UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA

Protocolo BitTorrent

AUTORES.

Alejandro Pardo Tapia

Diego Martínez Campos

Resumen

Con la masificación de Internet se ha hecho cada vez mas masivo el acceso a la información y contenidos, un contenido puede viajar en cuestión de minutos al rededor del mundo para ser accedido por millones de personas, este gran avance supone a la vez grandes desafíos tecnológicos especialmente en la compartición de archivos multimedia, que un archivo sea accesible por millones de personas significa al mismo tiempo que alguien debe suministrarlo. Bajo este escenario el paradigma clásico de cliente/servidor falla pues supone una sobrecarga en el servidor y un deterioro de la calidad del servicio a medida que el contenido se vuelve mas popular.

Las redes de pares (p2p) suponen un avance en esta materia aprovechando el ancho de banda de los usuarios que ya cuentan con el contenido para que estos ayuden en la difusión del material multimedia. De esta manera, contrario al sistema centralizado, a medida que el material se vuelve más popular, mejora la calidad del servicio dada la mayor cantidad de fuentes disponibles para la descarga.

Las redes p2p "puras" a su vez presentan otros desafíos como el problema de encontrar el contenido buscado y establecer las conexione entre usuarios que no están siempre conectados.

Bittorrent intenta dar solución a estos problemas con un sistema híbrido basado en redes independientes por cada archivo, en este trabajo se mostrará los principios de funcionamiento y arquitectura de la red, se mostrará su desempeño bajo distintas situaciones, se especificará el protocolo utilizado y se mostrarán algunos desarrollos hechos sobre esta misma para ofrecer nuevos servicios u/o mejorar los existentes.

BitTorrent en general ha demostrado ser una exitosa solución al problema de congestión en Internet demostrando una alta capacidad de escalabilidad y robustez en el servicio.

1. Introducción

BitTorrent es un protocolo de compartición de archivos p2ppar a par, sistema de red en que cada usuario funciona simultáneamente como cliente y servidor desarrollado inicialmente por Bram Cohen en el año 2001http://bramcohen.com/BitTorrent/ , esta pensado especialmente como una solución para archivos de gran tamaño. Funciona con un servidor central (tracker) que maneja el estado de la red, pero no requiere enviar el contenido en si, esta tarea esta destinada a los usuarios. Cuando un usuario quiere poner a disposición del publico un archivo crea un archivo .torrent con la información de este y lo publica en un tracker, luego un usuario cliente puede descargar este archivo para acceder a la red y empezar la descarga, cada archivo es dividido en pequeñas porciones (típicamente medio MiB) que son compartidas bajo la premisa de "la porción menos disponible primero", un usuario que tenga alguna parte completa puede a su vez ayudar al estado de la red compartiendo las porciones del archivo que ya posee, o compartiendo el archivo entero una vez que lo posea, de este modo se asegura una mayor redundancia de datos y se alivia la congestión del usuario que comparte el archivo original. Se estima que para el año 2005 el 30% del trafico por internet correspondía a esta red mientras que segun isoHunthttp://isohunt.com su servidor maneja mas de 1.4 millones de torrents que representan mas de un PetaByte de contenidos.

2. Protocolo BitTorrent

2.1 Problemas y enfoques de solución a sistemas de distribución.

Los servicios basados en el paradigma cliente/servidor tienen dos desventajas fundamentales.

A medida que aumenta la popularidad de un archivo, el trafico que el servidor aumenta en directa proporción, lo que obliga a aumentar la capacidad del servidor con el consecuente aumento en los costos o limitar el trafico de este a costa de un peor servicio para los usuarios.

La disponibilidad del contenido depende de la disponibilidad del servidor, esto significa que si por algún motivo el servidor falla, el contenido ofrecido deja de estar disponible.

P2p significa una mejora al primer problema porque aprovecha el ancho de banda de los usuarios para ofrecer el contenido, de esta forma cada usuario que ya posee una copia del archivo puede ofrecerlo a los demás permitiendo una disminución en el trafico por el servidor inicial.

El hecho de que múltiples usuarios puedan actuar como servidores de contenido permite una mayor robustez en el servicio dado que si un usuario se desconecta, puede haber otros desde los cuales conseguir el contenido deseado.

En una red p2p descentralizada, cada usuario que se conecta a la red comparte los archivos que dispone y a la vez tiene acceso a los archivos que poseen el resto de los usuarios. Al no haber un servidor que administre las conexiones y el contenido, cada usuario, para evitar saturar su computador, solo se conecta simultáneamente a una fracción de los usuarios de la red, por lo que puede conocer los contenidos ofrecidos por estos, de esta manera, cada cliente debe almacenar información sobre los contenidos a los que puede acceder directamente y a través de quién están disponibles, si lo que este busca no se encuentra indexado debe hacer una búsqueda, que significa preguntar a los pares conectados directamente por el contenido buscado, quienes a su vez revisan sus propias disponibilidades de contenidos y responden con

