# Caching Techniques for P2P Traffic

**Pablo A. Castillo**
0981 833 757
pabloacastillo@gmail.com

Universidad Autonoma de Asuncion
Introduccion a la Comunicacion de Datos[1]

**Resumen**

*Con el crecimiento exponencial del consumo de altos anchos de banda , ahora abiertos al publico general, los ISP se enfrentan con una presion considerable para identificar y ofrecer soluciones viables para manejar estos volumentes de trafico y aliviar el costo consecuente producto del mismo.*
*Las aplicaciones P2P generan una gran parte del trafico de internet hoy en dia. El alto volumen de este trafico (y el alto potencial benefico del cacheado) asi como los grandes tamaños de cache requeridos (y la posible controversia asociada al cacheado) solo subrayan el hecho que una eficiente politica de manejo del trafico P2P es escencial dentro de un ISP. Este posee caracteristicas que la distinguen muy particularmente de cualquier otro tipo de trafico y requiere de un analisis exclusivo en los metodos de deteccion y control para este tipo de redes.*
*El consumo total del ancho de banda de estas aplicaciones se acercan al 75% del consumo total de internet. Esto crea un dilema para todo ISP: como manejar el costo asociado y evitar incurrir en el crecimiento fisico de la red sin afectar el acceso de sus suscriptores al P2P.*

## 1. INTRODUCCION

P2P se refiere a la relacion en la que multiples dispositivos autonomos interactuan como iguales. Una red peer-to-peer es un tipo de red en la que los nodos actuan tanto como clientes (pidiendo datos) y/o como servidores (ofreciendo datos) y/o "servents" (clientes y servidores). La tecnologia P2P ofrece compartir tanto servicios como recursos de cada uno de sus nodos incluyendo informacion, archivos, ciclos de proceso y almacenado mediante un cambio directo entre nodos (sin uso de un servidor central). Las redes P2P permite compartir recursos que de otra manera quedarian sin uso en terminales indviduales.
1ra. generacion: mayo del 99 prescencio el lanzamiento de Napster. Este fue el primer fenomeno P2P de alcance global con su pico en febrero del 2001 con 29.4 milliones de usuarios registrado compartiendo 2.79 billiones de archivos en el mismo mes.
Napster causo gran descontento dentro de las productoras de contenido por diciembre del 99 comenzando la operacion de caza y cierre de Napster a fines del 2001 que puede ser atribuida a su pobre infraestructura centralizada en unos pocos servidores de indexado que distribuian el contenido.
2da. generacion: en el 14 de marzo del 2000 se publica en Slashdot un articulo de Nullsoft (previo a la compra por Warner) divulgando los secretos de un protocolo opensource estilo Napster. AOL (propietaria de Nullsoft) quito el articulo pero ya fue muy tarde. Los blueprints de la segunda generacion ya habian sido liberados.
Los desarrolladores de esta segunda generacion estuvieron muy atentos de no cometer el mismo error de Napster. Abrazaron un protocolo e infraestructura descentralizada que fue el punto flaco de la primera generacion. A pesar de funcionar esta red creaba grandes problemas de trafico masivo entre sus nodos y una performance relativamente pobre.
3ra generacion: 2001. Siendo concientes los desarrolladores de las serias debilidades de la estructura descentralizada se ponen a trabajar sobre una infraestructura hibrida. Este problema se soluciono con la introduccion de una topologia de red virtual jerarquica con supernodos (o ultrapeers). Esto ayudo a reducir el trafico en la red y mejoro la performance de busqueda.

---

1

Esta tercera generacion tenia como lideres a Kazaa, Grokster y los distintos Gnutella. Esto hizo del Kazaa Media Desktop la aplicacion mas bajada en la historia de Internet.

El metodo de cacheado es conceptualmente sencillo: cuando un cliente dentro de mi red se realiza una query de busqueda pasando primero por mi cacheserver. Si esta disponible tal archivo simplemente se redirecciona la peticion al cache que lo envia asi al cliente. Si el archivo aun no ha sido cacheado la ruta de la query sigue su curso normal hasta localizar el contenido y el cacheserver copia el contenido de la conversacion entre ambos nodos y lo almacena para su futura distribucion. Si bien es conceptualmente sencillo se enfrenta a serios problemas. Las diferentes topologias de las mismas redes suponen un problema en si. En bitTorrent por ejemplo esto soluciona el downstream, pero deja abierto aun el upstream del cliente y asi si bien baja el consumo de bajada el de subida se mantiene. Otro problema es detectar a los mismos paquetes P2P del resto del trafico normal. Si bien la inspeccion de paquetes por firmas es un metodo viable el desarrollo de los softwares P2P los han vuelto mas inteligentes ocultando esta firma de distintas maneras dentro del paquete por no mencionar el costo adicional de tener que analizar todos los paquetes atravesando el NAP. (Si, todos los paquetes ya que los softwares tambien incluyen capacidades de eleccion dinamica de puertos asi como la ocultacion bajo trafico real como si fuera una transmicion valida por puerto 80 o 25)
No es facil la vida del ISP. Esta se enfrenta a tres frentes principales a la hora de elegir una solucion valida:

- Controlar el impacto sobre la infraestructura de soporte y los recursos tecnicos.
- Controlar los costos operacionales, especificamente de acceso/transito en la red
- Mantener la experiencia del usuario utilizando estas redes para mantenerlo satisfecho.

## 2. DESARROLLO

### 2.1 El problema en si

Con el crecimiento y evolucion de las PCs en aumento, asi como tambien la velocidad y frecuencia de las conexiones a internet, ha hecho que el surgimiento y popularizacion de las redes P2P consuman actualmente la porcion dominante del ancho de banda de uso residencial. La evolucion rapida de Napster a Gnutella a Kazaa, de eDonkey a WinMX y bitTorrent ha aumentado dramaticamente la cantidad de datos transferidos sobre las redes de los proveedores

Esto tiene impacto directo sobre los ISP: por un lado las redes P2P son una de las razones principales para adquirir tanto la conexion como el pedido de mas y mas ancho de banda. Como contracara el trafico resultante ofrece muchas serias desventajas de servicio para los ISP.

