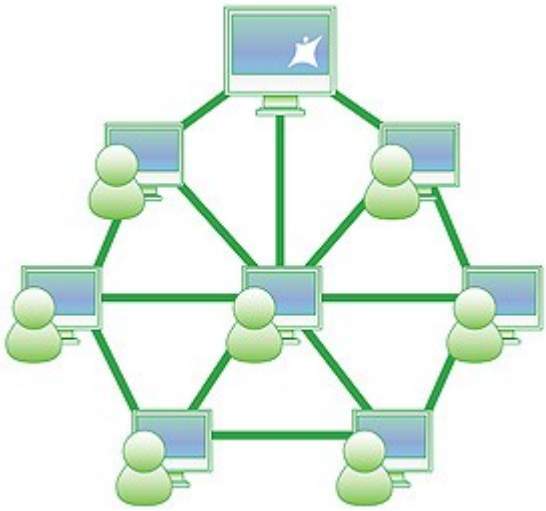
Clément HENRY
Alumno de Intercambio - 90001146

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Estudio de Escalabilidad de Bittorrent

Informe – Proyecto ELO322 – 08/07/15



I – RESUMEN:

Durante los últimos años, los archivos que circulan en el Internet han ganaron en volumen y el nombre de usuarios ha crecido también. El problema es que todas las arquitecturas que existen en el Internet no pueden adaptarse a este cambio. En este documente, se compara primero la escalabilidad de una arquitectura cliente-servidor clásica con la arquitectura Peer-to-Peer. En segundo lugar, se estudia como se podría simular virtualmente las redes P2P que tengan una arquitectura muy compleja. El protocolo usado en este documente para la arquitectura P2P es el protocolo Bittorent.

II – INTRODUCCIÓN

Hoy en día, los usuarios del Internet intercambian más en más archivos voluminosos. En efecto, hay un gran contenido multimedia (películas, música, Skype, ...) que circula en el Internet. Las aplicaciones deben adaptarse al crecimiento del numero de usuarios. Eso se llama la escalabilidad. La calidad del servicios proveído no debe cambiar con el crecimiento de la cantidad de usuarios. Hay dos arquitecturas principales en el mundo de las redes hoy día. Primero, el modelo Cliente-Servidor que es el más antiguo. Con esta arquitectura, todos los clientes deberían descargar un archivo en el servidor donde se encuentra éste. Eso plantea un problema de escalabilidad. Cuando el numero de usuarios va a aumentar, la salida del servidor va a comportarse como un cuello de botella. Segundo, tenemos la arquitectura Peer-to-Peer (P2P). En este modelo, cada cliente también se comporta como un servidor. En efecto, a medida que un cliente descarga un archivo, puede distribuir partes de este mismo archivo. En todo el documento, el protocolo de P2P usado es el protocolo Bittorrent. Entonces, vamos a ver primero como este protocolo funciona. En segundo lugar, vamos a comparar el P2P y la arquitectura Cliente-Servidor. Finalmente, vamos a ver como se puede simular el P2P. Para hacer eso, se usará la aplicación Peersim que permite simular sistemas P2P.

III – PRESENTACIÓN DEL PROTOCOLO BITTORRENT

El protocolo Bitorrent se basa en la arquitectura P2P. Permite intercambiar archivos entre diferentes hosts (peers). Un torrent es un grupo de peers que intercambian un archivo. En esta parte, vamos a ver en un primer tiempo como funciona el protocolo Bittorrent. En segundo lugar, se comparará este protocolo con la arquitectura Cliente-Servidor.