los enlaces necesarios en caso de éxito o preguntan a su vez a sus otros pares disponibles. Todo este intercambio de información implica un desperdicio de ancho de banda que debería utilizarse para la transferencia de datos en vez de mantener activa la red, este sistema limita considerablemente la escalabilidad de la red debido a la limitante del numero máximo posible de pares conectados simultáneamente, y la posibilidad de encontrar los datos requeridos, debido a que hacer una búsqueda exhaustiva por cada archivo significaría acceder a cada usuario conectado en búsqueda de la información lo que es impracticable si se consideran redes de miles de usuarios activos simultáneamente. Bittorrent mejora el problema con un sistema híbrido basado en redes dedicadas por cada archivo (o grupos de archivos) que se desee compartir. Existe un servidor central que contiene información de los usuarios conectados actualmente a la red al cual se conectan los nuevos clientes, éste indica las direcciones a las cuales conectarse para obtener el archivo pero no (necesariamente) envía el contenido en si, esto se hace a través de los usuarios de la red, esto evita la congestión en el servidor dado que la cantidad de datos que debe suministrar es significativamente menor a la que debería enviar si sirviera el contenido en si mismo, a la vez la congestión en los clientes disminuye dado a que ahora poseen una lista de clientes de los cuales se tiene certeza que poseen el archivo, no hay sobrecarga por búsqueda ni mantención de la red y además se tienen múltiples fuentes desde las cuales obtener el contenido por lo que la desconexión de un usuario no evita que el contenido siga estando disponible.

2.1 La red Bittorrent

Funcionamiento

La red bittorrent básicamente se caracteriza por dotar al cliente de un archivo obtenido desde un servidor tracker, el cual asigna a cada cliente “n” posibles destinatarios para realizar la descarga y de esta forma conexionarse a ellos para descarga descentralizada. La obtención de fichero: lo primero es habilitar un fichero, instalando un cliente de BitTorrent. Luedo Accedemos a la página web que contiene el link del fichero “.torrent” (.torrent es la extensión), finalmente se requiere que haya gente compartiendo (como mínimo 1).

2.2 Desarrollos y servicios basados en bittorrent

Multitracker Metadata ExtensionREF1 :

Es una mejora propuesta por John Hoffman consistente en agregar al archivo .torrent la clave announce-list que contiene una lista de trackers que mantienen simultáneamente información sobre el estado de la red, de esta manera, si un tracker falla, la red puede seguir funcionando

DHTREF2

Andrew Loewenstern propuso un sistema en el cual utilizando tablas hash es posible mantener una red activa aún en ausencia se un tracker activo, el algoritmo de routeo esta basado en el utilizado en la red kademlia.

Video por bittorrentREF3

LiveBT es una propuesta de mejora del protocolo bittorrent que permitiría a la red soportar videos bajo demanda, está desarrollada por una serie de investigadores de las universidades Fuzhou y la academia de ciencias, ambas en china

Similarity Enhanced Transfer (SET)REF4

David G. Andersen de la Universidad Carnegie Mellon ha propuesto una técnica que puede mejorar las redes p2p con pocos usuarios compartiendo un archivo en especifico, agregando a la red de descargas usuarios que tengan archivos que le sean parecidos, de este modo, al descargar desde varias fuentes varias versiones levemente modificadas del archivo original es posible reconstruir el contenido deseado, ésta técnica es especialmente interesante para el caso del contenido multimedia dada la gran cantidad de versiones distintas para una canción o video determinado que se pueden encontrar en la red

3. Conclusiones

Dada las condiciones que el protocolo bitTorrent opera el sistema es considerado escalable dado que ante un aumento de demanda de descarga de archivos beneficia directamente con un aumento de nodos de descarga que son nuevos potenciales clientes con archivos en descarga. También se considera un protocolo caracterizado por su robustez, ya que fallas en la red no producen caducación de la descarga, sino pausa en ella.

Según estudios representa uno de los protocolos p2p mas populares en la descarga de archivos, dada su tolerancia y aprovechamiento del tráfico.

Una vez obtenida la extensión de descarga, se asegura un rápido inicio de descarga ya que hay manejo de direcciones de nodos precisos que harán posible la descarga, sin cola de espera como resultado de servidor descentralizado.

Anexo

Especificación del protocolo

Sobre el archivo .torrent

El archivo .torrent está codificado mediante b-coding que permite mejorar el soporte multiplataforma, contiene un diccionario que puede variar de acuerdo a la implementación, pero las principales claves son:

info: Es un diccionario que describe el(los) archivos que contiene el torrent, puede tener dos formas, para torrents de un solo archivo y para torrents de multiples archivos. Contiene

piece length:(entero) numero de bytes de cada pieza.

pieces:(byte string) String consistente en la concatenación de todas las sumas de comprobación SHA1 (20 bytes) (una por pieza).

private:(opcional)(entero) indica si el cliente puede descubrir pares solamente a través del tracker especificado, esto es no están permitidas otras formas de descubrimiento de paresmas sobre esto en la sección de mejoras, si es un .torrent de un solo archivo sigue con:

name:(string) El nombre del archivo.

