Universidad de Buenos Aires
Facultad de Ciencias Económicas,
Cs. Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática

Trabajo Final

Tema
Redes Peer-to-Peer

Título
Implementación de un sistema para comunicación segura de voz y texto con tecnologías P2P

Autor:
Francisco Silva G.

Tutor:
Juan Devincenzi

2015
Cohorte 2014

Declaración Jurada

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Trabajos Finales vigente (v. 8.7.3) y se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual.

Firmado.

Ing. Francisco Silva Garcés.
DNI: 95.297.439

Resumen

El desarrollo de Internet, se gestó con la industria bélica a finales de la década de los 60, por entonces la idea esencial era establecer una red de nodos descentralizados, distribuidos territorialmente con caminos redundantes. Esta esencia del origen del Internet, se ha perdido a través del tiempo, convirtiéndose en una red de servicios centralizados, que ha crecido en proporciones inimaginables.

El presente trabajo propone volver a la esencia de los orígenes del Internet, para establecer comunicaciones descentralizadas, por medio de una red de pares (Peer-to-Peer) manteniendo los principios de seguridad. Para esto se estudiará el estado del arte de la tecnología Peer-to-Peer y se propondrán herramientas existentes para diseñar un sistema para comunicaciones por voz y texto que se pueda implementar como prototipo con tecnologías de hardware y software de código abierto.

Palabras clave: Peer-to-Peer, P2P, SIP, DHT, seguridad, comunicaciones, gobierno, activistas sociales, VPN, Raspberry PI, ciberactivismo, NAT, P2PSIP, ciberseguridad

Tabla de Contenido

Nómina de Abreviaturas

B2BUA	Back-to-Back User Agent
C/S	Client-Server
DHT	Distributed Hash Table
DNS	Domain Name System
HTTP	Hyper Text Transport Protocol
ICE	Interactive Connectivity Establishment
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange Protocol
IP	Internet Protocol
ISAKMP	Internet Security Association and Key Management Protocol
JXTA	Juxtapose
KBR	Key Based Routing
LAN	Local Area Network
LGPL	Lesser General Public License
LoLO-Peer	Lower Level Overlay Peer
MAC	Message Authentication Code
MANET	Mobile Ad hoc NETworks MGCP
MISE-P2PSIP	Middleware-Independent and SEcure Peer-to-Peer SIP architecture MSCML
NAT	Network Address Translation
NGN	Next Generation Networks
Node-ID	Node Identifier
NSA	National Security Agent
P2P	Peer-to-Peer
P2PNS	P2PSIP Name Service
P2PP	Peer-to-Peer Protocol
P2PSIP	Peer-to-Peer SIP
PBX	Private Branch eXchange
PCAN	Passive Content Addressable Network

PDP	Peer Discovery Protocol
PGP	Pretty Good Privacy
PIP	Peer Information Protocol
PKG	Private Key Generator
PKI	Public Key Infrastructure
PRP	Peer Resolver Protocol
RELOAD	Resource LOcation And Discovery RFC
RSA	Rivest-Shamir-Adleman
RSVP	Resource ReSerVation Protocol
RTCP	Real-Time Control Protocol
RTMs	Reputation and Trust based Models
RTP	Real-time Transport Protocol
RTSP	Real-Time Streaming protocol
RTT	Round Trip Time
RVP	RendezVous Protocol
S/MIME	Secure/Multipurpose Internet Mail Extension
SCTP	Stream Control Transmission Protocol
SDP	Session Description Protocol
SEP	Service Extensible Protocol
SIP	Session Initiation Protocol
SIPS	SIP URI Scheme
SMBR	Selective-Message Buddy Relaying
SMS	Short Message Service
SMTP	Simple Mail Transport Protocol
SRTP	Secure Real Time Protocol
STUN	Session Traversal Utilities for Network Address Translation
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTL	Time To Live
TURN	Traversal Using Relay NAT
UA	User Agent

UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VM	Virtual Machine
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
WAN	Wide Area Network
WG	Working Group
XML	Extensible Markup Languag

1 Introducción

1.1 Antecedentes

Más de 40 años de historia de Internet, y ahora más que nunca se está haciendo conciencia de lo frágil y vulnerable de su naturaleza para lograr comunicaciones seguras desde el punto de vista de la autenticación, confidencialidad e integridad. Las revelaciones recientes del ex-contratista de la NSA Edward Snowden han puesto de manifiesto en la conciencia pública esta realidad.

Tal vez en ciertas actividades diarias esta situación de alerta ante la violación desmedida de la privacidad, pasa sin dejar huella y no cambia significativamente la manera como se comunica la gente a través de Internet. Sin embargo, este atentado a la soberanía que ya nada tiene que ver con fronteras o límites geográficos, es un tema de relevante preocupación sobre todo para sectores de gobierno, activistas sociales, periodistas.

Peer-to-Peer es una tecnología que lleva al menos 40 años de estudio e investigación. El concepto en sí nació junto con Internet, como una red de nodos distribuidos, descentralizados, que comparten recursos al mismo tiempo. Existen múltiples investigaciones realizadas que han dado como resultado un sinnúmero de aplicaciones. Hoy en día uno de los usos más comunes de esta tecnología es la compartición de archivos, sin embargo se está dando un giro hacia la investigación de aplicaciones más trascendentes volcadas hacia la seguridad, sistemas de procesamiento y/o almacenamiento distribuidos, el anonimato, Cloud Computing, criptomonedas.

De aquí surge la necesidad de acercar estas tecnologías hacia usos y sectores específicos que no dependan de terceras partes, volviendo a los orígenes naturales de Internet en la industria bélica, como tecnología de comunicación descentralizada y resiliente, donde la comunicación se hace entre nodos sin intermediarios.

Las múltiples investigaciones realizadas en esta línea han dado como

resultado la generación de redes y aplicativos diversos basados en Peer-to-Peer, orientados a la confidencialidad de las comunicaciones.

En este trabajo se propone, bajo esta premisa, investigar e implementar una solución para comunicaciones seguras entre dos pares remotos, a través de una red privada Peer-to-Peer en base a tecnologías recientes ya desarrolladas, que pueda encapsularse en un solo dispositivo portable, de tecnología abierta.

1.2 Objetivos

El objetivo de este trabajo es presentar un prototipo utilizando software y hardware de código abierto para implementar un equipo de comunicaciones seguras por voz y texto entre pares ubicados en distintos puntos geográficos, por medio de la implementación de tecnologías Peer-to-Peer, aplicando los criterios de autenticación, confidencialidad e integridad.

Para la realización del trabajo se deberán alcanzar los siguientes objetivos:

- Investigar el estado del arte en tecnologías Peer-to-Peer y los productos/herramientas implementados.
- Realizar una selección de las herramientas/productos encontrados en base a los criterios de seguridad.
- Integrar las tecnologías idóneas en términos de seguridad para implementar un prototipo hardware y software para comunicación segura.

1.3 Enfoque y relevancia

Las comunicaciones están experimentando una crisis importante en términos de seguridad. Los mecanismos de autenticación, integridad y confidencialidad deben ser el eje fundamental de todo sistema de comunicación sensible, en esto las primitivas de cifrado de información cumplen un papel fundamental, sin perder de vista que en el mercado existen una variedad de productos comerciales que ofrecen

alternativas diversas y fantásticas.

A pesar de ello, no pasan del ámbito fantástico o incluso “mágico”, que como tal encierra misterio e incertidumbre. Está demostrado que la confianza en cajas negras es un acto de fe, depositado en los fabricantes y todo el sistema comercial-económico que tiene detrás, con múltiples intereses ajenos casualmente a la privacidad.

Con esto se propone soluciones exclusivas de código abierto en todos los aspectos, es decir software y hardware. Especialistas de la comunidad en criptografía coinciden que un criptosistema seguro debe ser de código abierto, a no ser que se cuente con uno propio. Esta sería una manera de encarar esta dualidad que presenta el Internet donde las comunicaciones son libres, abiertas, globalizadas por el poder de la tecnología, y al mismo tiempo limitadas, censuradas por los poderes de vigilancia de ciertas organizaciones.

Las múltiples investigaciones realizadas en tecnologías P2P en los últimos años han trascendido al campo académico, generándose un sinnúmero de bibliografía y proyectos que están a disposición en código abierto. Sin embargo sigue siendo un problema para encarar la conectividad entre pares en un ambiente real, es decir, detrás de un equipo intermedio tal como un router o firewall haciendo NAT. Esto genera inconvenientes a la hora de implementar una de red de pares totalmente descentralizada.

En un ambiente de alta seguridad como el que demandaría por ejemplo una comunicación entre gobiernos, podría presentarse la salvedad de implementar nodos con características especiales que faciliten esta conectividad, hosteado por los mismos gobiernos, que mantengan conectividad estable en Internet para crear una red privada, especializada.

Por otro lado, se podrá experimentar con equipos pequeños y con el poder computacional necesario y de costo muy reducido inferior a los US$50. Esto permitirá realizar un piloto o prototipo para materializar la solución propuesta.

Además, se encuentra a disposición algunos productos de software ya

desarrollados, tales como VPNs P2P, Clientes SIP, con los que se podrá realizar laboratorio para las pruebas de implementación sin necesidad de invertir tiempo en desarrollar algo nuevo, reduciendo el trabajo en implementación.

2 Marco Teórico

2.1 Peer-to-Peer

Empresas como telefónica de España vieron en esta tecnología, una oportunidad más que un problema afirmando que: *"Los usuarios españoles usan la banda ancha fundamentalmente para el P2P, aunque su uso trae quebraderos innegables de cabeza, hay que quedarse con sus aportaciones; de hecho gracias a ellos, se consigue uno de los fines fundamentales de la Sociedad de la Información: compartir información y conocimiento"* [1]

2.1.1 Origen y Descripción

Se dice que su origen fue a finales de la década de los 90s, iniciado por el revolucionario y polémico Napster que nació en 1999 para la compartición de archivos, específicamente MP3, entre usuarios. Sin embargo las redes P2P aunque ya existían, no eran conocidas hasta entonces. En 2 años Napster marcó una revolución sin precedentes en el mundo de la compartición de archivos y sobre esta tecnología surgieron múltiples aplicaciones. Era de suponer que algún día esto ocasionaría un problema, fue así que en el 2001 Napster fue demandada por las discográficas, por lo cual tuvo que convertirse en un servicio de pago y legal al siguiente año, entonces el P2P fue estigmatizado y encasillado.

Años más tarde debuta el servicio de música por *streaming*, es decir, ya no solo descargas sino reproducción en línea a través de un servicio de suscripción que dio sus inicios en el 2008 como una pequeña *start-up* de nombre Spotify, quien no podía costear servidores, para lo cual la tecnología P2P fue su mejor aliada. En el 2011, el 80% del tráfico que generaba su servicio era por medio de las redes P2P.

Es decir, cuando escuchamos música a través de Spotify, se dispone realmente de 3 fuentes: Servidores de Spotify, cache de la PC del usuario, o a través de otro usuario de Spotify por medio de la red P2P; por supuesto con el debido permiso de las principales discográficas con quien tiene acuerdos. [2]

Desde abril 2014, la pequeña *start-up* está dejando de a poco la red P2P para migrar a sus cientos de servidores. Se dice que esta red P2P es una de las más grandes de la Internet, contando a marzo del 2013 con más de 40 millones de usuarios registrados.[3]

Sin embargo, aquí entra un nuevo tema a considerar: muchos usuarios un poco más técnicos, o enterados y preocupados por su seguridad, están descontentos por el hecho de que el servicio no dice nada acerca de su modelo de red híbrida. Es decir, los usuarios no saben que se están descargando y subiendo archivos en su computador y que son un nodo dentro de una red P2P, de hecho el cliente en sus configuraciones no hay nada que lo diga o permita controlar este comportamiento, y por ejemplo decidir si no quiere ser parte de la red P2P y solo acceder a los servidores centrales.

La tecnología P2P ha sido empleada ampliamente, por ejemplo en la industria de los vídeo juegos utilizada exitosamente por años para los juegos en línea entre usuarios, la industria de los servicios de telefonía IP como skype, los servicios de difusión de vídeo, radio, almacenamiento automático de respaldos, las criptomonedas como bitcoin, litecoin, sistemas VPN, e incluso la compartición de recursos de almacenamiento o de cómputo al estilo Grid están siendo estudiadas y desarrolladas en plataformas P2P.

En 1987 surgen varios formatos digitales de audio, uno de ellos el MP3, el cual sería la tecnología que daría la luz a otra tecnología revolucionaria, el P2P. La difusión de este formato se extendería a tal punto que el término “MP3” llegó a superar al término “sexo” entre los buscadores de Internet; se puede decir que esta tecnología catapultó la serie de eventos antes descritos y desde entonces, la tecnología P2P ha seguido su desarrollo e investigación a nivel académico y comercial, a pesar que aún se la sigue conociendo por los usuarios como la revolucionaria tecnología para la compartición de archivos.

Los sistemas P2P han existido ya desde mucho tiempo atrás, tal vez desde los 70s. En ese entonces, IBM había desarrollado su LU6 (*Logical Unit* 6) de sus

sistemas de arquitectura de red. Estos incluían soporte para comunicación de aplicaciones P2P. [4]

Realmente los orígenes de esta tecnología, datan de los 60s junto con ARPANET. El objetivo de esta red física era compartir recursos y documentos entre diferentes unidades de investigación en los EEUU. En ese entonces no existía el paradigma cliente-servidor, por lo tanto los hosts eran tratados de igual manera, se puede decir que esta red era P2P, aunque no era auto-organizada ni estaba encima de otra red (overlay network).

Si somos un poco más estrictos y vamos un poco más atrás, por el siglo XIX encontramos los primeros sistemas P2P consumados; el teléfono y el telégrafo.

El término *peer-to-peer* o P2P, se refiere a la comunicación entre nodos, es decir, entre pares o entre iguales, donde dicha igualdad está dada en su comportamiento o naturaleza y no por sus características físicas, es decir, no existen clientes o servidores; todos los nodos se comportan de igual manera, son clientes y servidores al mismo tiempo (*servents*), aunque pueden tener características físicas distintas que son obvias entre dispositivos, por ejemplo, un celular, una laptop, un PC, una tablet o cualquier otro.

2.1.2 Comportamiento

Las redes P2P operan sobre otras redes, es decir, son redes superpuestas (*overlay network*) usualmente sobre internet (TCP/IP), y sus aplicativos tienen un comportamiento básico como se describe a continuación:

Descubrimiento: La primera tarea que requieren llevar a cabo los nodos, cuando ingresan a una red P2P, es descubrir los servicios, datos, nodos del que dicha red dispone para acceder al recurso deseado.

Ubicación: Todo nodo nuevo que se incorpore en la red P2P requiere obtener la ubicación del nodo que posee el recurso deseado. De la misma manera, el nuevo nodo debe reportar al resto su ubicación y los recursos que posee.

Obtención del recurso: Una vez ubicado el recurso y el nodo que lo posee, se inicia el intercambio o transferencia de datos. En este caso puede presentarse una gran variedad de recursos dependiendo del tipo de red P2P, ya que los recursos podrían ser archivos, conexiones, espacio de almacenamiento, ciclos de cpu, URI, etc.

A continuación veremos lo que hay detrás de las redes P2P, los desafíos que ha tenido que enfrentar, su naturaleza e implicaciones sociales.

2.1.3 Características

"Los sistemas peer-to-peer, son sistemas que consisten de nodos interconectados capaz de auto-organizarse en topologías de red con el propósito de compartir recursos tales como: contenido, ciclos de CPU, almacenamiento y ancho de banda, con la capacidad de adaptarse a fallas y a una población transitoria o temporal de nodos, mientras se mantiene una conectividad y rendimiento aceptables, sin requerir la intermediación o soporte de un servidor o autoridad centralizada global." [5]

Las principales propiedades que caracterizan a una red P2P :

Compartición de recursos: cada nodo contribuye en la compartición de recursos en todo el sistema P2P.

Interconexión: todos los nodos están interconectados con otros nodos en un sistema P2P.

Descentralización: el comportamiento de un sistema P2P está dado por las acciones colectivas de sus nodos, donde no hay un punto de control central.

Simetría: En la operación de un sistema P2P los nodos asumen roles idénticos, es decir, todos los nodos cumplen las mismas funciones.

Auto-organizado: El sistema P2P se mantiene organizado y mejora su organización con el tiempo en base a su conocimiento y operación local en cada nodo, donde ningún nodo domina el sistema.

Escalable: Este es un requisito fundamental en sistema P2P con millones de nodos simultáneos. Considerando que en consecuencia de tal tamaño, el crecimiento en el uso de los recursos por los nodos presenta un comportamiento que no es

precisamente lineal, sin embargo los tiempos de respuesta para proveer estos recursos pueden llegar a presentar un comportamiento lineal.

Estabilidad: Un sistema P2P debe ser estable a pesar de presentarse una tasa muy alta de nodos caídos, y por lo tanto ser capaz de routear determinísticamente entre el resto de nodos activos.

Una de las investigaciones que aportaron al desarrollo y mejora de las redes P2P son los trabajos en materia de tablas hash distribuidas (DHT), algoritmos matemáticos utilizados para ruteo y ubicación de objetos distribuidos por medio de hashing. A partir de ello se derivan algunos diseños DHT tales como Chord, Tapestry, Pastry, CAN, Kademlia entre otros.

2.1.4 Arquitectura

Si bien los sistemas P2P no son centralizados, cabe diferenciar entre sistemas descentralizados y distribuidos.

Los sistemas descentralizados no dependen de un único elemento como control central, sin embargo puede existir una entidad que controla representada por un grupo de nodos especiales, a diferencia de los sistemas distribuidos los cuales son la totalidad de sus nodos quienes mantienen el control en un esquema **auto-organizado**, donde no existen nodos dominantes.