1 – Funcionamiento de Bittorrent

El protocolo Bittorrent permite compartir archivos voluminosos. No vamos a ver la tablas de Hash acá. Si alguien quiere compartir un archivo, primero tiene que publicar un archivo .torrent conteniendo informaciones para el buen funcionamiento del protocolo. Estas datos son el nombre del archivo, su tamaño y las direcciones de sus trackers. Un tracker es un peer que conoce los otros peers que tienen partes del archivo. El archivo es dividido en pequeñas partes (Chunk en ingles) de 256KB cada una. Cuando una persona llega y quiere descargar el archivo, debe pedir a un tracker una lista de los peers que tienen una parte o todo el archivo. Después, el peer puede empezar a descargar. Hay diferentes eventos que pueden ocurrir durante la transmisión:

1. Churn: las personas pueden entrar en el torrent o dejarlo

2. Un peer pide periódicamente a los otros peers los chunks que no tiene. El mas raro antes.
3. tit-for-tat: Esto define a quien los peers deben enviar los chunks. El peer tiene una lista de los 4 peers que lo envían más datos. Esta lista se re-actualiza cada 10s. El peer enviá sus chunks a estos peers con su tasa de transmisión la más importante. Cada 30s, el peer elige de manera aleatoria un otro peer (optimistic unchoke) que integra el top 4.

2 – Comparación Cliente-Servidor y del protocolo Bittorrent

En esta parte, se habla de una ventaja importante del protocolo Bittorrent y de la arquitectura P2P en general sobre la arquitectura cliente-servidor. Esta ventaja es la escalabilidad, la capacidad de un servicio a proveer la misma calidad cuando el nombre de usuarios aumenta. Podemos ilustrar eso con la transmisión de un archivo.

Acá están la formulas teóricas para calcular el tiempo de transmisión de un archivo de tamaño D entre N clientes en una arquitectura cliente-servidor y en una arquitectura P2P que usa el protocolo Bittorrent.

Arquitectura Cliente-Servidor:

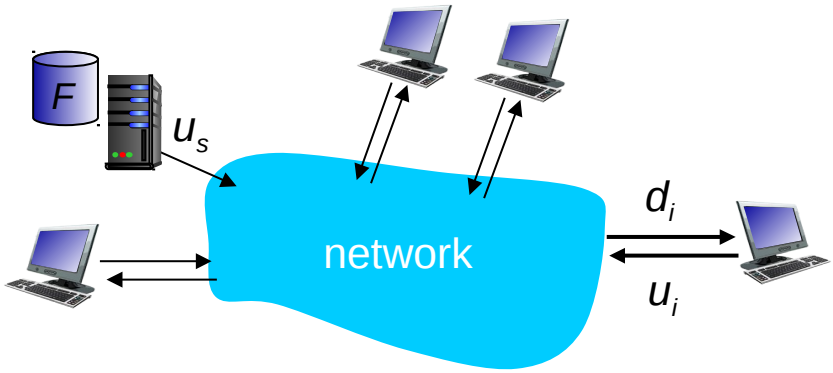


Figura 1: Arquitectura cliente-servidor [1]

Formula [1]: Sea ui, la tasa de transmisión de un cliente i en uplink, di en downlink y us la tasa de transmisión en uplink del servidor que contiene el archivo.

$$Tiempo\,de\,distribucion \geqslant max\left(\frac{D}{dmin}, \frac{N\,.\,D}{us}\right)$$

Arquitectura P2P con Bittorrent:

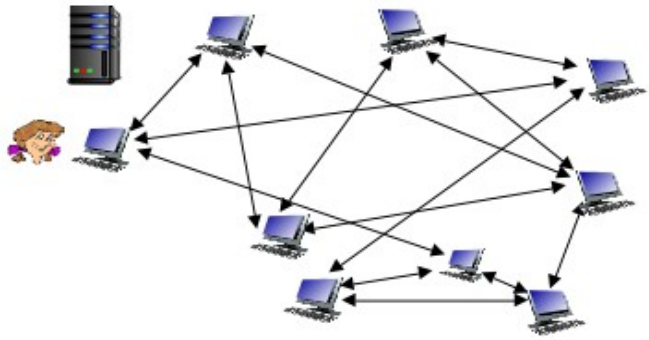


Figura 2: Arquitectura P2P [1]

Formula [2]: Sea C, la tasa de transmisión en uplink y downlink de cada peer, S_C el tamaño de un Chunk y S_F el tamaño del archivo a distribuir. Con esta formula, supongamos que un peer tiene el archivo al principio y que N-1 peers tienen que descargarlo.