Los problemas a afrontar por los ISP son varios:
*1) Costoso ancho de banda consumido: en un ISP tipico el consumo P2P puede llegar a representar el 60% de su trafico total*
*2) Costo adicional de redes: las redes p2p se conectan para bajar el mismo contenido tanto desde el otro lado del mundo como desde el vecino por igual.*
*3) No existencia de un modelo de negocios sobre suscripcion a redes: un cobro extra por consumo de redes p2p es inaceptable. Tanto por las politicas stealth de las mismas como por el hecho de que la mayor parte de sus usuarios consumen 24/7*
*4) Perdida de la equidad: hoy en dia un ISP competitivo con una red congestionada tendra a todos sus usuarios peleando entre si por los mismos recursos.*

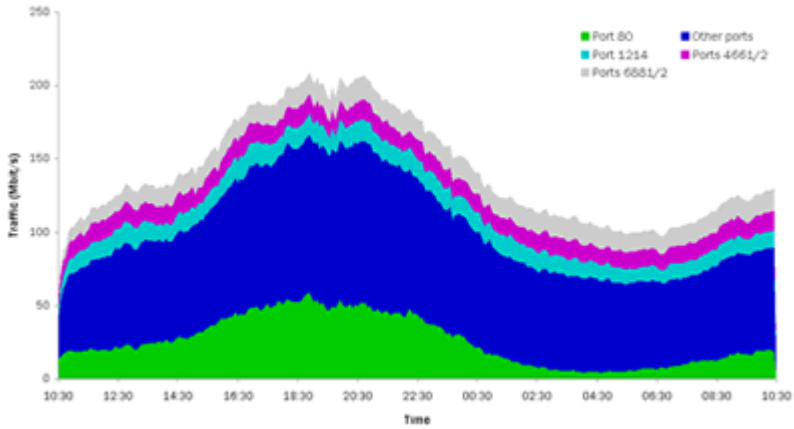
La degradacion de la experiencia del usuario:
Obtener y mantener una fuerte base de consumidores en un mercado ultra competitivo es muy difil. Degradar la experiencia del usuario no es aceptable.

- Tasas de transferencias mas bajas: al ser consumido el ancho de banda por trafico P2P, hay menos disponible para los demas usuarios.
- Distribucion desigual del ancho de banda: una desproporcionada parte de los suscriptores usan una gran parte del ancho de banda total con P2P. Este trafico trepa por encima del trafico legitimo de los demas usuarios bajando la performance de:

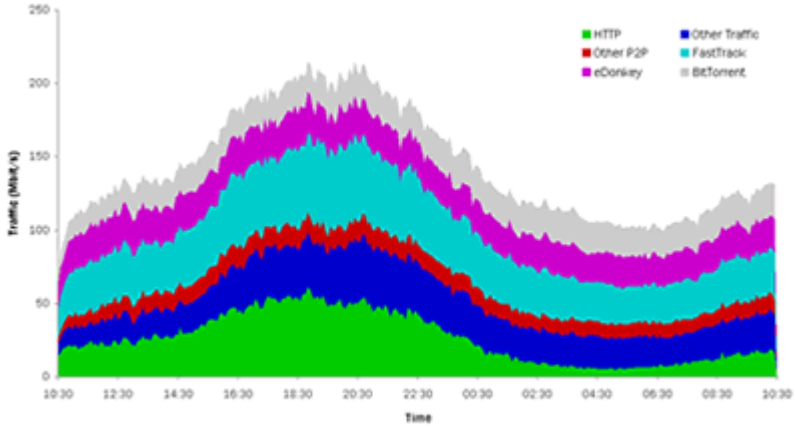
- Aplicaciones en tiempo real
- Navegacion
- Email

- Infraestrutura de red sobrecargada: una red congestionada tiene mas probabilidades de fallar mientras los equipos compiten entre si por los recursos. Cuando la red cae, los usuarios quedan, obviamente, sin servicio hasta que el proveedor pueda solucionar el problema.

Una solucion real al problema del trafico P2P debe estar dirigida tanto al ISP como al suscriptor del servicio. Las desventajas sobre las redes P2P son considerables, pero no imposibles de controlar.




Trafico por puerto




Trafico por protocolo

## 2.2 Deteccion de trafico P2P

Las circunstancias actuales nos llevan a una frustrante conclusion: una identificacion robusta del trafico P2P solo puede llevarse a cabo inspeccionando las transmiciones del usuario. Aun asi, la captura de paquetes y su analisis es un campo minado de problemas: legalidad, privacidad, tecnico, logistico y financiero. Peor aun, los intentos por ofuscar este tipo de trafico llevo a la tendencia cada vez mayor de encriptar estas comunicaciones (MUTE, Marabunda, TOR). De hecho, la frecuencia con que se introducen upgrades hacen de el analisis del trafico poco practico e ineficiente.
La mejor manera de detectar este abuso es con un analisis metodologico del patron de conexion de los paquetes en la capa de transporte y sin tener que depender de su contenido en si ni del consumo

del usuario.
El truco esta en la habilidad para identificar protocolos P2P sin depender del formato del paquete en si, lo que ofrece ademas una ventaja sobre las actualizaciones de los protocolos asi como los nuevos por venir.
Ventajas:

- *Hacer un profile del flujo y caracteristicas del comportamiento del P2P sin examinar el paquete del usuario*
- *Esta metodologia puede identificar e 99% del flujo P2P y limitas los falsos positivos a un 10%*
- *Es capaz de identificar el flujo que se escapa del analisis de consumo. Esta tecnica es capaz de detectar un 10% mas que anallizando el consumo en si.*
- *Usando los datos recolectados puede ofrecerse un estimado y tendencias del trafico P2P.*

### 2.2.1 Deteccion por firmas

El analisis trata de identificar caracteristicas de las palabras en los bits del paquete que potencialmente representa trafico P2P.
Cada tipo de red opera sobre protocolos no estandarizados, propietarios y de diseño personalizado. Asi, la identificacion requiere un analisis separado de varios protocolos diferentes para reconocer especificamente el formato de un paquete en cada caso.
Se buscan estos bits claves en los datos del nivel de aplicacion.

| PROTOCOLO P2P | CADENA | PROT. TRANS. | PUERTOS DEF. |
|---|---|---|---|
| eDonkey2000 | 0xe319010000<br>0xc53f010000 | TCP/UDP | 4661-4665 |
| FastTrack | "Get /.hash"<br>0x270000002980 | TCP<br>UDP | 1214 |
| BitTorrent | "0x13Bit" | TCP | 6881-6889 |
| Gnutella | "GNUT", "GIV"<br>"GND" | TCP<br>UDP | 6346-6347 |
| MP2P | GO!!, MD5, SIZ0x20 | TCP | 41170 |
| Direct Connect | "$MyN", "$Dir"<br>"$SR" | TCP<br>UDP | 411-412 |
| Ares | "GET hash:"<br>"Get shal:" | TCP | |

El proceso de deteccion es simple:
1) Si la fuente o destino utiliza uno de los puertos tipicos de P2P, la transmicion es estiquetada como P2P
2) Se compara los bits de datos dentro del paquete contra la tabla de cadenas. Si alguna concuerda se la etiqueta como P2P del protocolo correspondiente.
3) Si la transmicion esta etiquetada como P2P, tanto fuente como destino, son guardadas con sus direcciones dentro de una tabla. Toda la transmicion desde esa fuente y a ese destino seran etiquetadas como posible P2P aun si los pasos anteriores no arrojan positivos. Para evitar un error recursivo en la clasificacion de trafico no-P2P como P2P, se realiza este tipo de seguimiento solo en los IP que el paso 2) identifico como P2P.

