UNIVERSIDAD CÁTOLICA DE TEMUCO

FACULTAD DE CIENCIAS

TRANSVERSABILIDAD EN NAT/FIREWALL

por

Heinz Waldemar Herlitz Gatica

Trabajo de Título presentado a la
Facultad de Ciencias de la Universidad Católica de Temuco
Para Optar al Título de Ingeniero de Ejecución en Informática.

-Temuco, 2005 -

UNIVERSIDAD CATOLICA DE TEMUCO
FACULTAD DE INGENIERIA

COMISIÓN EXAMEN DE TÍTULO

Este Examen de Título ha sido realizado en la Escuela de Informática

Presidente Comisión: .....
Sr. Oriel Herrera Gamboa
Director de Escuela de
Ingeniería Informática

Profesor Guía: .....
Sr. Alejandro Mellado Gatica
Ingeniero de Ejecución en Informática
Magíster en Telecomunicaciones

Profesor Informante: .....
Sr. Luis Alberto Caro Saldivia
Ingeniero Civil en Informática

Secretario Académico .....
de la Escuela de Informática
Sr. Luis Alberto Caro Saldivia
Ingeniero Civil en Informática

Temuco, Septiembre de 2005

INFORME TRABAJO DE TÍTULO

TÍTULO : "Transversabilidad en NAT/Firewalls"
ALUMNO : Heinz Waldemar Herlitz Gatica

En mi condición de profesor guía de este trabajo puedo efectuar las siguientes observaciones:

- El trabajo está bien estructurado y se ha usado distintos elementos de acuerdo las necesidades de un prototipo.
- El presente trabajo es de alta complejidad y demuestra un manejo técnico avanzado.
- El trabajo se ha escrito de manera precisa y concisa.
- Es necesario que adjunte como anexo, fragmentos del código fuente para complementar el trabajo.

De acuerdo a estas consideraciones califico este trabajo con 7.0 (Siete coma cero)

_____

Alejandro Mauricio Mellado Gatica
Profesor Guía

Temuco, 19 de Agosto de 2005

INFORME TRABAJO DE TÍTULO

TÍTULO : "Transversabilidad en NAT/Firewalls"

ALUMNO : Heinz Waldemar Herlitz Gatica

En mi condición de profesor informante de este trabajo puedo efectuar las siguientes observaciones:

- El trabajo esta ordenado y bien estructurado y se han usado herramientas metodológicas adecuadas.
- Se aprecia un buen nivel de dedicación y un profundo dominio sobre el tema.
- La complejidad del trabajo es de un nivel avanzado, no obstante, el alumno logra motivar y enfocar su trabajo de manera amena y clara.

De acuerdo a estas consideraciones califico este trabajo con nota 7.0 (Siete coma cero)

_____

Luis Alberto Caro Saldivia
Profesor Informante

Temuco, 19 de Agosto de 2005

Dedicatoria

A mis Padres

Alex y Jeanette

ÍNDICE

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

Resumen

Día a día la crece la cantidad de usuarios que acceden a redes P2P (Peer to Peer) por medio de conexiones con un gran ancho de banda. Estas redes P2P ya no solo se utilizan para intercambio de archivos, sino que además se utilizan para la transmisión de VoIP (Voz sobre IP), MI (Mensajería Instantánea), videoconferencia, entre muchas otras.

Uno de los problemas más graves a los que se enfrenta un sistema P2P es la existencia de Firewalls y sobre todo, de entornos NAT (Network Address Translation), que son dispositivos que proveen de conexión a Internet a un gran número de equipos a través de una sola dirección.

En este trabajo se describen los protocolos estándares que actúan y el funcionamiento e implementación de conexiones P2P a través de NAT/Firewalls utilizando transversabilidad. Describiendo de una manera gráfica y detallada distintas técnicas para lograr la solución, mostrando además ventajas y desventajas de la solución.

Capitulo 1

1.1 Introducción

El uso de mecanismos NAT, (traducción de direcciones) permite usar una sola IP pública para toda una red privada. Pero esta práctica imposibilita el uso de muchas aplicaciones, que quedan relegadas a su uso en Intranets, dado que muchos protocolos son incapaces de atravesar los dispositivos NAT. Protocolos como RTP y RTCP (Protocolo de Transporte en Tiempo Real y Protocolo de Control en Tiempo Real) usan UDP con asignación dinámica de puertos (NAT no soporta esta traslación). Estos protocolos se utilizan para la transmisión de sonido y vídeo, muy utilizados hoy en día por un sinnúmero de empresas para realizar videoconferencias con otras sucursales.

Al buscar una solución para crear conexiones P2P a través de Firewalls con NAT, son muchas las alternativas que existen. Pero la idea principal es crear este tipo de conexiones sin modificar configuraciones en los firewalls, y además no entrar en gastos de nuevas tecnologías que utilicen elementos intermedios.

Este proyecto busca dar solución a la transversabilidad en firewalls, de forma en que solamente se necesite un servidor con IP pública, y una aplicación P2P que realice la transversabilidad. Bajo este esquema otras soluciones quedan fuera del tema por razones de costo e implementación y serán solamente mencionadas.

Para realizar esta transversabilidad es necesario analizar el funcionamiento de las redes P2P, los software orientados a esta tecnología, protocolos estándares de conectividad, tipos de NAT usados y métodos de transversabilidad existentes.

1.2 Objetivo General

Implementar la Transversabilidad en Firewalls, de manera que se pueda enviar un archivo mediante una conexión P2P, a través de firewalls o dispositivos que realicen NAT. Todo esto sin cambiar configuraciones en los Firewalls.

1.3 Objetivos Específicos

- Realizar un estudio de las redes P2P.
- Realizar un estudio de Traducción de Direcciones.
- Investigar los distintos métodos existentes actualmente para realizar transversabilidad.
- Analizar la estructura y funcionamiento de Protocolos Involucrados.
- Estudios de otros proyectos de transversabilidad relacionados.
- Implementar la Transversabilidad.

1.4 Justificación

Actualmente estamos bajo el funcionamiento de IPv4, con lo que tenemos 4,294,967,296 direcciones IP distintas para ser utilizadas públicamente. Aunque pareciera ser una cantidad suficiente para el total de computadoras y otros aparatos que necesitarían una dirección IP, el hecho es que estas han sido repartidas en diversas clases, dominios y usos, lo cual reduce las posibilidades de que cualquier persona pueda asignar una dirección IP pública, es decir identificada por el resto de Internet.

La solución es el protocolo de nueva generación IPv6, que otorga mucha mayor cantidad de direcciones, pero aún deberá pasar mucho tiempo para que se realicen las modificaciones necesarias en las redes de manera que sean compatibles con esta norma. Mientras tanto NAT es una alternativa previa al uso de IPv6, pero la utilización de NAT genera que muchas aplicaciones que trabajan bajo redes P2P no funcionen adecuadamente. Esto debido a que un NAT implica, en cierta medida, colocar un Firewall que protege a la red local, ya que al no saber los equipos externos la dirección privada de los sistemas internos, no solicita la conexión. Todo enlace hacia Internet deberá generarse bajo petición de los equipos locales.

En videoconferencia por ejemplo detrás de un NAT es el equipo local el que debe iniciar la llamada, cuando el sistema remoto contesta, es el equipo NAT quien le responde, y al no tener capacidades de negociación (H.323 para este caso), la llamada termina.

Dentro de las posibles soluciones a esta problemática expuesta tenemos:

- Ubicar un sistema que responda a las conexiones entrantes en la Zona Desmilitarizada (DMZ).
- Configurar el equipo NAT para enviar las peticiones entrantes en ciertos puertos hacia la IP privada del equipo local.
- Utilizar servidores duales (requiere inversión).

Todas estas soluciones involucran la adquisición de equipamiento, por lo tanto inversión, y configuración de equipos. La solución que se propone pretende posibilitar el uso de aplicaciones P2P de un modo seguro y transparente, facilitar la conectividad P2P a través de NAT sin configuraciones especiales , y permitir la construcción de futuros software P2P que funcionen en entornos con NAT.

Capitulo 2

2.1 Traducción de direcciones (NAT)

El crecimiento actual de las redes y los masivos cambios en seguridad han creado nuevas vías que hacen difícil el funcionamiento de algunas aplicaciones. La arquitectura original de Internet donde cada equipo se podía comunicar directamente con otro utilizando una IP única y global, ha sido remplazada por una nueva arquitectura de direcciones, consistente en una dirección global y muchas direcciones privadas interconectadas por NAT [1].

En esta nueva arquitectura como se ve en la figura 1, sólo los equipos que tengan dirección pública pueden ser fácilmente conectados por otros en la red, ya que tienen una única, global y ruteable dirección IP. Los equipos de las redes privadas pueden conectarse entre sí, o pueden establecer conexiones TCP o UDP con equipos que pertenezcan a la red pública.

Figura 1: Esquema de conectividad en Internet

Los dispositivos que realizan el NAT traducen la dirección y puertos de los paquetes que provienen de las redes privadas hacia las redes públicas [1], por lo tanto los dispositivos NAT generalmente permite sólo conexiones salientes y bloquea cualquier trafico entrante, a no ser que este configurado específicamente de otra manera.

Esta nueva arquitectura de direcciones permite la comunicación cliente / servidor en el caso típico donde el cliente se encuentra en una red privada y el servidor se encuentra en la red pública. Esta arquitectura hace difícil la comunicación directa entre dos equipos (que es importante para las redes P2P) que pertenezcan a distintas redes privadas. Con esto tenemos que muchas aplicaciones P2P que comparten archivos, realizan videoconferencia, y juegos en línea que no funcionan a través de entornos con NAT.

2.2 Cómo funciona el NAT

Cuando un cliente en la red interna contacta a un equipo público, envía paquetes IP destinados a ese equipo. Estos paquetes contienen toda la información de direccionamiento necesaria para que puedan ser llevados a su destino. NAT se encarga de estas piezas de información:

- Dirección IP de origen (Ej:192.168.0.1)
- Puerto TCP o UDP de origen (Ej: 12345)

Cuando los paquetes pasan a través de la pasarela de NAT, son modificados para que parezca que se han originado y provienen de la misma pasarela de NAT. La pasarela de NAT registra los cambios que realiza en su *tabla de estado*, para así poder:

a) Invertir los cambios en los paquetes devueltos.

b) Asegurarse de que los paquetes devueltos pasen a través del cortafuego.

Por ejemplo, podrían ocurrir los siguientes cambios:

- IP de origen: sustituida con la dirección externa de la pasarela (Ej:216.155.76.8)
- Puerto de origen: sustituido con un puerto no en uso de la pasarela. (Ej: 50050)

Ni la máquina interna ni el otro equipo de Internet se dan cuenta de estos pasos de traducción. Para el equipo interno, el sistema NAT es simplemente una pasarela a Internet. Para el equipo público, los paquetes parecen venir directamente del sistema NAT; ni siquiera se da cuenta de que existe la estación interna. Cuando el anfitrión de Internet responde a los paquetes internos de la máquina, los direcciona a la IP externa de la pasarela NAT (216.155.76.8) y a su puerto de traducción (50050). La pasarela de NAT busca entonces en la tabla de estado para determinar si los paquetes de respuesta concuerdan con alguna conexión establecida.

2.3 Tipos de NAT

Existen cuatro tipos de NAT [1]. Para las direcciones internas los tres primeros tipos de NAT mantienen una traducción de dirección independiente de la dirección de destino. El cuarto tipo de NAT creará una traducción independiente por cada conexión con el destinatario.

A menos que el NAT posea una tabla de traducción estática, la traducción se inicia cuando el primer paquete es enviado desde el cliente a través del NAT y éste será válido por una cierta cantidad de tiempo (generalmente un par de minutos), a menos que se sigan enviando paquetes a esa IP y puerto.

2.3.1 Full Conexión

En el caso de la conexión completa o llena, (Full Cone) ver figura 2, cuando la traducción se encuentra establecida, cualquier equipo que quiera alcanzar al cliente detrás del NAT, necesita sólo conocer la dirección y el puerto de donde el tráfico está siendo enviado. Por ejemplo un equipo detrás de un NAT con dirección 192.168.0.1 enviando y recibiendo en el puerto 5000, se ha traducido a la dirección externa 216.155.76.8 en el puerto 12345. Cualquier equipo de Internet puede enviar tráfico a esta IP externa y este tráfico será traspasado a la dirección cliente 192.168.0.1:5000.

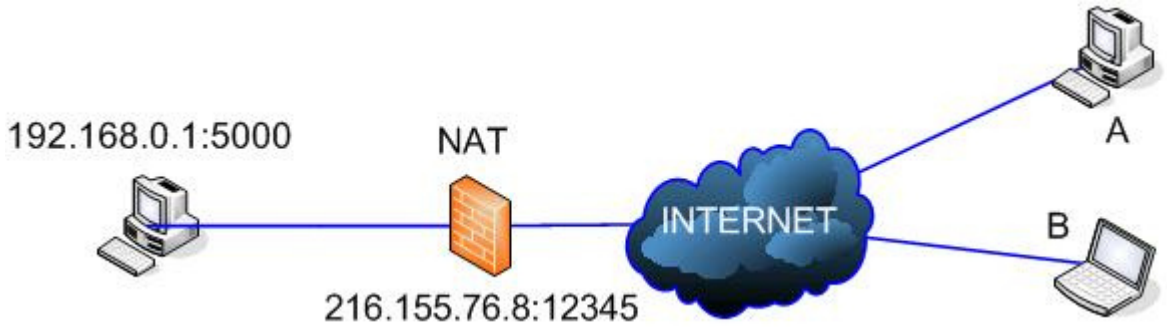

Figura 2: NAT Full Conexión

2.3.2 Conexión Restringida

En este caso de la conexión restringida, (Restricted Cone) ver figura 3, la IP y puerto externo son abiertos cuando el equipo de la red privada envía tráfico saliente a una dirección IP específica . Por ejemplo si el cliente envía paquetes hacia el equipo A, el NAT traduce la dirección privada 192.168.0.1:5000 a la dirección pública 216.155.76.8:12345, dejando que solamente el equipo A pueda enviar tráfico a esa dirección pública. El NAT bloqueará cualquier otro tráfico que venga de una dirección distinta.

Por lo tanto el equipo B podrá enviar tráfico hacia la red privada, solamente si antes el equipo que se encuentra detrás del NAT a enviado tráfico a este equipo B.

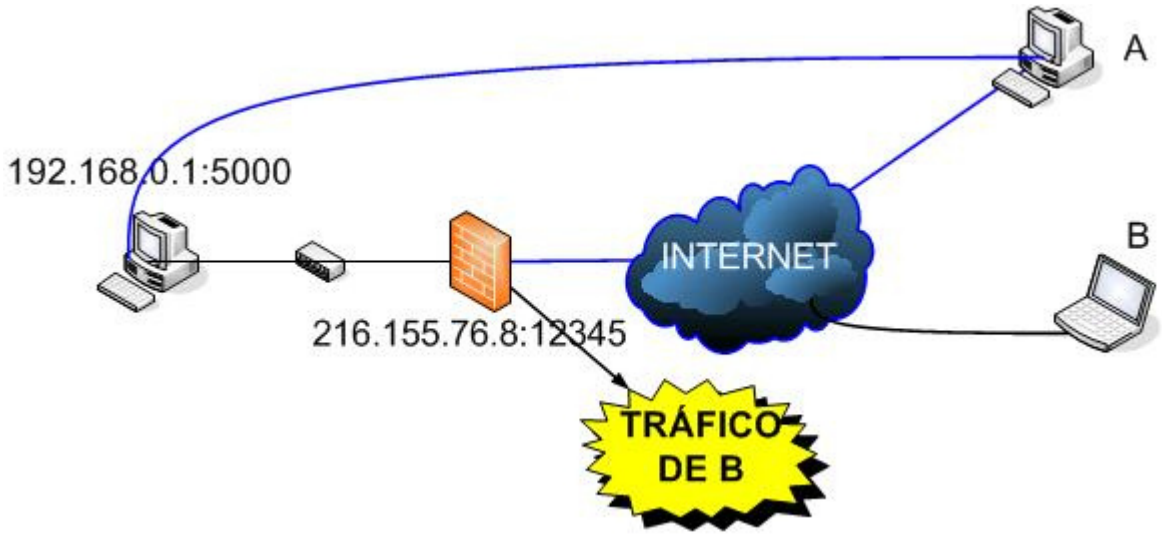

Figura 3 : NAT Conexión restringida

2.3.3 Conexión Restringida por Puerto

Una conexión restringida por puerto, (Port Restricted Cone) ver figura 4, es casi idéntica a la conexión restringida, pero en este caso el NAT bloqueará todo el tráfico, a menos que el cliente haya enviado antes tráfico a una IP y *puerto especifico,* solo entonces esa IP:PUERTO tendrán acceso a la red privada. Entonces en nuestro ejemplo si el cliente envía tráfico al equipo B puerto 10101, el NAT sólo permitirá tráfico proveniente desde 200.72.40.55:10101, ahora si el cliente envía a múltiples direcciones y puertos entonces estas direcciones podrán responder al cliente a la dirección 216.155.76.8:12345.

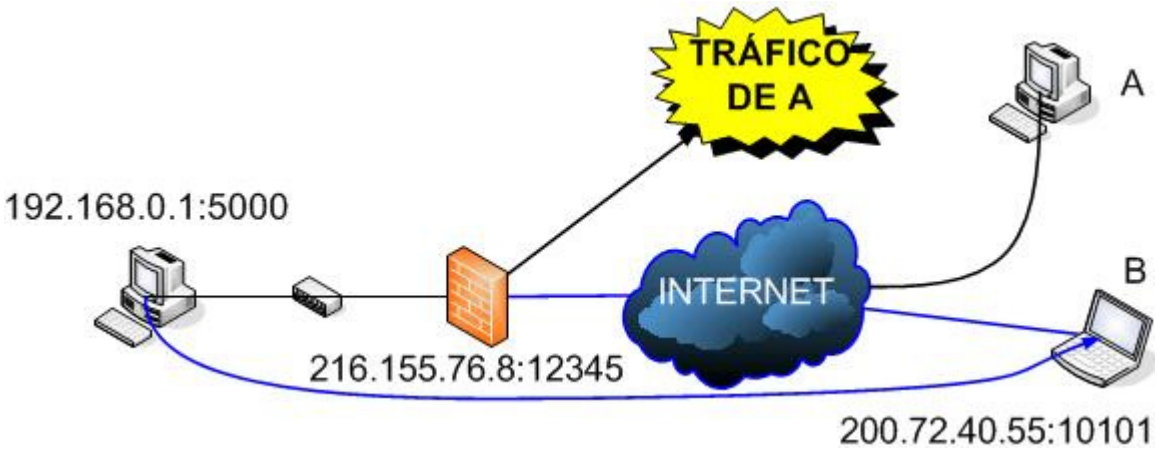

Figura 4 : Conexión restringida por puerto

2.3.4 Simétrico

El último tipo de NAT simétrico, (Symmetric NAT) ver figura 5, es diferente de los otros tres. Específicamente debido a que la traducción de la IP pública a la privada depende de la IP de destino donde ha sido enviado el tráfico. Para nuestro ejemplo si el cliente envía tráfico desde la dirección privada 192.168.0.1:5000 al equipo B, la dirección traducida sería la 216.155.76.8 con el puerto 12345, y si el equipo privado envía tráfico a otra IP pública distinta la dirección traducida para ese equipo sería la 216.155.76.8 con el puerto 45678.

