descarga y evitando la sobrecarga en un único servidor. Aquí hay un desglose más detallado de cómo funciona:

**1 Instalación del Cliente BitTorrent:**

El primer paso para utilizar la red BitTorrent es instalar un cliente BitTorrent en el dispositivo desde el cual se realizará la descarga. Hay una variedad de clientes disponibles, desde software de código abierto como qBittorrent hasta aplicaciones comerciales como µTorrent.

**2 Acceso al Archivo .torrent:**

Una vez que el cliente BitTorrent está instalado, el usuario necesita acceder a una página web que contenga el enlace al archivo .torrent del contenido deseado. Este archivo .torrent actúa como un mapa o guía para el cliente BitTorrent, indicándole cómo y dónde encontrar las partes del archivo deseado en la red.

**3 Verificación de Disponibilidad de Usuarios:**

Antes de iniciar la descarga, el usuario verifica que haya otros usuarios compartiendo el archivo deseado. Esto se hace a través del cliente BitTorrent, que se comunica con el servidor tracker asociado al archivo .torrent. El tracker es esencialmente un punto de encuentro para los clientes BitTorrent que desean compartir y descargar el mismo archivo. El tracker mantiene una lista de los clientes que tienen partes del archivo y coordina la conexión entre ellos.

**4 Asignación de Destinatarios y Descarga Descentralizada:**

Una vez que se confirma que hay usuarios compartiendo el archivo, el cliente BitTorrent obtiene una lista de posibles destinatarios (también conocidos como "peers") del tracker. Luego, el cliente establece conexiones con estos peers y comienza a descargar las partes del archivo de ellos de manera simultánea. A medida que se completan las descargas de partes del archivo, el cliente también comienza a compartir esas partes con otros peers que están solicitando el mismo archivo.

**5 Intercambio Continuo de Datos:**

El intercambio de datos entre los peers es bidireccional y constante. Cada cliente BitTorrent que participa en la descarga también actúa como un servidor, compartiendo las partes del archivo que ya ha descargado con otros clientes que las necesitan. Este proceso de compartir y descargar partes del archivo entre múltiples peers es lo que permite una descarga rápida y eficiente en la red BitTorrent.

**Posibles aplicaciones de BitTorrent**

**Multitracker Metadata Extensión (MME):**

Esta mejora en el protocolo BitTorrent añade una funcionalidad crucial al permitir la inclusión de una lista de múltiples trackers dentro del archivo .torrent. Esta característica proporciona una redundancia en la red, lo que significa que si un tracker falla, los clientes aún pueden encontrar y conectarse con otros trackers en la lista para mantener la continuidad de la descarga.

**DHT (Distributed Hash Table):**

DHT es un sistema innovador que propone eliminar la dependencia de un servidor tracker centralizado en la red BitTorrent. En lugar de ello, utiliza tablas hash distribuidas y un algoritmo de enrutamiento basado en Kademlia para permitir que los clientes se conecten directamente entre sí sin la necesidad de un tracker. Esto hace que la red sea más resistente a la censura y los fallos de los servidores centrales.

**LiveBT:**

LiveBT es una propuesta desarrollada por investigadores chinos que busca extender la funcionalidad de BitTorrent para soportar transmisiones en vivo y videos bajo demanda. Esta extensión permitiría a los usuarios disfrutar de contenido multimedia en tiempo real, como eventos deportivos o conciertos, utilizando la robusta infraestructura de la red BitTorrent para distribuir el contenido de manera eficiente y escalable.

**Similarity Enhanced Transfer (SET):**

SET es una técnica diseñada para mejorar las redes peer-to-peer (p2p) donde solo unos pocos usuarios están compartiendo un archivo específico. Utilizando versiones ligeramente modificadas del archivo original, SET permite a los clientes reconstruir el contenido deseado incluso si solo tienen acceso a una fracción del mismo. Esta técnica aumenta la eficiencia de la transferencia de archivos en redes con pocos usuarios compartiendo contenido similar.

**A día de hoy:**

BitTorrent es considerado escalable y robusto, ya que el aumento de la demanda de descarga de archivos se traduce en un aumento de nodos de descarga, y fallos en la red solo pausan la descarga. Es uno de los protocolos p2p más populares debido a su tolerancia y aprovechamiento del tráfico. Una vez obtenida la extensión de descarga, se asegura un rápido inicio de descarga sin cola de espera.

