Borrador de Descripción del Proyecto:
Sistema BitTorrent

El trabajo consiste en realizar una aplicación de **BitTorrent**[4], el cual consiste en un sistema de intercambio de archivos, de forma dinámica.

El usuarios que desee descargar un archivo (o conjunto de ellos) deberá obtener el archivo *.torrent* relacionado con los mismos, que da información sobre cuál servidor tiene información sobre la ubicación de usuarios con el archivo completamente descargado (*seeders*) y otros usuarios que están aun en proceso de descarga (*leechers*). De los primeros, puede obtener todas las partes que conforman al archivo, y de los segundos se pueden obtener algunas de ellas, pudiendo intercambiar partes de los archivos con los mismos.
Una vez obtenida dicha información, el cliente pasa a realizar las conexiones directamente con dichos usuarios, siendo (aunque no completamente) una aplicación esencialmente P2P (*peer-to-peer*).
Es importante notar que no sería un esquema eficiente el realizar toda la descarga desde un único *peer*, ya que, aunque sería la solución más sencilla, quedarían expuestas a varias falencias (de hardware, como el ancho de banda, o de software, debido a caidas de dicho peer). Por ende, es imprescindible realizar una coordinación entre los diversos pares, así como permitir un sistema de premios y castigos similares a los analizados en **Teoría de Juegos**, los cuales son pilares importantes en el estudio de este tipo de arquitecturas, dado que se fomenta que las descargas sean realizadas desde *leechers* y no únicamente desde *seeders* (puesto que estos desean descargar otros archivos, y esto puede disminuir el ancho de banda disponible). Dicho esquema se denomina *Tit-for-tat*[1], que busca obtener la **Eficiencia de Pareto**[2].

Una vez que un par (o *peer*) ha terminado de descargar un archivo, éste se lo informa al servidor que guarda la información de dicho *.torrent*, lo cual permite ser identificado como un *seeder* del cual puede descargarse dicho archivo.

En caso que se esté publicando un nuevo archivo, se genera el nuevo archivo *.torrent* con la dirección del servidor que vaya a guardar la información de dicho archivo (que muchas veces puede ser un servidor local creado, que contenga información de otros servidores de torrents), informándose en simultáneo como un *seeder* del mismo (o *seeder original*).
Una vez que se termina la tarea y se cierra la aplicación, se informa al servidor de cada archivo .torrent que no se descarga ni suben más archivos, para que deje de tenerlo en cuenta como *seeder* o *leecher*, según el caso. Si la salida es realizada de forma inesperada (i.e. caída del ISP, caída de la aplicación, apagado de la máquina, etc...), las demás aplicaciones podrán ir informando de la falta de comunicación con dicho *peer*, con lo cual el servidor puede decidir eliminarlo de la lista (notar que si un usuario decide no subir datos, los demás usuarios decidirán compartirle pocos datos debido al sistema de premios y castigos mencionado anteriormente).

El objetivo del trabajo es la implementación de dicho sistema, inicialmente sencillo y escalándolo utilizando protocolos estudiados por las diversas organizaciones[3] que hoy en día permite al **BitTorrent** ser uno de los sistemas de intercambio de archivos más usados en el mundo, analizando cómo influye cada protocolo a mejorar (o afectar)
los diversos aspectos de la arquitectura del sistema.

Referencias

[1] Tit-for-Tat (Toma y daca): Si una parte no cumple con su parte en favor del grupo, sino que actúa en su propio beneficio, debe ser castigado.
http://es.wikipedia.org/wiki/Toma_y_daca

[2] Pareto Efficiency: Dentro de un sistema, para que un punto mejore algún otro debe empeorar.
http://en.wikipedia.org/wiki/Pareto_efficiency
Combinando ambos términos, tenemos que, se premia aquellos que comparten archivos y se castiga a los que reducen la subida de archivos.

[3] Organización de estudio de protocolos para BitTorrent: http://bittorrent.org/.

[4] *Incentives Build Robustness in BitTorrent*, Bram Cohen, 2003. Fuente: BitTorrent.org, http://www.bittorrent.org/bittorrentecon.pdf.
En dicho paper se analiza las bases técnicas de un sistema de BitTorrent, resumidas en esta breve descripción.

[5] *Do incentives build robustness in BitTorrent?* Fuente: Stanford University.
http://sing.stanford.edu/cs303-sp11/papers/BitTyrant.pdf.
En este paper se analiza el problema principal de las aplicaciones P2P: que cada usuario viva *libremente*, sin contribuir al sistema, y como evitar que ésto suceda.

[6] *THE BITTORRENT P2P FILE-SHARING SYSTEM: MEASUREMENTS AND ANALYSIS*.
Fuente: Department of Computer Science, Delft University of Technology, the Netherlands.
http://www.ewi.tudelft.nl/over-de-faculteit/computer-science-engineering/
Sistemas que rondan BitTorrent (motores de búsqueda) y mediciones realizadas con diversos clientes, y con algunas variantes de protocolos (como la implementación de aseguramente de integridad de los datos).

[7] Distributed Systems — Peer-to-Peer, Allan Clark, School of Informatics University of Edinburgh. Fuente: http://www.inf.ed.ac.uk/teaching/courses/ds/handouts-2012-2013/part-6.pdf
Análisis de distintos aspectos y conceptos de aplicaciones P2P.

Referencias de información de Middlewares para aplicaciones P2P y BitTorrent

[8]Middleware and Distributed Systems, Peer-to-Peer Systems. Martin v. Löwis.
Fuente: Operating Systems and Middleware Group at HPI. https://www.dcl.hpi.uni-potsdam.de/teaching/mds/mds10_p2p.pdf.

[9] DISTRIBUTED HETEROGENEOUS
APPLICATIONS AND CORBA. Fuente: Linköping University, Department of Computer and Information Science. https://www.ida.liu.se/~TDDD25/lecture-notes/lect4.frm.pdf.
Middleware Corba, y explicaciones para aplicaciones P2P.
[10] Middleware for P2P architecture. Fuente: Donald Bren, School of Information & Computer Science. http://www.ics.uci.edu/~cs237/Fpresentationslides2014/middlewarep2p.pptx

Librerías encontradas:

[11] LibTorrent (librería en C++, con wrapper para Python): http://libtorrent.org.

[12] BitTornado (librería exclusivamente para Python): http://www.bittornado.com/