Uno de los primeros sistemas P2P, fueron los sistemas de noticias Usenet y el sistema de nombres DNS los cuales siguen un modelo jerárquico.

Los tipos de redes P2P son variados (ver figura 1), hay autores que los clasifican en modelos P2P puros, mixtos o híbridos; otros autores los clasifican en generaciones en base a los primeros aplicativos que se masificaron con fines de compartición de archivos, los cuales en su inicio utilizaban elementos centralizados; otros son más formales al dividirlos por su topología como **estructurados (*Structured*)** o **no-estructurados (*Unstructured*),** esta es la clasificación que se tomará en este trabajo.

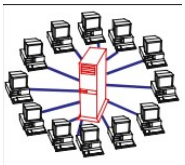
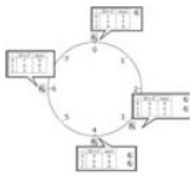
<table>
<tr><th>Cliente-Servidor</th><th colspan="4">Peer-to-Peer</th></tr>
<tr><td rowspan="4">1. Servidor es la entidad central y sólo provee de servicios y contenido.
2. Servidor es el sistema de mayor rendimiento.
3. Cliente es el sistema de menor rendimiento.
4. Cliente es la entidad que solicita y consumo los servicios.</td><td colspan="4">1. Los recursos siempre se comparten entre nodos.
2. Los recursos son accedidos directamente por los nodos.
3. Los nodos son proveedores y consumidores (prosumidores, servents)</td></tr>
<tr><td colspan="3">P2P no-estructurados</td><td rowspan="2">P2P estructurados</td></tr>
<tr><td colspan="2">1ra. Generación</td><td>2da. Generación</td></tr>
<tr><td>P2P Centralizado</td><td>P2P Puro</td><td>P2P Híbrido</td><td>Basado en DHT</td></tr>
<tr><td></td><td>1. No todas las características de P2P incluidas, al haber una entidad central.
2. Es necesario el servidor para proveer servicios.
3. Servidor hace de directorio/índice, los recursos de comparten entre los nodos.
Ej,: Napster.</td><td>1. No todas las características de P2P incluidas, al ser un sistema no muy escalable.
2. Cualquier nodo puede ser removido sin perder funcionalidad.
3. No requiere servidor central.
Ej.: Gnutella 0.4</td><td>1. Todas las características de P2P incluidas.
2. Cualquier nodo puede ser removido sin perder funcionalidad.
3. Entidad central dinámica.. (grupo de supernodos inter-conectados).
Ej.:Gunutella 0.6, Skype</td><td>1. Todas las características de P2P incluidas.
2. Cualquier nodo puede ser removido sin afectar el sistema.
3. No requiere servidor central.
4. Conexiones en la red son fijas.
Ej.: Chord, CAN, Tapestry</td></tr>
<tr><td>  </td><td>  </td><td>  </td><td>  </td><td>  </td></tr>
</table>

Figura 1. Resumen de evolución de los sistemas P2P. [6]

Como en muchas otras tecnologías, los avances en sistemas P2P han sido fuertemente catapultados por la demanda de aplicaciones, de ahí su evolución y clasificación variada, tal es el caso de Napster en la primera generación de sistemas P2P, el cual mantenía un servidor centralizado que hacía de directorio de nodos y recursos de esta manera un nodo sabía cómo conectarse a otros nodos y dónde ubicar los recursos, aunque el intercambio del recurso se hacía punto a punto entre nodos.

Esto presentó un serio problema técnico, y legal: al tener un solo punto de falla, ya que sin el servidor no se podía disponer del servicio, y al encontrarse el servidor dentro de la jurisdicción de EEUU que le valió el posterior cierre de sus

operaciones. Estos inconvenientes y la necesidad de dar continuidad a este tipo de servicio provocó el salto evolutivo al esquema P2P puro, prescindiendo del servidor, sin embargo se presentaron nuevos desafíos. De aquí en adelante estos desafíos han sido tratados de distintas maneras, mientras los sistemas P2P evolucionaban en estructuras más robustas.

Los nodos en una red P2P deben resolver los siguientes desafíos, que han sido constante tema de investigación: *1. cómo hacerse parte de una red P2P (bootstrapping), 2. cómo encontrar el recurso requerido y el nodo que lo contiene*, y *3. cómo distribuir información sobre sí mismos, es decir, su identificación para que puedan ser ubicados y los recursos que dispone*.

El primer desafío radica su complejidad en que de alguna manera un nodo nuevo debe ubicar a un nodo cualquiera conectado a la red P2P para poder ingresar. Considerando la posibilidad que muchos nodos pudieran estar detrás de un NAT, es necesario contar con nodos que tengan conexión permanente y estable, es decir una IP pública estática. Por lo tanto la existencia de un grupo de nodos con esta característica es vital, los cuales deberán ser conocidos por el resto de nodos. Este pequeño detalle hace una de las diferencias entre nodos comunes y supernodos.

Estos supernodos, pueden llegar a cumplir una función adicional de intermediario en el caso en que dos nodos comunes no puedan comunicarse entre sí, debido a las condiciones o configuraciones de los dispositivos que hacen NAT que se explicará más adelante, para lo cual los supernodos actuarían reenviando paquetes (relay) entre dos nodos comunes.

En cambio de la forma como se resuelvan los dos últimos desafíos, se pueden clasificar en redes P2P **estructuradas** o **no-estructuradas**. Las primeras, pueden proceder de una manera determinística por medio de algoritmos matemáticos que darán una estructura a la red, vinculando la identificación del nodo con la identificación del recurso en un espacio de identificación que se propagará de manera determinística en base a estos mismos algoritmos basados en DHT (Distribuited Hash Table).

En base a estos algoritmos los nodos distribuirán su información acerca de sus recursos y su identificación a un grupo de nodos, dependiendo de dichos algoritmos, que podrían ser los más cercanos, basados en sus hashes.

Las redes **no-estructuradas**, lo harán por métodos distintos que no definen una estructura, y podrían utilizar técnicas de *broadcast* limitadas por un TTL (*Time-to-Live*), siendo este *broadcast* total o segmentado. Cada nodo mantiene solo información de sus propios recursos, o en su defecto la información de los recursos de toda la red dependiendo del caso.

Las redes P2P **estructuradas** tendrán la ventaja de ser escalables, ya que cada nodo posee una tabla de ruteo, el cual se mantendrá pequeña independientemente del número de nodos en la red, existiendo cierta redundancia de esta información. Sin embargo requerirá readecuarse cada vez que un nodo entre o salga de la red, ya que su estructura se irá modificando dependiendo del algoritmo DHT utilizado, esto quiere decir que este tipo de redes tendrá un costo alto en cuanto al mantenimiento de las tablas hash.

Al contrario las redes P2P no-estructuradas no son tan escalables, ya que a medida que aumenta la cantidad de nodos la difusión de la información puede generar mucho tráfico, sin embargo la entrada o salida de nodos no requiere reorganización de la estructura.

Sobre estas dos clasificaciones esenciales podrían derivarse muchas más, combinándolas o agregando elementos en su arquitectura que van más allá de su topología brevemente descrita.

Entonces, algunos de los criterios que podríamos utilizar para decidir entre una red P2P estructura y no estructurada podrían ser: *1. qué tan dinámica es la red, es decir, con qué frecuencia los nodos entran o dejan la red y con qué frecuencia varían sus recursos y su identificación*, *2. qué tan grande será la red P2P.*

2.1.5 DHT (*Distributed Hash Table*) y los sistemas P2P estructurados

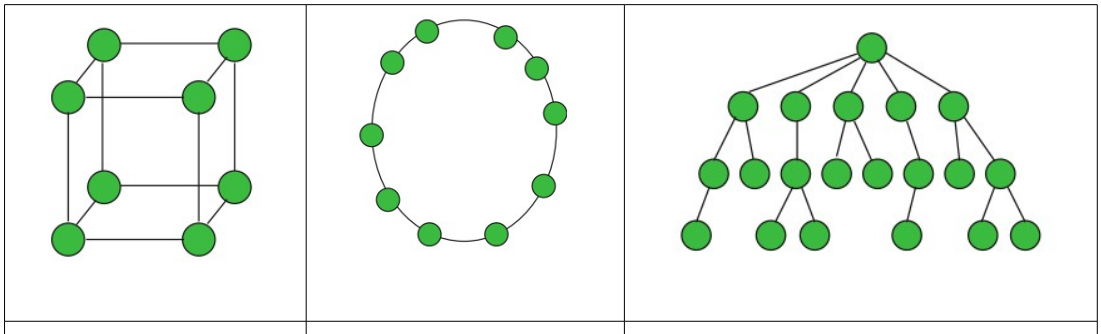
2.1.5.1 Introducción

Las tablas hash distribuidas son sistemas distribuidos descentralizados, tecnología que tiene más de 10 años de estudio y desarrollo en el ámbito académico. Los investigadores han propuesto una variedad de sistemas DHT con distintas mejoras, lo cual da una variada posibilidad de aplicaciones distribuidas.[7]

Las DHT tienen gran relevancia en los sistemas distribuidos, de hecho esta tecnología fue introducida para resolver serios inconvenientes en arquitecturas de gran escala basadas en el modelo Cliente/Servidor que llegaban a congestionarse ante una demanda extrema de recurso, por estar centralizadas.

La dualidad de su tecnología, fundamentada en algoritmos matemáticos de cierta complejidad, proporciona a su vez la simplicidad de su funcionamiento que se resume en sus dos operaciones básicas: GET para recuperar datos del sistema DHT y PUT para introducir datos en el sistema DHT.

Existe una amplia variedad de sistemas DHT que dan diversas estructuras topológicas, las cuales han surgido desde el ámbito académico, por lo tanto hay una diversidad de opciones de estructuras que se adaptan a diferentes necesidades de sistemas distribuidos, tales como: sistemas de archivos, sistemas de búsqueda, almacenamiento de gran escala, compartición de recursos, comunicaciones, entre otros.



CAN	Chord	Patry / Tapestry

Figura 2. Ejemplo de topologías redes P2P estructuradas basadas en DHT[4]

En la actualidad, tanto el ámbito académico como la industria han ido más allá de los aspectos teóricos y se han desarrolla plataformas y aplicaciones que son de amplia utilización en infraestructuras en producción.

En este apartado haré una breve introducción a los sistemas DHT, ya que su exploración implicaría más que un capítulo, lo cual saldría del propósito de este trabajo. Para profundizar se puede disponer de libros enteros dedicados al tema, lo cuales se incluyen en la bibliografía.

2.1.5.2 Operación

Las DHT heredan de las tablas hash propiedades importantes en cuanto a la búsqueda y localización de elementos con alta eficiencia, proveyendo un espacio de clave global y abstracto donde los recursos tienen una identificación única. Se entiende por recursos los datos y los propios nodos.

El espacio de clave se entiende por una tupla formada por **clave** y **valor** (K,V), donde la clave refiere al mapeo del dato por medio de una función hash que sirve como identificación, y el valor es el dato original que se está mapeando. Además cada nodo tiene una clave o ID único en el espacio DHT, entonces tanto los datos y el nodo que los contiene pueden ser mapeados en un sistema distribuido.

Aquí es donde entra su simplicidad al momento de operar, ya que en su diseño DHT con sus operaciones básicas puede recuperar información (V) del espacio DHT haciendo un GET de una clave determinada (K), o introducir información haciendo un PUT de un dato (V) utilizando su clave (K). Este modo elegante de operar será siempre el mismo independientemente de la estructura o tipo de DHT.

Entonces, como resultado de esta dualidad tecnológica de complejidad y simplicidad, los nodos en un sistema basado en DHT solo tiene que conocer a una pequeña porción del sistema (otros nodos) el cual le podrá proporcionar de manera

eficiente la totalidad de los recursos que mantiene, sin importar cuán grande pueda ser el sistema, persistiendo sus tres propiedades básicas:

Alta eficiencia: Debido a que un nodo no necesita conocer a todos los nodos de un sistema DHT, con solo una pequeña porción del sistema podrá localizar a la totalidad de los recursos del sistema. Es decir, conocer poco para obtener todo.

Descentralización: el sistema entero necesita a todos sus nodos para funcionar y cumplir la propiedad anterior, es decir, no existe un nodo central o un conjunto de nodos especiales sobre el cual funcione el sistema. Por lo tanto el sistema es, porque todos son.

Escalabilidad: En consecuencia de la propiedad anterior, un sistema DHT puede constituir a un sistema distribuido por el orden de los miles o hasta millones de nodos sin alterar su rendimiento. Sin embargo, un comportamiento usual como es la entrada y salida frecuente de nodos al sistema, puede derivar en un problema, el cual implica pensar en una gestión eficiente del mismo.

Entonces, la esencia de los sistemas DHT se reduce a resolver el problema de **dónde almacenar** y **cómo encontrar cierta información o dato** en un sistema distribuido que prescinde de un ente central que controle o coordine. Para esto se utilizará uno de los algoritmos DHT conocidos para explicar cómo se resuelve este desafío.

Espacio de direccionamiento, podría definirse de algunas maneras: como el tamaño máximo de un sistema DHT, la cantidad de direcciones que puede manejar el sistema DHT, o la cantidad máxima de nodos y/o elementos de datos permitidos; lo cierto es, que típicamente consiste de un valor entero muy grande en el orden de los 160 o 128 bits (n bits), dependiendo del algoritmo, que da un rango de valores desde 0 a 2^n-1.

Entonces la manera de cómo se aprovecha este espacio de direccionamiento, es decir, cómo éste se lo particiona es lo que hace la diferencia entre los distintos algoritmos DHT, dando en la mayoría de los casos una interpretación geométrica a su estructura, a manera de topología, como se muestra en la figura 2.

Por otro lado, los elementos de datos tienen asignado un ID único, cuyo valor es tomado del mismo espacio de direcciones; dicho ID puede ser elegido libremente, sin embargo lo más convencional es generarlo en base a una función hash, por ejemplo en el caso de que el elemento de dato sea un archivo, podría ser el hash del nombre del archivo, el de su contenido, o ambos.

Lo siguiente es determinar quién se hará responsable por ese elemento de dato, es decir quién lo almacenará o al menos sabrá dónde está almacenado. En esto radica la esencia de los sistemas DHT, que han dado vida a una variedad de sistemas y aplicativos distribuidos, en donde el ID del elemento de datos puede ser el hash de: un índice de datos, unas coordenadas geográficas, archivo binario, dirección IP/MAC, un SIP URI.

Toda esta eficiencia y elegancia de resolver la ubicación de un recurso que se quiere poner a disposición de todo el sistema, tiene además que resolver otros problemas a los cuales no son inmunes. Uno de ellos es evitar la sobre carga de uno o varios nodos con respecto a todo el sistema, condición que pudiera presentarse por al menos 3 situaciones:

1. *un nodo determinado gestiona una porción muy grande del espacio de direcciones, es decir, tiene mucha responsabilidad a su cargo*, o
2. *el nodo es responsable de una cantidad muy grande de elementos de datos dentro de su espacio de direcciones*, o
3. *entre los elementos de datos que un nodo es responsable, uno o algunos de esos elementos tienen mucha demanda*.

Para estas situaciones, los sistemas DHT deben manejar un mecanismo de balanceo de carga, ya sea transfiriendo responsabilidad de parte de su espacio de direcciones a otro nodo, o compartiendo la responsabilidad de su espacio de direcciones con otro nodo.

2.1.5.3 Mecanismo de ruteo

Las responsabilidades de un nodo en un sistema DHT radica en mantener una cantidad de direcciones del espacio de direccionamiento, con lo cual:

1. conocerá a una pequeña cantidad de nodos de todo el sistema, es decir, una vista parcial del todo, y *2. conocerá la ubicación de una cantidad de elementos de datos alojado por otros o incluso el mismo nodo*. Por lo tanto, los sistemas DHT implementan una variedad de mecanismos de ruteo (dependiendo del algoritmo), para poder ubicar recursos, ya sean nodos o elementos de datos.

Cuando un nodo demanda de un recurso, esta petición es hecha dentro de la porción de direcciones que conoce, es decir, a sus nodos conocidos. En consecuencia un nodo de ese grupo que recibe la petición la redirecciona a los nodos que éste a su vez conoce, en caso de que no sea responsable del recurso solicitado. Esto es un proceso recursivo que se repite hasta llegar al recurso solicitado. La cantidad de ejecuciones o saltos (s) realizadas se denota por O(m). La eficiencia de los sistemas DHT está dada en la cantidad de estos saltos, que está en el orden de O(log Z) donde Z es la cantidad de direcciones del espacio ($m=2^n-1$), manteniendo siempre una pequeña cantidad de saltos en relación al tamaño total del sistema.

Entiéndase por elementos de datos como el recurso solicitado, el cual el nodo responsable puede tener una referencia de su ubicación o el elemento de dato en sí. De aquí se derivan otra serie de mecanismos, que tienen que ver con la disponibilidad de los recursos, los cuales se detallarán en el presente trabajo.

2.1.5.4 Algoritmo Chord

Este es uno de los algoritmos entre los primeros y más conocidos, publicado en el 2001 por el Tecnológico de Massachusset, diseñado para abordar algunos de los desafíos que implica el mundo P2P. Su estructura topológica es circular, con su espacio de direcciones dispuesta al sentido de las manecillas del reloj. Los IDs tanto de los nodos como de los recursos, están ordenados ascendentemente en ese sentido

del círculo módulo 2^n, donde n determina el tamaño del espacio de direcciones.