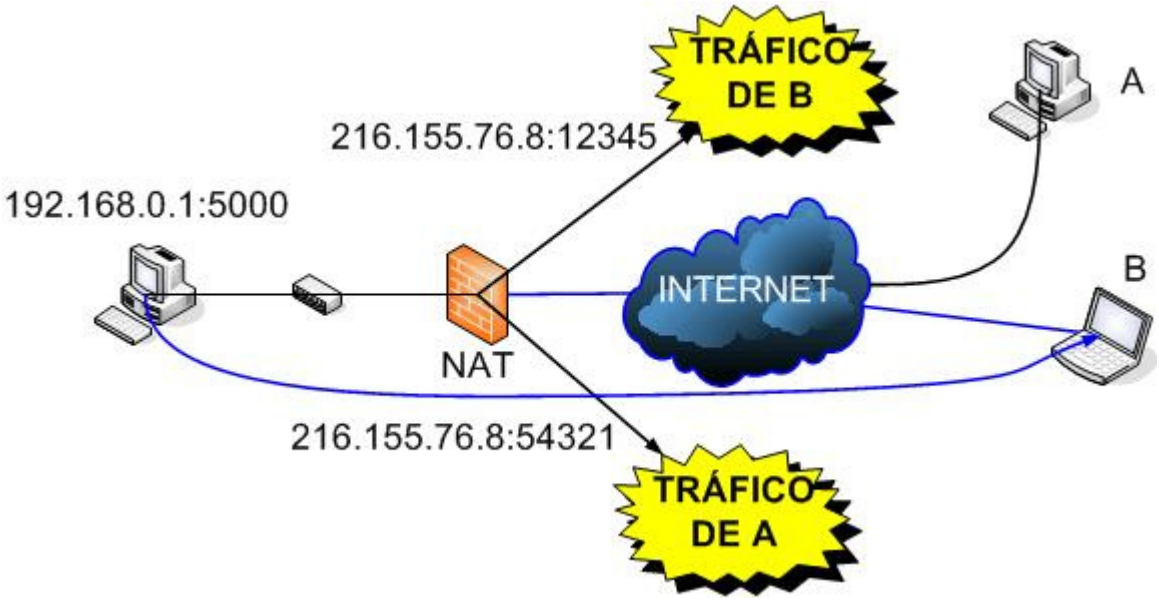

Figura 5: NAT Simétrico

El equipo B puede solamente responder a su dirección traducida (216.155.76.8:12345) y el equipo A a su dirección (216.155.76.8:45678). Si cualquier otro equipo envía tráfico a cualquiera de estas dos direcciones éste será rechazado.

Capitulo 3

3.1 STUN

STUN (Simple Traversal of UDP Through NAT) es un protocolo liviano que permite a las aplicaciones descubrir la presencia y tipos de NAT y/o Firewalls entre ellos y la Internet [2]. También provee a las aplicaciones la determinación de la dirección IP pública asignada por el dispositivo NAT. STUN trabaja con muchos NAT existentes, y no requiere ningún comportamiento especial. Como resultado se obtiene una red variada de aplicaciones que pueden funcionar a través de la infraestructura NAT.

Esta compuesto por un programa STUN servidor que tiene que estar disponible en la Internet, y por el cliente STUN que tiene que ser alojado dentro de la red que se encuentra tras el dispositivo NAT o Firewall. El cliente STUN es una entidad que genera peticiones STUN, y este cliente puede ser utilizado por un PC, o puede ser ejecutado como elemento de red, como por ejemplo un servidor de conferencias. El servidor STUN es una entidad que recibe peticiones STUN y envía respuestas STUN.

Dentro de las principales características a destacar de STUN tenemos:

- STUN activa un dispositivo para encontrar su dirección pública y tipo de NAT que da acceso a Internet.
- STUN opera en los puertos TCP y UDP 3478.
- STUN aún no es soportado por todos los tipos de dispositivos VoIP.
- STUN puede usar registros DNS SRV para encontrar servidores STUN unidos al dominio.
- El nombre del servicio es _stun._udp o _stun._tcp

3.2 Funcionamiento de STUN

Las peticiones STUN especifican los siguientes parámetros:

- Dirección de Respuesta
- IP Cambiada
- Puerto Cambiado

El servidor STUN enviará su respuesta a la dirección IP y puerto especificado en la dirección de respuesta. Si el campo no se encuentra definido, entonces el servidor enviará la respuesta a la dirección IP y puerto de donde recibió la petición. Si la dirección IP de cambio y el puerto de cambio no están definidos, el servidor STUN responderá desde la dirección IP y puerto donde el paquete inicial fue enviado. Si la dirección IP de cambio está definida, el servidor responderá desde una dirección IP diferente, y si el puerto de cambio está definido, el servidor responderá desde un puerto diferente. La respuesta STUN contiene la siguiente información:

- Dirección mapeada: La dirección IP y puerto del cliente que es vista por el primer servidor STUN afuera del NAT, una vez recibida la petición STUN.
- Dirección cambiada: La dirección IP que debería ser fuente de la respuesta recibida, si la petición tiene activado el flag de cambio de dirección IP.
- Dirección Fuente: La dirección IP y puerto donde la respuesta STUN fue enviada.

Usando una combinación de diferentes peticiones hacia el servidor STUN, un cliente puede determinar si el equipo se encuentra disponible a la Internet, si el equipo se encuentra detrás de un Firewall que bloquea UDP, si el equipo se encuentra detrás de un NAT y que tipo de NAT está detrás.

3.3 Detección automática del entorno NAT

Cuatro pruebas son requeridas para determinar en que situación se encuentra el cliente [3]. La siguiente tabla muestra los parámetros asignados y las respuestas que son esperadas por cada prueba. Asumiendo que existen dos servidores STUN disponibles, la dirección IP1 y IP2, y que pueden retornar respuestas desde los puertos 1 y 2.

Prueba	Destino	Cambia IP	Cambia Puerto	IP:Puerto Retorno
TEST I	IP1:1	N	N	IP1:1
TEST II	IP1:1	S	S	IP2:2
TEST III	IP2:1	N	N	IP2:1
TEST IV	IP1:1	N	S	IP1:2

Tabla 1: Pruebas para detectar tipo de NAT

En la figura 6 se describe gráficamente la secuencia que utiliza el programa STUN servidor para detectar el tipo de NAT en el equipo cliente.

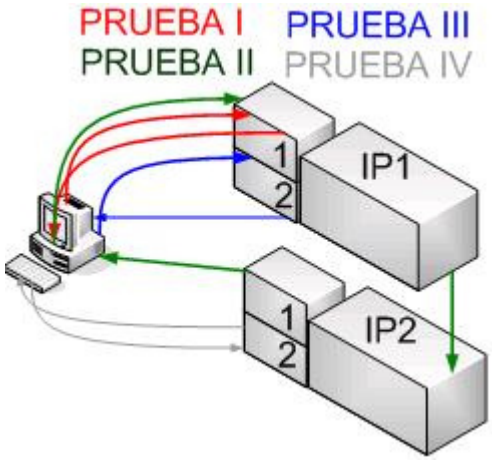

Figura 6: Secuencia de STUN para descubrir el NAT

En orden para descubrir el NAT de cliente, estas cuatro pruebas son desarrolladas de acuerdo a lo siguiente:

1. La prueba I es desarrollada, si no existe una respuesta recibida, entonces el cliente conoce que esta detrás de un Firewall que bloquea UDP.
2. Si existe una respuesta recibida, la dirección IP en el campo de dirección mapeada de la respuesta STUN, es testeada con la dirección que el cliente piensa que tiene.
3. Si la dirección coincide, la prueba II es ejecutada. Si no existe respuesta, el cliente se encuentra detrás de un NAT simétrico que permite UDP. La dirección se encuentra disponible en Internet, pero el Firewall solo permitirá tráfico UDP a un destino dado, una vez que el paquete es enviado fuera del destino. Si el cliente recibe una respuesta, entonces el cliente sabe que esta abierto a la Internet sin bloqueo.
4.Si la dirección IP en el paso 2 no es la misma, entonces la prueba II es ejecutada. Si el cliente recibe respuesta, entonces se encuentra detrás de un NAT FULL conexión .
5. Si no hay respuesta, el cliente ejecuta la prueba III y verifica la dirección IP que es retornada en la respuesta STUN en el campo de dirección mapeada, (proveniente de la IP2) contra la dirección mapeada que es retornada en la prueba I (proveniente de la IP1). Si las dos direcciones no son las mismas, entonces el cliente se encuentra detrás de un NAT simétrico.
6. Si las dos direcciones son las mismas, el cliente ejecuta la prueba IV (cambio puerto). Si la respuesta es recibida, entonces el cliente esta detrás de un NAT restrictivo. Si no existe respuesta, entonces el cliente se encuentra detrás de un NAT restrictivo por Puerto.

3.4 Utilización de STUN

El software STUN es usado por muchas aplicaciones como una herramienta para poder traspasar entornos con NAT, claro que como lo dice su acrónimo, sirve solamente en entornos que permiten el paso del protocolo UDP. Sin embargo se describirá el modo de uso, ya que es un método eficaz para lograr transversabilidad.

3.4.1 Cliente STUN

El cliente STUN bajo Linux se ejecuta colocando solamente la dirección IP del servidor STUN, una salida estándar del comando sería la siguiente:

```
Newton admin/stund> ./client 200.85.207.108
STUN client version 0.92
Primary: Symmetric Nat, random port, no hairpin
Return value is 0xc
```

Esta salida muestra la respuesta recibida desde el servidor. En la segunda línea se muestra la versión del cliente utilizada, la tercera línea indica el tipo de NAT en que se encuentra presente delante del equipo, en nuestro caso NAT Simétrico, finalmente es recibido un valor de retorno desde el servidor STUN.

3.4.2 Servidor STUN

El servidor STUN bajo Linux requiere dos interfaces de red, con sus respectivas direcciones IP públicas configuradas.

Si se tiene una sola dirección asignada se puede asignar la nueva dirección con el comando:

```
/sbin/ifconfig eth0:1 200.85.207.108 netmask 255.255.255.248 up
```

Una vez cuando las dos interfaces de red se encuentran arriba se puede ejecutar el comando que levanta el servidor STUN entregándole como parámetros las direcciones IP públicas disponibles en el equipo:

```
[hherlitz@kronos stund]# ./server -v -h 200.85.207.108 -a 200.85.207.109
STUN server version 0.94
Running with on interface 200.85.207.108:3478 with alternate 200.85.207.109:3479
Binding to interface 0x6ccf55c8
Opened port 3478 with fd 3
Binding to interface 0x6ccf55c8
Opened port 3479 with fd 4
Binding to interface 0x6dcf55c8
Opened port 3478 with fd 5
Binding to interface 0x6dcf55c8
Opened port 3479 with fd 6
```

La opción -v activa el "Verbose Mode", la cual permite ver los resultados de las peticiones recibidas desde el cliente, la opción -h asigna la dirección principal a utilizar por el servidor y la opción -a asigna la segunda dirección que el servidor utilizará para verificar el tipo de NAT.

Los resultados que nos entrega la salida del comando son las direcciones IP públicas y el puerto donde se encuentra corriendo el servidor. Después muestra los cuatro enlaces creados (Binding to interface) para las dos interfaces de red en los cuatro puertos (3478 y 3479 para cada interfaz) que necesita para funcionar.

Una vez que el servidor se encuentra funcionando, y recibe una petición STUN desde el cliente el servidor con la opción -v activada muestra la siguiente salida:

```
received on A1:P1
Got a request (len=28) from 200.54.173.254:51446
Received stun message: 28 bytes
ChangeRequest = 2
Request parsed ok
BindRequest does not contain MessageIntegrity
Request is valid:
    flags=2
    changeIp=0
    changePort=1
    from = 200.54.173.254:51446
    respond to = 200.54.173.254:51446
    mapped = 200.54.173.254:51446
Encoding stun message:
Encoding MappedAddress: 200.54.173.254:51446
Encoding SourceAddress: 200.85.207.108:3479
Encoding ChangedAddress: 200.85.207.109:3479
Encoding XorMappedAddress: 205.211.9.230:52499
Encoding ServerName: Vovida.org 0.94
```

Esta salida es solo una de las tantas peticiones que realiza el cliente STUN hacia el servidor, donde se puede apreciar la información que es recibida por el servidor, entre las que tenemos; dirección de origen, puerto, tamaño del mensaje STUN, tipos de flag activados (que son necesarios para determinar el tipo de NAT) y direcciones IP involucradas. Una vez que el servidor procesa todos los datos entregados por el cliente, este envía la respuesta al cliente con la información del tipo de NAT que se encuentra delante del cliente.

Capitulo 4

4.1 Introducción al P2P

Una de las clasificaciones de los sistemas de computación es llamada sistema distribuido, existen muchos tipos de sistemas distribuidos, en variadas escalas, como Internet, intranets, redes Lan, etc. Los sistemas distribuidos pueden ser organizados en un modelo P2P o en un modelo cliente servidor, como se ve en la figura 7.

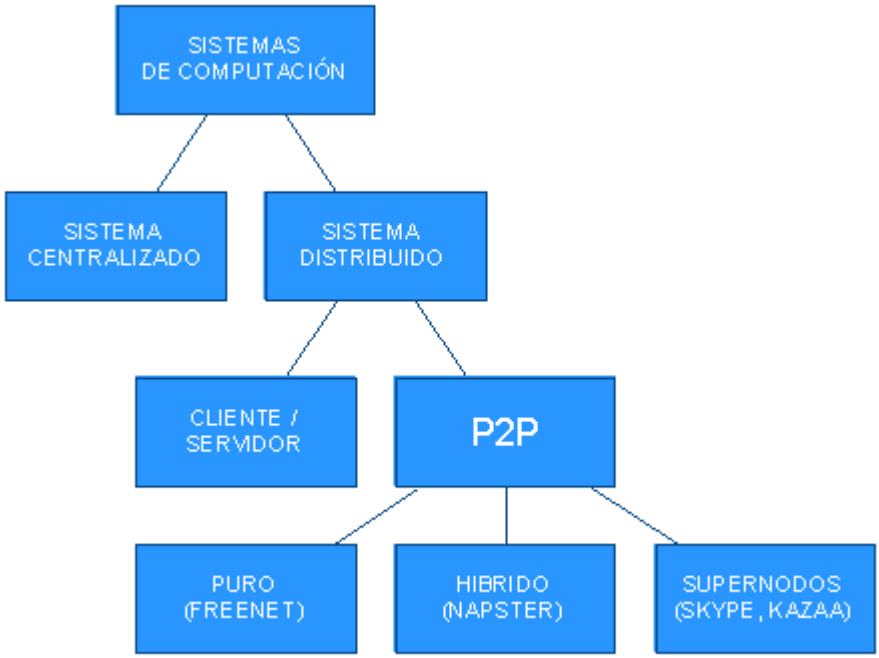

Figura 7: Clasificación de los sistemas de computación.

Dentro del modelo P2P tenemos un sistema P2P puro [4], en el cual no existe un servidor central. También tenemos un sistema P2P híbrido, donde existe un servidor central para obtener ciertos datos como la identidad del equipo a conectar. Finalmente tenemos un sistema P2P de supernodos, donde cada equipo puede aportar con información a otros equipos de la red.

4.2 Peer to peer

El término “peer to peer” (P2P) se refiere a una clase de redes y aplicaciones que emplean recursos distribuidos para desarrollar distintas funciones de una manera descentralizada. En general, es una red informática entre iguales, que no tiene clientes y servidores fijos, sino una serie de equipos que se comportan como clientes y como servidores de los demás equipos de la red. Cualquier equipo puede iniciar o completar una transacción compatible. Los equipos pueden diferir en configuración local, velocidad de proceso, ancho de banda de su conexión a la red y capacidad de almacenamiento. Este modelo es una alternativa al tradicional modelo Cliente/Servidor, ver figura 8.

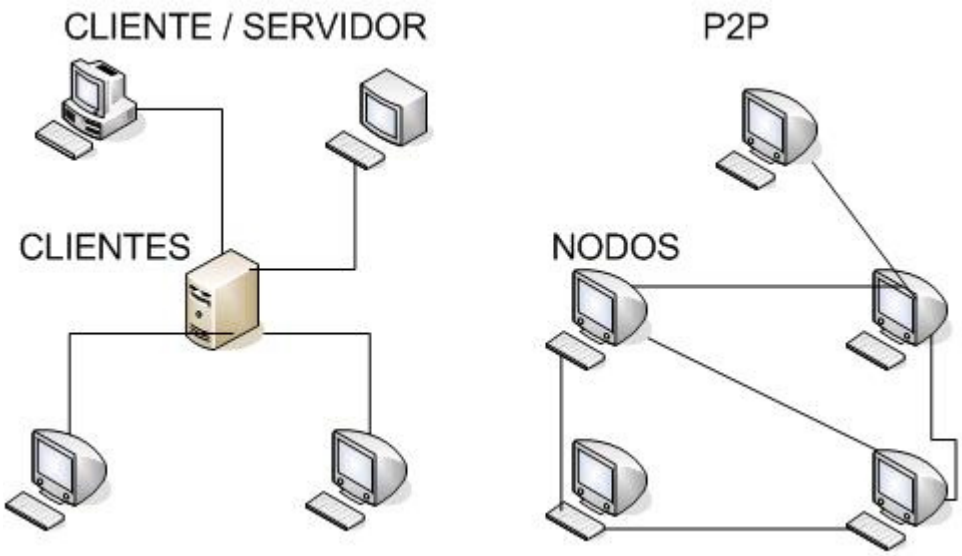

Figura 8 : Modelos P2P y Cliente / Servidor

Debido a que la mayoría de los computadores no tienen una IP fija, sino que le es asignada por el proveedor (ISP) en el momento de conectarse a Internet, no pueden conectarse entre sí porque no saben las direcciones que han de usar de antemano.

La solución habitual es realizar una conexión a un servidor (o servidores) con dirección conocida (normalmente IP fija), que se encarga de mantener la relación de direcciones IP de los clientes de la red, de los demás servidores y habitualmente información adicional, como un índice de la información de que disponen los clientes. Tras esto, los clientes ya tienen información sobre el resto de la red, y pueden intercambiar información entre sí, ya sin intervención de los servidores.

Con el actual crecimiento de las redes y los nuevos computadores, el P2P está recibiendo mucha atención para la investigación y desarrollo, encontrando dentro de los principales beneficios la escalabilidad, la eliminación de puntos centralizados y eliminando la necesidad de costear nueva infraestructura para la comunicación directa entre clientes.

4.3 Beneficios de la comunicación P2P

La comunicación peer to peer hace factible un amplio campo de nuevas capacidades y aplicaciones, como por ejemplo la búsqueda dinámica y distribuida, manejo y almacenaje de información en forma distribuida, procesamiento distribuido y paralelo. Además permite la comunicación personal, posibilitando por ejemplo que los empleados de una empresa se comuniquen fácil e intuitivamente con los clientes y compañeros de trabajo.

Con esta nueva arquitectura de comunicación se ve modificado el modo en que la gente está acostumbrada a trabajar, si bien el teléfono y correo electrónico son los medios de comunicación preferidos, los mensajes instantáneos son cada vez más utilizados. Las aplicaciones de este tipo también permiten el envío de archivos. Además las herramientas de colaboración hacen más fácil el trabajo en grupo para proyectos.

El modo de trabajar de las empresas también se ve modificado ya que P2P permite reducir costos de infraestructura. Cada máquina utiliza su capacidad de procesamiento.