Aun asi existen varias restriccion por las mismas caracteristicas del los protocolos P2P:

- *Captura del paquete: los monitores pueden capturar los primeros 16 bits del paquete del usuario (problemas legales en cuanto a privacidad impiden al ISP examinar mas alla). Esto impediria una deteccion 100% efectiva de todo el flujo P2P.*

- *HTTP request: muchos protocolos P2P usan HTTP request y response para transferir archivos, lo que lo hace imposible de distinguir del trafico HTTP normal con solo 16 bytes de lectura de paquetes.*
- *Encriptacion: el numero de protocolos utilizando SSL para transmitir archivos apenas comienza. Un paquete encriptado pasaria los metodos de deteccion.*
- *Trazas unidireccionales: algunas de las trazas pueden mostrar solo una direccion en el link de monitoreo. En este caso no puede identificarse el trafico P2P, ya que no hay datos.*

### 2.2.2 Deteccion por flujo

El Perfilado de trafico P2P (PTP: P2P Traffic Profiling) busca identificar el trafico P2P en puertos aritrarios sin inspeccionar el contenido de los paquetes.
Utilizando metodos heuristicos (antivirus) observando los patrones de conexion entre IPs fuente y destino puede ayudar a distinguir los falsos positivos y revelar propiedades distintivas del resto de las conexiones que pueden llegar a mostrar un patron similar.
Se utilizan dos metodos heristicos principales para la deteccion:
1) Examinar IP fuente-destino que usan TCP y UDP para transmitir datos.
2) Estudiando las caracteristicas de conexion entre los (IP, puerto)

- *Proceso de datos: se construye una tabla del flujo mientras observamos los paquetes atravesar el NAP, basado en 5-tuples, similar al metodo de inspeccion. Al mismo tiempo se estudian las caracteristicas de la conexion (IP, puerto) incluyendo los grupos de distintos IPs y puertos a los que un peer se conecta, el tamaño del paquete y el tamaño del flujo.*
- *Identificacion de posibles conexiones P2P: se etiquetan los flujos potenciales como P2P basandose en el uso de TCP/UDP y sus caracteristicas (IP, puerto).*
- *Falsos positivos: se eliminan los falsos positivos comparando las conexiones etiquetadas como P2P y se eliminan contra una tabla de identificacion de mail, DNS, malware*

| Puerto | Aplicacion |
|---|---|
| 135,137,139,445 | NETBIOS |
| 53 | DNS |
| 123 | NTP |
| 500 | ISAKMP |
| 554,7070,1755,6970,5000,5001 | streaming |
| 7000,7514,6667 | IRC |
| 6112, 6868, 6899 | juegos |

## 2.3 Los pares IP-TCP/UDP

Nuetro primer heristico identifica el origen-destino de un par IP que usen TCP y UDP. Seis de nueve protocolos P2P analizados usan ambos como capa-4 de transporte. Estos son eDonkey, Fasttrack(kazaa), WinMx, Gnutella, MP2P y Direct Connect.
Generalmente, control de trafico, querys y respuestas a querys usan UDP, y las transferencias reales TCP. Para identificar los host P2P se puede mirar el origen-destino que usan ambos protocolos. Mientras que el uso concurrente de TCP/UDP es determinante para detectar el trafico P2P, tambien es utilizado por otras aplicaciones como DNS o streaming. Para descartar de nuestras trazas que usan ambos protocolos, se examinan los pares de origen-destino contra el trafico TCP/UDP existente. Se descubrio que solo unas pocas aplicaciones usan ambos, TCP/UDP, como protocolos de transporte: DNS, NETBIOS, IRC, juegos y streaming que colectivamente usan unos pocos puertos como el 135, 137, 139, 445, 53, etc.
Exluyendo el flujo de esta tabla, el 98.5% restante usando TCP/UDP son etiquetadas como posible

P2P.
Resumiento, *si un origen-destino usa constantemente TCP/UDP como protocolos de transporte, se toma en consideracion el flujo entre estos peers mientras los puertos de origen-destino no se encuentren en la tabla.*