$$Tiempo\,de\,distribucion \geqslant \frac{S_F}{C} + \frac{S_C}{C}.\left(\lfloor \log_2(N) - 1 \rfloor + min\left(1, \frac{N_P}{2\lfloor \log_2(N) - 1 \rfloor} - 2\right)\right)$$

Ahora podemos comparar la escalabilidad de cada arquitectura con Matlab (Script en la anexo). El siguiente gráfico muestra las curvas del tiempo de distribución de un archivo siguiente el nombre de clientes para las 2 arquitecturas. La simulación se hace con un archivo de tamaño 10MB. El tamaño de un chunk es de 256KB. La capacidad en uplink y downlink de los clientes es de 10KB/s. Ademas, la capacidad en uplink del servidor que contiene el archivo es de 100KB/s. Se supone que el cuello de botella esté a la salida del servidor.

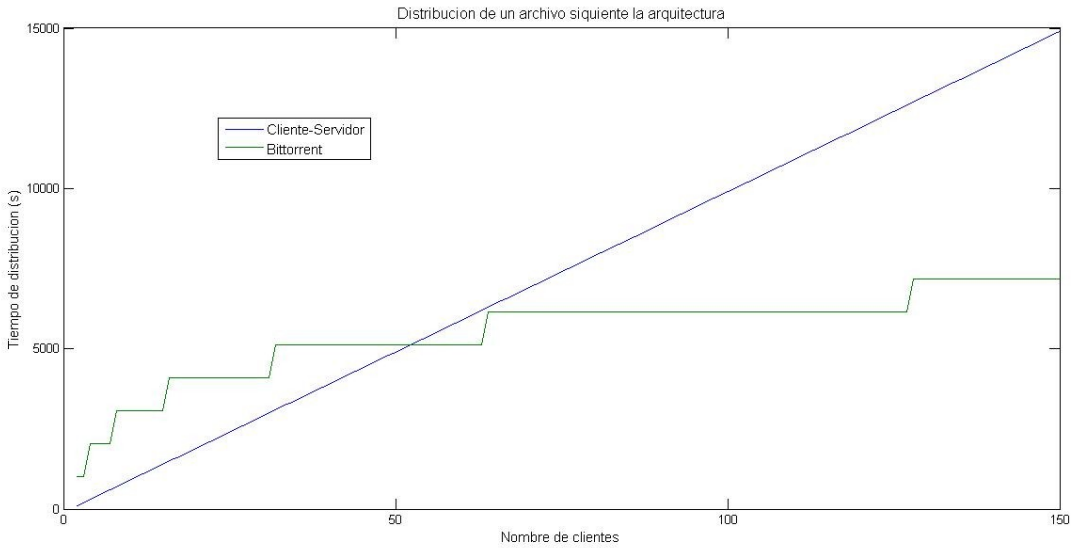


Figura 3: Distribución de un archivo siguiente la arquitectura

Se ve en la figura 3 que el tiempo de distribución del archivo en la arquitectura cliente-servidor tiene un crecimiento linear. Eso plantea problemas de escalabilidad por esta configuración. Al contrario, el tiempo de distribución aumenta en $\log_2(x)$ en la arquitectura Bittorent. Entonces, esta arquitectura muestra una mejora escalabilidad.

IV – SIMULACIÓN DE LA ESCALABILIDAD DE BITTORRENT

En esta parte, se simula el protocolo Bittorrent con una precisión mas grande. En efecto, la parte anterior solo muestra un aspecto muy simple de la arquitectura P2P con el protocolo Bittorrent. En realidad, esta arquitectura es difícil a estudiar. Una red P2P es demasiada grande para que se pueda implementar en una experiencia. Entonces, tenemos que utilizar una aplicación que permite simular este tipo de red virtualmente.

1 – Funcionamiento de Peersim

Peersim [3] es un simulador JAVA de red P2P basado en los eventos que ocurren el la red (event-based). Un evento puede ser mensajes intercambiados en un protocolo. Por ejemplo, con Bittorrent, Choke, Keep_Alive y Interested son eventos. Un módulo Bittorent [3] fue implementado al resto de la aplicación Peersim para simular especialmente el protocolo. Un archivo con los parámetros de simulación se pone en entrada y después se puede compilar y ejecutar Peersim. El funcionamiento de este módulo es enfatizado en [3].