Como cualquier otra tendencia de desarrollo, P2P puede usar un amplio rango de protocolos y tecnologías que varíen dependiendo de los objetivos específicos de los desarrolladores.

4.4 Tipos de aplicaciones P2P

Diferentes tipos de aplicaciones se han desarrollado en base a la tecnología P2P [4]. Se pueden destacar las referidas al manejo de información, es decir al hecho de compartir contenidos entre máquinas, el envío de mensajes instantáneos, las de colaboración , el acceso remoto a otras máquinas y el control de archivos, como se puede ver más ampliamente en la figura 9.

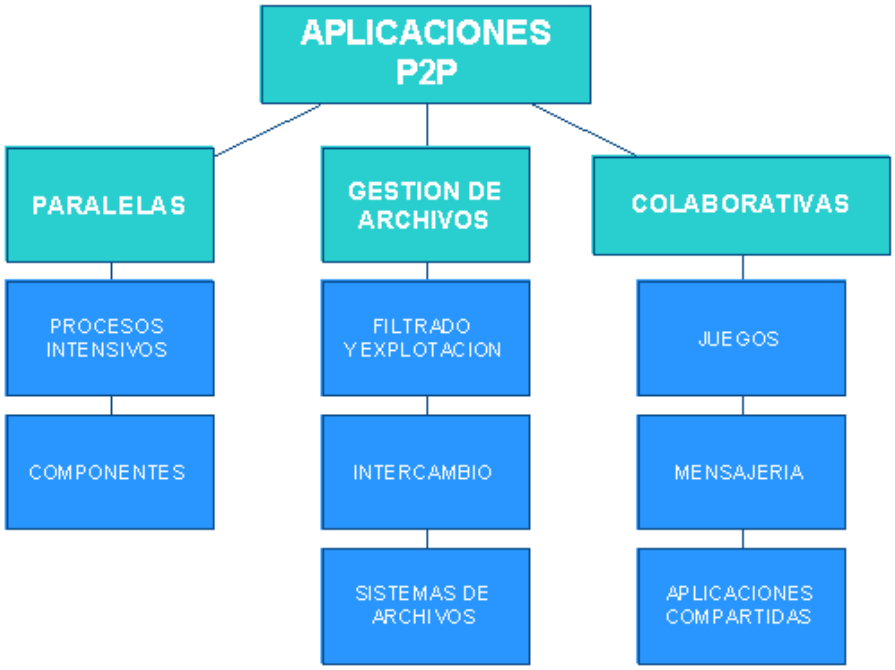

Figura 9: Clasificación de aplicaciones P2P

Dentro de los archivos digitales que se pueden compartir, han tenido mucho auge, los referentes a música de todo tipo provocando la reacción de las discográficas que vieron afectadas sus ventas. Cabe destacar que P2P se ha preocupado de garantizar seguridad en la transmisión de la información. Esta tecnología se utiliza también para la conexión de tecnologías dispares, de diferentes plataformas y recursos de computación.

4.5 Objetivos que buscan las aplicaciones P2P

a) Reducir el Costo de los Procesos

Este objetivo es fundamental para los sistemas P2P de computación distribuida, la idea principal es reducir el coste compartiéndolo entre los equipos.

b) Mejorar la escalabilidad y la fiabilidad

Este es una de las características que más tratan de destacar las aplicaciones P2P, y tiene que recibir especial atención, al no existir una autoridad central.

c) Permitir la agregación de recursos e interoperatibilidad

Este se refiere a que cada equipo del sistema P2P puede agregar sus recursos a la red.

d) Incrementar la autonomía

Todas las actividades asignadas a un equipo de una manera P2P, pueden ser ejecutadas localmente por los equipos sin intervención de otros.

e) Mejorar la privacidad y asegurar el anonimato

No es necesario que un usuario entregue información propia para algunos sistemas P2P (por ejemplo compartición de archivos en eMule)

f) Facilitar el dinamismo

Las aplicaciones P2P utilizan recursos que pueden ser utilizados de una manera continua y no estricta, lo que facilita la forma en que entran y salen estos recursos.

4.6 Modelos en P2P

A) Modelo Puro

Es una democratización total del grupo de equipos, cada nodo participante de la red posee las mismas capacidades que los demás, cualquier nodo puede iniciar una comunicación en la red. No existen recursos centralizados, por ende no existen los riesgos propios de tales sistemas, en los cuales se presentan puntos críticos de falla. En general, tales redes por su propia naturaleza implementan mecanismos ineficientes para descubrir otros equipos y localizar información.

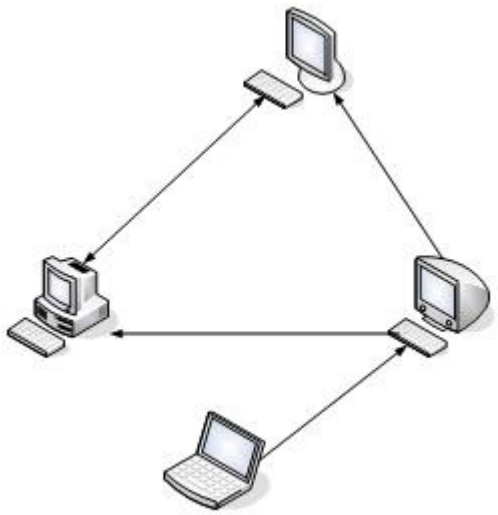

Figura 10: Modelo P2P Puro

B) Modelo Híbrido

Al realizar búsquedas primero se el nodo que inicia la negociación consulta a un servidor central para obtener meta información (en el paso 1 de la figura 11), y después con la información obtenida (paso 2) accede a ese nodo en particular (paso 3), finalmente el nodo conectado entrega la información al nodo de origen (paso4).

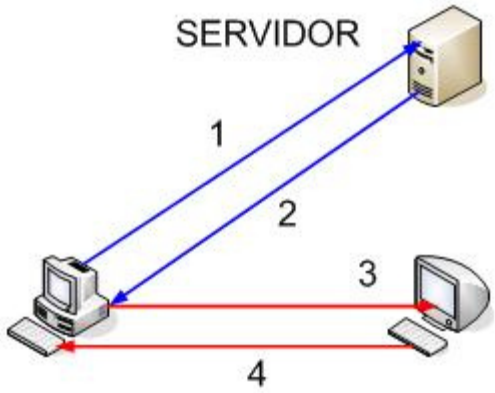

Imagen 11: Proceso de búsqueda de nodos en modelo Híbrido

Sistemas como Napster [5] tienen la característica que entre los usuarios compañeros de la red existen determinados equipos que están dedicados a una función determinada, es decir ofrecen ciertos servicios de forma centralizada. Tal es el ejemplo del sistema Napster, en el cual un servidor central contiene la base de datos con información acerca de que recursos comparten los usuarios. El concepto de hibridez deriva de que algunos equipos (nodos encaminadores) proporcionan alguna funcionalidad extra a los efectos de facilitar la interconexión entre compañeros.

Los nodos encaminadores, cuando actúan como catálogo de direcciones, pueden implementarse de dos formas:

1. Un compañero, en modo cliente, envía al nodo encaminador un requerimiento dado.
El nodo encaminador resuelve autónomamente la consulta y obtiene, generalmente, una respuesta consistente en que nodo activo (nodo a actuar en modalidad servidor) posee el recurso solicitado, y envía el requerimiento al nodo servidor seleccionado, donde éste último se comunica con el nodo origen de la consulta y le brinda el recurso solicitado. Lo anterior puede describirse como:

2. El nodo hace una petición, en modo cliente envía un requerimiento al nodo encaminador, éste devuelve la dirección del nodo que en modalidad servidor satisfacerá tal consulta, el nodo que realiza la petición se comunica con el nodo que oficiará de servidor enviándole su requerimiento de recurso, finalmente el nodo servidor satisface la petición.

c) Modelo con SuperNodos

Los nodos compañeros pueden ayudarse de un supernodo que brinda el servicio de descubrimiento de equipos. Las aplicaciones, al inicializarse y al finalizar, registran su presencia y ausencia notificando al servidor. Cualquier nodo puede consultar en cualquier momento al supernodo y obtener una lista de los nodos activos

Este modelo al utilizar supernodos [6], permite una mejor eficiencia en las búsquedas, y en las cargas. Aquí cada supernodo almacena información diferente al resto, de esta manera se logra que cada nodo haga crecer la red, Ejemplos que utilizan este modelo son Kazaa [5] y Skype [6].

Los supernodos se conectan entre si para mantener actualizados los índices de búsqueda, y estos se refrescan cada cierto tiempo. El modelo puede ser representado como en la figura 12.

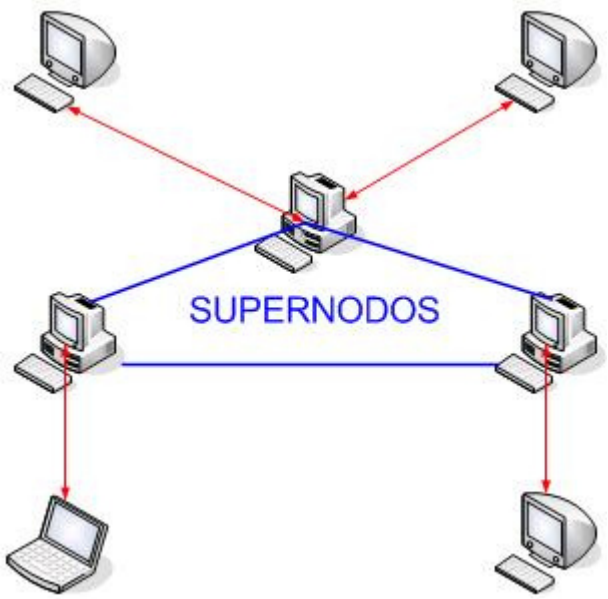

Figura 12: Búsqueda de nodos en modelo con SuperNodos

4.7 Componentes de un sistema P2P

En un sistema P2P podemos tener cinco capas [4], dentro de las que tendríamos capa especifica de las aplicaciones, especifica de clases, capa de robustez, capa de gestión de grupos y capa de comunicación. Se puede apreciar el esquema de capas en la figura 13.

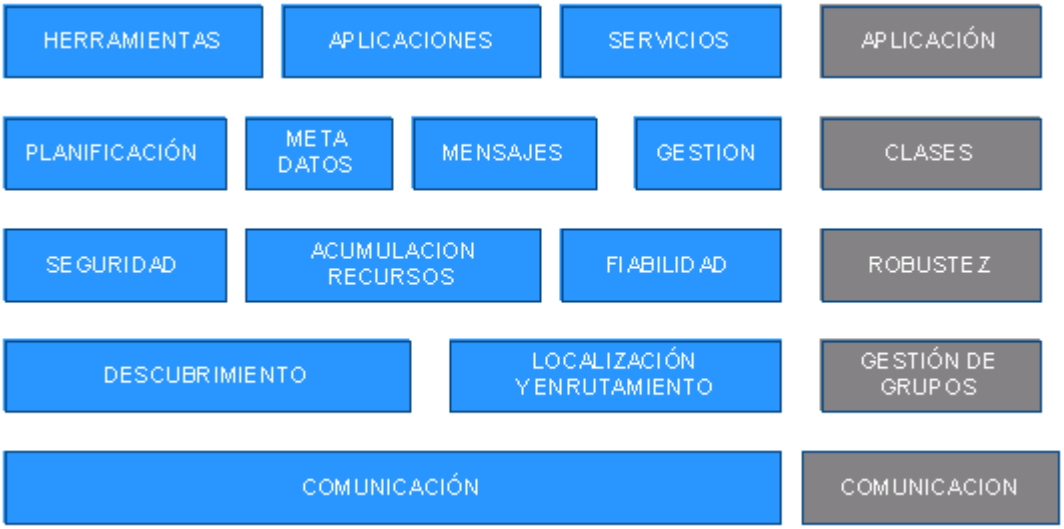

Figura 13: Componentes de un sistema P2P

a) Capa especifica de las aplicaciones

Implementa las funcionalidades de cada tipo de aplicación P2P. Estas se dividen en aplicaciones, herramientas y servicios.

b) Capa especifica de clases

Define un esquema para el almacenamiento de datos y metadatos, satisface requerimientos de los componentes servidor y cliente, Almacena información sobre el estado del nodo y la red (Ej. lista de nodos compañeros). Abstraen funcionalidades, dentro de las cuales tenemos:

Planificación: Se aplica en el computo intensivo paralelizable de las aplicaciones.

Metadatos: Se aplica a las aplicaciones en de gestión de ficheros.

Mensajes : Se usa en aplicaciones colaborativas.

Gestión : Permite gestionar la infraestructura subyacente.

c) Capa de robustez

Asegura la fiabilidad y estabilidad del sistema.

Seguridad: Bastante sensible ya que los nodos pueden funcionar como clientes y servidor, lo cual es un problema claro de seguridad. Si un cliente pasa a ser servidor puede alterar el sistema, solo las fuentes de confianza o autenticadas pueden ser servidor, por lo que se recurre a terceras pastes, la intervención del usuario o a centralizar la tarea de seguridad.

Acumulación de Recursos: Facilita las bases para que los nodos puedan interactuar y compartir información.

Fiabilidad: Complicado de gestionar en P2P, se tiene que recurrir a la redundancia. Las tareas se replican en varias máquinas para que en caso de fallar en una se pueda iniciar en otra distinta.

d) Capa de gestión de grupos

Encontrar nodos y mover datos a través de ellos, esto quiere decir descubrir, localizar y encaminar datos.

e) Capa de comunicación

Se busca superar la naturaleza dinámica de los nodos y lograr mantener la comunicación, ya que generalmente los grupos de nodos asociados a las redes P2P cambian.

Características:

- Envía y recibe mensajes de los componentes y a los componentes cliente y servidor.
- Administra mensajes duplicados, tiempos de vida de mensajes.
- Trata de optimizar la perfomance de la red.
- Acepta conexiones entrantes.
- Gerencia conexiones existentes.

4.8 Características de los sistemas P2P

- La primer característica que define a un nodo en un sistema P2P, es que cumple tanto el rol de cliente como de servidor.
- Los nodos participantes deberían ser autónomos. Cada uno regula su grado de participación en la red, definiendo los recursos que ofrecen y en que cantidad.
- No necesariamente los nodos deben tener una vista global del sistema. El comportamiento global emerge de las interacciones individuales.
- Todos los datos y servicios deberían ser accesibles por cualquier nodo.
- Pueden implementar un sistema de nombres alternativo al sistema DNS, que satisfaga sus propias necesidades.
- Los nodos en una red P2P pueden ingresar y salir constantemente de forma arbitraria. Pueden ser nodos de usuario final que no se encuentran permanentemente conectados a la red, es decir, su conectividad es variable ó bien nodos dedicados de alta disponibilidad.

- La ubicación de los recursos es dinámica, ya que depende del estado del sistema en un momento del tiempo.
- La red es un ambiente dinámico y heterogéneo, dado que la pueden conforman nodos con conectividad variable y de diversas plataformas de hardware y software. Todo nodo en una red tiene la posibilidad de que sobre éste se ejecute una aplicación P2P.

4.9 Sistemas P2P

Los sistemas P2P se pueden clasificar en cuatro categorías [4], según su estructura y función, como se ve en la figura 14: informática distribuida (Ej. SETI@home) [7], compartición de archivos (Ej. Napster, DC++) [5], sistemas colaborativos (Ej. Skype, Groove) [6] y plataformas (Ej. JXTA y servicios .NET) [8].

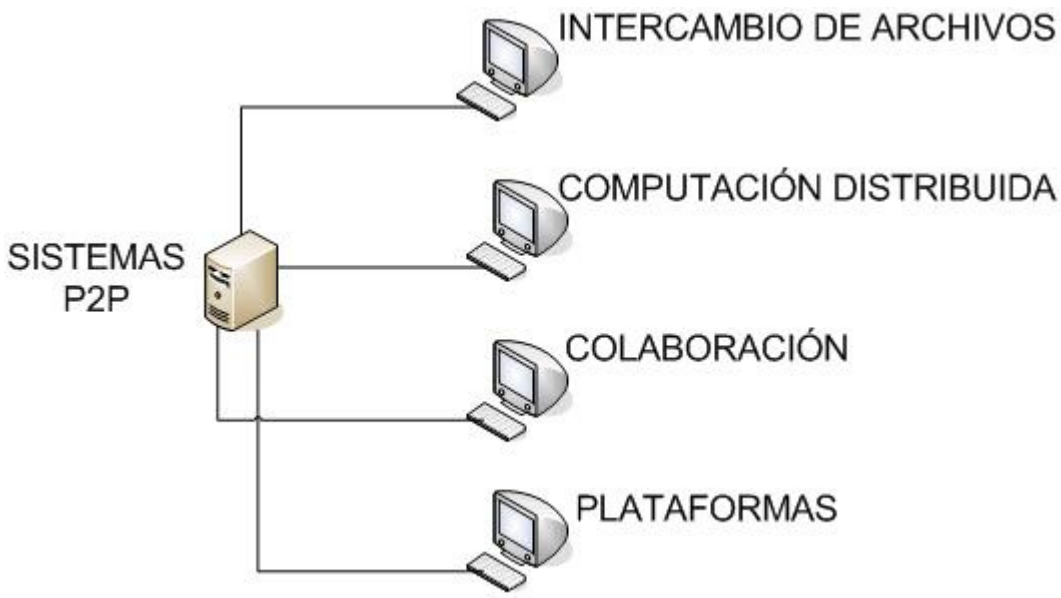

Figura 14: Esquema sistemas P2P

4.9.1 Computación Distribuida

Utilizada con mucho éxito en P2P, lo que hace es distribuir recursos y procesos a un gran número de individuos que se encuentren conectados (ver figura 15). Generalmente es una aplicación corriendo en cada equipo que entrega resultados a un servidor central. Esta trabaja de la siguiente manera, divide un problema computacional en pequeñas partes. El proceso de cada una de las partes (usando mecanismos fork-and-join) es realizado por cada equipo conectado a la red, enviando los resultados procesados al servidor central como se muestra en la figura 15. Este servidor central es el que se preocupa de distribuir los trabajos a los distintos equipos que están en la red.

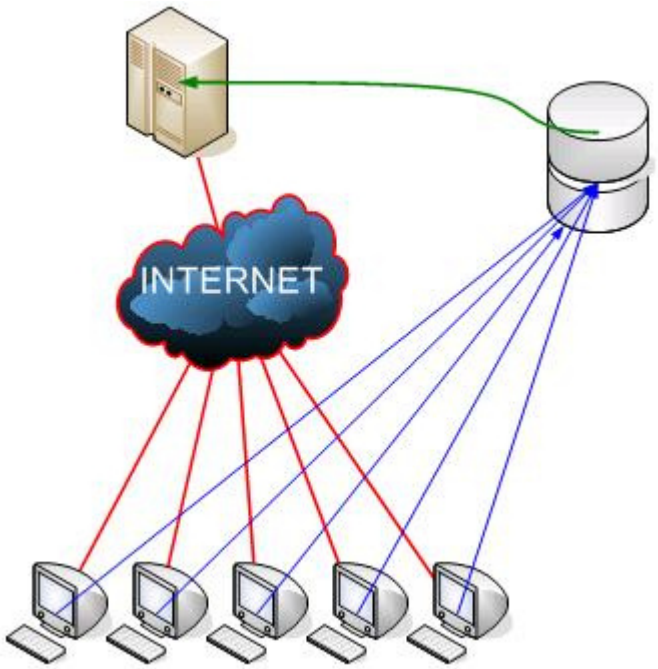

Figura 15: P2P Computación distribuida sobre Internet.

