FI-Torrent

Introducción

El presente trabajo tiene como objetivo el diseño y la implementación de un *cliente torrent*.
Un **cliente torrent** es una aplicación que permite a un usuario establecer una conexión tipo P2P para descargar archivos que otros usuarios (de la misma red de archivos) poseen y que están dispuestos a compartir para facilitar el intercambio de los mismos.

Concepto de redes Peer-to-Peer (P2P)

(Wikipedia - http://es.wikipedia.org/wiki/P2P y http://en.wikipedia.org/wiki/Peer-to-peer)

A grandes rasgos, una red informática entre iguales (*peer-to-peer*, punto a punto, o más conocida como P2P) se refiere a una red que no tiene clientes ni servidores fijos, sino una serie de nodos que se comportan simultáneamente como clientes y como servidores respecto de los demás nodos de la red.

Este modelo de red contrasta con el modelo cliente-servidor, el cual se rige mediante una arquitectura monolítica donde no hay distribución de tareas entre nodos, sólo una simple comunicación entre un cliente y un servidor, en la que el cliente y el servidor no pueden cambiar de roles.

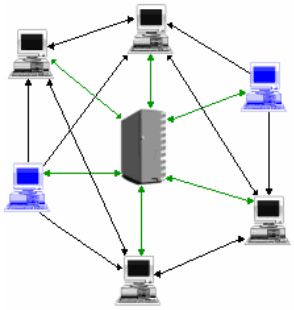
Las redes de ordenadores P2P son redes que aprovechan, administran y optimizan el uso de banda ancha que acumulan de los demás usuarios en una red por medio de la conectividad entre los mismos usuarios participantes de la red, obteniendo como resultado mucho más rendimiento en las conexiones y transferencias que con algunos métodos centralizados convencionales, donde una cantidad relativamente pequeña de servidores provee el total de banda ancha y recursos compartidos para un servicio o aplicación.

Las redes P2P son útiles para muchos propósitos, pero se usan muy a menudo para compartir toda clase de archivos que contienen: audio, video, texto, software y datos en cualquier formato digital. Este tipo de red es también comúnmente usado en telefonía VoIP para hacer más eficiente la transmisión de datos en tiempo real, así como lograr una mejor distribución del tráfico de la telefonía utilizando tecnología P2P.

Protocolo BitTorrent

(Wikipedia - http://es.wikipedia.org/wiki/BitTorrent_(protocolo) y http://en.wikipedia.org/wiki/BitTorrent_(protocol))

BitTorrent es un protocolo diseñado para transmitir archivos. De naturaleza P2P, los usuarios se conectan directamente entre sí para enviar y recibir porciones del archivo. Sin embargo, existe un servidor central (denominado *Tracker*) que coordina las acciones de los *peers*. El Tracker solo maneja conexiones y no posee ningún conocimiento acerca del contenido de los archivos que se distribuyen, permitiendo que un ancho de banda relativamente limitado permita soportar a un gran número de *peers*.



Para descargar un fichero primero se obtiene por medios convencionales un pequeño archivo con extensión .torrent. Este fichero es estático, por lo que a menudo se encuentra en páginas web o incluso se distribuye por correo electrónico. El archivo *torrent* contiene la dirección del *tracker*, el cual se encarga de localizar posibles fuentes con el fichero o parte de él.

Este servidor realmente se encuentra centralizado y provee estadísticas

acerca del número de transferencias, el número de nodos con una copia completa del fichero (*seeds*, indicados en color azul) y el número de nodos que poseen sólo una porción del mismo (*peers*, indicados en color negro).

El fichero o colección de ficheros deseado es descargado de las fuentes encontradas por el *tracker* y, al mismo tiempo que se realiza la descarga, se comienza a subir las partes disponibles del fichero a otros *peers*, utilizando el ancho de banda asignado a ello. Ya que la acción de compartir comienza incluso antes de completar la descarga de un fichero, cada nodo inevitablemente contribuye a la distribución de dicho fichero. El sistema se encarga de premiar a quienes compartan más, a mayor ancho de banda mayor el número de conexiones a nodos de descarga que se establecerán.