Por ejemplo, vamos a suponer un espacio de direccionamiento de n = 6 bits, es decir 2^6 = (64), donde las direcciones van de 0 a 2^6-1 (0 a 63). Esto quiere decir, que el espacio de direccionamiento del sistema Chord de ejemplo podría mantener un tamaño de 64 nodos y de 64 elementos de datos, cada uno ordenado y dispuesto a manera de anillo en el sentido del reloj, como se muestra en la figura 3, donde se representan los nodos como N_i y a los elementos de datos o recurso como K_i (siendo $0 \leq i \leq 2^6 - 1$).

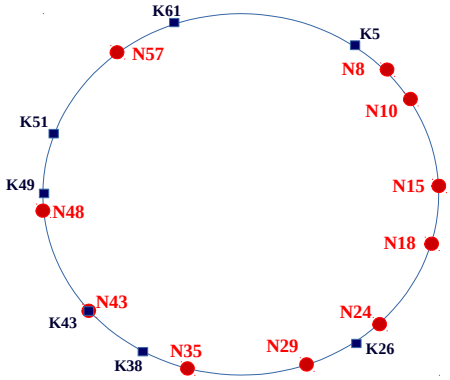


Figura 3. Espacio de direccionamiento Chord

Como se puede notar en la figura 3, tanto los nodos y los elementos de datos están ordenados en el anillo, y partir de ello viene lo clave del algoritmo Chord, en lo siguiente:

1. cada nodo N_i tiene una posición en el anillo y se define como nodo sucesor(N_i) al nodo siguiente de N_i en el sentido del reloj, y como nodo predecesor(N_i) al nodo siguiente en el sentido contrario al reloj.
2. cada tupla (k,v) donde (k) es el ID del recurso o elemento de dato (v), debe ser mantenido por un nodo cuyo ID sea: igual, o el mayor inmediato al ID del recurso k (N_i>=k). A este nodo se le denomina como nodo sucesor(K_i).

Entonces, un nodo N_i es responsable de un pequeño espacio de direcciones que va entre el ID de dicho nodo y el ID de su nodo predecesor(N_i).

3. cada nodo mantiene una tabla de rutas (*finger table*) de este pequeño espacio de direcciones del cual es responsable, dicha tabla de ruta mapea el ID de recurso (k) con el nodo sucesor(K_i) responsable. La tabla tiene un máximo de *n* entradas dado un espacio de direcciones de 2^n-1, es decir $1 \leq i \leq n$. Cada elemento *i* de la tabla de ruteo de un nodo mapea un recurso cuyo ID es el resultado de $N+2^{i-1}$ (donde N es el ID del nodo).

De esta manera, retomando el ejemplo, si el recurso K_{38} es requerido por el nodo N_{10}, este acudirá a su *finger table* el cual tendrá las siguientes entradas:

finger table nodo N_{10}		
i	**ID recurso (K)**	**ID nodo sucesor(K)**
1	$N_{10} + 2^{1-1} = 10 + 1 = 11 = K_{11}$	sucesor(K_{11}) = N_{15}
2	$N_{10} + 2^{2-1} = 10 + 2 = 12 = K_{12}$	sucesor(K_{12}) = N_{15}
3	$N_{10} + 2^{3-1} = 10 + 4 = 14 = K_{14}$	sucesor(K_{13}) = N_{15}
4	$N_{10} + 2^{4-1} = 10 + 8 = 18 = K_{18}$	sucesor(K_{18}) = N_{18}
5	**$N_{10} + 2^{5-1} = 10 + 16 = 26 = K_{26}$**	**sucesor(K_{26}) = N_{29}**
6	$N_{10} + 2^{6-1} = 10 + 32 = 42 = K_{42}$	sucesor(K_{42}) = N_{48}

Preguntará entonces por el recurso K_{38} el cual no lo contiene, entonces debe redireccionar la solicitud a otro nodo, para lo cual buscando en la tabla el recurso con su ID 38 cae entre la entrada 5 (K_{26}) y 6 (K_{42}). Como la entrada 6 excede al valor buscado se recurre a la entrada 5, el cual indica que el nodo sucesor(K_{26}) es N_{29}. Esto quiere decir, que la búsqueda continúa y el requerimiento es pasado al Nodo N_{29}.

Este nodo también tiene su *finger table*, con las siguientes entradas:

finger table nodo N_{29}		
i	**ID recurso (K)**	**ID nodo sucesor(K)**
1	$N_{29} + 2^{1-1} = 29 + 1 = 30 = K_{30}$	sucesor(K_{30}) = N_{35}
2	$N_{29} + 2^{2-1} = 29 + 2 = 31 = K_{31}$	sucesor(K_{31}) = N_{35}
3	$N_{29} + 2^{3-1} = 29 + 4 = 33 = K_{33}$	sucesor(K_{33}) = N_{35}
4	$\mathbf{N_{29} + 2^{4-1} = 29 + 8 = 37 = K_{37}}$	**sucesor(K_{37}) = N_{43}**
5	$N_{29} + 2^{5-1} = 29 + 16 = 45 = K_{45}$	sucesor(K_{45}) = N_{48}
6	$N_{29} + 2^{6-1} = 29 + 32 = 61 = K_{61}$	sucesor(K_{61}) = N_{8}

El sistema pregunta de nuevo por K_{38} al nodo N_{29} del cual no es responsable, entonces debe redireccionar la solicitud a otro nodo, el cual lo busca en la tabla y vemos que K_{38} cae entre la entrada 4 y 5. Entonces de la misma manera tomará la entrada 4, ya que la entrada 5 (K_{45}) excede al valor buscado. La entrada 4 indica como nodo sucesor a N_{43} y el proceso se repite recursivamente hasta encontrar el recurso solicitado. Para buena fortuna de nuestro ejemplo, el proceso termina en el nodo N_{43}, ya que este es el responsable del recurso K_{38}.

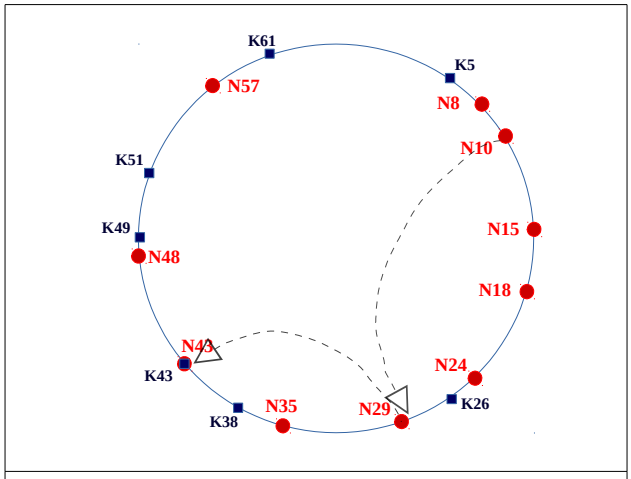


Figura 4. Saltos finger table Chord.

En el ejemplo, N_{10} para ubicar al recurso K_{38} tuvo que dar dos saltos hasta llegar a su destino final, como se ilustra en la figura 4.

2.1.5.5 Ingreso y salida de nodos

Una situación que estos algoritmos deben controlar eficientemente es el ingreso y la salida de nodos, debido a que las *finger table* antes descritas deben actualizarse en todos los nodos para mantener la estructura propia del algoritmo.

Cuando un nodo se une o deja el sistema, un protocolo de estabilización se hace cargo y mantiene actualizada las listas de nodos sucesores y predecesores. Este proceso se hace cada cierto tiempo, debido a la dinámica de estos sistemas, y su reacción es en cadena.

2.1.6 Primeras generaciones P2P, sistemas no-estructurados

Los sistemas de compartición de archivos han sido clasificadas en generaciones; la primera generación son diseños que combinan servidores centralizados con ruteo P2P, las segundas generaciones eran arquitecturas totalmente descentralizadas, la tercera generación podría incluir a los sistemas P2P de anonimato tales como I2P, Freenet, entre otros.

Los orígenes de los sistemas P2P, vienen de los aplicativos de compartición de archivos, que en sus inicios nacieron como sistemas centralizados con el legendario Naspter a finales de los 90s. Este tipo de sistemas mantenía de manera centralizada la meta-data acerca del contenido compartido y un directorio de nodos, es decir, qué archivos estaban disponibles y quién los tenía. Una vez obtenida esta información, los nodos se contactan directamente entre sí para transferir el archivo solicitado.

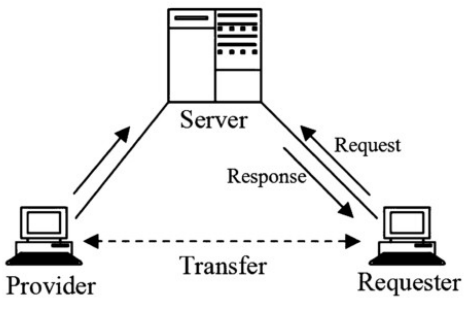


Figura 5. Modelo P2P Centralizado. (Napster) [5]

La segunda generación surge con el modelo distribuido de Gnutella en el 2000, un protocolo abierto que incluye: descubrimiento de nodos, búsqueda distribuida, transferencia de archivo. Los nodos primero deben iniciar la búsqueda de otros nodos para adherirse a la red, esta búsqueda inicial es a manera de *broadcast* a toda la red (Internet) hasta obtener respuesta, con un TTL definido. De la misma manera, para realizar una búsqueda del archivo requerido hará una solicitud que la esparcirá por los nodos hasta obtener respuesta. Gnutella reportó en el junio del 2005 1.81 millones de equipos conectados a la red.

Gnutella2 derivó en un modelo híbrido al igual que Kazaa, para solventar los problemas de un solo punto de fallo del modelo centralizado, y la sobrecarga en la red por el modelo totalmente distribuido. Estos disponían de un grupo de nodos (supernodos) que mantenían el directorio de recursos y nodos disponibles, analógicamente como lo hace un servidor en el modelo centralizado. Cuando un nodo requiere un recurso éste primero consulta al supernodo más cercano, quien responderá indicando quién almacena dicho recurso, caso contrario relega la solicitud a otro supernodo. Los supernodos entre sí harán lo mismo, difundir la solicitud por medio de *flooding* hasta ubicar el recurso solicitado.

A finales de los 90s, el tráfico de internet era mayormente web (65%) y ftp (10%). En el 2006 entre el 50% y 65% del tráfico de descarga, y entre el 75% y 90% del tráfico de subida, eran P2P. El líder de los aplicativos P2P, skype, reportó 8 millones de usuarios concurrente en el 2008, a finales del 2007 reportó 276 millones de usuarios suscritos al servicio. [5]

El modelo utilizado por skype, es híbrido por ser más eficiente dándole mayor carga a los supernodos, quienes son nodos promovidos dinámicamente según criterios como su capacidad de ancho de banda y poder de procesamiento. El componente centralizado de este modelo tiene como función de servidor de registro/login. En este tipo de modelo, es más eficiente derivar las búsquedas a los

supernodos, quienes tienen mayores capacidades.

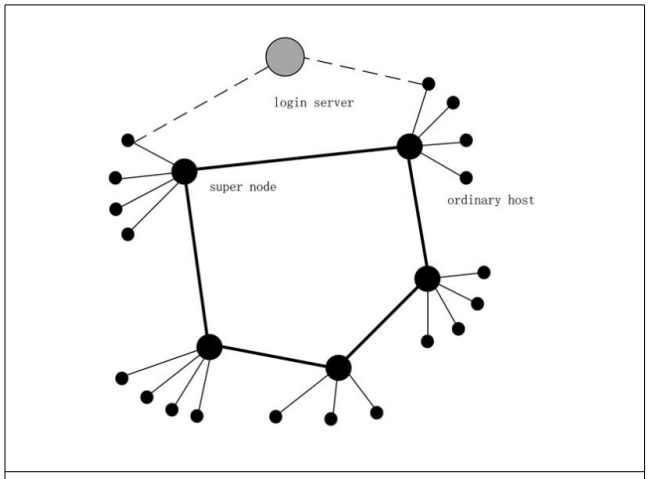


Figura 6. Arquitectura de red P2P de Skype [7]

Es aquí donde estas ideas ya implementadas en las primeras generaciones de P2P, empiezan a fusionarse entre el mundo estructurado y el no-estructurado. Entonces se hace conveniente, en una red con millones de nodos, implementar según sea el caso, un sistema híbrido con supernodos en un modelo estructurado basado en DHT y el resto del sistema en un modelo no estructurado.

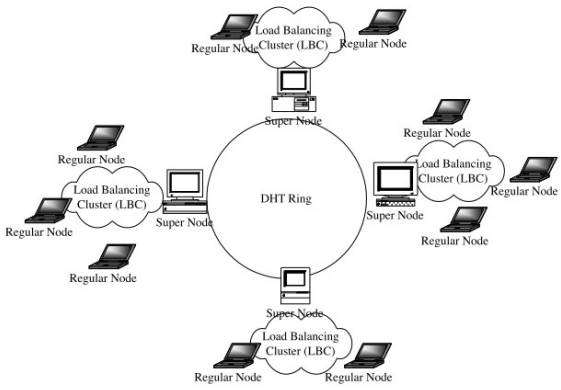


Figura 7. Arquitectura semi-estructurada. [8]

El proceso de *bootstrapping* es sin duda el primer problema que este modelo tuvo que resolver, al abandonar el modelo centralizado. Una alternativa para ello fue

tener nodos bien conocidos dentro de todo el sistema, cuyas direcciones estarían fijas en el software de cada nodo. Otra alternativa fue basada en hacer *flooding*, una especie de *broadcast* hacia la red (Internet) hasta que algún nodo responda.

2.2 NAT y Bootstrapping

Uno de los problemas fundamentales en un sistema Peer-to-Peer es establecer su estado inicial, es decir, el momento en que el primer par de nodos se conectan entre sí para la formación de un sistema P2P, y en consecuencia el mismo acto con respecto al resto de los nodos. A esto se le denomina *bootstrapping*.

En teoría esto no debe ser un problema, pues bastaría con que los nodos que inician conexión pudieran ser alcanzables entre sí, y el resto lo hará el conjunto de protocolos y algoritmos. Sin embargo, en la práctica la mayoría de nodos se encuentran detrás de un *firewall*, *router*, o cualquier dispositivo haciendo NAT.

NAT existe gracias a IPV4, como una manera de soportar la tremenda demanda de nodos ávidos de conexión, los cuales no pueden disponer de direcciones IP pública suficientes, debido a la pobre previsión en el diseño de IPV4 que solo ofrece una cantidad reducida de direcciones. Entonces aparecen las redes públicas, y las redes privadas.

Su relevancia en el mundo P2P radica en las distintas implementaciones de NAT que existen a nivel mundial, lo cual de una manera u otra facilitan o dificultan la conectividad entre nodos, de hecho si hacemos una búsqueda en Internet, podremos encontrar miles de papers que hacen referencia a "*Network Address Translation*", y más de media docena de RFCs haciendo referencia al tema.

2.2.1 Clasificación NAT

En cierta documentación técnica, se puede encontrar como tipología de implementaciones NAT la siguiente: 1. *Full cone NAT, 2. Address restricted cone NAT, 3. Port restricted cone NAT, 4. Symetric NAT*. Podemos decir, que los 3

primeros son del tipo NAT Asimétrico. Sin embargo, esta clasificación no describe claramente el comportamiento de los dispositivos NAT, por lo tanto no se la va a considerar en este trabajo.

En un sistema P2P las dificultades que proporcionan los NAT están en relación a los principios del P2P, ya que naturalmente los rompe debido a las siguientes condiciones:

- IP Publica desconocida para el software local.
- No son posibles las conexiones entrantes, solo salientes.
- Protocolos pueden incluir direcciones LAN en el payload del tráfico WAN.
- Particularmente malo, si ambos nodos están detrás de un NAT.

Naturalmente, algunos de los dispositivos NAT son fácilmente traversos, otros pueden llegar a ser imposiblemente amigables con P2P permitiendo conexiones con otro nodo solo a través de un nodo público que hace de funciones de intermediario (relay).

Entonces, en función de esto se puede establecer una clasificación de nodos en 3 tipos:

- Nodos públicos
- Nodos NAT traversables
- Nodos NAT no-traversables.

Para una mejor comprensión, se resumirá los distintos modos de operación de NAT en una clasificación basada en el RFC-5128, la cual resulta menos compleja en su comprensión que la tipología antes citada, y mucho más clara en función del comportamiento de un NAT.

Para esto, dicho documento presenta dos conceptos: **1. Mapeo del punto final (*Endpoint-mapping*)** y **2. Filtrado del punto final (*Endpoint-filtering*)**. [9]

[10]

El primero se refiere a la forma como el dispositivo NAT mapea, a través de sus puertos del lado WAN, su lado interno (*hosts* de la red privada) con los *hosts* externos que son accedidos (Internet), y cómo esos puertos son reutilizados. Esto último es la clave del comportamiento del NAT en este concepto.

El segundo se refiere a las reglas que aplica el dispositivo NAT para descartar los paquetes que vienen del exterior (Internet) que intentan acceder a su lado interno (*hosts* de la red privada). Este comportamiento es el que determina las propiedades de seguridad del equipo NAT, al establecer las reglas en las que descartará o no un paquete que viene del exterior para contactar a un *host* de la red privada.

Para un detalle más completo del comportamiento del NAT *Mapping* y *Filtering* se puede consultar el RFC-4787.

Entonces en base a estos conceptos se presentan ciertas situaciones que a continuación se detallan.

2.2.1.1 Mapeo del punto final (Endpoint Mapping)

Punto-final Independiente (*Endpoint-independent Mapping*) (EIM-NAT)

El dispositivo NAT reutilizará el mismo puerto mapeado para un equipo determinado de la red privada (IP:puerto), el cual ha enviado paquetes al exterior, independientemente de qué equipo (dirección IP pública) se trate. Es decir, una vez abierto un puerto en el lado WAN del NAT para mapear a un específico *host* de la red privada, siempre se reutilizará el mismo puerto para mapear dicho *host* con cualquier equipo de la red externa.