Pasos que sigue la computación distribuida:

1. Un trabajo de divide en subtareas.

2. Los equipos clientes acceden a participar.

3. Existe un planificador centralizado.

4. Cuando un equipo se encuentra inactivo, (por ejemplo protector de pantalla) realiza las subtareas que se le asignan.

5. Al terminar envía los resultados a una Base de Datos centralizada.

Entre los beneficios directos que se encuentran con la descentralización de los trabajos aprovechando redes de computadores P2P, se resuelven varios problemas al mismo tiempo:

Potencia de cálculo: Ninguna supercomputadora es capaz de competir con varios miles de máquinas (de todos los tamaños) conectadas a una red.

Tiempo: Las investigaciones que requieren el análisis de una gran cantidad de datos (como por ejemplo el genoma humano) se aceleran con proyectos de computación distribuida.

Almacenamiento: de la misma forma, el volumen de datos generados por algunos proyectos (como las señales de radio procedentes del espacio que analiza SETI@home) se puede distribuir entre los discos duros de millones de usuarios.

Dinero: No todas las empresas pueden permitirse una supercomputadora o una red propia para afrontar grandes retos . Una red distribuida que emplee recursos no utilizados reduce drásticamente el coste de grandes proyectos.

Colaboración: entre países, organismos o grupos de investigadores situados en distintos puntos del planeta para trabajar en un proyecto común.

a) SETI@home como sistema de computación distribuida

SETI@HOME [7] es el mejor ejemplo para este caso es el del Instituto SETI (Search for Extraterrestrial Intelligence Institute), ver la imagen del cliente en la figura 16. Para participar en este proyecto, el usuario baja por Internet un protector de pantalla que simplemente indica el momento cuando la computadora esta ociosa. Una vez que se activa este protector, el SETI le asigna a esa máquina el procesamiento de un trozo de las bitácoras de los datos de los radio-telescopios.

Almacenando y distribuyendo los datos de SETI@HOME

En Berkeley los datos son divididos en unidades de trabajo, separando las señales en 256 bandas de frecuencia, cada una aproximadamente de 10 KHz de ancho, a su vez cada banda es dividida en segmentos de 107 segundos, estos segmentos o unidades de trabajo están superpuestos por 20 segundos, asegurando que cada período de emisión este contenido enteramente en al menos un segmento. Las unidades de trabajo resultante son de alrededor de 350 Kb, la suficiente cantidad de datos para mantener a una típica computadora ocupada por varias horas, pero bastante pequeños para descargar con un módem lento en unos pocos minutos. La información sobre las unidades de trabajo y cintas, así como también muchos otros aspectos del proyectos son almacenados en tablas.

Un programa servidor "data/result" distribuye las unidades de datos a los clientes. Este utiliza un protocolo basado en HTTP a fin de que los clientes detrás de sistemas firewalls puedan contactar al servidor. Las unidades de trabajo son enviadas por round-robin en orden FIFO.

Un programa recolector de basura remueve las unidades de trabajo del disco y limpia la bandera llamada "en disco" de dichas unidades en la base de datos. SETI@home experimenta dos políticas para borrar las unidades de trabajo:

- Borrar las unidades de trabajo para las que se hayan recibido *N* resultados, donde *N* es el nivel de redundancia. Si el dispositivo que almacena las unidades de trabajo se llena, la producción de dichas unidades de trabajo es bloqueada, esto provoca que disminuya el rendimiento de todo el sistema.
- Borrar las unidades de trabajo que han sido enviadas *M* veces, donde *M* es ligeramente mayor que *N*, esto elimina la disminución en el rendimiento nombrada en el caso anterior, pero trae como consecuencia que algunas unidades de trabajo nunca tengan resultados completos, cumpliendo el nivel de redundancia. Actualmente SETI@home esta utilizando esta política.

En general SETI@home ha dividido su servidor en partes, tales como el programa que divide los datos en unidades de trabajo, el programa servidor "data/result" y el programa recolector de basura; estos son independientes unos de otros.

Actualmente SETI@home distribuye su software cliente figura 16 para 47 combinaciones diferentes de CPU y sistema operativo. Los usuarios pueden descargar este software desde el sitio Web de SETI@home (http://setiathome.ssl.berkeley.edu). Para Microsoft Windows y Apple Macintosh, el cliente se instala por defecto como un protector de pantalla, que procesa datos solamente cuando el protector de pantalla está activo, pero puede ser configurado para procesar constantemente. Para otras plataformas, el cliente básico esta basado en texto, en estas los usuarios generalmente lo corren en segundo plano.

Un programa de visualización gráfica similar a las versiones para Macintosh y Windows esta disponible para sistemas UNIX y Linux que corran las X Windows. demás una amplia variedad de aplicaciones de terceros han sido desarrolladas para mostrar los datos.

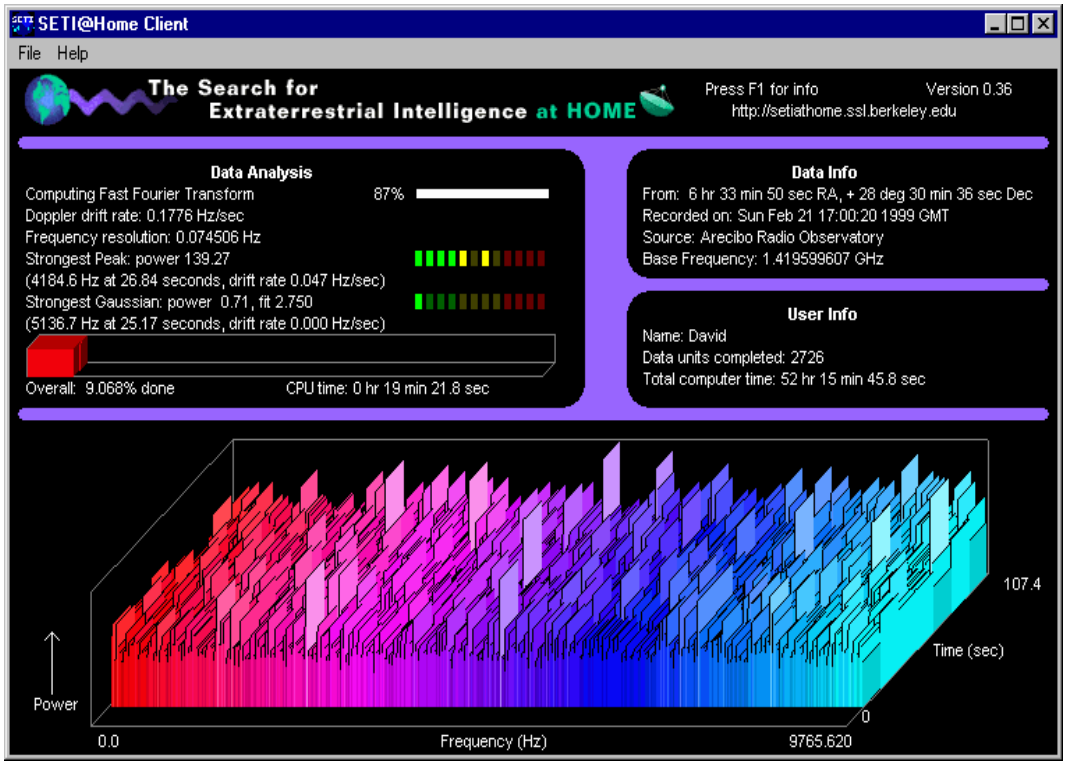

Figura 16: Cliente SETI@home

Este programa cliente en forma repetitiva busca una unidad de trabajo desde el equipo servidor data/result, la analiza y retorna el resultado que es una lista de señales candidatas. Dicho cliente necesita una conexión Internet solo mientras se comunica con el servidor. El programa periódicamente guarda su estado en un archivo en disco y lee este archivo cuando comienza a funcionar, de esta manera lleva a cabo su progreso aún si el equipo es apagado frecuentemente.

4.9.2 Compartición de archivos

El almacenaje y el intercambio de archivos [18] es el área más conocida por los usuarios que utilizan el P2P, además de ser la que ha crecido más rápidamente. Un archivo es guardado en algún equipo de la comunidad P2P y se encuentra disponible para cualquier otro peer (Figura 17). Lo que crea un área de intercambio de archivos. Si este archivo es bajado por algún peer y no se completo la transacción, esta transacción puede ser finalizada por algún otro equipo que contenga el archivo. La aplicación más conocida en cuanto a P2P para archivos compartidos es probablemente Napster. El sistema de Napster permite al usuario acceder a canciones que se encuentran almacenadas en los discos duros de otros usuarios. Napster mantiene una base de datos centralizada que contiene los títulos y la ubicación, pero no almacena las canciones. Cuando un usuario realiza una búsqueda de un título en especial, el sistema le muestra una lista de opciones de donde podrá, en base a su criterio personal, obtener la música.

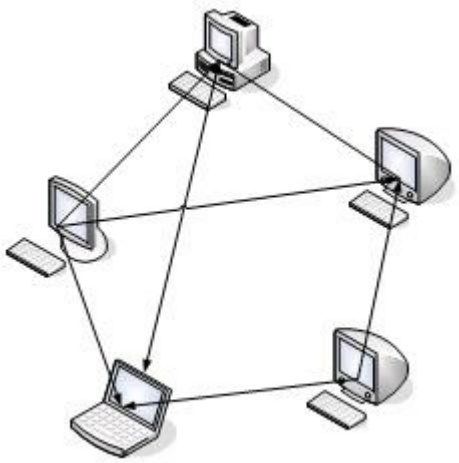

Figura 17: Esquema típico de compartición de archivos en P2P

Principales características de los sistemas de compartición de archivos

- Sin limitaciones, cuando de otra manera no se podría almacenar tanta información.
- Contenidos multimedia tienen gran tamaño.
- Disponibilidad de varias fuentes (redundancia de información).
- Anonimato para proteger al que publica y al que lee.
- Configurable para un mejor rendimiento (descarga desde servidores cercanos).
- Consumo de ancho de banda configurable, búsquedas, y seguridad.

a) eMule como sistema de compartición de archivos P2P

eMule [9] es un programa P2P libre de intercambio de archivos que utiliza la red eDonkey, pero ofrece más funcionalidades que el cliente eDonkey original, además de superarlo en popularidad desde hace ya un tiempo.

Las características que le distinguen son el intercambio directo de links entre sus equipos, el uso de un sistema de créditos, la recuperación rápida de partes corruptas. También destaca el hecho de que al ser un programa GPL cualquiera puede colaborar y mejorarlo libremente lo que ha motivado la proliferación de mods como el Phoenix, el webcache o el Morph (los mods no son más que modificaciones del proyecto original). También permite la aparición de proyectos independientes basados en su código como los clientes eMule para otras plataformas o el popular eMule Plus. Todo ello contribuye a una continua mejora de los programas. eMule es uno de los programas P2P más usados actualmente. Se caracteriza por su interfaz simple basada en pestañas, como se ve en la imagen 18. Estas son: "Conectar", "Servidores", "Tráfico", "Buscar", "Archivos compartidos", "Mensajes","IRC", "Estadísticas" y "Preferencias".

eMule se basa en un sistema de créditos por el cual quien más sube a la red más descarga. Los créditos se registran de forma descentralizada en todos los usuarios de la red, evitando así la posibilidad de falsearlos. De cada usuario se descargan partes de archivos (que pueden estar siendo descargadas en ese momento por otro usuario) ensamblándose al finalizar para formar el archivo completo.

Esta red P2P es más útil cuando los archivos a descargar son de gran tamaño; además, hay mucho material español y europeo en general ya que la red está principalmente popularizada en dicho continente.

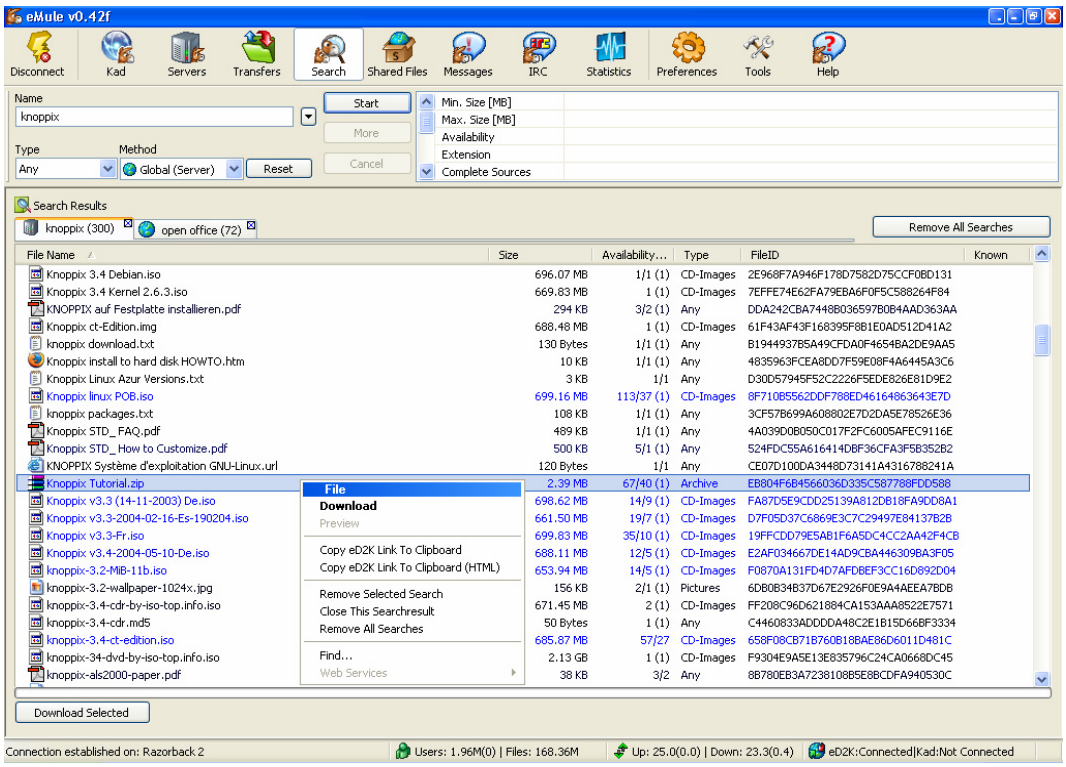

Figura 18: screenshoot eMule

Aunque el cliente eMule es una aplicación para sistemas Windows, su código es libre y de este se basan otros clientes como xMule o aMule.

Su amplia implantación, así como su carácter descentralizado lo han hecho el preferido por la mayor parte de los usuarios, dispuestos a "compartir contenidos". Esas mismas causas son las que han levantado la polémica sobre la necesidad o no de una legislación internacional que vele por la defensa de los derechos de propiedad intelectual y sancione actos que los puedan vulnerar.

4.9.3 Sistema P2P Colaborativo

Estas aplicaciones P2P permiten el establecimiento de comunicaciones entre usuarios de una manera colaborativa [6], o sea busca organizar espacios de trabajo en línea y compartir proyectos para grupos de trabajo. Dentro del rango de utilidades que se pueden encontrar están la mensajería instantánea, juegos en línea, telefonía IP y videoconferencia. Estas aplicaciones colaborativas son generalmente basadas en eventos, los peers forman grupos e inician las tareas. Los grupos pueden incluir dos o más peers colaborando directamente entre ellos. Cuando ocurre un cambio en un peer (Ej. un peer inicia un mensaje de Chat), el evento es generado y enviado al resto del grupo. Luego la capa de aplicación de cada peer es actualizada. El mejor ejemplo para este sistema P2P es Skype (ver figura 19), el cual da servicios de Voz sobre IP (VoIP) y mensajería instantánea. La principal característica que posee Skype es la solución frente a entornos NAT usando transversabilidad, se piensa que Skype utiliza una variante del protocolo STUN para determinar el tipo de NAT y firewall que se encuentra detrás del cliente.

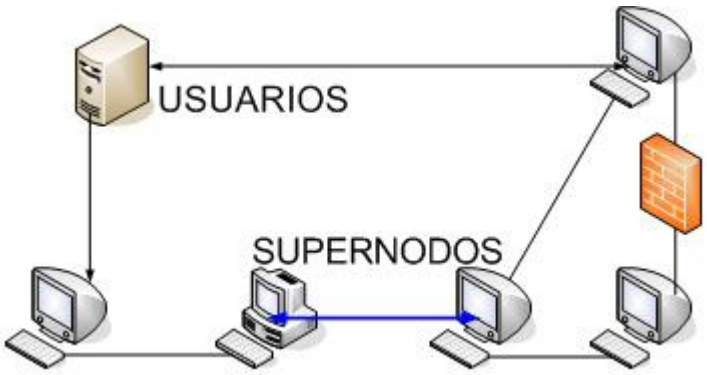

Figura 19: Esquema P2P Colaborativo Skype

a) Skype como P2P Colaborativo

Skype es un servicio de telefonía gratuita sobre Internet [6] que se comunica usando conectividad P2P. Usando un programa cliente similar al de cualquier software de mensajería instantánea como se puede ver en la imagen 20, que se encuentra disponible para un gran número de plataformas, un usuario Skype puede enviar y recibir mensajes, establecer llamadas e intercambio de datos con cualquier usuario perteneciente a la red de Skype. Skype utiliza una gran cantidad de técnicas para establecer comunicaciones P2P entre los clientes, esto para obtener una mejor calidad de las llamadas de voz. Al ser Skype un software P2P colaborativo, quiere decir que no necesita estar siempre conectado a un servidor principal, al contrario, esto es lo que hace a Skype ser un software robusto y tolerante a errores en la red.

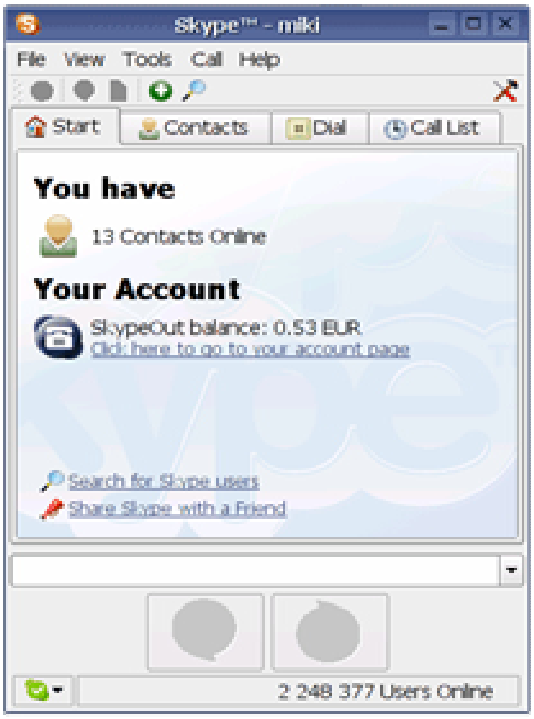

Imagen 20: screenshoot Skype

Servicios que entrega Skype

La principal utilidad vista en Skype son las llamadas telefónicas a través de Internet, además posee transferencia de archivos y mensajería instantánea. Todo cliente Skype viene provisto con los siguientes servicios:

- Realizar llamadas de voz a otros usuarios Skype.
- Realizar llamadas de multiconferencia.
- Realizar llamadas a teléfonos tradicionales (SkypeOut).
- Realizar llamadas desde teléfonos tradicionales (SkypeIn).
- Chat, a través de la mensajería instantánea.
- Transferencia de archivos entre distintas plataformas.
- Directorio general de participantes.