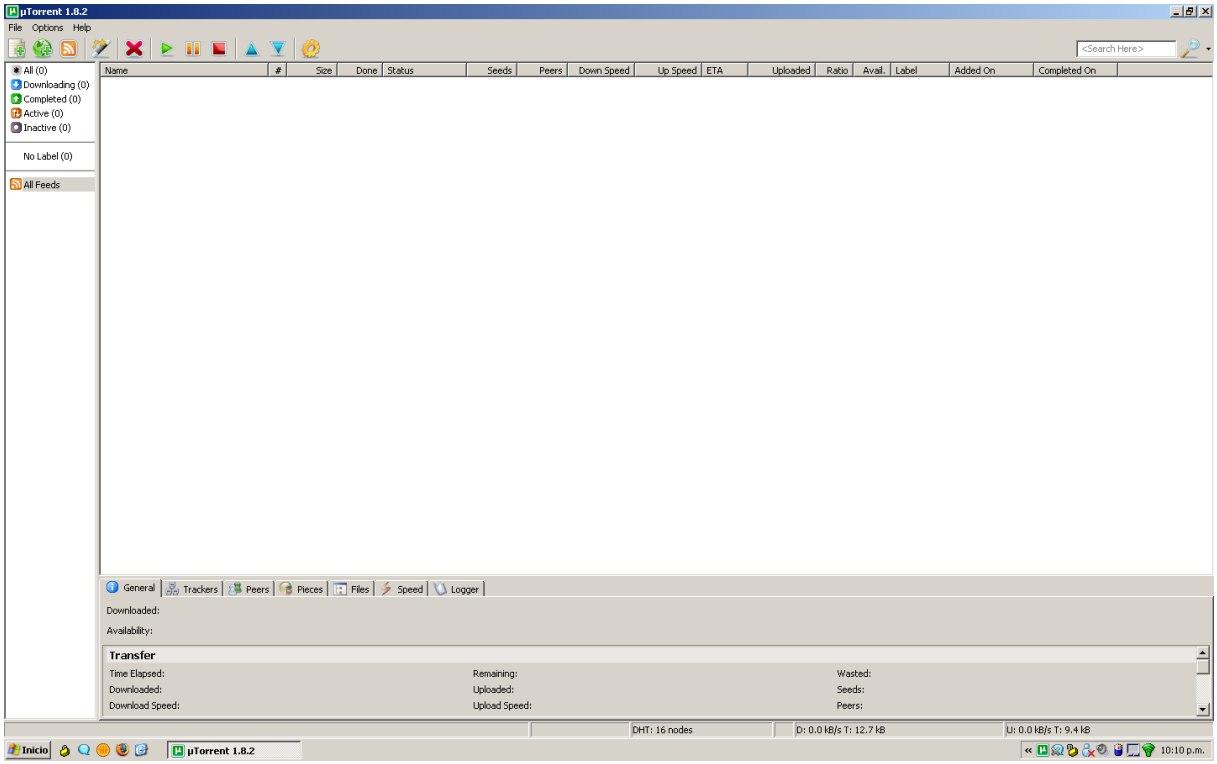
Cuando un usuario comienza la descarga de un fichero, BitTorrent no necesariamente comienza por el principio del mismo, sino que se baja por partes al azar. Si entre los usuarios conectados se dispone de cada parte del fichero completo (aún estando desparramado), finalmente todos obtendrán una copia completa de él. Por supuesto, inicialmente alguien debe poseer el fichero completo para comenzar el proceso. Este método produce importantes mejoras en la velocidad de transferencia cuando muchos usuarios se conectan para bajar un mismo fichero.

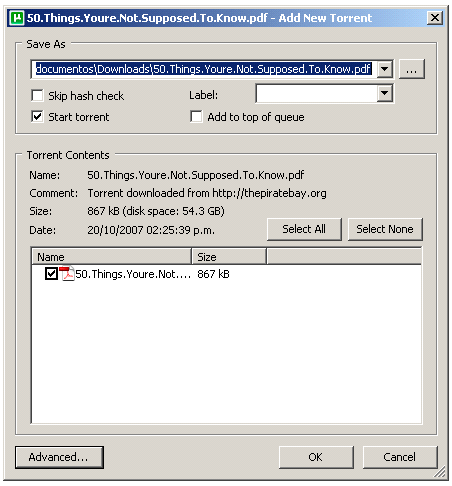
Clientes BitTorrent

(Wikipedia - http://es.wikipedia.org/wiki/Comparaci%C3%B3n_de_clientes_BitTorrent)

Existe una gran cantidad de clientes BitTorrent, casi todos ellos con opciones, interfaces y funcionamientos muy similares entre sí. Wikipedia mantiene una muy completa lista de clientes, junto con las características de cada uno de ellos.