Punto-final dependiente por dirección IP (*Endpoint Address-dependent Mapping*)

El dispositivo NAT reutilizará el puerto mapeado para un equipo específico de la red privada (IP:puerto), el cual ha enviado paquetes a un equipo específico del exterior (IP pública), sin importar de qué puerto se trate. Es decir, mientras no

cambie la dirección IP del equipo externo al que se está contactando, el puerto del lado WAN del NAT será reutilizado para mapear al mismo *host* de la red privada.

De esta manera, si el mismo *host* de la red privada quiere establecer conexión con un equipo distinto del exterior (una IP pública distinta), el dispositivo NAT abrirá un puerto distinto para realizar el mapeo.

Punto-final dependiente por dirección IP y puerto (*Endpoint Port-dependent Mapping*)

El dispositivo NAT reutilizará el puerto mapeado para un equipo específico de la red privada (IP:puerto), el cual ha enviado paquetes a un equipo específico del exterior especificando además su puerto (IP:puerto). Es decir, el equipo NAT sólo abrirá un puerto distinto en su lado WAN para mapear al mismo *host* de la red privada, si llegase a cambiar la IP del equipo externo contactado, su puerto, o ambos.

En algunas condiciones, dependiendo de la implementación de NAT, se aplicará preservación de puerto (*port preservation*), es decir, que el puerto utilizado por el lado WAN del NAT para el mapeo será el mismo número que el del *host* de la red privada. Sin embargo, en algunas situaciones esto podría producir colisiones y la implementación del NAT tratará de utilizar un puerto distinto. Existen además implementaciones de NAT que no aplican este criterio, es decir son “no *port preservation*”.

La duración de la sesión establecida para el puerto que está mapeando, dependerá de si se mantiene tráfico por él, caso contrario dependiendo de la implementación del NAT tendrá un tiempo determinado (típicamente 5 minutos) después del cual se cerrará.

2.2.1.2 Filtrado del punto-final (Endpoint Filtering)

Punto-final Indepeniente (*Endpoint-Independent Filtering*)

El equipo NAT permitirá el paso de los paquetes a través del puerto mapeado en su lado WAN de todo equipo del exterior, con la condición suficiente de que

cualquier equipo externo haya sido contactado por un *host* del lado interno del NAT, es decir, que permitirá paquetes de respuesta de cualquier equipo externo al *host* de la red privada, que haya establecido conexión con el exterior sin importar a qué dirección.

Punto-final Dependiente por Dirección IP (*Endpoint Address-dependent Filtering*)

El equipo NAT permitirá el paso de los paquetes a través del puerto mapeado en su lado WAN, de un equipo externo específico del exterior (solo IP) que haya sido contactado por un *host* del lado interno del NAT, es decir, que haya establecido inicialmente una conexión con dicho equipo externo.

Punto-final Dependiente por Dirección IP y Puerto (*Endpoint Port-dependent Filtering*)

El equipo NAT permitirá el paso de los paquetes a través del puerto mapeado en su lado WAN, de un equipo externo específico del exterior (IP:puerto) que haya sido contactado por un *host* del lado interno del NAT, es decir, que haya establecido inicialmente una conexión con dicho equipo externo considerando su dirección IP y su puerto.

2.2.2 NAT P2P amigable

Con base a los comportamientos antes descritos, para que un equipo NAT sea P2P amigable, es decir, no se requiera de un equipo externo en la red pública para que dos nodos detrás de un NAT puedan intercambiar información (*relay*), es necesario que cumpla las siguientes condiciones:

- Implementar mapeo de punto-final independiente EIM-NAT (*Endpoint-indepenent mapping*)
- Implementar cualquier tipo de Filtrado (*Endpoint Filtering*)

Estas condiciones corresponden al tipo de NAT Asimétrico que se presenta en alguna literatura técnica. Además de estas dos condiciones, siempre será necesario de un servidor externo que haga de función de registro (*rendezvous*), no de re-transmisión (*relay*).

Si la primera condición no se cumple (EIM-NAT), el servidor externo además de registro deberá hacer función de *relay*, y esto hace que el equipo NAT no sea P2P amigable. A esta condición es que se le denomina NAT Simétrico.

2.2.3 Bootstrapping

Este es uno de los desafíos P2P antes mencionados que implica el descubrimiento inicial de otros nodos del sistema P2P para poder unirse a la red. Con base a lo visto anteriormente, los nodos detrás de un NAT pueden tener problemas para iniciarse en una red P2P, e incluso transmitir datos entre nodos sobre todo si el dispositivo que hace NAT es simétrico.

Se hace necesaria la presencia de nodos (*rendezvous node*) que sean accesibles para cualquier tipo de nodo, sean estos fácilmente traversables o no traversables. Esto implica la existencia de nodos públicos, es decir, nodos que no están detrás de un NAT sino que poseen una IP pública estática.

Estos nodos no necesitan tener funciones especiales, simplemente tener IP pública. Sin embargo, para no caer en un esquema de centralización, la red deberá de disponer de más de un nodo público, tantos como se puedan para evitar un solo punto de fallo (descentralización). Una vez adentro del sistema los nodos podrán acceder al resto para mantenerse en la red.

Esto implica que los nodos conozcan cuáles son esos nodos públicos, que ya están predefinidos, y tener algún mecanismo para actualizar esta lista de nodos públicos en caso de que se incorpore uno nuevo a la red, utilizando direcciones IP predefinidas y estáticas, o por medio de DNS en sus registros SRV.

Existen otras técnicas, para prescindir de estos nodos públicos, como los implementados por Gnutella, el cual recurre al *flooding* para poder descubrir nodos en la red. Este método es NAT dependiente, pues una vez más, en el caso de nodos detrás de un NAT simétrico no podrían aplicar este mecanismo.

El *flooding* realiza un escaneo masivo hacia el Internet, hasta obtener resultado de algún nodo, supernodo, o nodos en general que respondan a la petición, algo similar a un mensaje de *broadcast ethernet*. Desde luego este procedimiento es lento y sobrecarga la red.

2.3 SIP (Session Initiation Protocol)

SIP (*Session Initiation Protocol*), es producto de las investigaciones realizadas en el laboratorio de ciencias de la computación de la Universidad de Columbia, a cargo de Henning Schulzrinne, profesor del Departamento de Ciencia de la Computación, a mediados de los 90. El mismo personaje fue coautor de RTP (*Real-Time Transport Protocol*) especificado en el RFC-3550 para la transmisión en tiempo real de datos por internet. También fue coescritor de RTSP (*Real-Time Streaming Protocol*) una propuesta de estándar especificado en el RFC-2326 para el control de *streaming* de contenido audio/video sobre la web.

Su RFC-3261 fue emitido en el 2001, el cual contiene las especificaciones del protocolo acompañado por los RFC-3262 al RFC-3265 el cual incluyen servicios complementarios al ecosistema SIP. Es a partir del 2001 que se puede ver un rápido crecimiento de la adopción de estos estándares por parte de los fabricantes de equipos de comunicación.

Una característica importante de SIP es que no define el tipo de sesión que se está estableciendo, sino que solo define cómo debe gestionarse la sesión. De esta forma el protocolo SIP es tan flexible que puede ser utilizado en una variedad de aplicaciones y servicios, tales como, juegos interactivos, música y vídeo bajo demanda, voz, vídeo y web *conferencing*. Además es independiente del protocolo de transporte (UDP, TCP, TLS/TCP).

2.3.1 Elementos SIP

SIP utiliza hasta cuatro componentes: *1. SIP User Agents*, *2. SIP Registrar Servers*, *3. SIP Proxy Servers*, *4. SIP Redirect Servers*; los cuales se describen brevemente a continuación:

SIP *User Agents* (UA): Vendría a ser el dispositivo del punto-final (*endpoint*), es decir el dispositivo que utilizará el usuario final para comunicarse, el cual puede ser: un teléfono, celular o cualquier dispositivo móvil, un *softphone*, etc. Con dicho dispositivo se crea o administra una sesión SIP.

SIP *Registrar Servers*: mantienen una base de datos de la ubicación de todos los User Agents (UA) en un dominio dado. Mantienen comunicación con el SIP *Proxy Server*, enviándole o tomando de él las direcciones IP de los UA participantes.

SIP *Proxy Servers*: acepta las solicitudes de sesión realizados por los SIP UA, además solicita al SIP *Registrar Server* información de direccionamiento del SIP UA receptor. Es decir, que se encarga de redireccionar la invitación de sesión al UA receptor, si es que este se encuentra en el mismo dominio, caso contrario lo hará hacia otro SIP *Proxy Server* del dominio al que pertenece el UA receptor.

SIP *Redirect Server*: permite al SIP *Proxy Server* dirigir las invitaciones de sesión a dominios externos.

Estos tres últimos componentes pueden hacer sus funciones en un mismo equipo o hardware. Sin embargo, el SIP *User Agent* (UA) para poder establecer una conexión con otro UA puede prescindir de estos tres últimos elementos estableciendo conexiones directas entre sí, conociendo sus direcciones IP.

La forma como se identifica a un UA es basado en un SIP URL, que tiene una estructura similar a la de una dirección de correo electrónico: sip:usuario@dominio

Características esenciales de SIP, entre otras:

- Reenvío de llamadas.

- Movilidad independientemente del terminal o de la dirección IP.
- Autenticación por parte del llamante o receptor
- Entrega de numeración por parte del llamante o receptor
- Conferencias *multicast*

En términos resumidos, un sistema SIP tendría dos componentes básicos: *el User Agent, y los Servidores*. El *user agent* es el sistema del extremo que actúa a nombre de alguien que quiere participar en una llamada. El *user agent* en general manejaría dos tipos de protocolos: el UAC (*User Agent Client*) y el UAS (*User Agent Server*), el primero utilizado para iniciar una llamada, el segundo para responder una llamada. En la figura 8, se ilustra el flujo del protocolo SIP en una llamada.

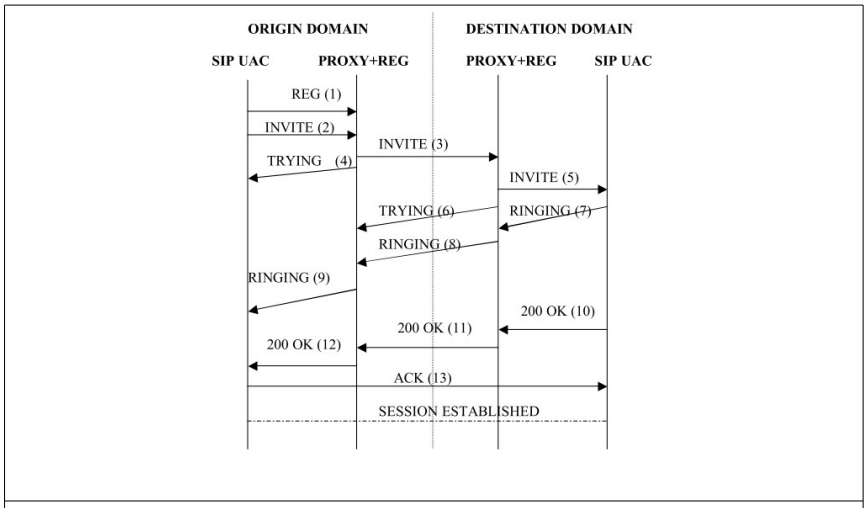


Figura 8: Flujo del protocolo SIP en una llamada.[11]

Cuando ambos protocolos están trabajando en un UA, este tiene activado su modo de operación Peer-to-Peer. Es decir, el UA hace funciones de cliente y servidor al mismo tiempo haciendo posible realizar llamadas directamente entre clientes (UA) sin necesidad de servidores intermedios. Un ejemplo de cómo se produce el intercambio de mensajes entre una comunicación directa entre dos UA sería como se ilustra en la Figura 9.

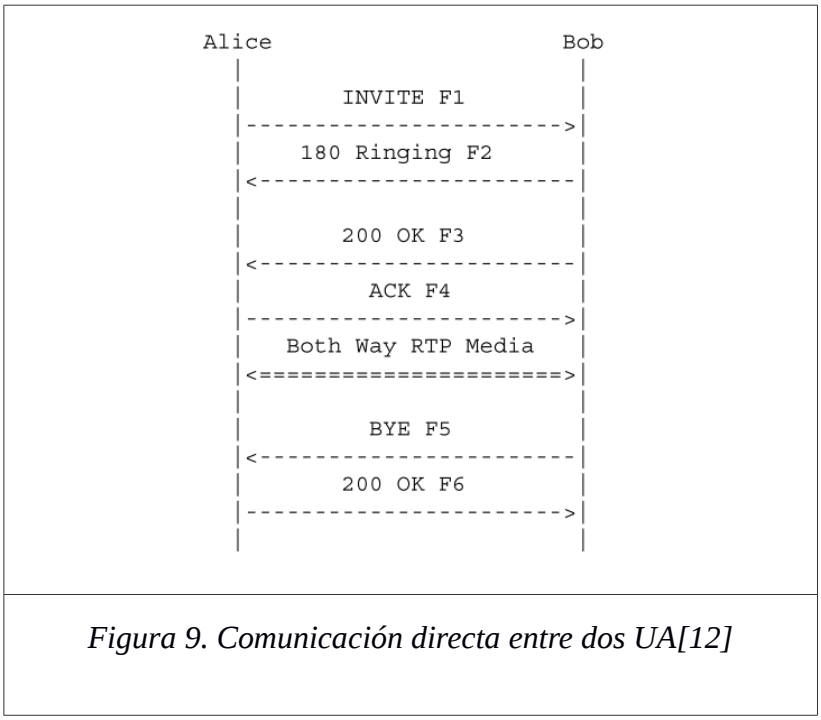


Figura 9. Comunicación directa entre dos UA[12]

SIP es un protocolo en la capa de aplicación diseñado para ser independiente de la capa de transporte, además hereda algunas características del protocolo HTTP y SMTP, por ejemplo las solicitudes SIP son hechas a manera de respuestas codificadas numéricamente, similares a HTTP, de la misma manera SIP mantiene una cabecera y sus mensajes se transmiten en texto claro.

Una descripción de los mensajes que utiliza el protocolo SIP puede verse en el RFC-3261.

2.4 SIP & P2P

El propósito de esta sección es dar a conocer en breve el estado del arte de las distintas posibilidades de esta interesante y novedosa integración. La integración de SIP con las tecnologías P2P son producto de investigaciones de al menos 10 años atrás, con el objeto de aprovechar las bondades de las técnicas de descentralización de servicios. La integración de tecnologías P2P con VoIP ya ha sido desarrollada por uno de los productos de mayor éxito en el Internet, Skype. Sin embargo, este

producto maneja protocolos propietarios y ofuscados.

Por lo cual existen distintas propuestas de estándares para integrar estas tecnologías, desde implementar extensiones al protocolo SIP o implementar nuevos comportamientos en la infraestructura de SIP antes descrita para convertirla a un modelo P2P sin tocar el protocolo, hasta integrar elementos y protocolos distintos ya existentes que pueden adaptarse para implementar un modelo descentralizado. La diferencia entre uno y otro radica en el nivel de interoperabilidad, compatibilidad con sistemas heredados y su portabilidad.

La Internet *Engineering Task Force* (IETF) propuso un estándar sobre esta integración, el cual llamó P2PSIP, que refiere a una arquitectura SIP descentralizada. Para esto fundó en el 2007 la P2P SIP Work Group (WG) el cual, a la fecha del presente trabajo, aún sigue vigente y en estudio.[13]

A la luz de este grupo de trabajo se han propuesto algunas soluciones, las cuales han sido discutidas para definir distintos mecanismos P2P SIP. Por otro lado la Columbia University, a cargo del mismo autor del protocolo SIP, realizó algunos de los primeros trabajos de investigación en el 2003 con su proyecto SIPpeer y el proyecto SOSIMPLE en el campo de los sistemas de comunicación P2P SIP.[14], [15]

Además del ámbito académico la industria también realizó lo suyo, empresas como Cisco, Nokia, HuaWei, entre otras. Muchos de esos trabajos están basados en extensiones del protocolo SIP, o son dependientes de un middleware P2P específico, lo cual los hacen poco interoperables.

La Universidad de Concordia en Canadá propone MISE-P2PSIP, una implementación independiente del middleware P2P, en base a una arquitectura compuesta por SIP proxies auto-organizados y SIP registrar servers distribuidos. [16] La interoperabilidad y la portabilidad son importantes, debido a que ayuda en el desarrollo de aplicativos utilizando librerías y herramientas ya existentes sobre SIP, por lo tanto el evitar extensiones al protocolo SIP y la posibilidad de la reutilización de sistemas P2P existentes, hacen de esta integración mucho más interesante.

Sin embargo, como podemos notar, SIP ha sido concebido con características P2P al permitir la posibilidad de comunicación directa entre los puntos finales (UAC & UAS), donde los elementos de proxy y registrar cumplen con la finalidad de búsqueda, ruteo y servicios adicionales. Estas dos primeras funciones, serían lo único que necesitaría un UA para establecer una red P2P, las cuales podrían implementarse con DHT.

Por lo tanto, existen dos tipos de implementaciones muy discutidas, es decir, ***P2P sobre SIP, o SIP utilizando P2P***. La primera implica utilizar el protocolo SIP para proporcionar un comportamiento P2P, lo cual podría ser un tanto ineficiente ya que para poder proporcionar un comportamiento P2P, SIP necesitaría emplear una cantidad excesiva de mensajes, lo cual podría presentar una sobrecarga para un sistema P2P. Según [16] para la fase de registro se necesitarían un intercambio de 10 o más mensajes SIP para unir un nodo a la red P2P.