2 – Simulaciones

Antes de simular, tenemos que definir los parámetros de simulación en un archivo (anexo 2) . Los parámetros principales son estos parámetros:

- Tiempo de simulación: simulation.endtime 10^6
- Tamaño de la red: network.size 30
- Tipo de protocolo de la capa transporte: init.net.transport urt
- Tamaño del archivo: protocol.bittorrent.file_size 100
- El máximo de nodos en el “swarm” de cada nodo: protocol.bittorrent.max_swarm_size 80
- El tamaño de la lista que envía un tracker: protocol.bittorrent.peerset_size 50
- Aumento máximo de la red en nodos: protocol.bittorrent.max_growth 20

Hay otros parámetros para definir la red y controlar el comportamiento de los trackers durante la simulación. Podemos encontrar la descripción de estos parámetros en [3].

Acá está una parte de la salida de la ejecución del módulo Bittorrent:

```
OBS: node 12(L)  pieces completed: 0          down: 0        up: 0 time: 10000
OBS: node 13(S)  pieces completed: 390        down: 0        up: 0 time: 10000
OBS: node 14(L)  pieces completed: 0          down: 0        up: 0 time: 10000
OBS: node 15(L)  pieces completed: 0          down: 0        up: 0 time: 10000
OBS: node 16(L)  pieces completed: 0          down: 0        up: 0 time: 10000
OBS: node 17(L)  pieces completed: 0          down: 0        up: 0 time: 10000
OBS: node 18(L)  pieces completed: 195        down: 0        up: 0 time: 10000
OBS: node 19(L)  pieces completed: 39         down: 0        up: 0 time: 10000
OBS: node 20(L)  pieces completed: 0          down: 0        up: 0 time: 10000
OBS: node 21(L)  pieces completed: 39         down: 0        up: 0 time: 10000
```

Cada nodo tiene un numero. Después de su nombre, podemos ver la cantidad de piezas del archivo que tiene cada nodo. Acá, uno de los nodo tiene 390 partes, eso significa que el nodo tiene todo el archivo. En efecto, 390*256KB~100MB (el tamaño del archivo). Ademas, eso se nota gracia al (S) cerca del nombre del nodo (Seed). Si el nodo no tiene todo el archivo, es un leecher (L). “Down” muestra el nombre de partes que el nodo descarga y “up” para upload. Finalmente, “time” es el tiempo de ejecución.

Se agrega nodos a medido del tiempo. En la siguiente figura, podemos ver la evolución del nombre de partes del archivo siguiente el tiempo para algunos nodos.