La similitud de GUIs puede apreciarse en las siguientes imágenes, capturadas de dos clientes populares: BitTorrent y uTorrent:



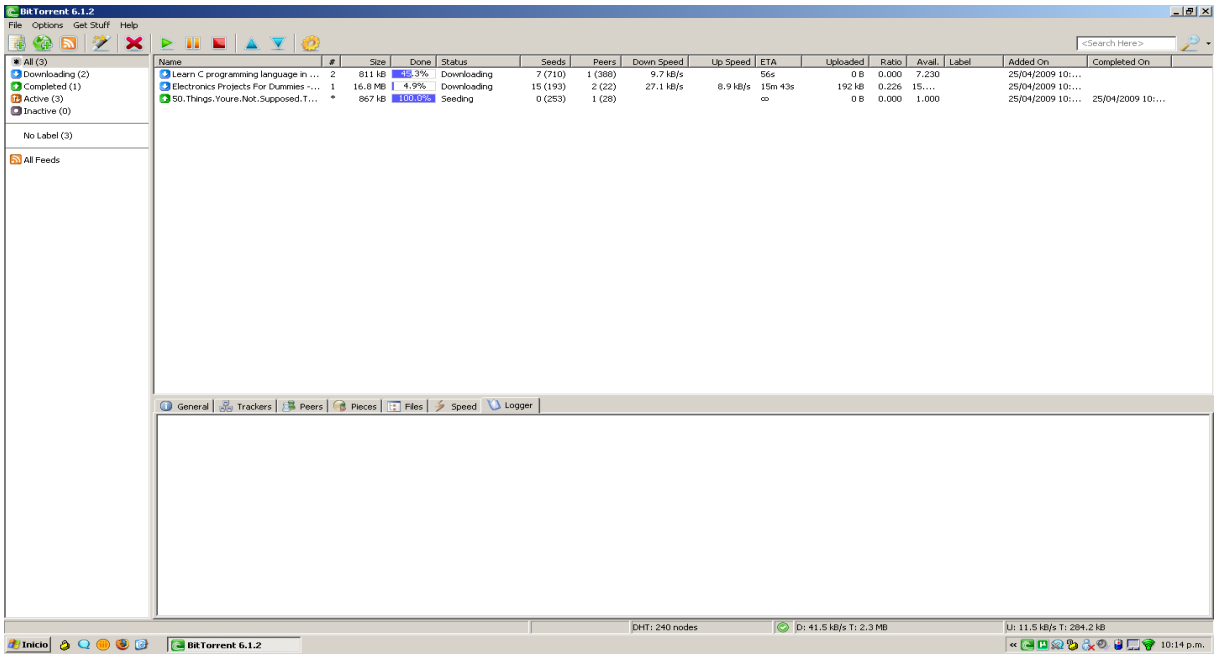


Como se indicó al describir el protocolo, la descarga se inicia al abrir un archivo con extensión .torrent, posiblemente descargado desde un sitio web.

Presionando en 'Advanced' se pueden activar o no algunas de las características adicionales o extensiones al protocolo que ofrece el cliente en uso.

A continuación, se comienza a enviar y recibir partes del archivo. La interfaz muestra en todo momento el estado de las transferencias en curso.

Algunos archivos pueden estar recibiéndose (*downloading*) o disponibles para otros usuarios (*seeding*). Se indica también otro tipo de información, como ser la velocidad de transferencia, el tiempo estimado para la finalización, la cantidad de bytes recibidos, la cantidad de *peers* disponibles, la cantidad de *seeds* disponibles, etc.



Descripción Técnica del Protocolo

En los siguientes links se describe en detalle el protocolo BitTorrent. Asimismo, se describen algunas de las extensiones disponibles, tanto oficiales como no oficiales.

http://wiki.theory.org/BitTorrentSpecification
http://www.bittorrent.org/beps/bep_0003.html

La aplicación

El cliente Torrent deberá estar implementado en C++ (Compilador ISO C++) para correr sobre las plataformas Linux y/o Win32.

Procesamiento de .torrent (parser bencoding).

Para poder compartir archivos es necesario disponer de un archivo de metadatos .torrent, estos archivos están codificados en bencoding. El cliente deberá obtener del archivo: nombre, tamaño y hash del archivo a descargar, la dirección del tracker e instrucciones para el cliente.

El bencoding permite codificar cuatro tipos de datos: números enteros, cadenas de bytes, listas y diccionarios, con el siguiente formato:

entero	i< num decimal >e
cadena de bytes	< longitud >:< secuencia de bytes >
lista	l< elemento1 >< elemento2 > . . . < elementoN >e
diccionario	d< key1 >< valor1 > . . . < keyN >< valorN >e

El archivo contiene al menos un diccionario con los campos *announce* e *info*.
Announce contendrá una cadena de bytes con la URL del tracker. En tanto que *info* tiene asociado un valor de tipo diccionario que contiene, entre otras, las siguientes entradas:
Name nombre del fichero.
Piece length entero que indica la cantidad de bytes que componen cada una de las partes del archivo a compartir.
Length entero que indica la longitud del archivo a compartir.
Pieces contiene una cadena que es el resultado de concatenar los códigos hash SHA1 de cada una de las partes del archivo. Los códigos son de 20 bytes cada uno.
Y otra información que resulta opcional como nombre y versión del programa que generó el archivo, fecha de creación y un espacio para comentarios, entro otros.