Skype es un software multiplataforma que corre actualmente en Windows XP, Windows 2000, Linux Mac OS X y Windows Mobile 2003.

Como funciona Skype

Skype provee una manera robusta y escalable en sus servicios, este diseño llamado "Supernodos con Arquitectura P2P", son las bases de la comunicación en Skype. Al contrario de que cada cliente se encuentre conectado a un servidor central para completar las llamadas , el software Skype interactúa directamente con otros clientes de la misma red para completar rápidamente las llamadas, lo que quiere decir que la participación de cada usuario Skype hace posible el funcionamiento de la red.

Comparando Skype a la telefonía tradicional donde todos los usuarios estaban conectados a switches que estaban distribuidos en distintos niveles para establecer comunicaciones locales, regionales y de larga distancia, los supernodos con arquitectura P2P son los que se preocupan de esta tarea.

La red de Skype permite este trabajo gracias a un índice global y distribuido que posee, en que los usuarios encuentran a otros para realizar las llamadas, enviar mensajes, etc. Todo esto sin servidores centrales. La Arquitectura de supernodos P2P ha sido usada satisfactoriamente por un gran número de aplicaciones P2P. El supernodo no es más que un cliente Skype regular que provee asistencia a la red de Skype con direccionamiento de contactos y ayudando a encaminar las llamadas. Este servicio llamado "Índice Global", permite que la gama de servicios Skype funcionen hasta los clientes que parecen no accesibles. Cuando un cliente Skype pasa a ser un supernodo, este acepta conexiones desde un pequeño grupo de otros usuarios Skype con el propósito de mantener la exactitud del índice global. Toda la actividad realizada por el supernodo es completamente transparente para el usuario, por lo tanto si un usuario no puede recibir conexiones entrantes, por que por ejemplo se encuentra detrás de un NAT, éste nunca será escogido para realizar las tareas de un supernodo.

Entornos firewalls y NAT en Skype

La mayoría de las soluciones de VoIP son diseñadas para entornos empresariales, y muchos usuarios no pueden establecer llamadas VoIP sin reconfigurar sus routers y firewalls, por que como se ha dicho anteriormente la red posee un firewall restrictivo o una pasarela NAT.

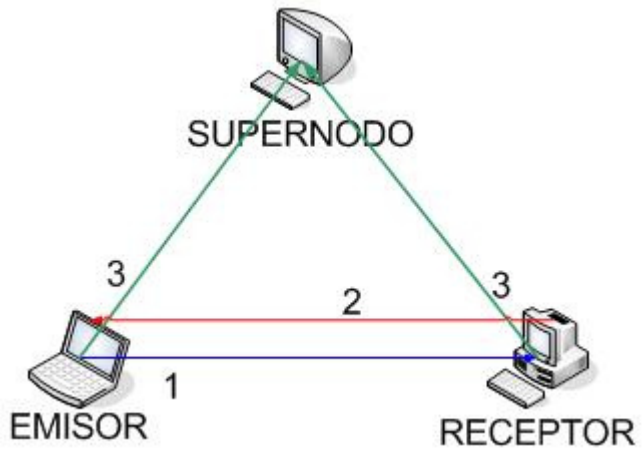

Figura 21 : Conectividad Transversal con Skype

La arquitectura P2P de Skype soluciona este problema permitiendo a los usuarios que se encuentran detrás de un NAT o firewall, gracias a un encaminamiento transparente, con la ayuda de supernodos que poseen dirección pública. Esto quiere decir que cualquier usuario de Skype puede establecer llamadas VoIP sin la necesidad de reconfigurar un router o firewall.

Como se muestra en el paso 1 de la figura 21, cuando dos clientes Skype quieren comunicarse primero se tratan de comunicar directamente,. si las llamadas se encuentran protegidas por un firewall, entonces el equipo al que se llama pregunta al índice global si puede iniciar una conexión al revés hacia el equipo, como se muestra en el paso 2, si estas conexiones fueron satisfactorias entonces la llamada se completó con una conexión directa entre los peers. Sin embargo si ambas llamadas se encuentran bajo un NAT restrictivos, se requiere un supernodo que sea alcanzable por los dos equipos y éste utilice la técnica relaying para comunicar a los dos peers. Como se ve en el paso 3 de la figura 21, cuando la conexión es establecida el tercer equipo actúa como Proxy de la llamada. Las llamadas, los mensajes de texto y la transferencia de archivos son encriptados entre los peers.

Un óptimo de Skype

Los administradores de red pueden impactar positivamente en la calidad de las comunicaciones P2P de Skype, permitiendo el paso de paquetes TCP y UDP, para un mejor funcionamiento de Skype. Skype puede funcionar de acuerdo a las siguientes reglas:

1. Conexiones TCP salientes deben ser permitidas para puertos 1024 y superiores .
2. Conexiones TCP salientes deben ser permitidas para los puertos 80 y 444.
3. Conexiones UDP salientes deben ser permitidas para los puertos 1024 y superiores. Para que UDP sea útil para Skype, el NAT debe permitir respuestas para enviar datagramas UDP de al menos 30 segundos o mayores.
4. El NAT debe proveer de una traducción consistente, de manera de que la traducción de direcciones salientes sea usualmente la misma para los paquetes UDP consecutivos.

A pesar de todo lo visto hasta ahora, el uso de UDP es opcional, lo que quiere decir que Skype trabajará bien sin la opción de transmitir mensajes UDP, aunque si se utiliza UDP la calidad de la llamada será mucho mejor en promedio.

4.9.4 P2P como infraestructura

Varias compañías compiten por imponer sus proyectos de infraestructura para soportar aplicaciones P2P. Microsoft propone su estructura de trabajo denominado .NET y Sun Microsystems a JXTA [8] . Otra compañía denominada Groove Networks presenta una plataforma descentralizada llamada groupware y ha incorporado servicios P2P, tales como mensajería instantánea e intercambio de archivos.

Los desarrolladores de sistemas P2P se encuentran construyendo aplicaciones desde el principio. Tienen que investigar, desarrollar, construir prototipos, probar y esperar ser artífices de sus éxitos y fracasos sobre esta nueva tecnología. A los efectos de consolidar la computación distribuida se necesitará desarrollar APIs y compiladores robustos desarrollados específicamente para tal ambiente. Compañías como SUN a través de su plataforma JXTA y Microsoft con .NET están intentando brindar ambientes de desarrollo unificado para construir tales aplicaciones.

a) JXTA como P2P de infraestructura

La oferta de SUN para desarrollar aplicaciones distribuidas en un entorno P2P. Esta busca integrar uniformemente Datos, ancho de banda, recursos de cálculo.

Características de JXTA

- Código abierto y participativo (http://www.jxta.org).
- Todas las aplicaciones del entorno utilizan la misma plataforma de comunicaciones.
- El modelo de comunicación de cada aplicación puede ser diferente.
- Posee Independencia del sistema operativo.

4.10 Seguridad en redes P2P

La seguridad es un campo poco explorado aún en las redes P2P. Dado que este modelo distribuido y cooperativo se basa en la confianza es necesario lograr mecanismos de seguridad adaptados específicamente a P2P que aseguren la autenticidad y privacidad de las comunicaciones.

Existen dos aspectos sobre este dominio, uno es la seguridad a aplicar a los grupos, donde la confianza debe ser fundamental, y se deben implementar técnicas que tiendan a denegar todo acceso a recursos a aquellos usuarios que no son de confianza o usuarios mal intencionados. El otro aspecto es contemplar la protección contra ataques de virus, gusanos, intrusos, etc.

La seguridad es un importante problema en ambientes P2P, el posible impacto en un ambiente de computación distribuida donde personas mal intencionadas pueden entregar resultados fraudulentos o estropear algún proyecto, es un opción a no omitir. La utilización de firmas digitales y encriptación debería resolver la mayoría de los problemas de seguridad.

4.11 Metas futuras de los sistemas P2P

Anonimato y hospedaje descentralizado de bloques de información (una misma información, partirla y distribuirla en *n* nodos) serán los principios de diseño en las nuevas aplicaciones, actualmente existen unos pocos proyectos que utilizan esta tecnología (como por ejemplo SETI@home). El derecho a la privacidad se garantizará implementando como una cualidad entandar de cualquier nueva aplicación.

Definir maneras o formas más fáciles de encontrar información compartida, ya sea a través de normalización de formas de almacenamiento, definición de metadatos y sistemas de búsqueda distribuida.

Actualmente el acceso general a espacios de almacenamiento y a ciclos de CPU se realiza bajo un riesgo de seguridad importante por parte de los usuarios. La computación distribuida necesitará definir nuevos entornos de trabajo, protocolos de comunicación y lenguajes de programación que minimicen los riesgos (ejecución de código no seguro acceso a información no compartida) en tales equipos.

Los modelos de encaminamiento en redes P2P aportan la posibilidad del encaminamiento por difusión (multicast). Implementándose mucho más fácil y con menos requisitos que la definida a nivel de red en Internet. Tal característica será mayormente difundida a los efectos de lograr comunicaciones más veloces y que utilicen un ancho de banda menor.

Capitulo 5

5.1 Técnicas de Transversabilidad

5.1.1 Relaying

Esta es una de las técnicas más confiables para establecer sesiones P2P a través de NAT [10], pero la menos eficiente, y se basa simplemente en hacer que la transmisión parezca una típica comunicación Cliente/Servidor, esto a través de la retransmisión. Supongamos que tenemos dos clientes A y B, que han iniciado algún tipo de comunicación, ya sea TCP o UDP, con un servidor con dirección IP pública C, como se ve en la figura 22. La dirección IP de C es la 216.155.80.25 y el puerto de comunicación será el 12345. Como se puede ver en la figura, los clientes pueden encontrarse distintas redes, y los respectivos NAT bloquearán una conexión directa entre ellos. Entonces para que se logre enviar un mensaje al cliente B, el cliente A envía un mensaje al servidor C, donde ya tiene una conexión establecida, y éste a la vez reenvía la información al cliente B, que anteriormente había establecido una conexión al servidor C.

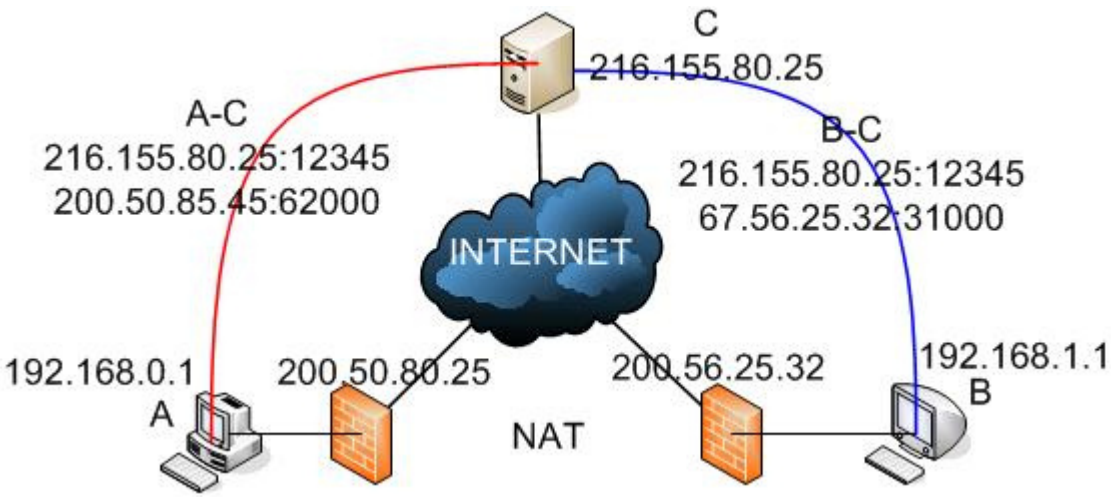

Figura 22: Utilizando Relaying

Con esto se logra una comunicación directa entre los peers utilizando al servidor público C como mediador entre ellos.

El reenvío de la información usando la técnica relaying, funciona solamente cuando los dos clientes pueden establecer una conexión con el servidor público, la desventaja que presenta esta solución es que consume poder de procesamiento del servidor , y mucho ancho de banda de la red, por lo que la comunicación entre los peers depende en cierta medida del servidor público. A pesar de estas desventajas no existe una técnica más efectiva que esta para realizar transversabilidad en entornos con NAT.

5.1.2 Conexión al Revés

La llamada conexión al revés (Connection Reversal) [10] se utiliza cuando uno de los peers se encuentra detrás de un entorno NAT y el otro se encuentra en la red pública, como muestra la figura 23. Frente a este esquema la comunicación del cliente que esta dentro del entorno NAT hacia afuera es inmediata y directa, pero si el equipo que se encuentra en la red pública quiere comunicarse directamente con el que se encuentra dentro del entorno NAT, no puede pues el NAT rechaza la petición. Lo que se hace en este caso es pedir al equipo privado que se conecte al equipo público, esto a través de un servidor que sirva como mediador, que debe poseer dirección IP pública, realizando así una conexión al revés. Como se muestra en la figura M, el equipo A puede establecer directamente una comunicación con el equipo B, pero B debe solicitarle al equipo A, a través de C que sea él quien se conecte mediante una conexión al revés.

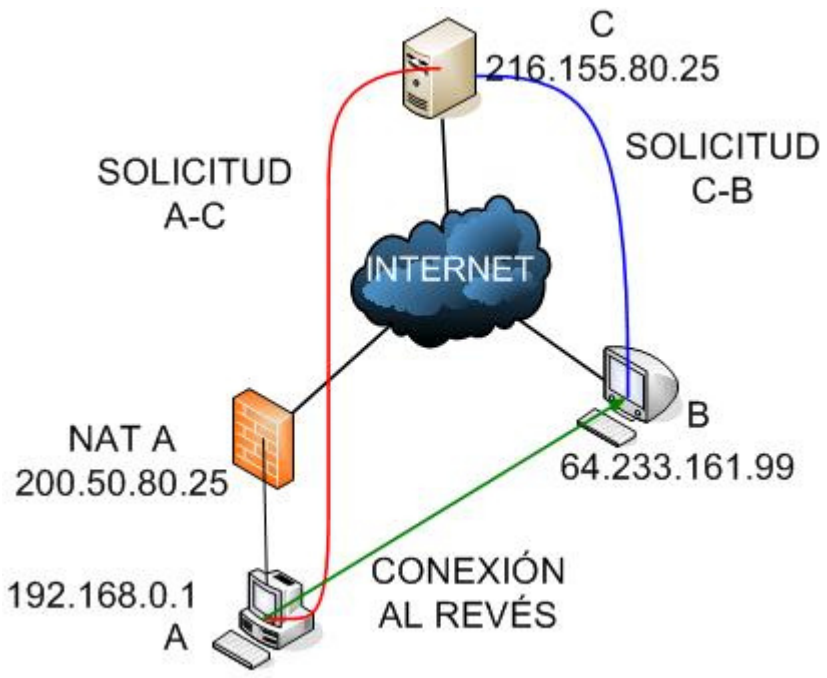

Figura 23: Conexión al revés.

5.1.3 UPD Hole Punching

Esta técnica permite establecer sesiones UDP entre dos peers que se encuentras detrás de entornos con NAT [10], esto como en las técnicas anteriormente mencionadas se logra con la ayuda de un servidor mediador con IP pública. Este servidor público asume que los dos clientes ya han tenido una conexión UPD con el mismo. Cuando cada cliente se registra con este servidor mediador, éste graba dos direcciones con sus respectivos puertos por cliente. Las direcciones a guardar son la que el cliente cree que tiene para comunicarse con el servidor y la que el servidor observa desde la red pública. Por lo tanto el servidor público guarda *dirección:puerto privado* y *dirección:puerto público.* El servidor puede obtener la dirección privada dada por el mismo cliente cuando este le solicitó el registro, y puede obtener la dirección pública desde la cabecera del paquete UPD. Si el alguno de estos dos clientes no se encuentra en un entono con NAT entonces las direcciones deberían ser las mismas.

Supongamos que tenemos dos clientes A y B que se encuentran detrás de distintos entornos NAT, y tenemos un servidor público C. Si el cliente A quiere establecer una sesión UDP directamente con el cliente B, el proceso de Hole Punching sería el siguiente:

1. El equipo A inicialmente no sabe como alcanzar al equipo B, entonces el equipo A le solicita al servidor público C que lo ayude a establecer una sesión UDP con el equipo B.

2. El servidor público C responde a el equipo A con un mensaje conteniendo las direcciones pública y la dirección privada de el equipo B, al mismo tiempo el servidor C usa la sesión UDP que tiene con el equipo B para enviarle a el equipo B un mensaje conteniendo la dirección pública y la dirección privada de el equipo A. Una vez hecho esto, ambos cliente conocen las direcciones de sus contrarios.

3. Cuando el equipo A recibe la dirección pública y la dirección privada de de el equipo B, el equipo A comienza a enviar paquetes UDP hacia las dos direcciones de el equipo B, con lo que abre una compuerta (Hole Punching) para recibir un mensaje desde el equipo B. Al mismo tiempo el equipo B comienza a enviar paquetes UDP hacia el equipo A, donde la conexión directa funciona gracias al agujero que se abrió en el NAT del equipo A.

A continuación se presentarán los distintos escenarios de red en que puede ser utilizada la técnica UPD Hole Punching. El primero que es el más fácil, consiste en dos clientes frente al mismo NAT. Luego la más común dos clientes frente a distintos NAT y finalmente clientes frente a múltiples niveles de NAT.

Esquemas de red bajo el cual se puede utilizar UDP Hole Punching

a) Utilizando UPD Hole Punching en clientes con NAT común

Supongamos que tenemos dos clientes desconocidos entre sí, que se encuentran bajo el mismo NAT como se muestra en la figura 24.

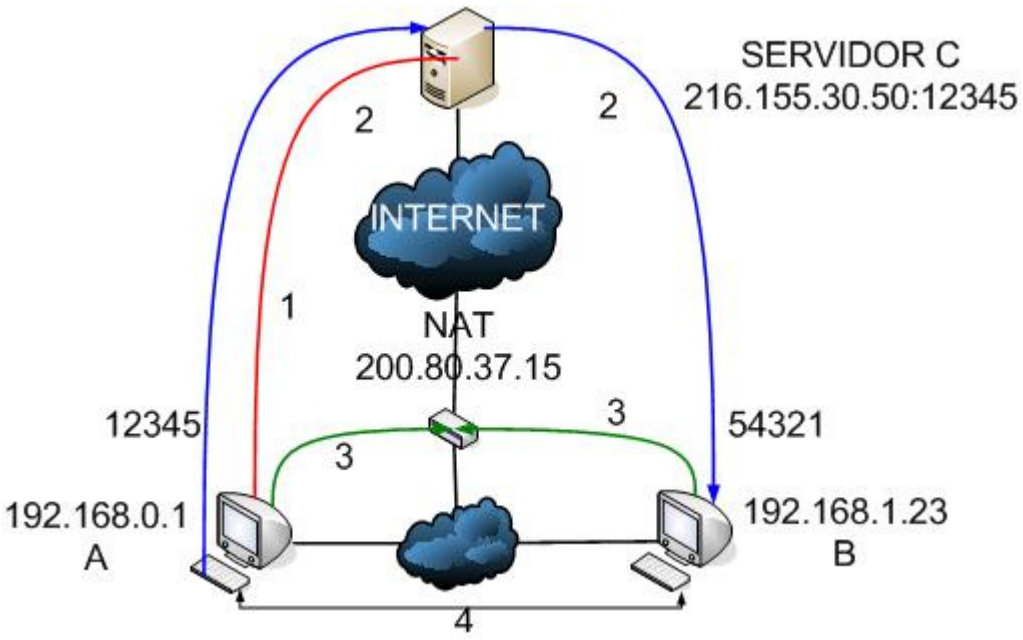

Figura 24 : Proceso UDP Hole Punching bajo un NAT común

El cliente A tiene establecida una sesión UDP con el servidor C, y el NAT le asignó a A el puerto 12345, también el cliente B tiene establecida una sesión UDP con el servidor C y el NAT le asignó el puerto 54321.