El estándar para SIP descentralizado aún en proceso de estudio y desarrollo, por parte de la IETF desde el 2007, propone implementar SIP utilizando P2P como sistema de búsqueda y ruteo, haciendo un uso de SIP prescindiendo de servidores. [17]. En este tipo de implementación, el protocolo SIP como tal se mantendría intacto, lo que varía es la manera como se rutean los mensajes utilizando técnicas P2P.

Otras iniciativas basadas en SIP utilizando P2P que han trabajado en esta integración son: SIPPEER (*A SIP-based P2P Internet Telephony Client Adaptor*) es una propuesta de la Univerisdad de Columbia, el proyecto SIPDHT2 como una implementación *open-source* basada en las librerías sofia-sip utilizando DHT, P2PNS (*P2P Name Service*) un sistema de nombres distribuido que provee un eficiente servicio de resolución de nombres para SIP, RELOAD (*Resource LOcation And Discovery*) propuesto por el *Working Group* de la IETF basado en DHT. [16]

Es un denominador común que las propuestas basadas en utilizar P2P como recurso tecnológico para búsqueda y ruteo son las más trabajadas en la mayoría de proyectos. En un modelo de SIP basado en P2P, cada nodo toma responsabilidad por

las tareas de ruteo y localización del resto de los nodos, en este caso los UA.

Por otro lado, están las propuestas de P2P sobre SIP tales como: Kademlia dSIP por parte de la Universidad de Parma; SOSIMPLE, un sistema de VoIP y mensajería instantánea basada en SIP/SIMPLE y P2P propuesto por la William&Mary College.

En el 2009 la Universidad de Columbia (Canadá) liberó OpenVoIP, un sistema open-source de VoIP y mensajería instantánea sobre P2P, el cual corre en PlanetLab, una red P2P experimental. [18]

SIP utilizando P2P				
Propuesta	**Autor(es)**	**Características Básicas P2P**	**Independiente del P2P middleware**	**SIP estándar sin extensiones**
RELOAD	IETF-WG	SI	SI	NO
Basado en DHT	Columbia University, IETF.	SI	NO	SI
P2PNS	University of Karlsruhe	SI	NO	SI
P2PSIP Jerárquico	Beijing University	SI	NO	NO
P2P sobre SIP				
Propuesta	**Autor(es)**	**Características Básicas P2P**	**Independiente del P2P middleware**	**SIP estándar sin extensiones**
DSIP	IETF, University of Parma	SI	SI	NO
SOSIMPLE	College ofWilliam and Mary	SI	SI	NO
MISE-P2PSIP	Univerisdad de Milan	SI	SI	SI

Tabla 1. Propuestas de integración P2P & SIP [19]

Sin embargo muchas de estas propuestas no han terminado de evolucionar, y en la actualidad no existe un estándar de integración de estas tecnologías. A medida que las propuestas van surgiendo, nuevos requerimientos deben tomarse en cuenta, como por ejemplo la seguridad en las comunicaciones.

En este trabajo, se plantea una propuesta basado en elementos de seguridad como base de las comunicaciones, para una cantidad limitada de nodos en un modelo P2P.

2.5 VPN P2P

Las redes privadas virtuales (*virtual private network*) son infraestructuras de red que permiten a los elementos de una red privada comunicarse entre sí de manera segura sobre redes públicas como Internet. [20]

Existen dos implementaciones de VPN como las más difundidas de protocolo abierto, tales como: IPSec, y las basadas en SSL/TLS (OpenVPN). Para ofrecer un mecanismo que asegure las comunicaciones sobre Internet sin depender de un elemento centralizado, los investigadores y las comunidades de software libre integraron estas tecnologías con protocolos P2P. En esencia, las VPN P2P son redes virtuales privadas que prescinden de un servidor central para poder establecer la red.

A medida que el acceso a Internet se fue masificando en los hogares en la década de los 90, los servicios que ofrecía la red eran cada vez más centralizados y la Internet fue perdiendo su esencia P2P con la que fue gestada. A finales de los 90s junto con esta tendencia fueron surgiendo aplicaciones para compartición de recursos con técnicas P2P, además de la necesidad de agregarle seguridad a la red que desde sus inicios nació rota. Es decir, nunca se contempló, casi de manera deliberada, la posibilidad de proteger la información que por ella pasaba haciéndola insegura desde sus cimientos por diseño.

En la década siguiente, los investigadores y las comunidades de software libre iniciaron proyectos para migrar algunas tecnologías al paradigma P2P, tales como sistemas de: video conferencia, compartición de archivos, chat, incluso las VPN.

Los avances realizados sobre las VPN bajo el paradigma P2P son el eslabón que complementa esta convergencia de tecnologías, a tal punto que incluso las Botnet no se quedaron atrás. Las Botnet P2P más recientes, tales como Zeus, Storm; implementan técnicas utilizadas en las VPNs para asegurar su infraestructura.

Existe una variedad de productos VPN P2P que pueden evaluarse, todos motivados por conseguir una internet libre de la censura y el espionaje, tales como:

- CJDNS cuyo nombre lleva las iniciales de su autor (Caleb James DeLisle) [21] mantiene la red hyperboria basada en esta tecnología. Está compuesta por 3 componentes principales que trabajan en conjunto: switch, router, y CryptoAuth; los cuales tienen como protocolo base IPV6.
- GNUvpe (GNU *Private Ethernet*) que permite emular una red segura sobre Ethernet utilizando mecanismos de clave pública y privada, por medio de las librerías de OpenSSL.[22];
- N2N una VPN P2P de capa 2 sobre capa 3 que a diferencia de la anterior, esta maneja únicamente clave simétrica entre todos los nodos, utilizando el algoritmo Twofish para cifrar. Su código base es muy ligero ideal para dispositivos móviles, y no mantiene dependencias con librerías externas como OpenSSL.[23];
- Tinc es una VPN P2P basada en clave pública y privada para la autenticación entre nodos, y clave simétrica para cifrado de datos. Liberada su primera versión en el 2003, actualmente implementa una adaptación de TLS 1.2 con una suite reducida de cifrado fuerte.
- GNUnet un framework que mantiene como objetivo central la seguridad utilizando tecnologías P2P Estructuradas.[24];
- FreeLan, PeerVPN son otros ejemplos de VPN P2P.

Contrarrestar la vigilancia masiva y la violación a la privacidad son el terreno de estas tecnologías, las cuales han dado origen a la existencia de redes P2P seguras (VPN P2P), tales como: Hyperboria, basada en cjdns; LibreVPN red P2P Argentina, basada en tinc, GNUnet basada en el framework del mismo nombre; en la que se mueven entre otros: periodistas, activistas de los derechos humanos; quienes necesitan evadir toda posibilidad de vigilancia que pueda atentar con las libertades de su accionar, e incluso sus vidas.

2.6 P2P or not to P2P

Definitivamente, esta tecnología representa una nueva perspectiva desde diferentes ópticas en cuanto a las alternativas de interacción humana frente a la tradicional y consumada interacción centralizada.

No existe tecnología alguna que no tenga incidencias en un contexto social, ya sea para bien o para mal. Innegablemente el desarrollo y evolución de esta tecnología plantea cuestiones de índole legal, económicas, sociológicas, políticas. Sobre todo ahora que son temas tan discutidos la privacidad, integridad, soberanía.

Por lo tanto, los sistemas P2P deben entenderse no simplemente como un "sistema de compartición de archivos", sino como una plataforma distribuida que da un nivel de abstracción a la compartición de recursos, que tiene una clara conexión con sus efectos sociales.

En los temas legales, se presenta una curiosa ironía que raya en lo hilarante, pues uno de los impactos más polémicos de los sistemas P2P desde finales de los 90's, ha sido indudablemente en los *copyrights* para lo cual se tomaron muchas acciones legales diligentes y radicales, entre una de ellas el cierre de Napster.

En los últimos 5 años, el comportamiento social sobre los *copyrights* ha dado un giro producto de esta revolución tecnológica. Una muestra de ello es el giro de los modelos de negocio que se están adaptando a la filosofía libre de la esencia del Internet, como el caso de Spotify que provee acceso a música por *streaming* de manera gratuita y legal, a finales de los 90's esto era impensable.

Hoy en día, uno de los derechos violentados más discutidos no son los *copyrights*, sino que están relacionados a las personas. Es decir, la privacidad de las comunicaciones de las personas, la protección de los datos personales, la vigilancia y el espionaje masivo a nivel mundial, violentada por las leyes y los poderes que viene siendo lo mismo, como justificación de por ejemplo los *copyrigths*, la seguridad de las naciones, el terrorismo, etc. y es ahora que irónicamente es la tecnología quien

toma acciones diligentes y radicales por medio de los sistemas P2P.

Desde luego, los *copyrights* representan muchos miles o millones de dólares de toda una cadena de poseedores de propiedad intelectual, tal vez por esto las tecnologías P2P no evolucionaron en el mundo comercial sino en el académico. Podríamos entonces abstraer los sistemas P2P de las concepciones tecnológicas, y recurrir a la metáfora de "*una forma de participación y organización humana*".

Entonces, desde el enfoque de este trabajo los sistemas P2P son considerados como una manera de establecer una plataforma tecnológica que mantenga la confidencialidad y la autonomía, prescindida de un ambiente centralizado que pudiera controlar, generando censura o violación de la privacidad.

Sin embargo, a este nivel los impactos multidimensionales de estas tecnologías pueden llegar a ser tan controversiales y discutibles como el libre albedrío, pues los usos pueden ser múltiples al igual que sus propósitos. Los sistemas P2P han evolucionado para bien o para mal, una muestra de ello son las Botnet, las cuales basadas en estas tecnologías son objeto de investigación y hasta hoy en día existen muchas de ellas que no han logrado desmantelarse y aún siguen generando pérdidas económicas, o incluso violentando la privacidad.

3 Propuesta de implementación

"Tenemos los medios y la tecnología para terminar con la vigilancia masiva, sin necesidad en lo absoluto de acciones legislativas o cambios de políticas. Básicamente adoptando cambios como hacer del cifrado un estándar universal – donde todas las comunicaciones estén cifradas por defecto – podemos poner fin a la vigilancia masiva no sólo en los Estados Unidos, sino en todo el mundo." (Edward Snowden)

Se pretende presentar en este capítulo la implementación del piloto de un sistema de comunicación seguro de voz y texto, que utiliza el paradigma Peer-to-Peer y aplica los conceptos antes mencionados. El objetivo central es mantener la seguridad de las comunicaciones en términos de privacidad, confidencialidad e integridad. Se descarta por lo tanto la presencia de servidores que de alguna manera controlen la comunicación que se esté produciendo entre emisor y receptor.

La solución propuesta es establecer una red privada como base (VPN) sobre la cual fluirán todas las comunicaciones entre los nodos. Esta base es fundamental ya que mantendrá los conceptos de P2P antes descritos, considerará las posibles implementaciones de NAT como se describe en (2.2), detrás del cuál estarán los nodos y mantendrá la comunicación cifrada punto a punto.

3.1 Caso de uso propuesto

Un posible caso de uso podría ser en el ámbito de Gobierno, donde el Jefe de Estado y el Ministro de Relaciones Exteriores mantendrán comunicación con cada uno de los Embajadores del País alrededor del mundo. Cada uno de los Embajadores y las autoridades mencionadas deberán poseer un equipo de comunicación personal, el cual se conectará a la red privada cifrada sin depender de un servidor como gestor de comunicaciones. Cada Embajador que desee comunicarse con sus autoridades o viceversa o los Embajadores entre sí, deberán marcar la extensión correspondiente asignada, la misma que todos podrán acceder desde su dispositivo personal.

Los equipos trabajarán en modo P2P, y estarán auto-organizados, de tal manera que el ingreso y la salida a la red de cualquiera de estos dispositivos no

implicarán cambios manuales en la configuración de los demás.

Se utilizarán componentes de software libre que ya existen, los cuales serán adaptados a la arquitectura que se propone siendo uno de los principios fundamentales de la propuesta mantener tecnologías abiertas, bajo el precepto de soberanía tecnológica.

3.2 Arquitectura y Componentes

Existen una variedad de diseños posibles, sin embargo se optará por la simplicidad de la implementación, conservando los requerimientos de seguridad y el objetivo central que es establecer comunicación de voz y texto entre pares.

Se deberá implementar los componentes sobre un hardware abierto, que podría ser un Raspberry Pi, o cualquiera similar de fabricación nacional. Para el piloto que se levantará en máquinas virtuales, se implementarán los siguientes componentes:

- Sistema operativo Ubuntu 14.04 (UserAgent)
- Sistema operativo Debian 7.6 (NAT device, Supernodos)
- Tinc (VPN Peer-to-Peer Unstructured)[25]
- Avahi (Implementación de Zeroconf para Linux)[26] [27]
- Ekiga 4.0.1 o Blink (SIP User Agent con soporte para Zeroconf)[28]
- Protección contra DDoS para supernodos (Port-Knocking). [29]

Como se muestra en la figura 10, la “capa P2P” de la arquitectura está soportada por la combinación de dos herramientas, la VPN Peer-to-Peer y el sistema Zeroconf para publicación de servicios. Ambos elementos comparten funcionalidades con su capa inferior y superior respectivamente para construir una red superpuesta autónoma y auto-organizada.

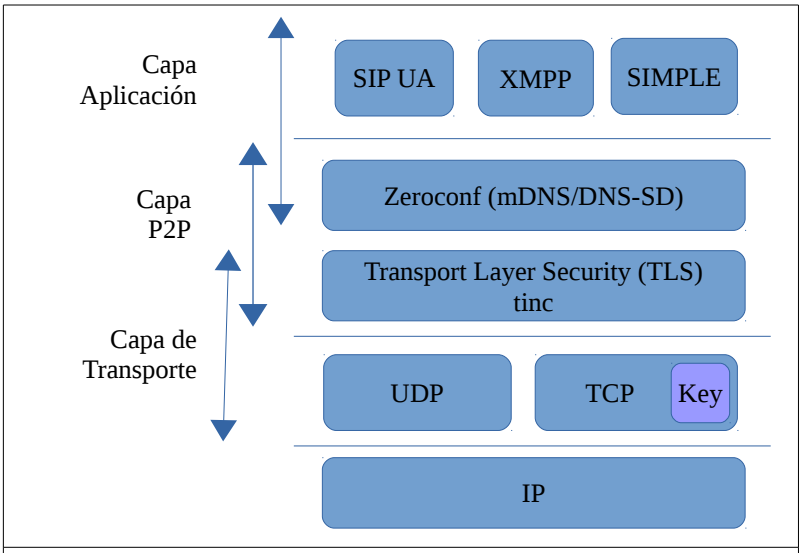


Figura 10. Arquitectura propuesta. (nodos comunes UA)

La “capa aplicación” estará constituida esencialmente por el UA (User Agent) que estará basado en protocolos estándares para el servicio de VoIP y Chat, tales como SIP, XMPP, SIMPLE. Esta capa además comparte con la “capa P2P” el servicio de Zeroconf, esto implica que el UA soporte esta tecnología para tener la capacidad de publicar su propio URI a la red, y descubrir los URI del resto de UA conectados a la red.

La “capa de transporte” (TCP/UDP) sobre la cual estará funcionando la VPN P2P comparte con la “capa P2P” una adaptación del protocolo TLS 1.2 para el cifrado de las comunicaciones. Sin embargo es la “capa P2P” a través del protocolo propio de tinc quien constituirá el núcleo de la red P2P.

Para solventar los inconvenientes de compatibilidad de dispositivos NAT que pudieran presentarse se contará con nodos públicos (supernodos), los cuales serán alojados por cada una de las instituciones participantes en la red P2P (en el caso propuesto serían las Embajadas, Cancillería, Presidencia). Estos serán los puntos de ingreso en la red, como se indica en la figura 11.

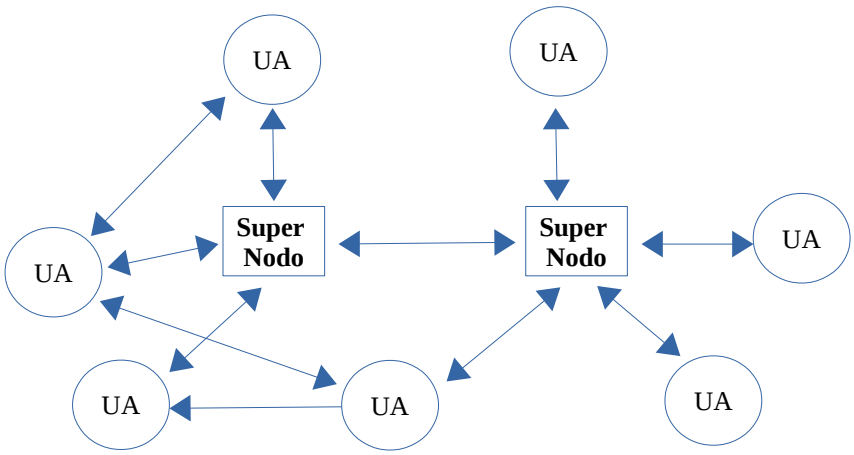


Figura 11. Topología VPN P2P – no-estructurado con supernodos (rendevouz)

3.2.1 Ekiga (SIP User Agent)

SIP puede verse como una aplicación de arquitectura P2P, donde cada UA forma una red superpuesta auto-organizada para localizar y comunicarse con otros nodos. El diseño hará uso de las capacidades naturales P2P de los User Agents, tal como se describe en (2.3), para establecer las comunicaciones entre nodos a través de un cliente estándar (SIP UA). Para la capa de aplicación se seleccionó Ekiga en su versión 4.0.1 como SIP UA, ya que tiene incluido soporte para Avahi (Zeroconf) más depurado que sus versiones anteriores. Otra alternativa de cliente podría ser Blink, que viene con soporte para Bonjour, y ZRTP para cifrado de paquetes RTP.