Los archivos de torrent para descarga de multiarchivos tienen un formato diferente. (Ver especificaciones del protocolo bittorrent: **http://wiki.theory.org/BitTorrentSpecification**)

Capacidad de actuar como cliente y servidor

Protocolo http : Comunicación con Tracker
Al parsear el .torrent se obtiene, entre otras cosas, la URL del tracker al cual conectarse para acceder a la lista de clientes.

La consulta al tracker se realiza mediante el protocolo HTTP, a la cual se le añaden al final una serie de parámetros consistentes en pares de valores de la forma param = valor separados por caracteres &.

Para la consulta se utiliza el método GET.

Parámetro	Descripción
info_hash	contiene el identificador de 20 bytes resultante de calcular el código hash SHA1 de la codificación bencoding del valor asociado a la entrada info del fichero .torrent.
peer_id	Sirve para identificar al agente de usuario mediante una secuencia de 20 bytes. Dicha secuencia puede ser generada siguiendo un criterio totalmente

	arbitrario.
port	Valor numérico con el puerto en el cual el cliente atiende a las conexiones de los demás clientes.
uploaded	La cantidad de bytes cargados hasta el momento, numero decimal
downloaded	La cantidad de bytes descargados hasta el momento, numero decimal
left	Cantidad de bytes que faltan para terminar la descarga.
event	indica tres posibles valores (started, stopped y completed) El valor started se utiliza en la primera consulta, mientras que completed es usado cuando el cliente dispone del recurso completo. El valor stopped sirve para indicar la terminación de la ejecución de un cliente al tracker.
ip	Opcional.
numwant	Opcional. Numero de peers que el cliente quisiera recibir del tracker.

El tracker responde con un documento codificado en bencoded (un diccionario) con las siguientes claves para indicar que usuarios que pueden proporcionar alguna parte del archivo requerido.

Clave	Descripción
failure reason	Indica, mediante un string, un mensaje de error
interval	Numero de segundos que deben transcurrir entre dos consultas sucesivas al tracker.
complete	Numero de seeders
incomplete	Numero de leechers
peers	
-peer id	String que identifica el peer
-ip	String con la dirección IP del un determinado peer (IPv6 o IPv4) o DNS
-port	Entero que indica el puerto de dicho peer.

Protocolo bittorrent : Message-flow con peers.

Los clientes se comunicaran entre si para intercambiar partes del archivo a compartir, para eso se utiliza una serie de mensajes, pero antes de que comiencen a intercambiar mensajes, es necesario transmitir una secuencia inicial con el objeto de identificarse y verificar que el recurso al que se quiere acceder es el correcto. Esto se conoce con el nombre de handshake y tiene la siguiente estructura:

pstrlen(1 byte)	pstr	reserved(8 bytes)	info hash(20 bytes)	peer id(20 bytes)

Contiene la longitud y el nombre del protocolo, el código hash SHA1 de la sección info del fichero .torrent y el id del cliente.

Una vez que se haya recibido esta secuencia de iniciación, el agente decidirá si va a atender la conexión o cerrarla, en función del valor info hash. Si ambos valores no son idénticos, se cerrara la conexión, sino comenzará el intercambio de mensajes

Los mensajes tienen la siguiente estructura:

Longitud (4 bytes)	id (4 bytes)	payload (longitud variable, depende del mensaje)

Los mensajes disponibles son:

choke Indica a un cliente que sus peticiones serán ignoradas

unchoke Derogación de lo dispuesto por el mensaje choke

interested Declaración del interés por alguna parte ofrecida por el cliente remoto

not interested Derogación de lo dispuesto por el mensaje interested

have Información a un agente de que una parte ha sido completada

bitfield Información de partes ya obtenidas

request Solicitud de un bloque

piece Transmisión de un bloque

cancel Petición de cancelación de transmisión de un bloque

(Ver el detalle en las especificaciones del protocolo bittorrent:
http://wiki.theory.org/BitTorrentSpecification)

Una vea terminada la descarga de cada una de las partes del archivo compartido se debe verificar la integridad comparando el codigo hash SHA1 con el contenido en el archivo .torrent que refiere a dicha parte, de no coincidir se desecha y se descarga nuevamente.