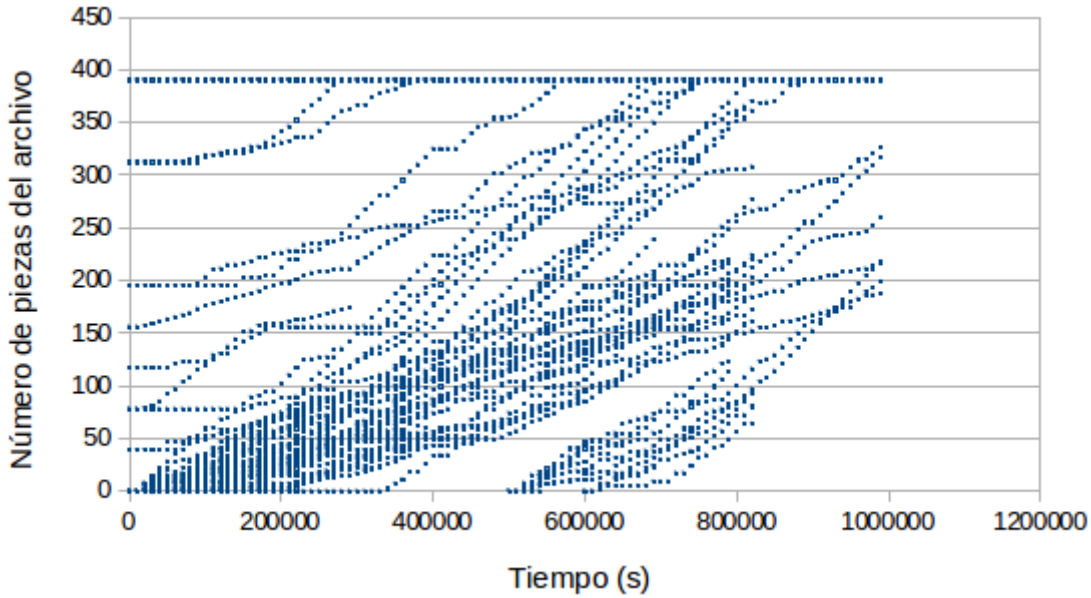


Figura 4: Distribución del archivo

Podemos ver en la figura 4 los diferentes comportamientos de los nodos. Los que empiezan a 390 son seeders. Si la evolución de un nodo queda constante durante un rato, eso significa que este nodo es “choked”, tiene que esperar que un otro nodo “unchock” a él.

V – CONCLUSIÓN

Así, el protocolo Bittorrent es un medio adecuado para compartir archivos voluminosos con una grande cantidad de usuarios. En efecto, la arquitectura tiene una grande escalabilidad en comparación de la arquitectura cliente-servidor. Sin embargo, los protocolos P2P son difíciles a simular porque se fundan en una arquitectura muy compleja. Entonces, Peersim provee una buena herramienta para simular y experimentar este tipo de arquitectura.

VI – BIBLIOGRAFÍA

[1] Computer Networking, A Top-Down Approach. J.F. Kurose and K.W. Ross. 2012.
[2] Scalability of the BitTorrent P2P Application. K. Eger and U. Killat.
[3] A BitTorrent module for Peersim. F. Frioli, M. Pedrolli. February, 2008.

VII – ANEXOS

1 – Script Matlab para calcular el tiempo de distribución

```
function [Tcs,Tp2p]= time(Sf,Sc,C,Cs,N)

%Sc: Size of the file
%Sf: Size of a Chunk
%C: Capacity in Downlink and Uplink of each client
%N: Number of clients who want to download the file
%Cs: Capacity in Uplink of the server which has the file

VectN=[2:N];

A=[1,VectN/(2*floor(log2(VectN-1)-1))-2];

Tp2p=((Sf+Sc)/C)*(floor(log2(VectN)-1)+min(A));

Tcs=((VectN-1)*Sf/Cs);

figure
plot(VectN,Tcs,VectN,Tp2p)
title('Distribucion de un archivo siguiente la arquitectura')
legend('Cliente-Servidor','Bittorrent')
xlabel('Nombre de clientes')
ylabel('Tiempo de distribucion (s)')


end
```

2 – Archivo de configuración

```
#Config file for BitTorrent extension

random.seed 1234567890
simulation.endtime 10^6
simulation.logtime 10^3

simulation.experiments 100

network.size 30
network.node peersim.core.GeneralNode

protocol.urt UniformRandomTransport
protocol.urt.mindelay 10
protocol.urt.maxdelay 400

#BE AWARE: the value "max_swarm_size" must be greater than
#the value "peerset_size", since I have to be sure
#that the space for the neighbor nodes is enough.

protocol.bittorrent peersim.bittorrent.BitTorrent
protocol.bittorrent.file_size 100
protocol.bittorrent.max_swarm_size 80
protocol.bittorrent.peerset_size 50
protocol.bittorrent.duplicated_requests 1
protocol.bittorrent.transport urt
protocol.bittorrent.max_growth 20

init.net peersim.bittorrent.NetworkInitializer
init.net.protocol bittorrent
init.net.transport urt
init.net.newer_distr 80
init.net.seeder_distr 15

control.observer peersim.bittorrent.BTObserver
control.observer.protocol bittorrent
control.observer.step 10000

control.dynamics peersim.bittorrent.NetworkDynamics
control.dynamics.protocol bittorrent
control.dynamics.newer_distr 60
control.dynamics.minsize 10
control.dynamics.tracker_can_die 1
control.dynamics.step 100000
control.dynamics.transport urt
control.dynamics.add 5
control.dynamics.remove 5
```