3.2.2 Avahi (Zeroconf – mDNS/DNS-SD protocolo de publicación y descubrimiento de servicios)

Las capacidades de registro y ruteo que en una infraestructura SIP convencional lo haría el SIP Proxy Server y el SIP Registrar, serán reemplazadas por el sistema de publicación y descubrimiento de servicios Avahi, que es una implementación de Zeroconf[1], similar y compatible a Bounjor de Apple. Para esto, el

1 Zeroconf (Zero Configuration Networking IETF Working Group 1999), inicialmente propuesto por Apple, es una tecnología que permite crear de manera automática una red IP, sin la necesidad de utilizar servidores o configuraciones especiales. Esto incluye, direccionamiento IP, resolución de nombres, y descubrimiento de servicios.

User Agents debe tener capacidad para hacer uso de dicha tecnología para publicar su URI y descubrir los URI del resto de User Agents en la VPN P2P. [30]

3.2.3 Tinc (VPN Peer-to-Peer)

Tinc es un demonio que permite crear una red privada virtual (VPN) multipunto, sin necesidad de definir específicamente los túneles por cada nodo, es decir, que sólo requiere se especifiquen los puntos finales y el demonio se encarga de armar los túneles generando automáticamente un full-mesh (figura 12). De este modo la VPN estará formada por el conjunto de nodos y supernodos que construirán la red superpuesta (overlay network) utilizando el modelo no-estructurado.

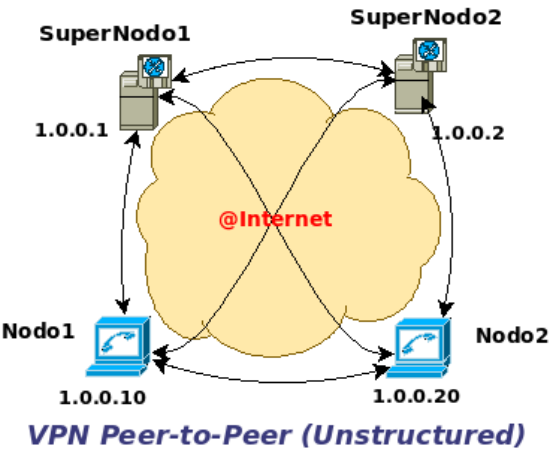


Figura 12: Red superpuesta en modo full-mesh.

Los supernodos en la propuesta planteada sólo harán de facilitadores para el ingreso de los dispositivos a la red P2P, no tendrán funciones adicionales además de compartir la información necesaria para ayudar a los nodos a encontrarse entre sí. Sin embargo, son esenciales ya que si todos los supernodos son atacados o vulnerados, la red entera estaría inaccesible. Estos supernodos, al estar expuestos al Internet con sus respectivas IP públicas, deberán estar detrás de los equipos de seguridad de la Institución donde se esté alojando, tales como firewall, IPS.

3.3 Montaje del Laboratorio

El laboratorio se armó con al menos 6 máquinas virtuales, para poder simular un ambiente de dos puntos remotos que están detrás de un dispositivo NAT, y dos supernodos que serán los nodos de entrada, como se ilustra en la figura 11.

Cada punto remoto consta del dispositivo NAT (nat1 o nat2), y el nodo de usuario (nodo1 o nodo2). El ambiente debe simular la red privada de cada punto remoto y la red externa (Internet) en donde estarán comunicados los supernodos y las interfaces WAN de los dispositivos NAT.

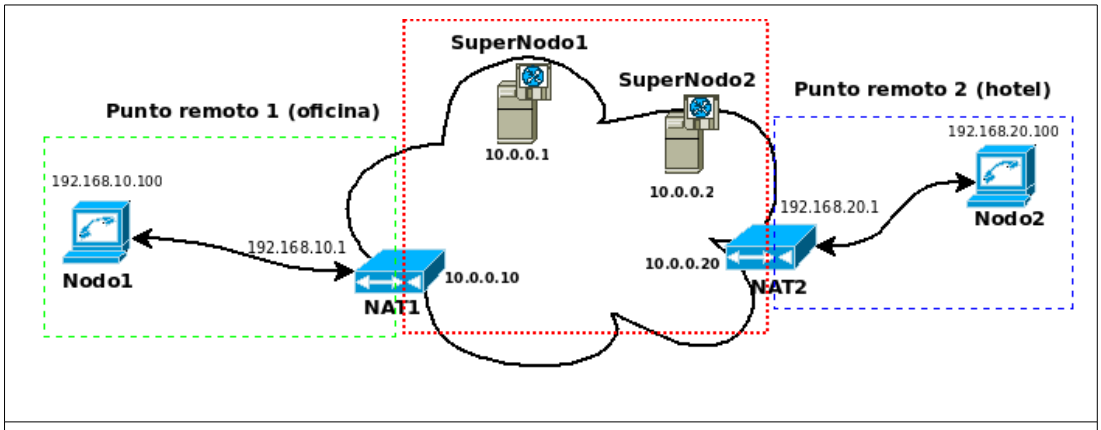


Figura 13: Ambiente simulado en Laboratorio.

En la Figura 13, se muestran tres dominios de red marcados con cuadros de líneas punteadas. El cuadro rojo representa el dominio de Internet, el cuadro verde y azul denotan el dominio del punto remoto 1 y 2, respectivamente. En el Anexo A se describen las configuraciones realizadas en virtualbox para configurar los distintos dominios.

Los requisitos de las máquinas virtuales:

Nodos 1 y 2	Dispositivo NAT 1 y 2	SuperNodos 1 y 2
Ubuntu Desktop 14.04 Ekiga 4.0.1 Avahi 1 Interface de red. Tinc 512 MB RAM	Debian 7.6 1 interfaces de red LAN 1 interface de red WAN 200 MB RAM	Debian 7.6 1 interface de red. Tinc 200 MB RAM

3.3.1 Modo de trabajo

Se trabajó usando scripting para ejecutar las configuraciones individuales que corresponden a cada máquina virtual. Debido a que constantemente se estaban realizando pruebas de instalaciones y configuraciones, se procedió a trabajar con 2 máquinas virtuales base.

Una máquina virtual base se creó con sistema operativo Debian 7.6 que emularán los SuperNodos y dispositivos NAT, y una segunda máquina se creó con sistema operativo Ubuntu desktop 14.04 para emular los nodos de usuario final.

De aquí en adelante, se trabajó con máquinas clonadas a partir de estas dos máquinas base; cada máquina clonada evidentemente tenía los mismos programas instalados y los script de configuración para personalizar la máquina clonada según el equipo que corresponda emular (nodo1, nodo2, supernodo1, supernodo2, nat1 o nat2). En el anexo B se describen al detalle las configuraciones base de cada una de las dos máquinas virtuales.

Se procedió además a crear los script necesarios para gestionar automáticamente las máquinas virtuales, es decir, clonarlas, encenderlas, apagarlas, generar snapshot, eliminar las máquinas clonadas. Debido a que estas tareas eran procesos repetitivos que llevaban al menos 10 minutos por cada vez que se requería regenerar todo el ambiente, después de realizar cambios en las configuraciones de las máquinas base, se decidió generar estos script automatizados para agilizar el trabajo. Estos scripts se detallan en el anexo A.

3.4 Seguridad en los Nodos

Más allá de la seguridad externa, estos nodos tendrán su propio mecanismo de mitigación contra DDoS, que sería el ataque más común al cual estarían expuestos. Para esto, los nodos deberán evitar ser vistos o detectados por el escaneo de puerto, que podría hacerse por medio de ofuscación utilizando puertos no tradicionales. Pero esto traería un problema adicional, ya que los nodos comunes

seguramente tendrían problemas para atravesar los equipos de seguridad perimetral de la infraestructura donde se encuentren.

La dinámica de los nodos comunes, quienes típicamente podrían estar conectándose desde cualquier lugar (hotel, oficina, casa, etc) hacen necesaria la utilización de un puerto común, que no suele ser filtrado por los equipos de seguridad, y este sería el puerto 80. En este caso lo único que queda, es hacer que los **supernodos** hagan buen uso de su denominación y tengan superpoderes y se hagan invisibles.

3.4.1 Mitigación de DDoS en supernodos

Para esto, se sugiere incluir un mecanismo de autenticación entre los dispositivos (nodos comunes) y los supernodos, a más bajo nivel implementando una técnica propuesta en [29], para modificar el stack TCP/IP del kernel Linux e incluir en el protocolo, en su fase inicial de *handshake*, un secreto compartido (PSK) reutilizando el campo de "TCP SQN *number*" del TCP *payload*. Este mecanismo sugerido conocido como "*Port knocking*" es un método que permitiría a los supernodos ser menos visibles en el Internet.

La idea base de este superpoder, sería que los supernodos reciban conexiones por TCP por parte de los nodos comunes que solicitan hacer *bootstrapping* y que no responda a dichas solicitudes desde la pila TCP, es decir, que no respondan positivamente a un TCP SYN al menos que un paquete particular de "knock" haya sido recibido primero.

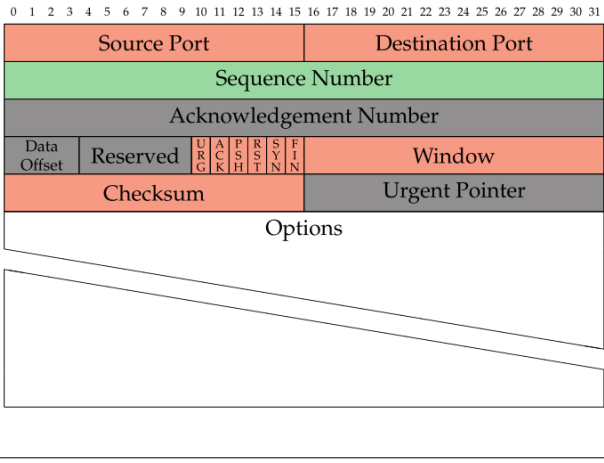


Figura 14: Campos TCP Header, según RFC-793. ISN da la posibilidad de un token de longitud 2^{32}

Este paquete "*knock*" es un token de autorización embebido en el campo "*sequence number* (ISN)" del paquete TCP SYN. Esto podría hacerse parchando el kernel Linux con las librerías propuestas en la tesis de Julian Kirsch (2014).

Esta técnica sólo permitiría conexión hacia los supernodos, de quienes puedan autenticar a través de esta modificación del *stack* TCP. Siendo los supernodos nodos de función específica, esta exclusividad no sería un problema sino un mecanismo ideal para mitigar ataques como DDoS.

3.4.2 Seguridades en Tinc (VPN-P2P)

Tinc establece dos tipos de conexiones UDP y TCP, una de ellas para los datos cifrados (UDP) y otra para los meta-datos de su propio protocolo (TCP), tales como información de ruteo, claves de sesión.

Para la gestión de la seguridad Tinc 1.1.x implementa SPTPS (*Simple Peer-to-Peer Security*) un protocolo propio basado en TLS 1.2, pero simplificado. Utiliza una suite reducida de cifrado fuerte de entre todas las opciones. Los nodos autentican unos a otros utilizando ECC con claves de 521 bits, utiliza Diffie-Hellman con ECC ephemeral de 521 bits para intercambio del secreto perfecto. Para el cifrado de las

comunicaciones utiliza AES-256-CTR, y HMAC-SHA-256 para autenticación de mensajes. [25]

La autenticación de los nodos está basada en infraestructura de clave pública en modo anillo de confianza (*web of trust*). En esta propuesta, se asume que los supernodos son conocidos, y están ya definidos. Cada nuevo nodo que se une a la red, solo necesita conocer a estos supernodos (IP y clave pública) para poder ingresar a la red y poder establecer comunicación directa con cualquier otro nodo. En el Anexo C se detallan las configuraciones de tinc para los nodos y supernodos.

3.4.3 Cifrado de los dispositivos

Es importante que la seguridad esté enfocada, además de la arquitectura del sistema, en los extremos, es decir en los nodos ya que estos podrían convertirse en el eslabón más sensible. Por lo tanto, cada uno de los nodos deberá tener su unidad de almacenamiento cifrado e implementar al menos doble factor de autenticación, por token y clave. Si uno de los dispositivos cae en malas manos, los equipos necesitarán de la clave de acceso además del token para poder iniciar su sistema operativo el cual estará cifrado en su unidad de almacenamiento.

4 Conclusiones

En este trabajo hemos visto las distintas tecnologías esenciales involucradas en la propuesta y cómo éstas han ido evolucionando hasta nuestros días. En la propuesta de implementación planteada se recurre a una arquitectura sencilla, en la que se asume una cantidad pequeña de nodos (entre 500 y 1000), que podría ser eficiente para el caso sugerido en un ambiente específico.

Es de notar que en un ambiente público con millones de usuarios las arquitecturas P2P basadas en modelos híbridos, que combinan tecnologías no-estructuradas y DHT, serían más eficientes al utilizar lo mejor de cada una, considerando que en este tipo de ambientes los sistemas estructurados son más eficientes al ubicar un recurso, y que un nivel alto de entrada y salida de nodos tiene mínima afectación en los sistemas no-estructurados.

Se pretende con esta propuesta de implementación demostrar que existen herramientas que pueden integrarse para construir una red descentralizada de comunicaciones de voz y texto, para evadir las interceptaciones, censuras, recolección de datos y vigilancia masiva que sufren los servicios centralizados. Este modelo sencillo al prescindir del componente Servidor de la infraestructura SIP, tiene ciertas limitaciones como por ejemplo:

- No es posible la funcionalidad de conferencia entre varios usuarios al mismo tiempo, a no ser que el User Agent lo soporte.
- No se contempla la gestión de la calidad de servicio.

La variedad de herramientas desarrolladas para crear redes o servicio bajo el paradigma P2P, deja un espacio para evaluar dichas tecnologías y experimentar distintas alternativas de implementación. Se puede apreciar que hay una tendencia muy fuerte por los desarrolladores e investigadores de explorar estas tecnologías más allá del ámbito académico.

Un trabajo que resultaría interesante es evaluar la distintas VPN P2P

existentes, para determinar los niveles de seguridad que estas implementan y los servicios que mejor se podrían integrar. Por otro lado, P2P SIP es otra tecnología propuesta y desarrollada desde el 2006 que se podría evaluar, mejorar y analizar hasta qué punto ambas tecnologías P2P (SIP y VPN) estarían duplicando esfuerzos o podrían compartir tecnologías para una integración eficiente.

5 Anexos

5.1 Anexo A: Configuración de dominios de red en VirtualBox

VirtualBox permite varios tipos de configuración de los adaptadores de red, para el piloto se utilizaron: Red Interna (*Internal Network*), Adaptador solo-anfitrión (*Host-only Adapter*).

Red Interna, permite conectividad sólo entre las máquinas virtuales que estén dentro de ese tipo de configuración, formando un dominio privado al cual aún el *host* (máquina física) no tendrá acceso. Este tipo de adaptador es único, viene ideal para emular el dominio de internet (cuadro rojo en figura 12).

Adaptador solo-anfitrión, se comporta de manera similar a Red Interna, pero además el *host* físico puede acceder a las máquinas virtuales que está dentro de esta configuración. Esta opción permite crear múltiples dominios, de tal manera que VirtualBox Manager maneja internamente un servidor DHCP el cual puede ser activado a voluntad. Cada dominio genera una interface de red virtual en el sistema operativo del *host* de nombre vboxnetN (N es el número de interface). Este tipo de adaptador viene bien para emular los dominios privados de los puntos remotos.

Siguiendo la figura 12, los dispositivos **nat1** y **nat2** virtualizados en Debian 7.6, tendrán la interface LAN en modo “adaptador solo-anfitrión” con vboxnet1 y vboxnet2 respectivamente, y la interface WAN en modo “Red Interna” con intnet.