Nuetro segundo heuristico monitorea el patron de conexion entre ambos nodos tomado el IP, puerto de ambos (origen y destino)
Desde el caso judicial contra Napster, la presencia de redes P2P centralizadas ha casi desaparecido y las redes distribuidas o hibridas. Para conectarse a estas redes distribiudas, cada cliente P2P mantiene un cache al iniciar. Dependiendo de la red, el cache puede contener la direccion IP de otros peers, servidores y supernodos/ultrapeers. Este cache de conexiones facilita la conexion inicial a los nuevos peers a la red P2P ya existente.
En cuanto se establece una conexion a uno de los host del cache, el nuevo host A informa al supernodo su direccion IP y numero de puerto al que va a aceptar conexiones de otros peers. El host A tambien provee otra informacion especifica sobre cada protocolo P2P pero eso ahora resulta irrelevante.
Mientras que en la primer generacion de las redes P2P el puerto de escucha estaba bien definido las nuevas versiones de todos los clientes P2P permiten al usuario configurar un puerto al azar. El supernodo debe propagar la informacion (IP, puerto) del nuevo peer A al resto de la red. Este par, es escencialmente el ID del nuevo peer que los otros nodos deberan utilizar para comunicarse con el.
En resumen, cuando un host P2P inicia una conexion, tanto TCP como UDP, al peer A, el puerto de destino tambien sera informado al peer A, y el puerto de origen un numero al azar escogido por el cliente..
Normalmente los peers mantienen cuando menos una conexion TCP a cada uno de los otros peers, pero tambien puede haber trafico UDP al mismo peer. Teniendo en mente que las multiples conexiones entre distintos peers es poco usual en nuestras comunicaciones, consideramos que ocurre cuando 20 peers se conectan al peer A. Cada peer seleccionara un puerto de origen temporal y se conectara al puerto de escucha de A. El peer A estara asociado con 20 IPs y puertos distintos. En otras palabras, el numero de IPs conextados a A sera igual al numero de puertos en uso.
Consideremos que ocurre en HTTP. Como en P2P, cada host se conecta a un par especificado (IP, puerto). Usualmente un webserver creara mas de una conexion para soportar conexiones concurrentes y asi bajar en paralelo. En resumen, el trafico web tendra un ratio mayor que el P2P en cuanto a numero de IPs conectadas versus numero de puertos en uso.

```
PROCEDURE PTP                                    //PTP
FOR every src-dst IP pair in FT DO
      IF TCP/UDP pair THEN
            P2PIP.insert(srcIP)                  // heuristica TCP/UDP
            P2PIP.insert(dstIP)
FOR all flows in FT do
      IF src IP or dst IP in P2PIP THEN
            print flow
      //found by TCP/UDP pairs
            P2PIP.insert (srcIP)                 //put both IPs in P2P
            LIST
            P2PIP.insert (dstIP)
      ELSEIF DNS heuristic is true THEN
            RejectedPairs.insert(src Pair)
            pair=={IP,port}
            RejectedPairs.insert(dst Pair)
      ELSEIF src and dst IP not in MailServers THEN
            FOR src and dst Pair DO
                  IF pair in P2PPairs THEN
                        print flow               //found in previus interval
                        P2PPairs.insert (src pair)    //put both
                        pairs in P2PPairs list
```

```
                        P2PIP.insert(src pair)
                  ELSE IF pair not in Rejected THEN
                        Update sets for pair
                        IPPort.insert(pair)
                  ELSE IF pair in Rejected THEN
                        Rejected.insert(src pair)
                        Rejected.insert (dst pair)

FOR pairs in IPPort DO  //examine pairs thar were added
                        //during previous intervals and have not been yet
classified
IF IP not in MailServer and pair not in Rejected THEN
      IF IP in P2PIP or pair in P2PPairs THEN
            P2PPairs.insert(pair)
            print all flows pair
      ELSE
            diff = pair.IPSet.len - pair.PortSet.len
            IF diff<2 or (diff < 10 and port in KnownP2PPorts) THEN
                  IF Check_if_Mailserver==true THEN
                        MailServer.insert(IP)
                  ELSE IF Check_if_Malware==true THEN
                        Rejected.insert(pair)
                  ELSE IF Port-History heuristic==true THEN
                        Rejected.insert(pair)
                  ELSE
                        P2PPairs.insert(pair)
                        print all flows of pair
      ELSE IF diff >10 THEN
      Rejected.insert (pair)
```

## 2.4 Cacheado

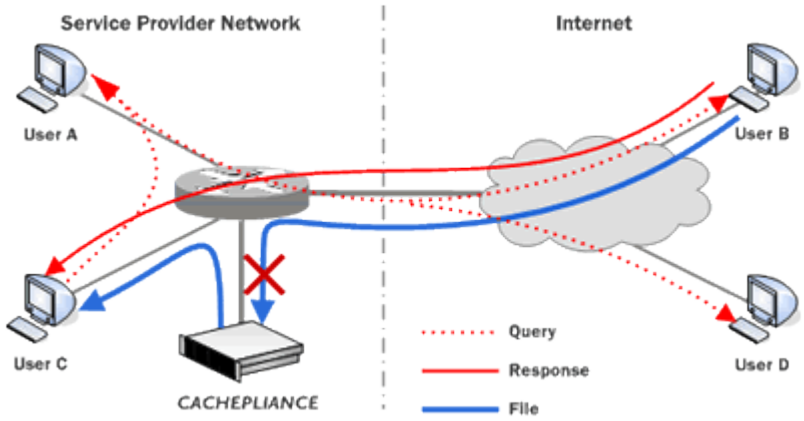
Tenemos dos necesidades principales para nuestro cache:
*1) debe ser transparente (no hay cambios en el protocolo de bajada)*
*2) debe ser pasivo (no debe originar pedidos de descargas).*
Para satisfacer los pedidos debe contener el rango completo de los archivos. A esto lo llamaremos "escenario de cache completo P2P".
Por otro lado, los rangos que no esten totalmente cacheados no seran ofrecidos. Para salvar esta ineficiencia podemos tomar dos alternativas:
El escenario de cacheo P2P parcial donde permanece pasivo pero dejar de ser transparente modificara el protocolo de bajada para negociar con el cliente el download de los subrangos del rango pedido. Alternativamente el cache puede volverse activo y pedir un download completo del rango faltante.

Otra pregunta es: *debe el cache ignorar las cancelaciones de los usuarios?*. Usualmente un cliente cancela el download cuando encuentra una mejor fuente de descarga. Para este caso el cache dejara de recibir la informacion, por otra parte el cache puede continuar la descarga por su cuenta y completar la descarga para ofrecerla mas adelante. Este comportamiento podria considerarse predictivo y anticipar futuros pedidos.
Si el objetivo del cacheado es reducir el trafico externo de la red se debera tomar una decision sobre como se tratara la politica de las cancelaciones.

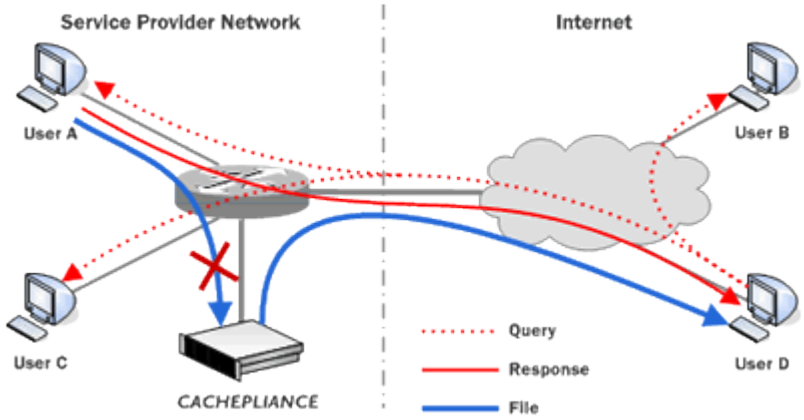
### 2.4.1 Como funciona

Salvando el Downstream
*1. Usuario C envia una query a la red P2P y localiza el archivo requerido en la computadora del usuario B*
*2. Usuario C envia un pedido para bajar el archivo desde el usuario B y el usuario B comienza a*