Entonces si el cliente A utiliza la técnica Hole Punching para establecer una sesión con el cliente B, usará al servidor C como mediador.

Los pasos a seguir son los siguientes:

1.- El cliente A envía al servidor C un mensaje de petición de conexión hacia el cliente B.

2.- C responde a A con las respectivas direcciones pública y privada de B (200.80.37.15:54321 y 192.168.1.123), al mismo tiempo C envía hacia B las direcciones pública y privada de el cliente A (200.80.37.15:12345).

3.- Luego ambos clientes comienzan a enviar datagramas UDP directamente a su contrario. Los mensajes pueden no alcanzar su destino, dependiendo si el tipo de NAT soporta *Hairpin Translation*. Los paquetes podrían alcanzar a los respectivos clientes, sin embargo la conexión no será tan rápida si pasan a través del NAT.

4.- Finalmente los clientes deben identificar que ambos pertenecen a la misma red privada (ambos poseen la dirección IP pública 200.80.37.15), por lo tanto pueden establecer una comunicación privada directa (192.168.0.1 - 192.168.1.23).

b) Utilizando UPD Hole Punching en clientes con distintos NAT

Supongamos ahora que tenemos dos clientes desconocidos entre sí, bajo un NAT diferente como se muestra en la figura 25. Además tenemos un servidor C que actúa como mediador entre ambos peers.

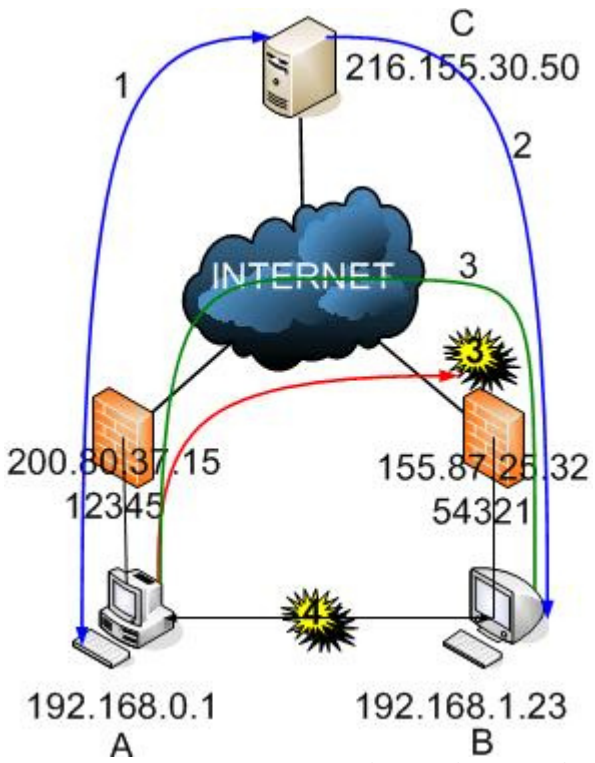

Figura 25: Proceso UDP Hole Punching bajo un NAT diferente

Al igual que en el ejemplo anterior A y B ya tienen iniciada una comunicación UDP con el respectivo servidor mediador C. El NAT A asigna al cliente la dirección 200.80.37.15 y puerto 12345 para establecer la comunicación con el servidor C. Por otra parte el NAT B asigna la dirección 155.87.25.32 y puerto 54321 para establecer la comunicación con el servidor C.

1.- El cliente A envía un mensaje de registro hacia C, reportando su dirección privada, al mismo tiempo el servidor obtiene la dirección pública revisando el encabezado del mensaje UDP, por otra parte el cliente B envía también un mensaje de registro hacia C reportando la misma información.

2.- Ahora el cliente A comienza el proceso de Hole Punching para establecer una comunicación UDP directamente con el equipo B.

Primero el equipo A solicita a C la dirección de B, una vez que esta dirección es enviada, el equipo C envía hacia B la dirección de A.

Cuando finaliza este paso, los equipos A y B conocen las direcciones de sus contrarios.

3.- Con el paso anterior completo los equipos comienzan a enviarse datagramas UDP. Ya que A y B se encuentran en diferentes redes privadas y sus respectivas direcciones no son globalmente ruteables puede o no que los paquetes UDP alcancen su destino, esto dependiendo del tipo de NAT que posean.

4.- Consideremos que el primer mensaje que A envió hacia B, cuando este mensaje sale del NAT A, el NAT avisa que existe un paquete UDP saliendo, la dirección de procedencia del paquete es la misma que posee con el servidor C, pero la dirección de destino es diferente. Si el NAT se comporta como se espera, traducirá todas las sesiones salientes de la dirección privada 192.168.0.1 a la dirección publica 200.80.37.15:12345, aquí es donde el primer mensaje que envía A hacia B crea un agujero en el NAT de A para una nueva sesión UDP, identificada por las direcciones 192.168.0.1, 155.87.25.32:54321 en la red privada de A, y por las direcciones 200.80.37.15:12345, 155.87.25.32:54321 en la Internet.

Si el mensaje que envía A hacia B alcanza el NAT de B antes que el paquete de B hacia A cruce el NAT de B, el NAT de B puede interpretar el paquete de A como no solicitado y rechazarlo. El primer mensaje de B hacia la dirección pública de A al igual que en el proceso anterior abre un agujero en el NAT de B para una nueva sesión UDP identificada por las direcciones 192.168.1.23, 200.80.37.15:12345 en la red privada de B, y por las direcciones 155.87.25.32:54321, 200.80.37.15:12345 en la Internet.

Una vez que el los mensajes de los clientes A y B cruzaron los respectivos NAT, los agujeros en los NAT quedan para que la comunicación UDP sea directa e inmediata, y por lo tanto puede proceder con normalidad. Una vez que finaliza la transferencia el timeout en los respectivos NAT cerrará los Hole Punching que fueron abiertos.

b) Utilizando UPD Hole Punching en clientes con múltiples niveles de NAT

En algunas topologías donde existen múltiples niveles de dispositivos NAT, dos clientes no pueden establecer una óptima comunicación P2P, ya que no existe conocimiento sobre la estructura de topología usada. Considerando este ultimo escenario, como se ve en la figura 26.

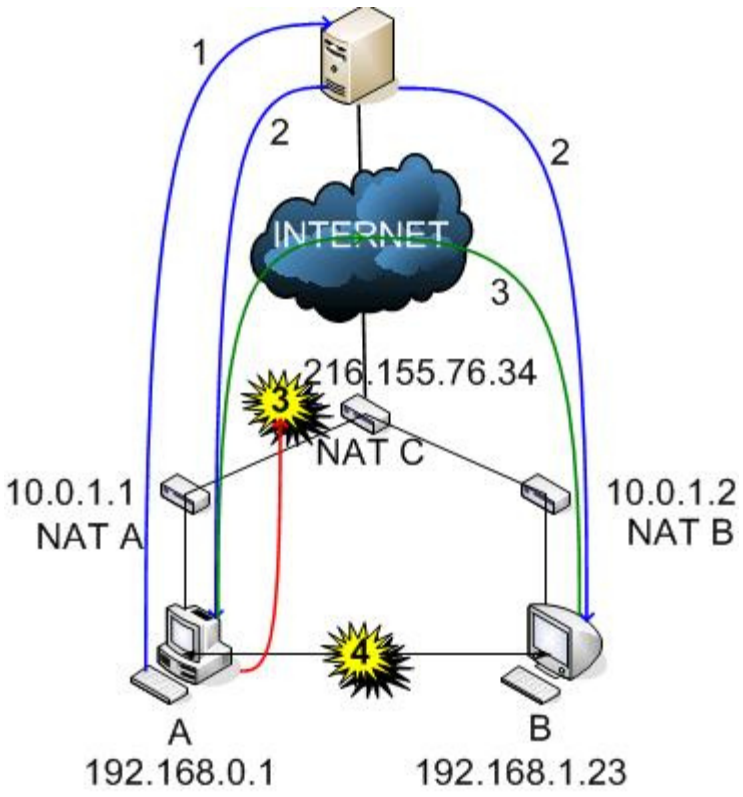

Figura 26: Proceso UDP Hole Punching frente a múltiples niveles de NAT

Supongamos que tenemos un NAT C, que es un gran entorno NAT creado por un ISP (Internet Service Provider) para multiplexar a los clientes con unas pocas direcciones IP. El NAT A y el NAT B son clientes que quieren multiplexar la cantidad de equipos sobre la dirección que les da el ISP. Solo el servidor D y el NAT C poseen dirección IP pública y ruteable.

Las direcciones IP "públicas" usadas por el NAT A y el NAT B, son realmente direcciones privadas del ISP. Mientras que las direcciones de los clientes A y B dependen de sus respectivos NAT. Cada cliente inicia una sesión saliente con el servidor D, causando que el NAT A y el NAT B establezcan una traducción pública - privada. Además causando que el NAT C establezca una traducción pública - privada para cada sesión. Ahora supongamos que el cliente A y el cliente B tratan de establecer una conexión P2P directa vía UDP Hole Punching. El encaminamiento óptimo debería ser desde el cliente A hacia el NAT B (en la dirección semi-pública), designada por la dirección IP 10.0.1.2:55000, y para el cliente B el envío de un mensaje hacia el NAT A (en la dirección semi-pública) 10.0.1.1:45000. Desafortunadamente los clientes A y B no tienen manera de saber cuales son esas direcciones, ya que el servidor D solo observa las direcciones públicas globales de los clientes (216.155.76.34:62005 y 216.155.76.34:62000). Si los clientes A y B tuvieran alguna manera de conocer las direcciones semi-públicas de cada uno, no existe una garantía completa de que podrían ser usadas, ya que estas direcciones asignadas por el ISP pueden entrar en conflicto con las direcciones privadas de los clientes (Ej: la dirección del NAT A y el cliente A podrían tener la misma dirección).

Cuando el equipo A envía un paquete UDP hacia la dirección global de B (216.155.76.34:62005), el NAT A traduce el datagrama de la dirección 192.168.2.5 a la dirección 10.0.1.1:45000.

El datagrama ahora alcanza el NAT C, Si el NAT C actúa como se piensa, traducirá la dirección de origen y destino en el datagrama y envía el datagrama de vuelta a la red privada, ahora con el origen 215.155.76.34:62000 y destino 10.0.1.2:5000. El NAT B, finalmente traduce el datagrama de destino como un datagrama de la red privada de B y el datagrama alcanza a B.

TimeOuts en UDP

Cuando se utiliza UDP como protocolo de transporte, no hay manera de determinar el tiempo de vida de una sesión cruzando un NAT. La mayoría de los NAT asocian un tiempo ocioso con traducciones UDP, cerrando el agujero creado en el UPD Hole Punching después de cierto tiempo. Lamentablemente no existe un estándar para determinar este tiempo, algunos NAT poseen un timeout cortos de aproximadamente 20 segundos.

Si una aplicación necesita mantener activa una sesión UDP vía UDP Hole Punching, ésta tendrá que enviar periódicamente paquetes para que la traducción permanezca en el NAT y no desaparezca. Muchos NAT asocian estos timeouts con sesiones UDP individuales, definidas por equipos particulares, entonces al enviar paquetes UDP para que la sesión permanezca con vida, solo la mantendrá a ella, pero no a las otras sesiones P2P activas.

Es posible realizar algunas acciones para no enviar tráfico excesivo, con el objetivo de mantener varias sesiones P2P activas, una de estas es detectar que si la sesión UDP no se encuentra activa se vuelve a establecer el proceso de UDP Hole Punching "ha pedido".

5.1.4 TCP Hole Punching

Establecer conexiones P2P transversales a través de TCP, cuando ambos host se encuentran detrás de entornos NAT, es ligeramente más complicado que para UDP, pero TCP Hole Punching [10] es muy similar a nivel de protocolo.

A pesar de que este método a través de TCP no es muy conocido, ya que es soportado por algunos pocos dispositivos NAT, cuando el NAT lo soporta, TCP Hole Punching puede llegar a ser tan rápido como UDP Hole Punching y mucho más confiable.

Con lo mencionado hasta ahora, la comunicación P2P a través de TCP cruzando entornos NAT, puede ser mucho más robusta que para UDP. Además el protocolo TCP permite determinar el tiempo de vida preciso de una sesión TCP en particular.

Sockets y Reusabilidad de puertos TCP

El principal desafío que se tiene al crear aplicaciones que utilicen TCP Hole Punching, no es el protocolo , sino la programación de las API. Esto por que las API de Berkeley que trabajan con sockets fueron diseñadas para dar solución al problema de la conectividad cliente / servidor, las API permiten que un flujo TCP sea usado para iniciar una conexión saliente vía connect(), o escuchar por conexiones entrantes vía listen () y accept(), pero no ambos.

Abriendo flujos P2P con TCP

Supongamos que un cliente A quiere establecer una conexión TCP con un cliente B. Los clientes A y B mantienen una conexión TCP activa con un servidor C. El servidor registra cada dirección pública y privada de cada cliente, a nivel de protocolo TCP Hole Punching actúa igual que para UDP:

- El cliente A usa su conexión activa TCP que tiene establecida el servidor C, para que lo ayude a conectarse a B.
- El servidor C, responde hacia el cliente A con la dirección pública y privada de B, al mismo tiempo envía hacia B la dirección pública y privada de A.
- Desde el mismo puerto TCP, que el cliente A y el cliente B utilizan para comunicarse con el servidor C, El cliente A y B tratan asincrónicamente de establecer una conexión a la dirección pública de su contrario, mientras simultáneamente escuchan por conexiones entrantes en los respectivos puertos TCP locales.
- El cliente A y B esperan que sus conexiones tengan éxito, si una de estas conexiones falla, se puede enviar un error como "host no alcanzable" y el nodo simplemente trata de reconectar después de un cierto periodo de tiempo.
- Una vez que la conexión TCP es establecida, los hosts se autentifican entre si, para verificar que la conexión sea la correcta.

A diferencia de UDP, donde se necesita solo un socket de comunicación, para comunicarse con el servidor principal y muchos peers, con TCP cada cliente debe manejar varios sockets a un puerto TCP local, como se muestra en la figura 27.

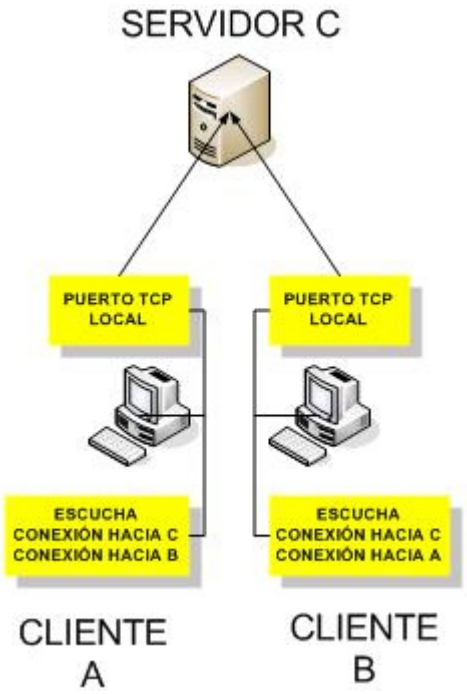

Figura 27: Múltiples conexiones en TCP Hole Punching

Resultados usando Hole Punching

A continuación se muestran los porcentajes de funcionamiento de las técnicas UDP y TCP Hole Punching, según un estudio realizado por Bryan Ford [10], donde se presentaron los NAT más usados:

UDP HOLE PUNCHING	**82% de los NAT**	**TCP Hole Punching**	**64% de los NAT**
Linsys	98,00%	Linksys	87,00%
Netgear	84,00%	Windows	52,00%
Widnows	94,00%	Netgear	64,00%
Linux	81,00%	Linux	67,00%

Tabla 2: Porcentaje de funcionamiento en NAT

5.1.5 Transversabilidad de conexiones TCP a través de Spoofing

¿ Que es Spoofing ?

Al Spoofing se conoce como una creación de tramas TCP/IP utilizando una dirección IP falseada. Al principio esto fue utilizado por hackers, para lograr tener acceso a ciertos host restrictivos. La idea básica es muy sencilla: desde un equipo, se simula la identidad de otra máquina de la red para conseguir acceso a recursos de un tercer sistema que ha establecido algún tipo de confianza basada en el nombre o la dirección IP del host suplantado, el Spoofing sigue siendo usado en la actualidad, aunque ahora se le están dando fines para lograr conectividad P2P.

Llevando el Spoofing al P2P

Como se ha visto, en el Spoofing entran en juego tres máquinas: un nodo A, un nodo B, y un nodo suplantado que tiene cierta relación con el nodo B. Para que se pueda establecer un tipo de relación sabemos que necesitamos falsear la comunicación con el objetivo, y además evitar que el nodo suplantado interfiera en la comunicación. Existen diferentes técnicas para lograr un Spoofing, pero en el caso de la comunicación P2P necesitamos un Spoofing de dirección IP.

En un escenario típico de conectividad, un nodo envía una trama SYN a su objetivo indicando como dirección origen la de esa tercera máquina (que podría encontrarse fuera de servicio) y que mantiene algún tipo de relación de confianza con el otro nodo. En este caso el nodo objetivo responde con un SYN+ACK a la tercera máquina, que simplemente lo ignorará por estar fuera de servicio (si no lo hiciera, la conexión se resetearía y la conexión no sería posible), el nodo original enviará ahora una trama ACK a su objetivo, también con la dirección origen de la tercera máquina.

Para que la conexión llegue a establecerse, esta última trama deberá enviarse con el número de secuencia adecuado; el nodo original deberá predecir correctamente este número: si no lo hace, la trama será descartada, si lo consigue la conexión se establecerá y podrá comenzar a enviar datos a su objetivo.

Podemos comprobar que una conexión P2P a través de NAT utilizando Spoofing no es inmediato; de entrada, el nodo original deberá hacerse una idea de cómo son generados e incrementados los números de secuencia TCP, y una vez que lo sepa, se podrá engañar al dispositivo NAT, utilizando estos números para establecer la comunicación, cuanto más robusta sea esta generación por parte del nodo objetivo, más difícil lo tendrá el nodo origen para realizar la comunicación P2P con éxito.

Además, es necesario recalcar que al usar el Spoofing para establecer una comunicación P2P, se realiza de una manera ciega, ya que el nodo original ve en ningún momento las respuestas que emite su objetivo, ya que estas van dirigidas a la máquina que previamente ha sido deshabilitada, por lo que debe suponer qué está sucediendo en cada momento y responder de forma adecuada en base a esas suposiciones. Quizás por estas razones es que es una técnica muy poco usada, por lo menos en la mayoría de las aplicaciones que poseen transversabilidad.