length:(entero) Largo del archivo en bytes.

md5sum:(opcional)(string) suma MD5 del archivo (32 caracteres hexadecimales)

si es un .torrent de multiples archivos sigue con:

name:(string) El nombre de la carpeta donde están todos los archivos

files: una lista de diccionarios, una por cada archivo, cada diccionario de la lista contiene las siguientes entradas:

length:(entero) Largo del archivo en bytes.

md5sum: (opcional)(string) suma MD5 del archivo (32 caracteres hexadecimales)

path: Lista que contiene uno o mas strings que representan el directorio y el nombre del archivo, cada elemento representa una carpeta mientras que el ultimo el archivo final

announce: (string) La URL del tracker

creation date:(opcional)(entero) La fecha de creación del torrent en formato standard UNIXsegundos desde 1-enero-1970 00:00:00 UTC)

comment:(opcional)(string) cadena de texto con comentarios del autor de .torrent

created by:(opcional)(string) Nombre y versión del programa usado para hacer el .torrent

encoding:(opcional)(string) El formato de codificación usado para generar la sección pieces de la entrada info en el .torrent

Sobre los mensajes entre los pares

cada cliente utiliza una conexion TCP con un handshake inicial de la forma

handshake: <pstrlen><pstr><reserved><info_hash><peer_id>

donde:

pstrlen: largo del string pstr
pstr: string identificando el protocolo
reserved: 8 bytes reservados que generalmente son ceros
info_hash: hash SHA1 provisto por el archivo .torrent
peer_id: cadena de 20 bytes usada para identificar el cliente está compuesta por
-CodigoDelClienteVersion-NumerosAleatorios>

por ejemplo una cadena handshake puede ser la siguiente

<19><BitTorrent protocol><00000000><f5s4fc10#.df48(r6sd8><-TR7973-215487654846>

una vez establecida la conexión los clientes se envían datos de control que tienen la siguiente estructura:

<length prefix><message ID><payload>.

Donde leng prefix es un valor de cuatro bytes big-endian, el message ID es un byte decimal mientras que payload depende del tipo de mensaje. Los posibles mensajes a enviar son los siguientes

keep-alive: <len=0000> Este mensaje tiene el único propósito de mantener la conexión abierta si no se han enviado mensajes en una cierta cantidad de tiempo, generalmente dos minutos

choke: <len=0001><id=0> Este mensaje indica si el cliente ha sido bloqueado

unchoke: <len=0001><id=1> Indica si se ha desbloqueado al cliente

interested: <len=0001><id=2> Para indicar que el se está interesado al alguna parte del archivo que el cliente posee

not interested: <len=0001><id=3> Para indicar que el cliente no posee partes del archivo en la cuales se esté interesado

have: <len=0005><id=4><piece index> Es un índice indicando que partes del archivo ya se posee, de este modo los clientes pueden hacer una mejor elección sobre que parte ofrecer.

bitfield: <len=0001+X><id=5><bitfield> Este mensaje se envía inmediatamente después del handshaking e indica los trozos del archivo que de posee actualmente, cada bit del campo payload representa una pieza, si está ya se posee el bit es seteado a uno o en caso contrario el bit es cero. X indica el largo del campo bitfield, si este está mal calculado, el cliente debería ignorarlo y cerrar la conexión.

request: <len=0013><id=6><index><begin><length> Este campo se usa para pedir un trozo del archivo, index indica el numero del trozo, begin es para indicar un offset a partir del comienzo del trozo en cazo de que ya se posea una parte de este y length indica cuantos bytes se están solicitando

piece: <len=0009+X><id=7><index><begin><block> Este mensaje se utiliza para enviar el contenido en si, index indica el numero de la pieza enviada, begin indica el offset desde el comienzo de la pieza mientras que block son los datos enviados

cancel: <len=0013><id<=8><index><begin><length> Este mensaje se usa para cancelar un pedido por alguna sección de datos en especifico, los campos de payload son equivalentes a los del mensaje request

port: <len=0003><id=9><listen-port> Este mensaje es enviado por clientes bittorrent que implementan DHTmas sobre esto en la próxima sección para indicar en que puerto del cliente se está escuchando

Referencias

[REF1] http://www.bittornado.com/docs/multitracker-spec.txt

http://bittorrent.org/beps/bep_0012.html

John Hoffman febrero del 2008

[REF2] DHT Protocol

Andrew Loewenstern enero del 2008

[REF3] LiveBT: Providing Video-on-Demand Streaming Service over BitTorrent Systems

Jianming Lv Xueqi Cheng Qing Jiang Jing Ye Tieying Zhang Iming Lin Lei Wang

Diciembre del 2007 Chinese Acad. of Sci., Beijing;

[REF4] http://www.cs.cmu.edu/~dga/papers/nsdi2007-set/

Exploiting Similarity for Multi-Source Downloads using File Handprints

Himabindu Pucha and David G. Andersen and Michael Kaminsky

abril del 2007 Universidad de Cambridge.