Funcionalidad mínima requerida:

- El cliente deberá averiguar qué otros clientes poseen los datos buscados, conectarse a ellos para realizar la descarga de dichos datos y al mismo tiempo atender peticiones de otros clientes.
- Debe permitir realizar múltiples transferencias simultáneas como así también el uso de torrents multiarchivo. Se debe permitir suspender y reanudar descargas.
- Se debe implementar un sistema de premiación, dándole mayor cantidad de conexiones a nodos de descarga para quienes compartan más.
- Se deberá implementar una estrategia de descarga de pieces para lograr así lograr una transferencia más eficiente. Recomendado rarest-first.
- El cliente deberá contar con una interfaz grafica (GUI) con al menos la siguiente funcionalidad:

Listado de transferencias (up y down), barra de progreso para enviados y recibidos, poder ver los clientes con los cuales se esta interactuando, poder acceder al listado de archivos a compartir, el estado y el detalle de las transferencias, posibilidad de configurar la cantidad de transmisiones simultaneas.

- Además de la interfaz grafica, se debe poder utilizar el programa por línea de comandos (logrando de esta manera una total independencia de la GUI)

No se pide la implementación de un tracker. Para realizar las pruebas se utilizará un tracker ya existente.

Características opcionales

Esta es una serie de características aplicables al trabajo que no son de carácter obligatorio. Algunas de ellas pueden tener alta complejidad.

- Priorization / Selective Download
- Torrent Publishing
- Peer Exchange - http://en.wikipedia.org/wiki/Peer_exchange
- Super Seeding - http://en.wikipedia.org/wiki/Super-seeding
- Distributed Hash Table - http://en.wikipedia.org/wiki/Distributed_hash_table
- Protocol Encryption - http://en.wikipedia.org/wiki/BitTorrent_protocol_encryption
- Web Seeding
- Multi Tracker Support

Modalidad de trabajo

Grupos de trabajo

El grupo de trabajo del TP debe estar integrado por 3 alumnos regulares de la cátedra. Sólo se contemplarán grupos de distinta cantidad de integrantes por razones de fuerza mayor y bajo expresa autorización del cuerpo docente.

Calendario Sugerido

Las fechas indicadas corresponden al inicio de la semana.

	Alumno 1 (GUI + Parser + SHA-1)	**Alumno 2 (Comunicaciones)**	**Alumno 3 (Lógica)**
Semana 1 (05/05)	Análisis del protocolo bittorrent		
Semana 2 (12/05)	Investigación de widgets a utilizar en la GUI, SHA-1 y creación de parser becoder.	Cliente-servidor multithread	Análisis de la lógica
Semana 3 (19/05)	Programa de prueba de interfaz gráfica sin funcionalidad de red – Procesamiento de .torrent. (de 1 o mas archivos)	Protocolo http : Comunicación con Tracker. Protocolo bittorrent : Message-flow con peers.	Implementación de la lógica: Manejo de *peers status.* Manejo de *pieces.* Downloads – Uploads. Múltiples transferencias simultáneas. Command Line.
Semana 4 (26/05)	GUI completa. Integración.	Servidor completo. Integración.	Lógica completa. Integración.
Semana 5 (02/06)	Testing y documentación.		
Semana 6 (09/06)	**PREENTREGA**		
Semana 7 (16/06)	Realización de correcciones en base a los resultados de la preentrega		
Semana 8 (23/06)	**ENTREGA FINAL**		

Bibliotecas / Librerías

Toda funcionalidad que no figure en la siguiente lista deberá ser implementada por los alumnos:
1. ISO C++
2. STL
3. Bibliotecas específicas del SO para el manejo de threads y sockets. Los sockets deben ser BSD compatibles (es decir, bloqueantes).
4. Para la interfaz gráfica y para el instalador se pueden usar GTK o gtkmm .
Si el grupo desea usar bibliotecas no incluidas en la lista, deberá consultar con los docentes (se recomienda solicitar la autorización por escrito).

Entrega

El software deberá constar de instalador. El código fuente deberá contener comentarios claros y estar impreso en un apéndice del informe.

Detalle del informe:

Definición del proyecto

- Enunciado
- Cronograma
- División de tareas
- Reporte de avance
- Mejoras
- Conclusión

Documentación Técnica

- Análisis (pantallas, descripciones)
- Diseño (notas técnicas, estructuras de datos (uml), interfaces, políticas adoptadas)

Documentación del usuario

- Forma de instalación
- Forma de uso (pantallas, explicación)
- Pruebas