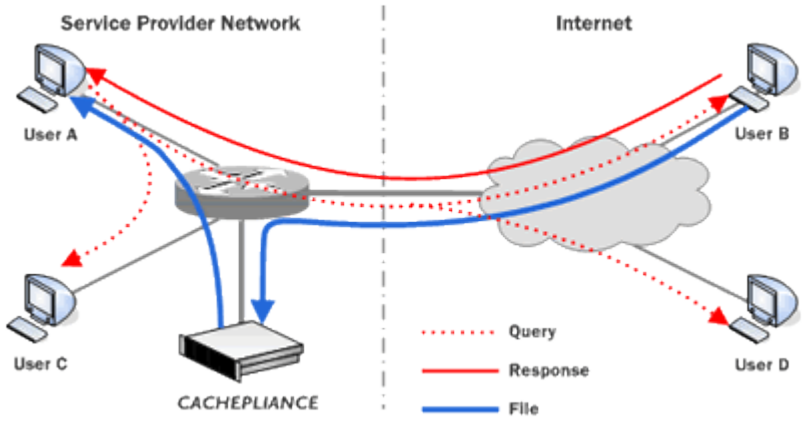
*enviarlo.*
*3. El cache server detecta la transferencia y establece si el archivo se encuentra en cache*
*4. El cache server de manera transparente termina la transferencia desde el usuario B y transmite el archivo al usuario C desde el cache*




Salvando el Upstream
*1. El usuario D envia una query a la red P2P y localiza el archivo deseado en la computadora del usuario A*
*2. Usuario D envia el archivo pedido por el usuario A y A comienza a descargar el archivo.*
*3. El cache server detecta la transferencia y establece si el archivo solicitado se encuentra en cache.*
*4. El cache server termina la conexion de el usuario A de manera transparente y sirve el archivo al usuario D desde el cache*

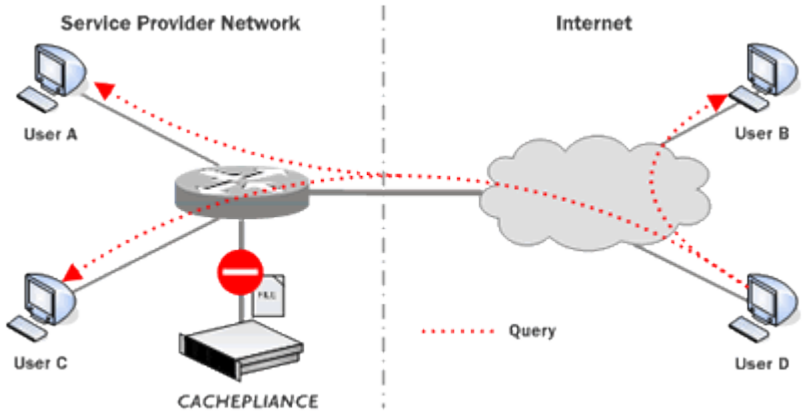



Construyendo la libreria
*1. Usuario A envia una query a la red P2P y localliza el archivo deseado en la computadora del usuario A*
*2. El usuario A sirve el archivo al usuario B y el usuario B comienza a descargarlo.*
*3. El cache server detecta la transferencia y establece si el archivo se encuentra en cache (en este caso no)*
*4. El cache server controla la transferencia entre B y A y simultaneamente salva el archivo en cache*



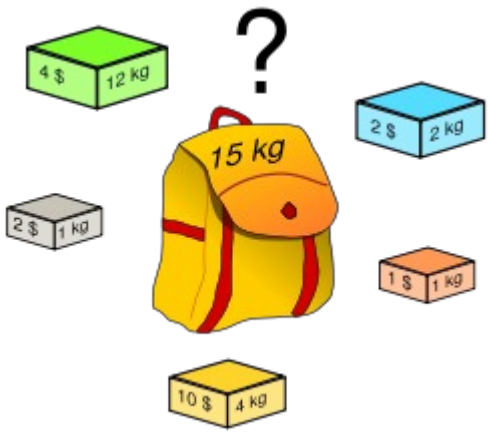

Manteniendo la integridad

*1. El usuario D envia una query a la red P2P para localizar un archivo*
*2. El resultado de la busqueda resulta nulo*
*3. El cache server tiene una copia del archivo solicitado pero como no se ha detectado una transferencia no sirve el archivo en cache*




### 2.4.2 El problema Knapsack

El problema knapsack es un problema de optimizacion de combinaciones. Deriva su nombre del problema de maximizacion de los elementos escenciales que caben en una bolsa a ser cargada por una persona. Dada una serie de items, cada uno con un costo y un valor, determinar el numero de dichos itemnes a ser incluidos en la coleccion tanto asi que el costo es el menor posible para la maxima carga.
La pregunta del problema es: puede un valor de cuando menos X ser alcanzado sin exceder el costo Y?




### 2.4.3 Debe una politica de cacheado tomar los rangos o los archivos?

Una politica de reemplazo del contenido cacheado puede verse como un problema tipo knapsack. El conjunto de archivos tiene que ser maximizado hasta un cierto costo satisfaciendo una cierta utilidad.
Un cache P2P que guarda rangos de archivos en lugar de archivos completos puede beneficiarse de esto, porque los rangos de un archivo son mas pequeños que el archivo completo en si y ofrece una mayor flexibilidad a una politica de reemplazos.
El problema se centra en que los rangos de archivos no son fijos sino variantes y una sola descarga puede incluir el rango de hasta la mitad del archivo.
Esto presenta dos variantes: en una las cancelaciones del cliente son ignoradas y el rango completo del pedido es solicitado. La segunda variante seria guardar el rango del archivo hasta que fue cancelado.
Las cancelaciones por parte del cliente llevan a un incremento del pedido de pequeños rangos de archivos y un incremento por bajar el rango completo del archivo.
Las investigaciones concluyen en que generalmente los rango de pedidos son cortos y piden cualquier rango del archivo. Aparte, las cancelaciones de usuario tienden a incrementar la cantidad de pedidos pequeños.