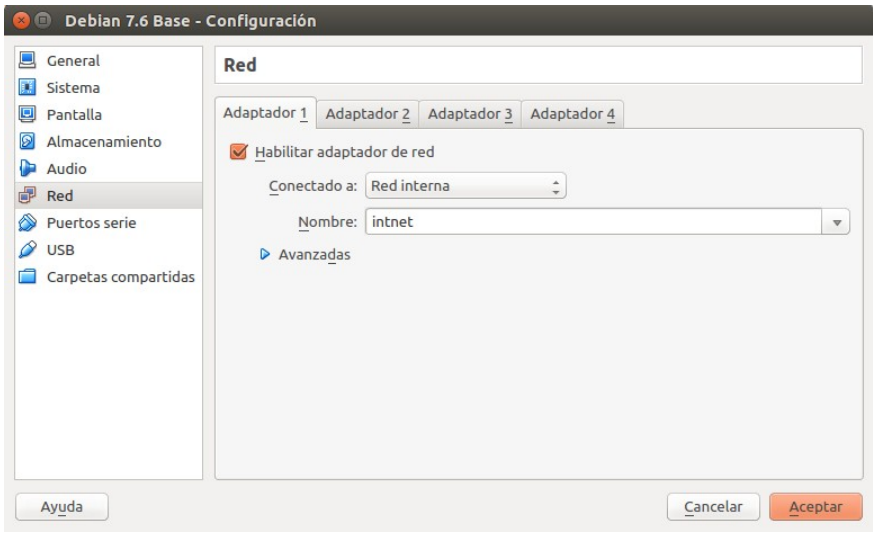


Figura 15: Adaptardor WAN – VirtualBox – nat1, nat2, supernodo1, supernodo2

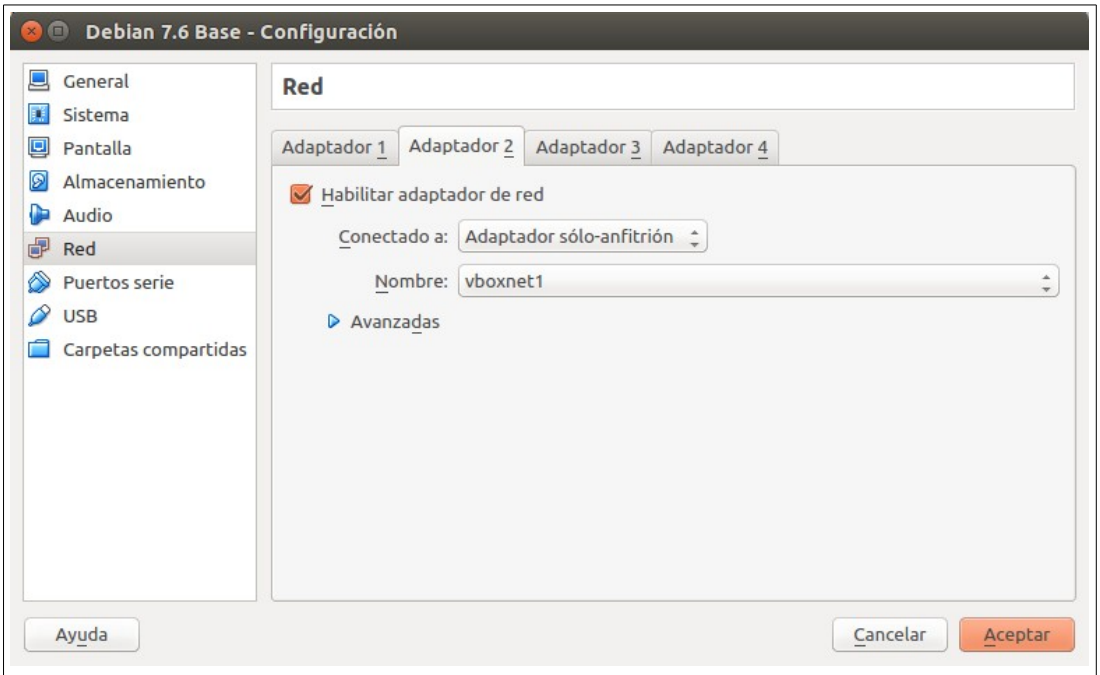


Figura 16: Adaptardor LAN – VirtualBox – nat1

Los supernodos tendrán sólo una interface de red, configurada como “Red Interna”.

Los nodos comunes, **nodo1** y **nodo2**, tendrán una sola interface de red configurada como "adaptador solo-anfitrión" con vboxnet1 y vboxnet2 respectivamente, quedando los dominios de la siguiente manera:

Tipo de Adaptador	Dispositivo/interface	Dirección IP
intnet	supernodo1	10.0.0.1
	supernodo2	10.0.0.2
	nat1 (WAN)	10.0.0.10
	nat2 (WAN)	10.0.0.20
vboxnet1	nat1 (LAN)	192.168.10.1
	nodo1	192.168.10.100
vboxnet2	nat2 (LAN)	192.168.20.1
	nodo2	192.168.20.100

El dominio de la red externa (internet) también podría emularse utilizando el Adaptador de tipo "puente" en lugar de "red interna" si se dispone de otro *host* físico para emular el punto remoto 2 (nodo2 y nat2), conectado a un *switch*.

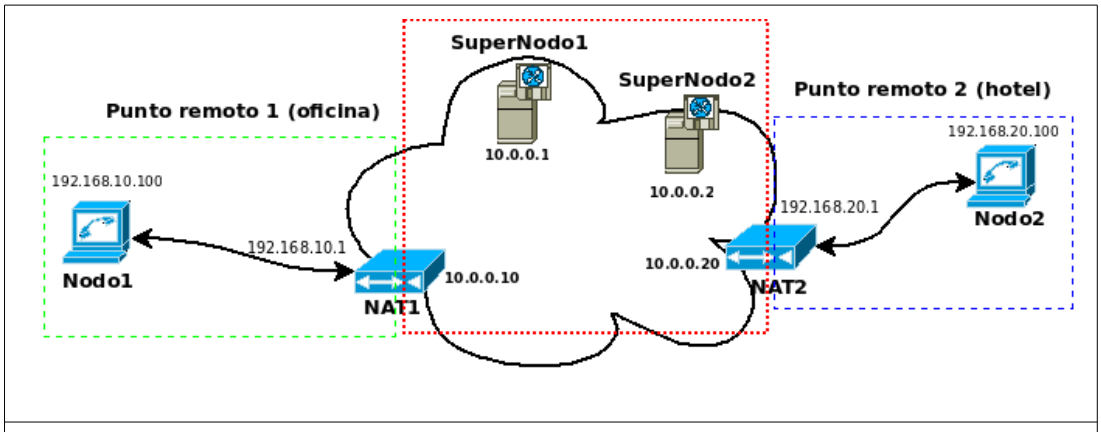


Figura 17: Ambiente simulado en Laboratorio.

5.2 Anexo B: Configuración Base de máquinas virtuales

Configuración máquina virtual base Debian 7.6.

Paquetes que se deben instalar

```
apt-get install tinc
apt-get install avahi*
```

Archivos complementarios:

iptables.rules

```
# Generated by iptables-save v1.4.14 on Tue Feb 24 21:23:55 2015
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -o eth0 -j MASQUERADE
COMMIT
# Completed on Tue Feb 24 21:23:55 2015
# Generated by iptables-save v1.4.14 on Tue Feb 24 21:23:55 2015
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -i eth0 -j ACCEPT
-A INPUT -i eth1 -j ACCEPT
-A FORWARD -i eth1 -j ACCEPT
COMMIT
# Completed on Tue Feb 24 21:23:55 2015
```

/etc/hosts

```
127.0.0.1       localhost
#127.0.1.1      debian76.laboratorio.edu        debian76
192.168.0.106   debian76.laboratorio.edu        debian76
10.0.0.1        supernodo1
10.0.0.2        supernodo2
```

```
10.0.0.10       nat1
10.0.0.20       nat2
10.0.0.30       nat3
192.168.10.1    nat1priv
192.168.20.1    nat2priv
192.168.30.1    nat3priv
192.168.10.100  nodo1 nodo1.local
192.168.20.100  nodo2 nodo2.local
192.168.30.100  nodo3 nodo3.local
1.0.0.10        vpnodo1 vpnodo1.local
1.0.0.20        vpnodo2 vpnodo2.local
1.0.0.30        vpnodo3 vpnodo3.local
```

Para activar las configuraciones del dispositivo **nat** correspondiente (nat1, nat2, nat3, ….) se elaboró un script al cual se le pasa por parámetro el número del nat que se desea configurar, por ejemplo para el **nat2**:

 # . setnat.sh 2

/root/nat/setnat.sh [numero-de-nat]

```
if [ $# -eq 0 ];
then
        echo "Debe incluir parámetros";
else
# configuracion de nombre y direccion IP del nodo.
       export NATNAME="nat"$1
        export NATNAMEPRIV="nat"$1"priv"
        export nat1pub="10.0.0.10"
        export nat2pub="10.0.0.20"
        echo $NATNAME
        hostname $NATNAME

# desactivar networ-manager para configuracion manual
        /etc/init.d/network-manager stop
# configuracion de reglas para NAT
        iptables-restore < iptables.rules
# configuracion manual de interfaces de red
        ifconfig eth0 $NATNAME  netmask 255.255.255.0
        ifconfig eth1 $NATNAMEPRIV netmask 255.255.255.0
        #route add -net 192.168.10.0 netmask 255.255.255.0 gw 192.168.10.10 dev eth1
# activacion manual ip_forward para habilitar modo de ruteo
        echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
fi
```

Para activar las configuraciones de los **supernodos** (sueprnodo1, supernodo2, supernodo3 …) se elaboró un script al cual se le pasa por parámetro el número del supernodo que se desea configurar, por ejemplo para el **supernodo1**:

 # . set-sn.sh 1

/root/sn/set_sn.sh [numero-de-supernodo]

```
if [ $# -eq 0 ];
then
       echo "Debe incluir parámetros";
else
       export NODONAME="supernodo"$1
       echo $NODONAME
       hostname $NODONAME.local

# desactivar networ-manager para configuracion manual
       /etc/init.d/network-manager stop
# configuracion manual de interfaces de red
       ifconfig eth0 10.0.0.$1  netmask 255.255.255.0
# desactivar interface eth1 que no se utilizara en los supernodos
       ifconfig eth1 down
       #route add -net 0.0.0.0 netmask 0.0.0.0 gw 192.168.$10.1 dev eth0

# copia archivos de configuracion de la vpn correspondiente del supernodo
       cp tinc.conf.supernodo$1 /etc/tinc/laboratorio/tinc.conf
      # copia script personalizado para levantar interface tun0 del nodo
       cp tinc-up.supernodo$1 /etc/tinc/laboratorio/tinc-up
      # copia clave privada previamente generada del nodo
       cp rsa_key.priv.supernodo$1 /etc/tinc/laboratorio/rsa_key.priv
       cp supernodo* /etc/tinc/laboratorio/hosts/
       cp ../nodo/nodo? /etc/tinc/laboratorio/hosts/

# reiniciar el servicio avahi
       /etc/init.d/avahi-daemon restart

# levantar el servicio tinc
       tincd -D -n laboratorio

fi
```

Configuración máquina virtual base Ubuntu 14.04

Paquetes que se deben instalar

```
apt-get install tinc
apt-get install avahi*
apt-get install ekiga
```

Archivo complementario:

/etc/hosts

```
127.0.0.1       localhost
#127.0.1.1      debian76.laboratorio.edu        debian76
192.168.0.106   debian76.laboratorio.edu        debian76
10.0.0.1        supernodo1
10.0.0.2        supernodo2
10.0.0.10       nat1
10.0.0.20       nat2
10.0.0.30       nat3
192.168.10.1    nat1priv
192.168.20.1    nat2priv
192.168.30.1    nat3priv
192.168.10.100  nodo1 nodo1.local
192.168.20.100  nodo2 nodo2.local
192.168.30.100  nodo3 nodo3.local
1.0.0.10        vpnodo1 vpnodo1.local
1.0.0.20        vpnodo2 vpnodo2.local
1.0.0.30        vpnodo3 vpnodo3.local
```

Para activar las configuraciones de los **nodos** (nodo1, nodo2, nodo3 …) se elaboró un script al cual se le pasa por parámetro el número del nodo que se desea configurar, por ejemplo para el **nodo3**:

 # . setnodo.sh 3

/root/nodo/setnodo.sh [numero-de-nodo]

```
if [ $# -eq 0 ];
then
        echo "Debe incluir parámetros";
else
```

```
        export NODONAME="nodo"$1
        echo $NODONAME
        hostname $NODONAME.local

# desactivar networ-manager para configuracion manual de red
        stop network-manager

# configuracion manual de interface de red
        ifconfig eth0 $NODONAME  netmask 255.255.255.0
        ifconfig eth1 down

# configuracion de gateway por defecto (dispositivo nat)
        route add -net 0.0.0.0 netmask 0.0.0.0 gw 192.168.$10.1 dev eth0

# configuracion del nombre del nodo en Ekiga
        cp -p %gconf.xml.nodo$1
/home/francisco/.gconf/apps/ekiga/general/personal_data/%gconf.xml

# reiniciar servicio gconfd-2 para que Ekiga tome la configuracion del nuevo nombre
        killall gconfd-2

# copia archivos de configuracion de tinc para el nodo
        cp tinc.conf.nodo$1 /etc/tinc/laboratorio/tinc.conf
        cp tinc-up.nodo$1 /etc/tinc/laboratorio/tinc-up
        # copia de clave privada del nodo en el directorio de configuracion
        cp rsa_key.priv.nodo$1 /etc/tinc/laboratorio/rsa_key.priv
        # copia datos de host del nodo correspondiente, y de todos los supernodos
        cp nodo$1 /etc/tinc/laboratorio/hosts/
        cp ../sn/supernodo* /etc/tinc/laboratorio/hosts/
# reinicia el servicio de avahi
        /etc/init.d/avahi-daemon restart
fi
```

Para la gestión de las máquinas virtuales, se crearon los siguientes script:

1. clonar todas la máquinas virtuales del laboratorio.

setvms.sh

```
# crear snapshot Base
vboxmanage snapshot "Debian 7.6 Base" take "Sistema Base"
vboxmanage snapshot "Ubuntu Mate" take "Sistema Base"

# Clonar snapshot
```

```
vboxmanage clonevm "Debian 7.6 Base" --snapshot "Sistema Base" --options link  --name
"Debian 7.6 super1" --register
vboxmanage clonevm "Debian 7.6 Base" --snapshot "Sistema Base" --options link  --name
"Debian 7.6 nat1" --register
vboxmanage clonevm "Debian 7.6 Base" --snapshot "Sistema Base" --options link  --name
"Debian 7.6 nat2" --register
vboxmanage clonevm "Ubuntu Mate" --snapshot "Sistema Base" --options link  --name
"Ubuntu Mate nodo1" --register
vboxmanage clonevm "Ubuntu Mate" --snapshot "Sistema Base" --options link  --name
"Ubuntu Mate nodo2" --register

# Modificar configuracion VMs
vboxmanage modifyvm "Debian 7.6 super1" --memory 200 --nic1 intnet --intnet1 intnet
--boot1 dvd --boot2 disk
vboxmanage modifyvm "Debian 7.6 nat1" --memory 200 --nic1 intnet --nic2 hostonly
--intnet1 intnet --hostonlyadapter2 vboxnet1 --boot1 dvd --boot2
disk
vboxmanage modifyvm "Debian 7.6 nat2" --memory 200 --nic1 intnet --nic2 hostonly
--intnet1 intnet --hostonlyadapter2 vboxnet2 --boot1 dvd --boot2
disk
vboxmanage modifyvm "Ubuntu Mate nodo1" --memory 600 --nic1 hostonly
--hostonlyadapter1 vboxnet1 --boot1 dvd --boot2 disk
vboxmanage modifyvm "Ubuntu Mate nodo2" --memory 600 --nic1 hostonly
--hostonlyadapter1 vboxnet2 --boot1 dvd --boot2 disk
```

2. encender todas las máquinas virtuales clonadas.

startvm.sh

```
# Encender máquinas virtuales clonadas

vboxmanage startvm "Debian 7.6 super1"
vboxmanage startvm "Debian 7.6 nat1"
vboxmanage startvm "Ubuntu Mate nodo1"
vboxmanage startvm "Debian 7.6 nat2"
vboxmanage startvm "Ubuntu Mate nodo2"
```

3. apagar todas las máquinas virtuales clonadas.

apagarvms.sh

```
# apagar maquinas virtuales clonadas
```

```
vboxmanage controlvm "Debian 7.6 super1" poweroff
vboxmanage controlvm "Debian 7.6 nat1" poweroff
vboxmanage controlvm "Debian 7.6 nat2" poweroff
vboxmanage controlvm "Ubuntu Mate nodo1" poweroff
vboxmanage controlvm "Ubuntu Mate nodo2" poweroff
```

4. eliminar todas las máquinas clonadas virtuales.

limpiarvms.sh

```
# Eliminar las máquinas virtuales clonadas

vboxmanage unregistervm "Debian 7.6 super1" --delete
#vboxmanage unregistervm "Ubuntu Mate super1" --delete
vboxmanage unregistervm "Debian 7.6 nat1" --delete
vboxmanage unregistervm "Debian 7.6 nat2" --delete
#vboxmanage unregistervm "Debian 7.6 nodo1" --delete
vboxmanage unregistervm "Ubuntu Mate nodo1" --delete
#vboxmanage unregistervm "Debian 7.6 nodo2" --delete
vboxmanage unregistervm "Ubuntu Mate nodo2" --delete

# Eliminar los snapshot de las máquinas virtuales bases
vboxmanage snapshot "Debian 7.6 Base" delete "Sistema Base"
vboxmanage snapshot "Ubuntu Mate" delete "Sistema Base"
```

5.3 Anexo C: Configuración Tinc

Supernodo1

/etc/tinc/laboratorio/tinc.conf

```
Name = supernodo1
AddressFamily = ipv4
Interface = tun0
ConnectTo = supernodo2
ConnectTo = supernodo3
ConnectTo = supernodo4
```

/etc/tinc/laboratorio/hosts/supernodo1

```
Address = 10.0.0.1
Subnet = 1.0.0.1/32

-----BEGIN RSA PUBLIC KEY-----

MIIBCgKCAQEAyP1bN+qTy7JbCrd6ABr1cory4jcU0gm15aB/HoXh/cFeAY9+6pbU
a5KGEHRROSmLml3tRkSqPlPQnoIv1Q2TXS49HWvzEjM2APL1qOfXU2A0wtA97Din
Up8S8Vytbq3444wonZCFaGhIw/RemObfOvVLhzCyeIRmXxzu/xVPz5zJrkuPwnVQ
vi3l0hBAZqckQtj1aawf27rQtkN9xxJHoOZpDUnfA7HtJpzywuWc3gmLAXr463ap
3o11ugS/2othGkx2JyPKg3QENOdRS2REEbAnC2SJyh65KyhdaqJggANuqfN2nRlD
Hvl4TOyXQMApN5tA9Q2gQtLinwueSeWjGQIDAQAB

-----END RSA PUBLIC KEY-----
```