**Especificación del protocolo :Archivo .torrent**

El archivo .torrent está codificado mediante b-coding y contiene:

- info: Describe los archivos contenidos, incluyendo la longitud de cada pieza y las sumas de comprobación SHA1.

- announce: URL del tracker.

- Optional: Información adicional como la fecha de creación, comentarios del autor, y el formato de codificación.

Cada cliente utiliza una conexión TCP con un handshake inicial seguido por mensajes de control que pueden incluir:

- Keep-alive: Mantiene la conexión abierta.

- choke/unchoke: Indican si el cliente ha sido bloqueado o desbloqueado.

- interested/not interested: Indican interés en partes del archivo.

- have: Indica las partes del archivo que se poseen.

- bitfield: Indica los trozos del archivo que se poseen actualmente.

- request/piece/cancel: Para pedir, enviar o cancelar la solicitud de partes del archivo.

- port: Indica el puerto del cliente para DHT.

**Explicación de nuestra Implementación:**

## **Torrentool: Una biblioteca de Python para trabajar con archivos torrent**

Utilizamos la librería Torrentool que proporciona herramientas para trabajar con archivos torrent. Te permite crear, leer, modificar e interactuar con archivos torrent utilizados para compartir archivos peer-to-peer (P2P).

**Componentes:**

* Interfaz de línea de comandos (CLI): Te permite interactuar con Torrentool a través de comandos de terminal. Requiere de que esté instalado el paquete “click”. Este se puede instalar junto con las funcionalidades de la CLI usando pip install torrentool[cli].
* Utilidades de torrent: Estas utilidades proporcionan funcionalidades para crear, leer y modificar archivos torrent.
* Utilidades de Bencoding: Los archivos torrent utilizan un formato de codificación específico llamado Bencoding. Torrentool incluye herramientas para codificar y decodificar archivos torrent.

**Uso de la CLI:**

* **Creación de un archivo torrent:**
  + torrentool torrent create /home/my/files\_here/video.mkv**:**

**Crea un archivo torrent a partir del archivo de video especificado.**

* + torrentool torrent create /home/my/files\_here**:**

**Crea un archivo torrent a partir de todo el directorio, agrega rastreadores abiertos para una distribución más amplia y lo prepara para su publicación en un servicio de almacenamiento en caché de torrents.**

**Aplicación en nuestro código:**

1. Direccionamos correctamente la ruta del archivo torrent:

archivo\_torrent = 'C:\\Users\\aquinooyedro\\Primer.torrent'

1. Comprobamos que el archivo torrent existe

if os.path.exists(archivo\_torrent):

1. Si existe, se procede a abrirlo y modificarlo con la librería torrentool

my\_torrent = Torrent.from\_file(archivo\_torrent)

1. En este caso, queremos ver el tamaño de un archivo, por lo que lo hacemos con la siguiente línea de código:

print("Tamaño total del archivo en bytes:", my\_torrent.total\_size)

1. Ahora deseamos ver el link magnético de torrent de este caso, usamos la siguiente líne a de código:

print("El enlace magnético del archivo es:", my\_torrent.magnet\_link)

1. Procedemos a modificar el comentario de torrent:

my\_torrent.comment = 'No me gustan los torrents'

1. Y luego de haber hecho todos estos cambios, los guardamos:

my\_torrent.to\_file()

1. No podemos olvidarnos de comprobar de qué es lo que ocurriría si no existiese el archivo torrent, por lo que en lo que queda del condicional agregamos los siguientes comandos:

else:

print("El archivo torrent no existe en la ruta especificada:", archivo\_torrent)

1. Creamos el archivo torrent que no existe con su ubicación correspondiente:

new\_torrent = Torrent.create\_from('C:\\Ubicación')

1. Luego, establecemos el tracker del torrent:

new\_torrent.announce\_urls = 'udp://tracker.openbittorrent.com:80'

1. Y finalizando, guardamos nuestro nuevo archivo tipo torrent:

new\_torrent.to\_file('another.torrent')

1. Como dato adicional, podemos intentar ver cuanto pesa el archivo torrent:

print("Tamaño total del nuevo archivo en bytes:", new\_torrent.total\_size)