Como habíamos mencionado, el principal problema que puede plantear el equipo que realiza el NAT, es que la secuencia de predicción de números de secuencia TCP, o sea que presente un esquema de generación robusto, como lo son la mayoría de Unix (aunque muchos de ellos no lo hagan por defecto). Otro problema que se puede presentar son las relaciones de confianza basadas en la dirección IP o el nombre de las máquinas, sustituyéndolas por relaciones basadas en claves criptográficas; el cifrado y el filtrado de las conexiones que pueden aceptar los nodos en cuestión.

Capitulo 6

6.1 Implementación de Transversabilidad

Como solución al problema del envío de un archivo utilizando transversabilidad. Se propuso crear una aplicación P2P, que implementa tres soluciones que resuelven este problema, todo esto utilizando conexiones TCP. Esta misma aplicación se puede utilizar en distintos esquemas de redes, detrás de entornos con Firewalls o dispositivos NAT.

Esta aplicación debe primero realizar una conexión a un servidor central, que va a ser el que se preocupara de mantener la relación entre las direcciones de los distintos nodos que se conecten a este entorno P2P. Dentro de las técnicas usadas tenemos:

1. Conexión TCP utilizando Relaying.

Como describe el capitulo anterior, esta técnica se basa en que el servidor actúa como un Proxy de comunicación, ya que éste traspasa la información recibida por el nodo que inicia la comunicación hasta el nodo que actúa como receptor.

Principal Ventaja:

La gracia que posee esta técnica es que funciona sobre cualquier entorno NAT, a pesar de que ambos clientes se encuentren detrás de entornos NAT, o en múltiples niveles de NAT, esto debido a que la comunicación es realizada hacia un servidor con dirección IP pública, el traspaso de información funciona hacia los dos sentidos sin problema alguno, ya que una vez establecida la comunicación con el servidor, no se requiere que cada nodo intente conectarse a su igual.

Principal desventaja:

Un problema que acarrea esta técnica es que sobrecarga el servidor, esto por que el traspaso de comunicación consume recursos, como procesador y memoria del equipo, además de que consume ancho de banda de la red.

2. TCP utilizando Conexión al Revés.

En este esquema de solución, el nodo que se encuentra detrás del entorno NAT, no puede recibir conexiones directas, por lo tanto, el otro nodo, que no se encuentra bajo NAT, le solicita a través del servidor, que sea éste quien se conecte directamente a él.

Principal Ventaja:

Esta técnica como realiza una conexión TCP directa, sin un servidor intermedio que traspase la información, es muy rápida, y permite un acceso directo y seguro a la información: Ya que si por alguna razón el servidor que ayudó a iniciar la comunicación no responde, la comunicación permanece.

Principal Desventaja:

Este esquema funciona solamente cuando uno de los nodos se encuentra detrás de un entorno NAT, por lo tanto si ambos nodos están detrás de Firewalls, este esquema no funciona, por que ambos tratan de conectarse a su contrario, pero les responde el equipo que esta haciendo NAT, y como éste no tiene capacidad de negociación que posee el servidor, la conexión es rechazada.

3. Conexión directa por detección de dirección.

En este esquema los peers reciben una respuesta del servidor que ayudó a iniciar la comunicación, la cual establece que como la dirección de origen es la misma, los dos nodos se encuentran bajo una red privada común, y bajo este esquema es posible establecer una comunicación TCP directa entre ellos, sin atravesar el NAT que se encuentra sobre ellos.

Principal Ventaja:

La velocidad de la conexión es muy alta, al no tener que pasar a través del NAT. Esta conexión actúa como una conexión servidor / servidor entre los dos peers.

Principal Desventaja:

Si alguno de los nodos posee un Firewall local, que bloquee cualquier petición de conexión, la comunicación no se establece. Este método funciona únicamente si ambos peers se encuentran bajo la misma dirección.

6.2 Modo de funcionamiento del cliente P2P

Al ejecutar la aplicación desde la consola de la forma ./clientep2p &, tenemos nuestra aplicación corriendo como se aprecia en la figura 28.

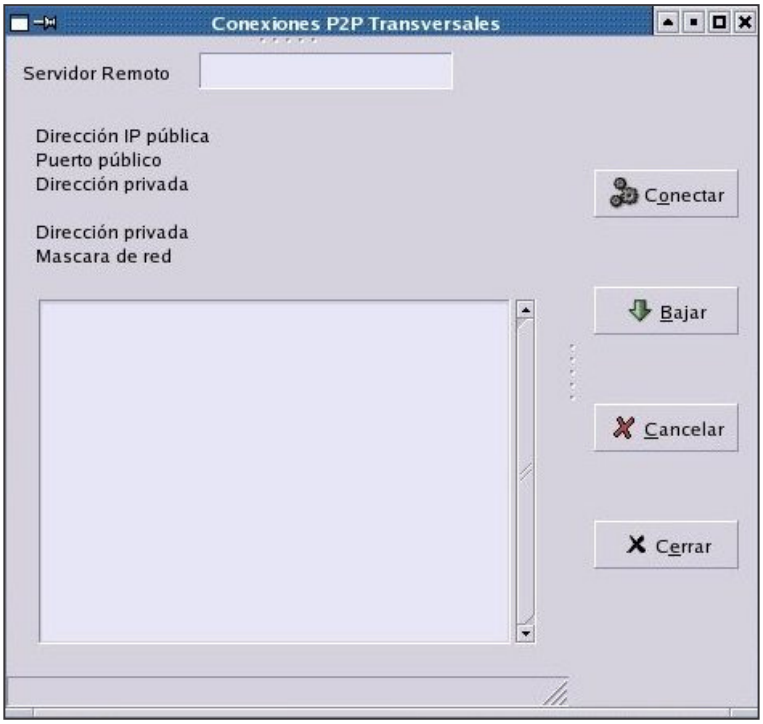

Figura 28 : Screenshoot clientep2p

Al lado del cuadro de texto donde dice "Servidor Remoto", se le indica al cliente cual va a ser la dirección IP del servidor público, que va a mantener la relación entre los peers. Las dos listas de texto contendrán información de la conexión y la lista de archivos compartidos por el otro nodo.

Una vez ingresada esta dirección podemos presionar el botón conectar, éste quedara presionado hasta que un nuevo nodo se conecte a la red.

Una vez que otro nodo ha iniciado el mismo proceso, los datos de la conexión, y la información de cada nodo es enviada a su contrario. Cuando la información es recibida por los dos nodos la aplicación se verá como en la siguiente figura:

Figura 29: Screenshoot clientep2p conectado

La información enviada va contenida en un paquete de 1024 bytes, en este paquete tenemos dirección privada, dirección publica y el listado de archivos compartidos. Los archivos compartidos se encuentran en una carpeta con el nombre shared. Después que el proceso anterior es finalizado se puede escoger que archivo obtener desde el nodo remoto, una vez seleccionado el archivo presionamos el botón bajar, para que se abra la nueva ventana, que va a contener los tipos de técnicas de transversabilidad utilizadas para bajar el archivo, como se ve en la figura.

Figura 30: bajando archivo

Estando abierta esta nueva ventana podemos indicarle a la aplicación que técnica de transversabilidad utilizar. Si la técnica no funciona envía un mensaje de la posible causa y solicita que se intente con otra. Como se aprecia en la siguiente figura:

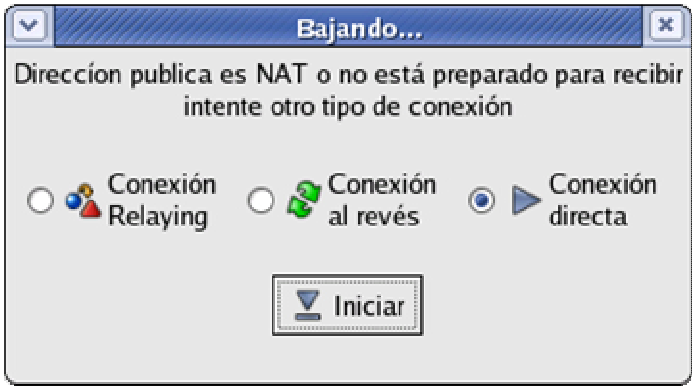

Figura 31: cliente remoto no preparado

Si la conexión es satisfactoria, el archivo procede a enviarse en paquetes de 1024 bytes, el primer paquete contiene el nombre y el tamaño del archivo a enviar, con esto los dos peers conocen el tamaño del archivo y el receptor comenzará a leer cada 1024 bytes, hasta que llegue al termino del archivo, donde se procede a leer la diferencia. Finalmente cuando es recibido se despliega el mensaje de recepción satisfactoria.

Capitulo 7

7.1 Chequeando el tipo de NAT con NAT Check

El programa Natcheck [11], escrito por Bryan Ford, detecta si la traducción que se realiza a cierto host es compatible con protocolos P2P. Si una red está teniendo acceso al Internet de detrás un traductor de la dirección de red de una cierta clase, se necesita conocer el comportamiento de diverso NATs, en términos de cómo, y de si apoyan cierta técnica para permitir una comunicación del tipo P2P entre los nodos anfitriones, particularmente cuando ambos puntos finales se encuentran detrás de entornos NAT.

7.1.1 La Técnica

Supongamos que hay tres anfitriones que se comunican: A, B, y C. El nodo A es un servidor con dirección IP pública, que puede ser visto desde Internet permanentemente, que actúa como relacionador para los otros dos nodos. (por ejemplo, nodo B y el nodo C, reciben las respectivas IP de los otros nodos a través de este servidor A.) B y C se encuentran detrás de entornos con NAT.

Para iniciar una conexión peer to peer con el nodo C, el nodo B envía un mensaje donde solicita una petición de conexión hacia C, usando el nodo A, como habíamos dicho, el nodo A envía un mensaje de respuesta que contiene la dirección IP del nodo C, y el número de puerto UDP de acceso, según el nodo C , además la dirección IP de C y el puerto UDP observado según el nodo A. (Si C se encuentra detrás de un NAT, entonces estas dos combinaciones dirección y puerto serán diferentes.) Al mismo tiempo el nodo A envía al nodo C un mensaje que contiene la dirección IP de B y el puerto de acceso UDP.

Nuevamente las direcciones entregadas por B y los observados por el nodo A serán diferentes si el nodo B se encuentra detrás de un NAT.

Ahora el nodo B y el nodo C, cada uno saben que desean iniciar una conexión uno con otro, y saben cada dirección pública del otro. El nodo B y el nodo C ahora comienzan enviar mensajes UDP directamente el uno al otro, en cada una de las direcciones disponibles.

Si el nodo B y el nodo C se encuentran detrás del mismo entorno NAT, entonces podrán comunicarse con uno directamente entre sus direcciones privadas.

En el caso más común donde los nodos B y C se encuentran detrás de distintos entornos NAT, como se había mencionado originalmente, las direcciones IP privadas son inútiles, ya que es imposible conectarse a direcciones privadas de diferentes dominios.

Las combinaciones de dirección IP y puerto UDP, observadas por A, se pueden utilizar, en este caso para establecer una comunicación directa. Aunque inicialmente la primera reacción del NAT de B, será filtrar cualquier paquete UDP, que llegue desde el nodo C.

El caso en si es muy simple: enviar un datagrama hacia el exterior abre, automáticamente, una regla para aceptar la respuesta. Así que basta con que ambos extremos empiecen a enviar datagramas a las respectivas direcciones IP publicas de los otros peers, con los puertos correctos (origen y destino) hasta que empiecen a recibir los datagramas del otro "peer".

Hay un requisito importante que el NAT debe poseer para que esta técnica trabaje: el NAT debe trabajar de modo que asignen solamente dirección pública y puerto para cada nodo privado. Una sesión en terminología Internet esta definida por las direcciones IP y puerto de cada punto final, así la comunicación del nodo B con el nodo A, se considera una sesión, si la comunicación entre el nodo B con el nodo C sea distinta.

Si el NAT de B, asigna un puerto UDP público, para los nuevos intentos de sesión del nodo B con el nodo C, la técnica anteriormente mencionada, no trabajará por que los mensajes del nodo C hacia el nodo B serán dirigidos al puerto UDP incorrecto.

El RFC 3022 sugiere que los entornos NAT se comporten de esta manera para que se pueda establecer una correcta comunicación P2P, manteniendo solo dirección IP pública y puerto para la dirección privada y puerto, y que esto sea independiente del número de las sesiones activas que implican esta traducción.

Este comportamiento es no solamente correcto para lograr una buena compatibilidad con usos de UDP, sino que además ayuda a establecer de manera ordenada la traducción hacia los nodos privados. Mantener el nodo público con un puerto constante no afecta en ninguna manera la seguridad de la red, por que el tráfico entrante se puede todavía filtrar, sin importar como se traducen las direcciones.

7.1.2 Que chequea en NAT

El programa natcheck.c es básicamente un programa que realiza constantes ping a puertos UDP conocidos, en dos servidores públicos y accesibles desde Internet. Ambos servidores corren el mismo programa "natserv.c", a esto le sumamos un tercer nodo corriendo otro programa, llamado "natbouncer.c".

Siempre que cada uno de los primeros dos servidores reciba una petición UDP, se envía no sólo una respuesta directa remitente de esa petición, sino también envía un mensaje al tercer servidor, que alternadamente "despide" la respuesta de nuevo al cliente original.

Para determinar si el dispositivo NAT, esta funcionando manteniendo una relación dirección pública y puerto – dirección privada puerto, el programa natcheck.c del cliente inicia una secuencia de pings simultáneos a los primeros dos servidores (en caso de que algunas de las peticiones o de las respuestas se pierdan) y comprueba que la dirección del cliente y el puerto del UDP según lo dicho por ambos servidores es igual. Si el NAT asigna un puerto público nuevo para cada nueva sesión, entonces el puerto del nodo de origen será diferente, por lo tanto abría que actualizar el NAT.

Las respuestas del tercer servidor se utilizan para comprobar solamente si el NAT filtra correctamente el tráfico entrante no solicitado. Puesto que el cliente nunca envía cualquier mensaje al tercer servidor, si el NAT está poniendo correctamente una funcionalidad de firewall.

7.2 STUNT

Transversalidad simple para UDP y TCP a través de NAT (STUNT) [12], que extiende la funcionalidad de STUN, ya que incluye la funcionalidad TCP. Es un protocolo liviano, que permite a las aplicaciones que corren detrás de entornos NAT, determinar la dirección IP pública, y las propiedades de asignación de puertos, reglas de filtrado y timeouts asociados con conexiones TCP a través de NAT.

Conociendo estos parámetros, se permiten establecer sesiones entre dos hosts detrás de NAT. Como resultado las aplicaciones P2P pueden funcionar existiendo una infraestructura NAT, sin sacrificar los beneficios que trae TCP.

7.3 Detectando la Transversabilidad

Para verificar que la transversabilidad, esta siendo realizada, habría simplemente que utilizar una herramienta que muestre el estado de las conexiones, y verificar dirección IP de origen y puerto utilizado. Para realizar esto puede utilizar una herramienta llamada Tcpdump (disponible para la mayoría de los UNIX y también para Windows, llamada Windump) son programas cuya utilidad principal es analizar el tráfico que circula por la red. Se apoya en la librería de captura pcap, la cual presenta una interfaz uniforme y que esconde las peculiaridades de cada sistema operativo a la hora de capturar tramas de red.

Lo primero que debemos averiguar cuando estamos usando el Tcpdump, es las interfaces que queremos escuchar. Por defecto cuando se ejecuta sin parámetros, en los Linux se pone a escuchar en la eth0, mientras que en Windows hay que especificarla la interfaz donde quiere escuchar. Para averiguar la interfaces en cualquier Unix recurrimos al comando ifconfig -a el cual nos da una lista de las interfaces que tenemos, así como sus parámetros de configuración.

```
eth0    Link encap:Ethernet  HWaddr 00:08:A1:73:F9:B6
        inet addr:192.168.0.1  Bcast:192.168.0.255  Mask:255.255.255.0
        inet6 addr: fe80::208:a1ff:fe73:f9b6/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:9410 errors:0 dropped:0 overruns:0 frame:0
        TX packets:10217 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:5423498 (5.1 Mb)  TX bytes:1663630 (1.5 Mb)
        Interrupt:10 Base address:0xde00
```

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:270 errors:0 dropped:0 overruns:0 frame:0
          TX packets:270 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:13476 (13.1 Kb)  TX bytes:13476 (13.1 Kb)

ppp0      Link encap:Point-to-Point Protocol
          inet addr:200.126.116.19  P-t-P:200.126.116.1  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1492  Metric:1
          RX packets:8795 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9685 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:5181071 (4.9 Mb)  TX bytes:1417307 (1.3 Mb)
```

Usando los datos del comando anterior, en el Linux si queremos escuchar en la interfaz eth0, usaremos tcpdump -i eth0, que producirá la siguiente salida:

```
[root@leithold tcpdump-2005.07.20]# ./tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
```

Una vez que tenemos el programa corriendo, interpretamos la salida ejecutando nuestro clientep2p, y conectándonos al servidor que mantiene las relaciones de los host. Luego se verá algo como lo siguiente:

17:48:44.655465 IP 200-126-116-19.bk8-dsl.surnet.cl.33647 > 200.85.207.108.clip2p: S 564613916:564613916(0) win 5808 <mss 1452,sackOK,timestamp 184530630,nop,wscale0>

Que corresponde a la solicitud del nodo local 200.126.116.19 hacia el nodo que mantiene la relación entre los peers, en este caso 200.85.207.108. Después deberíamos obtener la respuesta del servidor con el listado de archivos y la información del otro nodo P2P (dirección IP pública y privada), se vería así:

17:48:45.245264 IP 200.85.207.108.clip2p > 200-126-116-19.bk8-dsl.surnet.cl.33648:.ack 1025 win 7168 <nop,nop,timestamp 54972227 18456604>

Finalmente al realizar el traspaso de archivo utilizando relaying, deberíamos obtener múltiples respuestas del tipo:

17:57:35.225101 IP 200.85.207.108.clip2p > 200-126-116-19.bk8-dsl.surnet.cl.33647: . ack 4545619 win 63000 <nop,nop,timestamp 55024865 18983160>

Las cuales corresponderían al envío de cada paquete de 1024 bytes, provenientes del otro clientep2p, que pasa a través del servidor. Si estamos transfiriendo un archivo, utilizando conexión directa, deberíamos tener como dirección de origen la del nodo y no la del servidor de relación. En ese caso podría ser así:

18:45:59.168058 IP 216.155.77.42.1863 > 200-126-116-19.bk8-dsl.surnet.cl.33706:P 102:109 (7) ack 247 win 65289 <nop,nop,timestamp 2308004021887869>

CONCLUSION

El futuro de las redes P2P crece considerablemente, y cada vez son más las aplicaciones que utilizan esta tecnología, y al mismo tiempo aumenta el número de empresas que las usa. El problema del establecimiento de conexiones P2P a través de NAT se mantendrá, mientras no exista un método fijo para la traducción de direcciones, que sea amigable a la mayoría de las técnicas de transversabilidad. Por lo tanto, lo que se debe hacer al desarrollar nuevas aplicaciones P2P, es crear un conjunto de técnicas que apliquen distintos métodos de transversabilidad, de modo que si algún método falla, exista uno de contingencia que de solución al problema. De esta manera, cualquier técnica que se utilice, debería ir acompañada del uso de "Relaying", ya que es la única que funcionará a través de todos los tipos de NAT.

La utilización en conjunto de las técnicas "Conexión al Revés" y "Relaying", son suficientes para que cualquier software P2P pase a través de entornos con traducción de dirección. Aunque teóricamente la utilización del Hole Punching, es la técnica de propósito general mejor evaluada para establecer conexiones P2P, ya que funciona tanto para UDP como TCP, y puede ser implementada sin tener privilegios especiales o información especifica de la topología de red. A pesar de que funciona en un gran número de dispositivos NAT, no funciona en el 18 % de estos, para el caso por UDP, por lo que se debe buscar obligatoriamente una técnica que alternativa, y en este caso la utilización de relaying a través de un SuperNodo es la mejor opción.

Finalmente aunque estas técnicas ayudan en la mayoría de los casos, van a seguir siendo utilizadas hasta que no se implemente completamente el uso de IPv6.

ANEXOS

A.1 TCP funcionamiento y características

El TCP es un servicio de entrega confiable [13], orientado a conexiones. Los datos son transmitidos en segmentos. Orientado a conexiones significa que una conexión debe establecerse antes de que el servidor intercambie datos. La confiabilidad es lograda asignando un número de secuencia a cada segmento transmitido. Se utiliza una confirmación para verificar que los datos fueron recibidos por el otro servidor. Para cada segmento enviado, el servidor que recibe debe regresar una confirmación (acknowledgment, ACK) dentro de un periodo específico de bytes recibidos. Si una ACK no es recibida, los datos son retransmitidos. El TCP está definido en el RFC 793 [13].

El TCP utiliza comunicaciones de flujo de bytes (byte-stream), donde los datos dentro del segmento TCP son tratados como una secuencia de bytes sin límites de registro o de campo. La tabla 3 describe los campos claves en la cabecera TCP.

Campo	**Función**
Puerto origen	El puerto TCP del servidor que envía.
Puerto destino	El puerto TCP del servidor destino.
Número de secuencia	El número de secuencia del primer byte de datos en el segmento TCP.
Número de confirmación	El número de secuencia del byte que el que envía espera recibir del otro lado de la conexión.
Ventana	El tamaño actual de la memoria intermedia TCP en el servidor que envía este segmento TCP para almacenar segmentos que llegan.
Suma de verificación TCP	Verifica la integridad de la cabecera TCP y de los datos TCP.

Tabla 3. Campos clave en la cabecera TCP.

A.2 Puertos TCP.

Un puerto TCP proporciona una localización específica para entregar los segmentos TCP. Los números de puertos por debajo de 1024 son puertos bien conocidos y están asignados por la Autoridad de Número Asignados de Internet (Internet Assigned Numbers Authority, IANA). La tabla 4 lista algunos puertos TCP bien conocidos.

Número de puerto TCP	Descripción
21	FTP
22	SSH
23	TELNET
80	HTTP

Tabla 4. Puertos TCP bien conocidos.

A.3 La negociación TCP de tres vías (Three-Way Handshake).

Una conexión TCP es inicializada a través de un intercambio de tres vías. El propósito del intercambio de tres vías es sincronizar el número de secuencia y los números de confirmación de ambos lados de la conexión, intercambiar los tamaños de ventana TCP e intercambiar otras opciones TCP tales como el tamaño máximo de segmento. Los siguientes pasos delinean el proceso:

El cliente envía un segmento TCP al servidor con un Número de Secuencia inicial para la conexión y un tamaño de Ventana indicando el tamaño de la memoria intermedia en el cliente para almacenar los segmentos que lleguen del servidor.

El servidor envía de regreso un segmento TCP conteniendo su Número de Secuencia inicial elegido; una confirmación del Número de Secuencia del cliente y un tamaño de Ventana indicando el tamaño de la memoria intermedia en el servidor para almacenar los segmentos que lleguen del cliente.

El cliente envía un segmento TCP al servidor conteniendo una confirmación del Número de Secuencia del servidor. El TCP utiliza un proceso de intercambio similar para terminar una conexión. Esto garantiza que ambos servidores hayan terminado de transmitir que todos los datos hayan sido recibidos. Gráficamente se vería como en la figura 31.

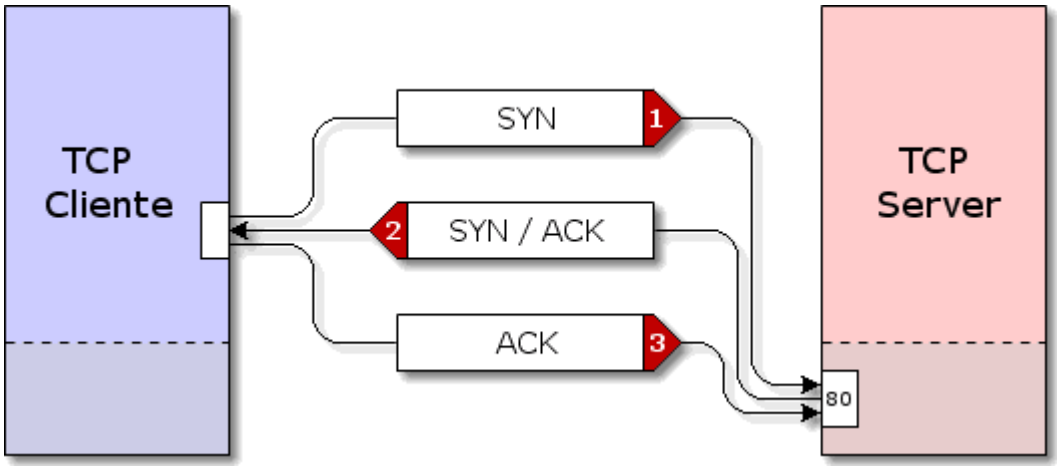

Figura 31:Three-Way Handshake

A.4 UDP Funcionamiento y Características

El UDP proporciona un servicio de datagrama [14] sin conexión que ofrece entrega no confiable, de mejor esfuerzo de los datos transmitidos en los mensajes. Esto significa que la llegada de los datagramas no está garantizada; ni que la entrega de los paquetes esté en la secuencia correcta. El UDP no se recupera de la pérdida de datos utilizando retransmisión. El UDP está definido en el RFC 768 [14].

El UDP es utilizado por aplicaciones que no requieren confirmación de la recepción de los datos y que típicamente transmiten pequeñas cantidades de datos en un momento dado. El servicio de nombres de NetBIOS, el servicio de datagramas de NetBIOS y el Protocolo Simple de Administración de Redes (Simple Network Management Protocol, SNMP) son ejemplos de servicios y aplicaciones que utilizan el UDP. La tabla 5 describe los campos clave en la cabecera UDP.

Campo	Función
Puerto origen	Puerto UDP del servidor que envía
Puerto destino	Puerto UDP del servidor destino.
Suma de verificación UDP	Verifica la integridad de la cabecera UDP y de los datos UDP
Número de confirmación	El número de secuencia del byte que el que envía espera recibir del otro lado de la conexión.

Tabla 5. Campos claves en la cabecera UDP.

A.5 Puertos UDP

Para usar el UDP, una aplicación debe proporcionar la dirección IP y el número de puerto UDP de la aplicación destino. Un puerto proporciona una localización para los mensajes que se envían. Un puerto funciona como una cola de mensajes multiplexada, significando que puede recibir múltiples mensajes a la vez. Cada puerto está identificado por un número único.

Es importante notar que los puertos UDP son distintos y separados de los puertos TCP, incluso aunque algunos de ellos usen el mismo número. La tabla 6 lista algunos puertos UDP bien conocidos.

Número de puerto UDP	Descripción
53	Petición de nombre para el Sistema de Nombres de Dominio (Domain Name System, DNS)
69	Protocolo Trivial de Transferencia de Archivos
137	Servicio de datagrama NetBIOS
161	Protocolo Simple de Administración de Redes

Tabla 6. Puertos UDP bien conocidos

B.1 Por que es tan usado el NAT

Conexiones con módem a Internet

La mayoría de los ISP (Proveedor de Servicios de Internet) le dan una sola dirección IP cuando se conecta con ellos. Puede enviar paquetes con cualquier dirección que le plazca, pero sólo obtendrá respuestas a los paquetes con esa IP de origen. Si se desea utilizar varias máquinas diferentes (como una red casera) para conectar a Internet a través de un enlace, necesita NAT.

Varios servidores

Puede que se quiera cambiar el destino de los paquetes que entran en una red. Con frecuencia esto se debe (como antes), a que sólo tiene una dirección IP, pero se desea que la los nodos externos sean capaces de llegar a las máquinas detrás de la que tiene la IP "real". Si se rescribe el destino de los paquetes entrantes, a través de NAT, se podrá conseguir. Una variante común de esto es el balanceo de carga, en la cual se toma un cierto número de máquinas, repartiendo los paquetes entre ellas. Este tipo de NAT se llamó reenvío de puerto (port-forwarding) en anteriores versiones de Linux.

Proxy transparente

Hay veces donde se deseará simular que cada paquete que pase por su máquina Linux esté destinado a un programa en la propia máquina. Esto se utiliza para hacer Proxy transparente: un Proxy es un programa que se pone entre su red y el mundo real, filtrando las comunicaciones entre ambos. La parte transparente se debe a que su red nunca tendrá por qué enterarse de que está comunicándose con un Proxy.

B.2 Configuración básica de un servidor NAT (Network Address Translation) con IPTABLES para pasarela a Internet en una Lan

En Linux tenemos una herramienta que no es tan visual, pero es más potente que cualquier software para Windows. Esta herramienta se llama IPTABLES, y es una herramienta de control, es decir, nos sirve como router y lógicamente también como firewall, ya que podemos establecer perfectamente las peticiones que queremos denegar si algún nodo se intenta conectar por algún puerto peligroso. Esto lo haremos a través de pequeñas reglas trabajando, en modo superusuario.

Se mostrará un ejemplo básico para una Lan pensando en una conexión ADSL por módem con IP DINÁMICA, y dicho ejemplo es fácilmente adaptable por ejemplo, a una RDSI o cualquier otra línea. Supongamos que tenemos un nodo servidor con la dirección IP privada 192.168.0.1 y el cliente (o clientes) de 192.168.0.2 en adelante (configuradas de forma estática). Hay que decir que todos los kernels a partir de 2.4.x traen soporte para iptables, y si se instalan desde las herramientas de configuración de RedHat, Mandrake, etc... no hay que preocuparse de cómo activarlo en el arranque. En los kernels de la versión 2.0.x se usaba ipfwadm, y en los 2.2.x ipchains. Para comprobar que efectivamente tenemos iptables cargado, podemos ejecutar:

```
[root@leithold sender]#/sbin/chkconfig --list | grep iptables
```

Siempre y cuando tengamos la herramienta grep instalada. Aparecerá una línea con un formato similar:

```
iptables 0:desactivado 1:desactivado 2:activo 3:desactivado 4:activo 5:desactivado
6:desactivado
```

Suponiendo ya que tenemos IPTABLES correctamente instalado, vamos a establecer unas reglas básicas. Solamente se pretenderá conocer lo básico y tener el ruteo funcionando, pero IPTABLES admite reglas de una complejidad considerable, además de un alto número de parámetros. Para tener información sobre ellos sólo hay que teclear "man iptables".

Primero vaciamos todas las reglas que pudiera haber anteriormente:

```
iptables -flush
iptables -table nat -flush
iptables -delete-chain
iptables -table nat -delete-chain
```

Activamos FORWARDING y MASQUERADING :

```
iptables -table nat -append POSTROUTING -out-interface ppp0 -j MASQUERADE
iptables -append FORWARD -in-interface eth0 -j ACCEPT
```

Como se ve se a supuesto que la tarjeta de red eth0 es la que lleva a la red local. Si se tuviera otra tarjeta de red, porque estuviera conectada al módem, habría que asegurarse de que le introducimos la correcta. Es muy importante no confundir la numeración de las tarjetas de red. Atención a que se presupone que cuando nos conectamos se crea la conexión ppp0, aunque esto tiene variantes. Solo hay que mirar con el programa que usemos para conectar. Si por ejemplo es raspppoe, se ve con el comando adsl-status.

Ahora se activa el soporte de "packet forwarding" en el kernel:

```
echo 1 >/proc/sys/net/ipv4/ip_forward
```

Para asegurar de que todo va bien, podemos hacer una comprobación:

cat /proc/sys/net/ipv4/ip_forward

Esto debe imprimir en pantalla un 1. Bien, ahora haremos:

service iptables save

Con ellos guardamos las reglas que hemos definido.

Con esto ya tenemos nuestro NAT funcionando, ahora deberíamos poder conectar desde un nodo de la Lan asegurándose de que tenemos nuestro Gateway o Puerta de Enlace correctamente configurado. Simplemente se trata de decirle a los nodos de la Lan la pasarela para realizar sus peticiones.

En Linux: Dependiendo de la utilidad que usemos para configurar la red según la distribución (en casi todas es Linuxconf, entre otras), pero básicamente consiste en buscar el campo Puerta de Enlace (Gateway) y colocar la dirección IP del nodo que hace el NAT, por ejemplo 192.168.0.1. Después buscamos DNS y ponemos la misma: 192.168.0.1. Finalmente se debe reiniciar la red.

C.1 Principales funciones en C utilizadas

Usada por el nodo relacionador de equipos para abrir un socket.

```
int AbreSocket (char *Servicio)
{       struct sockaddr_in Direccion;
        struct sockaddr Cliente;
        socklen_t Longitud_Cliente;
        struct servent *Puerto;
        int Descriptor;
        Descriptor = socket (AF_INET, SOCK_STREAM, 0);
        if (Descriptor == -1)
                return -1;
        Puerto = getservbyname (Servicio, "tcp");
        if (Puerto == NULL)
                return -1;
        Direccion.sin_family = AF_INET;
        Direccion.sin_port = Puerto->s_port;
        Direccion.sin_addr.s_addr =INADDR_ANY;
        if (bind (Descriptor,(struct sockaddr *)&Direccion,
                        sizeof (Direccion)) == -1)
        {       close (Descriptor);
                return -1;
        }
```

```
        if (listen (Descriptor, 1) == -1)
        {       close (Descriptor);
                return -1;
        }
        return Descriptor;
}
```

Usada por el nodo relacionador de equipos y por los nodos clientes para aceptar peticiones de conexión.

```
int AceptaCliente (int Descriptor,char n)
{       socklen_t Longitud_Cliente;
        struct sockaddr_in Cliente;
        int Hijo;
        Longitud_Cliente = sizeof (Cliente);
        Hijo = accept (Descriptor, (struct sockaddr *) &Cliente, &Longitud_Cliente);
        if (n == 1)
                { strcpy(ip1,inet_ntoa (Cliente.sin_addr));
                  port1 = Cliente.sin_port;
                }
        else
                { strcpy(ip2,inet_ntoa (Cliente.sin_addr));
                  port2 = Cliente.sin_port;
                }
```

```
        if (Hijo == -1)
                return -1;
        return Hijo;
}
```

Usada por los nodos clientes para establecer una conexión con el nodo relacionador de equipos.

```
int AConexion (char *Host_Servidor,char *Servicio)
{       struct sockaddr_in Direccion;
        struct servent *Puerto;
        struct hostent *Host;
        int Descriptor;
        Puerto = getservbyname (Servicio, "tcp");
        if (Puerto == NULL)
                return -1;
        Host = gethostbyname (Host_Servidor);
        if (Host == NULL)
                return -1;
        Direccion.sin_family = AF_INET;
        Direccion.sin_addr.s_addr = ((struct in_addr *)(Host- h_addr))->s_addr;
        Direccion.sin_port = Puerto->s_port;
        Descriptor = socket (AF_INET, SOCK_STREAM, 0);
       if (Descriptor == -1)
                return -1;
```

```
        if (connect (Descriptor,(struct sockaddr *)&Direccion,
                    sizeof (Direccion)) == -1)
                return -1;
        return Descriptor;
}
```

Usadas por todos los nodos participantes, para leer y escribir en los sockets.

```
LeeSocket (int fd, char *Datos, int Longitud)
{       int Leido = 0, Aux = 0;
        if ((fd == -1) || (Datos == NULL) || (Longitud < 1))
                return -1;
        while (Leido < Longitud)
        {       Aux = read (fd, Datos + Leido, Longitud - Leido);
                if (Aux > 0) Leido = Leido + Aux;
                else {  if (Aux == 0)
                                return Leido;
                        if (Aux == -1)  {   switch (errno)
                                                {       case EINTR:
                                                        case EAGAIN:
                                                                usleep (100);
                                                                break;
                                                        default:
                                                        return -1;
                                                }
                        }  }
```

```
        }
        return Leido;
}
int Escribe_Socket (int fd, char *Datos, int Longitud)
{       int Escrito = 0, Aux = 0;
        if ((fd == -1) || (Datos == NULL) || (Longitud < 1)) return -1;
        while (Escrito < Longitud) {
           Aux = write (fd, Datos + Escrito, Longitud - Escrito);
           if (Aux > 0) Escrito = Escrito + Aux;
                else {  if (Aux == 0)
                                return Escrito;
                        else
                                return -1;
                         }
        }
        return Escrito;
}
```

REFERENCIAS

[1] RFC 1631 - The IP Network Address Translator (NAT)

http://www.faqs.org/rfcs/rfc1631.html

[2] RFC 3489 - STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NAT)

J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy.

http://www.faqs.org/rfcs/rfc3489.html

[3] NAT Traversal in SIP

Baruch Sterman, David Schwartz

http://www.sipcenter.com/sip.nsf/html/WEBB5YN5GE/$FILE/SIPNATtraversal.pdf

[4] Peer – to - Peer Computing

Dejan S. Milojicic, Vana Kalogeraki

HP Laboratories

http://www.hpl.hp.com/techreports/2002/HPL-2002-57.pdf

[5] Programas de Intercambio de Ficheros

Carlos Moyano Garalut

http://studies.ac.upc.edu/FIB/CASO/seminaris/2q0102/M3.ppt1

[6] Skype – Guide for Network Administrators

http://www.skype.com/security/guide-for-network-admins.pdf

[7] Proyecto SETI@HOME

http://setiathome.berkeley.edu/tech_news.php

[8] Proyecto JXTA™

http://www.jxta.org/

[9] Funcionamiento de eMule

http://www.emule-project.net/home/perl/help.cgi?l=17

[10] Peer-to-Peer Communication Across Network Address Translators

Bryan Ford, Pyda Srisures

http://www.brynosaurus.com/pub/net/p2pnat.pdf

[11] NAT Check

Bryan Ford

http://midcom-p2p.sourceforge.net/

[12] Simple traversal of UDP through NATs and TCP too (STUNT)

Saikat Guha, Paul Francis.

http://nutss.gforge.cis.cornell.edu/

[13] RFC 768 - User Datagram Protocol

http://www.faqs.org/rfcs/rfc768.html

[14] RFC 793 - Transmission Control Protocol

http://www.faqs.org/rfcs/rfc793.html

Lectura

Traditional IP Network Address Translator (Traditional NAT)

http://www.faqs.org/rfcs/rfc3022.html

TCP connections for P2P apps: A software approach to solving the NAT problem.

Jeffrey L. Eppinger.

http://reports-archive.adm.cs.cmu.edu/anon/isri2005/CMU-ISRI-05-104.pdf

NAT and peer-to-peer networking

Dan Kegel. Julio 1999.

http://www.alumni.caltech.edu/~dank/peer-nat.html

Traversing Firewalls and NATs with Voice and Video Over IP

WainHouse research

http://www.tandberg.net/collateral/white_papers/WR-trans-firewalls-nats.pdf

RFC 1180 - TCP/IP tutorial

http://www.faqs.org/rfcs/rfc1180.html