### 2.4.4 Politicas de reemplazo basicas

Para esta seccion se realizaran algunas comparaciones con un caso mas estudiado: web caching
Una politica de reemplazo generalmente puede ser definido por la regla de comparacion que compara dos itemes cacheados (dos archivos en una politica basada en archivos y dos rangos en una politica basada en rangos). Bajo dicha regla se pueden organizar los objetos de mayor a menor y eliminar el de menor valor.
Cada objeto cacheado (rango o archivo) tiene varios atributos como momento de acceso (ultima vez que fue accedido) o tamaño. Esas caracteristicas son las que usaremos para nuestra politica de reemplazo.
La forma de reemplazo mas simple usando reglas de comparacion es la de Ultimo Recientemente Accedido (LRU: last recently used) y tamaño minimo (MINS: minimun size). Sus negaciones serian Mas Recientemente Accedido (MRU: Most Recently Used) y tamaño maximo (MAXS: maximun size) que tambien seran incluidos en la evaluacion.
Tamaño Doblemente Codicioso (GDS: Greedy-Dual Size) es una politica de reemplazo simple y muy usada en web caching. GDS incorpora una forma simple las caracteristicas mas importantes de un objeto: su historial de acceso, su tamaño y su ultimo acceso.

### 2.4.5 Politicas de Reemplazo Especializado

Las politicas basicas no explotan toda la informacion disponible en el cache. Por ejemplo, una

archivo guardado en cache consiste de varios rangos con saltos entre si. Una importante pieza de informacion es cuanto del archivo total esta guardado en cache. Los objetos guardados pueden tener los siguientes atributos avanzados:
1. Tamaño maximo: *el tamaño maximo de objeto. Para archivos seria mayor que para solo sus rangos.*
2. Bytes transmitidos: *la cantidad de informacion que ha sid enviada a un cliente de este objeto. Esto puede tener en cuenta las cancelaciones de usuario: cuando un objeto esta siendo servido, la cantidad de bytes servidos antes de de la cancelacion del usuario es añadida a la cantidad de bytes transmitidas del objeto.*
3. Tiempo de acceso escalado: *un numero que toma en cuenta la seccion actualizada de un objeto. Cuando un objeto es accedido, el peso entre su ultimo acceso anterior y el presente es medido contra la porcion del objeto que ha sido pedido y enviado y este numero es añadido al tiempo de acceso escalado. Si los pedidos son siempre de objetos completos, como en web caching, es equivalente a una politica de LRU.*

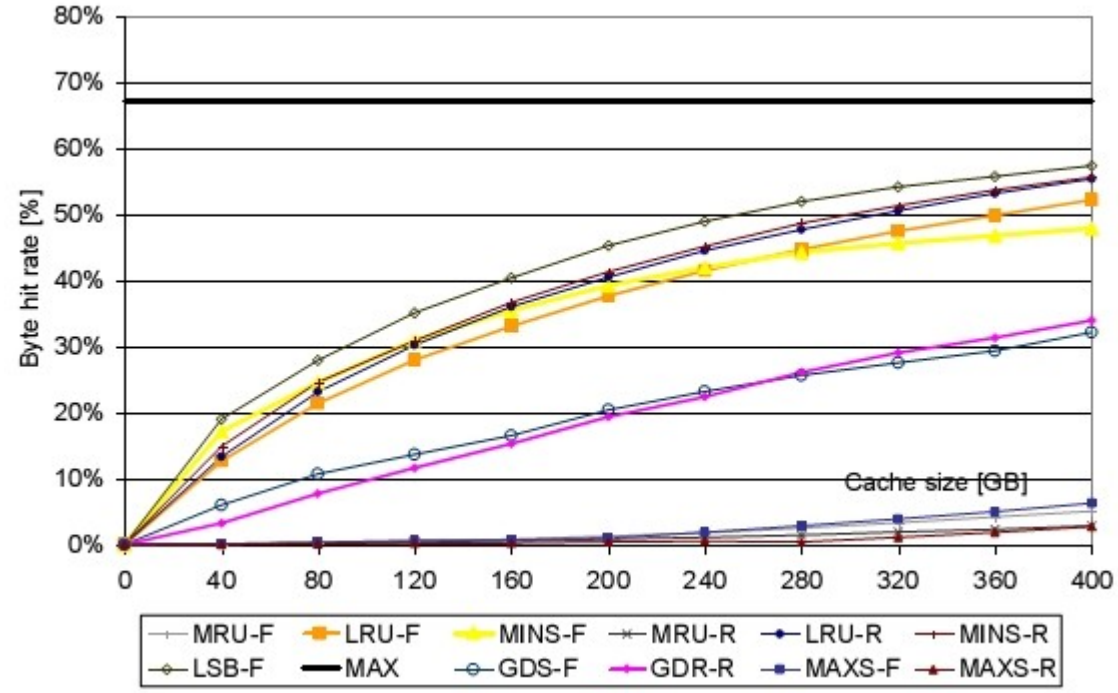
La primer politica especializada a presentar sera basada en archivos que tomara en cuenta la porcion la porcion dle archivo almacenada en cache. Si el cache almaceno la mayor parte del archivo entonces tiene mejores posibilidades de cumplir todos los rangos que le seran solicitados. La politica sera llamada Tamaño Minimo Relativo (MINRS: Minimun Relative Size). Removera del cache ls archivos con el menor grado de cacheado relativo al tamaño completo del archivo.
Otra posibilidad es tomar en cuenta como la mayor parte de los datos son servidos a los clientes a partir de un objeto en cache. Para web caching esto seria equivalente a la politica de uso frecuente (LFU). En cacheo de P2P se deben tomar en cuenta tambien las cancelaciones de los usuario y se debera cambiar el tamaño cando un rango nuevo es añadido. La politica de Bytes Menos Enviados (LSB: Least Sent Bytes) y Bytes Relativamente Menos Enviados (LRSB: Least Relative Sent Bytes) usa los bytes transmitidos de un objeto. Este atributo crece cuando el objeto es servido a un cliente, por la cantidad de bytes descargados antes de que el usuario abortara. LRSB divide esa cantidad por el tamaño maximo del archivo.

#### 2.4.6 Comparacion de Politicas de Reemplazo

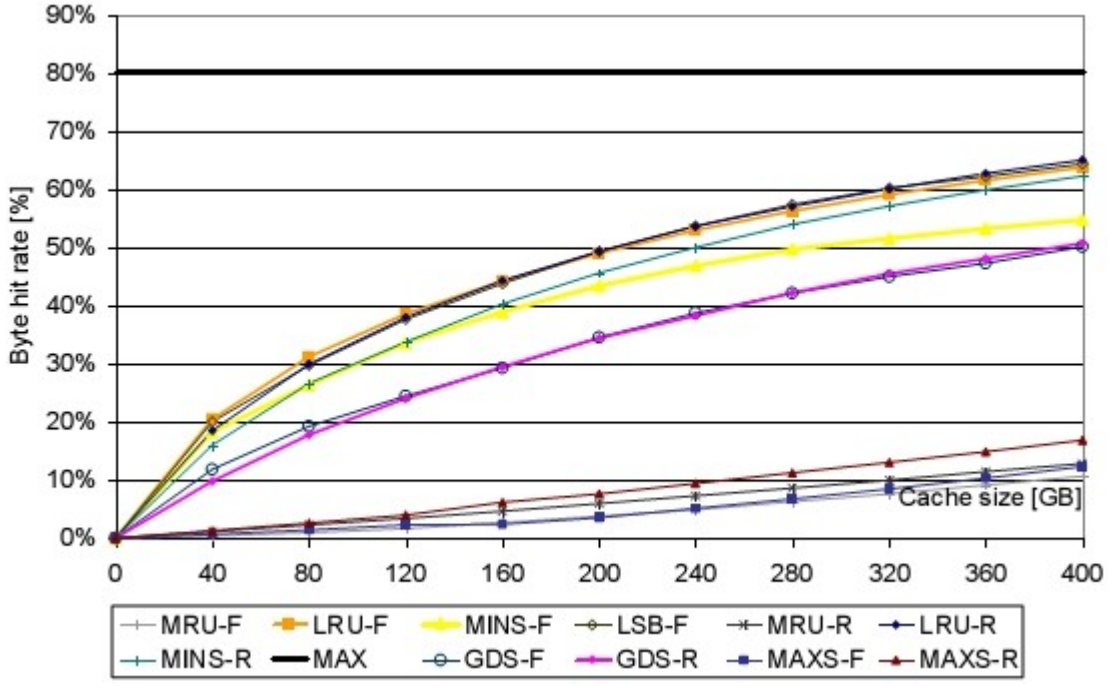
Las graficas presentan el grado de exito de las diferentes politicas de reemplazo para proporcionar la maxima optimizacion del tamaño en cache/trafico solicitado.
El resultado de las comparaciones con objetos completos (sufijo -F) contra los objetos granulados (sufijo -R).
Como resultado vemos que una politica de cacheado de objetos completos muestra una buena performance bajo LRU. Por otro lado, la performance de MINS es sorprendentemente buena, mientras la de MAXS es bastante pobre.
Una explicacion posible es considerar como el cache determina que ocurrio una solicitud y la distribucion de comienzos de rangos de pedido: el cache necesita tener el rango completo en orden para ofrecer elelemento pedido. Tengamos en cuenta que los rangos pedidos estan distribuidos a todo lo largo del archivo por lo tanto las entrada del cache que son grandes tienen una mejor posibilidad de ser servidas. La politica que remueve los objetos mas grandes se comporta de manera pobre mientras que la politica de remover los objetos pequeños trabaja muy bien.
La pobre performance de GDS puede ser explicada de manera similar. GDS prefiere eliminar los objetos grandes por lo pagara la misma multa que MAXS.
MINRS trabaja de manera mas pobre que MINS probablemente porque esta politica puede discriminar el porcentaje de completado del objeto lo cual genera una gran cantidad de bytes en cache. Para elementos grandes los nuevos rangos son relativamente pequeños en comparacion con el tamaño objeto completo y seran removidos primero por MINRS

Para cache completo, la mejor performance fue obtenida por LSB. Esta politica tiene la ventaja de

que considera la informacion disponible por las cancelaciones de usuario. Su buen desempeño indica que existe una localizacion identificable en las cancelaciones del usuario. Probablemente algunos archivos grandes son cancelados con mas frecuencia en peers lentos que otros archivos. El resultado destaca la superioridad de LSB en contraste con los resultados obtenidos por LRU, una politica de frecuencia similar a LFU, y MINS que fue analizado en un cache P2P en vivo. En el estudio, LRU e comporto ligeramente mejor que en la politica basada en frecuencia en el trafico saliente mientras que ambas politicas se comportaron de manera similar en el trafico entrante.




Cacheado completo




Cacheado Parcial

## 2.5 Otras soluciones al problema

#### 2.5.1 Comprar mas ancho de banda

Pareceria la solucion logica, a mayor requirimientos mayor ancho de banda contratado, mas aun si la empresa esta en crecimiento. Extrañamente esta es una de las peores soluciones posibles y solo contiene el problema por un tiempo muy bajo para luego aumentarlo.
Los clientes, al notar el aumento en el ancho de banda, son practicamente alentados a bajar mas con lo que el trafico crece a la par que el ancho de banda aumenta y el problema se mantiene.
Esto sin contar los problemas en costes monetarios para el ISP y el cliente

#### 2.5.2 Bloquear el trafico P2P

Consiste en buscar y cerrar los puertos en el NAP que esten utilizando los nodos P2P.
El objetivo es bajar el consumo impidiendo el uso.
En la practica esto puede resultar muy problematico. Reduce la experiencia del usuario y es una solucion desfasada ya que la gran mayoria de los programas P2P permiten la seleccion dinamica de puertos con el proposito expreso de evitar este tipo de medidas. Estas tecnicas dificultan al ISP poder blokear todo o gran parte del consumo desmedido pero el peor problema es el empobrecimiento del servicio al usuario que puede haber contratado el servicio con el proposito casi exclusivo del uso de este tipo de redes. (O sea, si el 60% de mi consumo es por el P2P, bloquearlo va a causar gran infelicidad en mis clientes)

#### 2.5.3 Utilizar cache de red

Un cacheo del trafico P2P permite al ISP mantener cierto cache del material mas solicitado de manera local sin empobrecer la experiencia del usuario ni consumiendo recursos excesivamente.
Cuando un usuario realiza una query solicitando ciertos datos esta pasa primero por mi cache, si la informacion se encuentra es provista al usuario sin necesidad de salir de mi backbone. Si la informacion no se encuentra aun en cache, se realizara la conexion con los seeds y se transmitira al usuario a la vez de realizar una copia dentro del cache.
Esta solucion minimiza de manera importante el consumo del ancho de banda manteniendo este consumo dentro de mi red local lo mas posible y mas importante aun: deja la experiencia del usuario sin tocar si no es que la mejora.
Aun con estas ventajas el cacheado es controversial ya que el ISP estaria ofreciendo muy probablemente contenido ilegal lo que hara que un proveedor piense entre mantener la calidad de su servicio al coste de posibles costes en abogados.
Otro problema del cacheado es que no hace nada por solucionar la congestion causada por el upstream. Los usuarios de redes externas siguen consumiendo los recursos de nuestra red mientras bajan contenido de nuestros suscriptores.

#### 2.5.4 Implemetar limites de consumo

Un limite de consumo produce un sobreprecio (en caso de exceso) sobre el consumo total del cliente (usualmente medido en GB).
Este tipo de planteamiento limita el consumo del suscriptor a una cierta cantidad mensual pudiendo separar a los clientes en diferentes capas (planes) de acuerdo a su consumo.
A los niveles mas bajos se les provee lo suficiente para navegar y utilizar su mail. El alto trafico de un usuario P2P requerira de un mayor consumo mensual y pagara una cuota mas cara. Cobrando por consumo el ISP puede recuperar algo del costo adicional causado por los usuarios con alto trafico. El costo de sobrepasar el limite de consumo desalienta a la mayor parte de los suscriptores de excederse de su capa.
Aun asi, esto deja el upstream sin ser solucionado y la red continua congestionada, aunque limitada.
Adicionalmente, el limite de consumo no enfrenta el problema del sobretrafico P2P, solo limita su accion a un consumo determinado. La objecion mas importante podria saltar sobre el hecho de que un usuario no siempre esta consiente de la manera en que utiliza su ancho de banda y no sabe como limitar (tecnicamente) su download y upload de las redes P2P con el costo adicional el RP para el ISP. ("Mi ISP me cobra por un consumo que no realice y me esta robando","Yo no pude consumir

tanto", etc)

#### 2.5.5 Shapear el trafico P2P

Shapear se refiere al procesar, buferear y priorizar el trafico atravesando el NAP. Esto permite al ISP darle una prioridad alta a el trafico no P2P, dejando el ancho de banda restante a este tipo de red. Cada paquete es analizado dentro del NAP y clasificado basado en ciertos parametro encontrados en el paquete. Basado en la prioridad dada a cada categoria de trafico, el paquete entra en una cola de espera para ser transmitido. En un ambiente de shapeo P2P, los paquetes de esta red son enviados ultimos consumiendo el sobrante del ancho de banda priorizando al resto.
Ciertamente tiene sus ventajas; un ISP puede ganar cierto control sobre su red reduciendo los costos del trafico P2P hasta el punto de blokearlo totalmente (un kb por hora para este tipo de paquetes). Aun asi, este tipo de tacticas, puede ser evadida implentadas por los softwares clientes disfrazando el pedido de otro tipo de trafico. La comunidad desarrollando el P2P ha desarrollado varias tecnicas para evitar este tipo de shapeo:
* Offset variable dentro del paquete: el identificador no siempre esta en el mismo lugar del paquete todo el tiempo.
* Reversear matematicamente la informacion del paquete: como al equilibrar una equacion se cambia de lado y se cambia de signo sin perder la logica de la informacion. De esta manera se puede enmascarar la naturaleza del paquete
* Contatenar el identificador en varias conexiones
* Contatenar el identificador dentro de multiples paquetes: el identificador del paquete solo puede ser obtenido luego de examinar varios paquetes
* Identificador artificialmente repartido defragmentando los paquetes.

Cada una de estas acciones pueden ser implementadas para evadir el shapeo de manera bastante efectiva.
Aun pudiendo detectar exitosamente el 100% de los paquetes existe una desventaja significativa. Empobrecer la experiencia de los usuarios de toda la red y no solo a los de P2P.
El shapeo requiere inspeccionar cada paquete atravesando el NAP para determinar su prioridad. Esto introduce una latencia extra en la red y un tiempo de procesamiento mas alto mientras cada paquete es inspeccionado, categorizado y colocado en la cola correcta de espera.

#### 2.5.6 Utilizar una Politica de manejo por estados

La politica de manejo de estados (Stateful Polict Management) es una tecnica que controla tanto el downstream como el upstream P2P. Del lado del downstream, redirecciona el trafico atraves de la red por el camino menos costoso, mientras el upstream es controlado por un limite de conexiones externas a la red.
Este se logra con una profunda inspeccion del paquete. El termino "stateful" se refiere a una aplicacion y conexion a la cual se le debe mantener el rastro en el tiempo como informacion especifica para identificar el protocolo y las subsecuentes acciones y operacion a ser tomadas en el protocolo. La inspeccion de estados es crucial para una inteligente clasificacion y ruteado del trafico ya que provee un entendimiento panoramico de la conversacion entre ambos clientes.
Esta conciencia sobre el layer-7 facilita el redireccionado del flujo del trafico P2P. El agente de redireccion guarda el rastro de todo el trafico plano, consistente generalmente de querys, envio de pedidos, envio de terminaciones y lista de contenido compartido. Como el agente "escucha" la conversacion P2P, el redireccionamiento puede facilitar el trafico entre suscriptores en lugar de dejar al cliente conectarse a cualquiera en la red al azar.
Aun sufre el problema de la latencia por el bufereado, inspeccion, proceso y re-enrutamiento de las conexiones pero cuando menos no afecta la experiencia del cliente.

## 3. Referencias

http://www.cachelogic.com/

**Characterizing Peer-to-Peer Traffic across Internet** - *Yunfei Zhang, Lianhong Lei, Changjia Chen (School of Electronics and Information Engineering Beijing Jiaotong University, Beijing, China, 100044)*

**Cache Replacement Policies Revisited: The Case of P2P Traffic** - *Adam Wierzbicki, Nathaniel Leibowitz, Matei Ripeanu, Rafal Woniak*

**Meeting the Challenge of Today's Evasive P2P Traffic: Service Provider Strategies for Managing P2P Filesharing** - *Sandvine Incorporated*

**Cross-Layer Peer-to-Peer Traffic Identification and Optimization Based on Active Networking** - *I. Dedinski, H. De Meer , L. Han , L. Mathy , D. P. Pezaros , J. S. Sventek , Z. Xiaoying (Department of Mathematics and Computer Science University of Passau - Passau, Germany / Computing Department Lancaster University - Lancaster, UK / Department of Computing Science University of Glasgow - Scotland, UK )*

**World Wide Web Acceleration: Caching Techniques and Peer-to-Peer networks** - *Christoph Olbrich*

**Are file swapping networks cacheable?: characterizing p2p traffic** - *Nathaniel Leibowitz, Aviv Bergman, Roy Ben-Shaul, Aviv Shavit (Expand Networks, Tel-Aviv, Israel)*

**Transport Layer Identification of P2P Traffic** - *Thomas Karagiannis (UC Riverside) Andre Broido (CAIDA,SDSC) Michalis Faloutsos (UC Riverside) Kc claffy(CAIDA,SDSC)*