Supernodo2

/etc/tinc/laboratorio/tinc.conf

```
Name = supernodo2
AddressFamily = ipv4
Interface = tun0
ConnectTo = supernodo1
ConnectTo = supernodo3
ConnectTo = supernodo4
```

/etc/tinc/laboratorio/hosts/supernodo2

```
Address = 10.0.0.2
Subnet = 1.0.0.2/32

-----BEGIN RSA PUBLIC KEY-----

MIIBCgKCAQEA52d2NBNPDtJMyMufQUpeBoFYDDiwUp3V+6hD9vauGPY98BBvzZEe
OIygJALZ9hoqKn9gcjZ7BCHVhhTLVqsigzBoEGTjfJn9VVV70QQBuyb94fALlh6b
xt8XK/yaXyP33NXhpoZWsh6GH/f0eWQzBPwaiTGNkReGFvusnpAIE4ES4JWsvCh5
B4vXj6+FSH7R4LeaL50lB/BR2bUVQInVyQ8eQyaDYcbPFuvl9ih+RwT1WTkxkP7s
0ZchpbbtZpg8c2RzSXZ1KZWpaei2C+8lHmqjMYkY7B8mTV+O8S0oCg9MWU5+W09n
```

```
81HJcVKbLIdVvB9NEmmyLqxCLxl7yhCvawIDAQAB

-----END RSA PUBLIC KEY-----
```

Nodo1

/etc/tinc/laboratorio/tinc.conf

```
Name = nodo1
AddressFamily = ipv4
Interface = tun0
ConnectTo = supernodo1
ConnectTo = supernodo2
ConnectTo = supernodo3
ConnectTo = supernodo4
```

/etc/tinc/laboratorio/hosts/nodo1

```
Subnet = 1.0.0.10/32

-----BEGIN RSA PUBLIC KEY-----

MIIBCgKCAQEAy1JHtweSmSBPYTVm1VC1GNiQ2SSjLB9x5J7HXJyW4ya8pWnShpo+
13qDRRnirDqyxUB1QECv7oHpd9a8Mbg23ajjvtsa8A6iHup+D2xGBb1ucjjrreGt
mhNguefe9W6ruQCilP1xYjNCdgsU47Tv08reycDniz+fEaUL6UJVvzk7VUGGMf3m
qgQQhcvzEgZJ3F8BfcoBr8Wujdvhe1BwdTyPZmVkzh5xFk6sV4B4gHiroXWTgMir
XA+7Ac4UQrGzrGVLumEkkqEEZNKEgoOa0JCV6PpjDyafjtYVxtXwz9jhTDT3c4o1
UJh5z87wnr/H9zocFGQ+U10XglwUHLU/DQIDAQAB
-----END RSA PUBLIC KEY-----
```

Nodo2

/etc/tinc/laboratorio/tinc.conf

```
Name = nodo2
AddressFamily = ipv4
Interface = tun0
ConnectTo = supernodo1
ConnectTo = supernodo2
ConnectTo = supernodo3
ConnectTo = supernodo4
```

/etc/tinc/laboratorio/hosts/nodo2

```
Subnet = 1.0.0.20/32

-----BEGIN RSA PUBLIC KEY-----

MIIBCgKCAQEA0B5TbB2Gp+kYBl0Ga35RBk33RapZFS/Kshf32HhoT3flUK2cfDaZ
lG/jKILRj4xC/mlS2NkRzdaq3zWd18tYCoLAjb4hSlpDwQ+o+9jNc621OgJuvlK9
2nqgBKOrRBwpPHfXc6kCibjRrkt2nJirBlR+74RJj5h6C/gqRPzSuQObkW0OwA78
```

```
If9Ju/qFFmF3m/d4XjAe0KY2qwHiyOY8FbUjVK7vdFT0nic4CD5Iv3YFG6ca+dkE
2yHDx3D2UaasVrrD7OIS/J4sSyG6xckmHzF8w8h5fKUEON1ri8jSaNU4pIvmocAg
kd2GQcOPILj6ksuW5NoJXxom9K770tJ6mwIDAQAB

-----END RSA PUBLIC KEY-----
```

Para la generación de las claves pública y privada de cada nodo, se debe ejecutar el siguiente comando en cada uno de ellos:

```
# tincd -K

Generating 4096 bits keys:
..............++++++ p
..............++++++ q

Done.

Please enter a file to save private RSA key to [/etc/tinc/rsa_key.priv]:
Please enter a file to save public RSA key to [/etc/tinc/hosts/nodo1]:
Appending key to existing contents.
Make sure only one key is stored in the file.
```

Todos los dispositivos de usuario (nodos comunes) tienen especificados una lista de supernodos a los cuales se podrán conectar (IP pública), por medio del parámetro “ConnectTo”. Además deberán tener en su directorio “/etc/tinc/laboratorio/hosts” el archivo que tiene las definiciones (IP, y clave pública) de los supernodos:

“/etc/tinc/laboratorio/supernodo1” del supernodo1

“/etc/tinc/laboratorio/supernodo1” del supernodo1

“/etc/tinc/laboratorio/supernodo1” del supernodo1

….

etc

Los supernodos proporcionan información de conexión del resto de nodos a toda la red P2P, por esto los nodos comunes no necesitan conocerse entre sí, basta con conocer a los supernodos. Una vez que los nodos comunes obtienen esta información buscan conectarse directamente con el nodo requerido.

Es decir, tinc funciona en modo (*web of trust*) donde se establece que todo nodo que conozca a un supernodo de la red y además posea su clave pública, se asume que es confiable.

En esta configuración se utilizaron los protocolos por default, sin embargo se recomienda utilizar el protocolo propio de tinc (SPTPS) que utiliza ECC y AES.

5.4 Anexo D: Configuración Avahi

La siguiente configuración corresponde a los parámetros mínimos que se deben configurar en todos los nodos que formarán parte de la red P2P. Algo importante es limitar la difusión de los servicios a la interface de la VPN creada por el demonio tinc, en este caso corresponde a **tun0**.

Esto se define con el parámetro **allow-interfaces=tun0**, que al estar especificado tinc entiende que el resto de interfaces no estarán permitidas.

/etc/avahi/avahi-daemon.conf

```
[server]
use-ipv4=yes
use-ipv6=no
allow-interfaces=tun0
allow-point-to-point=yes

[wide-area]
enable-wide-area=yes

[publish]
publish-addresses=yes
publish-workstation=yes
publish-domain=yes
```

5.5 Anexo E: Configuración User Agent

El User Agent Ekiga 4.0.1, se le debe configurar el nombre del usuario, en el menú Edit → preferencias.

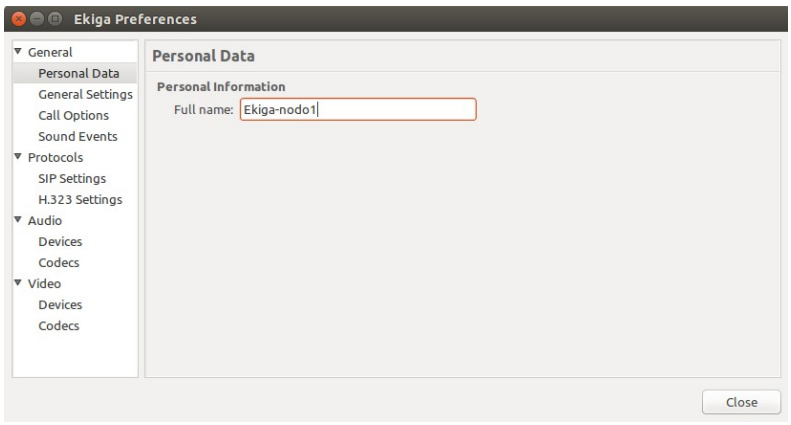


Figura 18: Ekiga 4.0.1

Otra alternativa de software es **Blink SIP Client**, que utiliza ZRTP como mecanismo de cifrado del flujo de datos de voz. Este mecanismo utiliza Diffie-Hellman como protocolo para intercambio de claves previo a establecer el flujo de datos RTP (Real-time Transport Protocol). Esto le agregaría una capa más de seguridad al cifrar dentro de la VPN la transmisión de voz entre nodos.

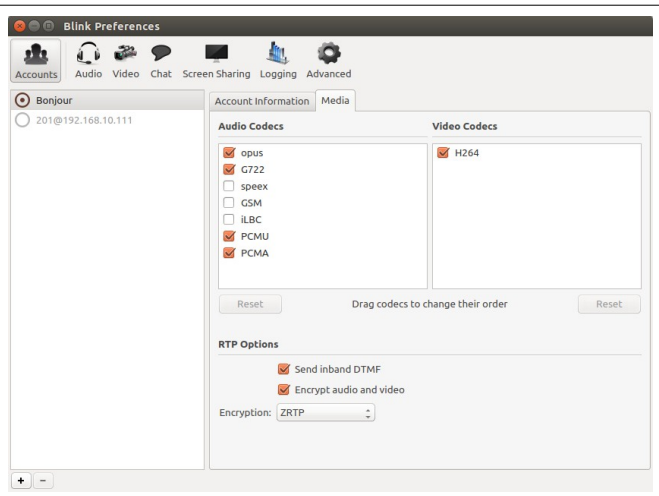


Figura 19. Blink SIP Client.

6 Bibliografía

6.1 Específica

[1] R. Tejedor, «Domine las Redes (P2P)», *Peer-to-Peer”*, *Editor. Alfaomega*, p. 327, 2007.

[2] «Spotify: A Massive P2P Network, Blessed by Record Labels | TorrentFreak.» [En línea]. Disponible en: http://torrentfreak.com/spotify-a-massive-p2p-network-blessed-by-record-labels-110617/. [Accedido: 27-oct-2014].

[3] «Spotify To Phase Out Its P2P Network, and Run Its Own Servers | Digital Trends.» [En línea]. Disponible en: http://www.digitaltrends.com/music/spotify-shuts-p2p-network-prevent-music-eating-bandwidth/. [Accedido: 27-oct-2014].

[4] D. Korzun y A. Gurtov, *Structured Peer-to-Peer Systems*. Springer, 2013.

[5] X. Shen, H. Yu, J. Buford, y M. Akon, *Handbook of peer-to-peer networking*. Springer, 2010.

[6] K. W. Ralf Steinmetz, *Peer-to-Peer Systems and Applications*, vol. 3485. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[7] H. Zhang, Y. Wen, H. Xie, y N. Yu, *Distributed Hash Table: Theor, Platforms and Applications*. 2013.

[8] Y.-K. R. Kwok, *Peer-to-peer Computing: application, Architecture, Protocols, and Challenges*. 2012.

[9] C. Jennings y F. Audet, «RFC-4787: Network Address Translation (NAT) Behavioral Requirements for Unicast UDP», pp. 1-29, 2007.

[10] P. Srisuresh, B. Ford, y D. Kegel, «RFC-5128: State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)», pp. 1-32, 2008.

[11] M. Fajandar, «Implementing an Authorization model in a SIP User Agent to secure SIP sessions», Bombay University, 2000.

[12] A. Johnston, S. Donovan, R. Sparks, C. Cunningham, y K. Summer, «RFC 3665 Session Initiation Protocol (SIP) Basic Call Flow Examples», 2003.

[13] «Peer-to-Peer Session Initiation Protocol (p2psip) - Charter», 2015. [En línea]. Disponible en: http://datatracker.ietf.org/wg/p2psip/charter/. [Accedido: 01-mar-2015].

[14] D. a Bryan y B. B. Lowekamp, «SOSIMPLE : A SIP / SIMPLE Based P2P VoIP and IM System», pp. 1-6, 2004.

[15] K. Singh y H. Schulzrinne, «SIPPEER: A Session Initiation Protocol (SIP)-based Peer-to-Peer Internet Telephony Client Adaptor.»

[16] Y. Houngue, «A Middleware-Independent and Secure Peer-To-Peer Sip Architecture (Mise-P2Psip)», UNIVERSITÀ DEGLI STUDI DI MILANO, 2012.

[17] «Peer-to-Peer Session Initiation Protocol (Active WG)», 2007. [En línea]. Disponible en: http://tools.ietf.org/wg/p2psip/. [Accedido: 27-feb-2015].

[18] S. A. Baset, G. Gupta, y H. Schulzrinne, «OpenVoip: An Open Peer-to-Peer VoIP and IM System», 2008.

[19] S. A. Ahson y I. Mohammad, *SIP Handbook. Service, Technologies and Security Os Session Initiation Protocol*. 2009.

[20] H. Dudani, «Virtual Private Networks for Peer-to-Peer Infrastructures», Universidad Técnica de Darmstadt, 2012.

[21] «CJDNS Whitepaper.» [En línea]. Disponible en: https://github.com/cjdelisle/cjdns/blob/master/doc/Whitepaper.md.

[22] M. Brown, «Using the GNU Virtual Private Ethernet», n.º February, pp. 1-13, 2011.

[23] L. Deri y R. Andrews, «N2N: A layer two peer-to-peer vpn», *Resilient Networks Serv.*, pp. 53-64, 2008.

[24] «GNUnet | GNU’s Framework for Secure Peer-to-Peer Networking.» [En línea].

Disponible en: https://gnunet.org/. [Accedido: 14-dic-2014].

[25] I. Timmermans y G. Sliepen, «Manual: Setting up a Virtual Private Network with tinc», 2014.

[26] «AboutAvahi – Avahi.» [En línea]. Disponible en: http://avahi.org/wiki/AboutAvahi. [Accedido: 07-abr-2015].

[27] A. Sikarwar, «Implementation of SIP URI Service Discovery», p. 3, 2008.

[28] «ImTelephonyAvahiApplication – Avahi.» [En línea]. Disponible en: http://avahi.org/wiki/ImTelephonyAvahiApplication. [Accedido: 03-mar-2015].

[29] J. Kirsch, «Improved Kernel-Based Port-Knocking in Linux», Universidad Tecnica de Munich, 2014.

[30] A. Srivastava, J. W. Lee, y H. Schulzrinne, «Implementing Zeroconf in Linphone», pp. 1-4, 2011.

6.2 General

- Barrio, L. B. (2005). A P2P Framework for VoIP applications. Recuperado a partir de http://bibing.us.es/proyectos/abreproy/11069/fichero/VoIP2P+v12.pdf
- Bryan, D. a., Lowekamp, B. B., & Jennings, C. (2005). SOSIMPLE: A serverless, standards-based, P2P SIP communication system. Proceedings - First International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications, AAA-IDEA 2005, 2005, 42-49. doi:10.1109/AAA-IDEA.2005.15
- Cohrs, K. M. (2008). Implementation and Evaluation of the Peer-to-Peer-Protocol (P2PP) for P2PSIP. Georg-August University.
- DeLisle, C. J. (2013a). [liberationtech] CJDNS hype. Recuperado a partir de https://mailman.stanford.edu/pipermail/liberationtech/2013-July/010037.html
- DeLisle, C. J. (2013b). cjdns Whitepaper · GitHub. Recuperado a partir de https://github.com/cjdelisle/cjdns/blob/master/doc/Whitepaper.md
- Deri, L., & Andrews, R. (2008). N2n: A layer two peer-to-peer vpn. Resilient Networks and Services, 19. Recuperado a partir de http://link.springer.com/chapter/10.1007/978-3-540-70587-1_5

- Dudani, H. (2012). Virtual Private Networks for Peer-to-Peer Infrastructures. Universidad Técnica de Darmstadt. Recuperado a partir de http://sit.sit.fraunhofer.de/smv/publications/download/Virtual_Private_Networks_for_Peer-to-Peer_Infrastructures.pdf
- GauthierDickey, C., & Grothoff, C. (2008). Bootstrapping of Peer-to-Peer Networks. 2008 International Symposium on Applications and the Internet. Ieee. doi:10.1109/SAINT.2008.15
- GNU Telephony » Blog Archive » RaspberryPi Secure VoIP access points with GNU SIP Witch. (s. f.). Recuperado 28 de marzo de 2015, a partir de http://planet.gnu.org/gnutelephony/?p=56
- Jennings, C., & Lowekamp, B. (2012). A SIP Usage for RELOAD, 1-14. Recuperado a partir de http://tools.ietf.org/pdf/draft-ietf-p2psip-sip-14.pdf
- Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., & Schulzrinne, H. (2012). REsource LOcation And Discovery (RELOAD) Base Protocol. Internet Draft, 156. Recuperado a partir de http://tools.ietf.org/pdf/draft-ietf-p2psip-base-19.pdf
- Johnston, A., & Donovan, S. (2008). RFC-5359 Session Initiation Protocol Service Examples, 1-170. Recuperado a partir de http://www.rfc-editor.org/rfc/pdfrfc/rfc5359.txt.pdf
- Juste, P. S., Jeong, K., Eom, H., Baker, C., & Figueiredo, R. (2014). TinCan: User-Defined P2P Virtual Network Overlays for Ad-hoc Collaboration, 01(2), 1-15.
- Lupu, C., Control, A., Faculty, C., & Independent, S. (2012). Peer-To-Peer Virtual Private Networks: Multiple Supernodes Feature for N2N.
- NAT TRAVERSAL IN PEER-TO-PEER ARCHITECTURE. (2004). Recuperado 17 de agosto de 2014, a partir de http://www.sce.carleton.ca/~fgagnon/Publications/NATTraversal_TR_SCE-12-04.pdf
- Papageorgiou, P. (2001). A comparison of H. 323 vs SIP. University of Maryland. Recuperado a partir de http://www.ktl.elf.stuba.sk/~chromy/SpS2/referaty/01 - A Comparision of H.323 vs SIP.pdf
- Richardson, J. (s. f.). DUNDi , So Easy A Caveman Could Do It ! Recuperado a partir de http://www.voip-info.org/storage/users/813/47813/images/1654/DUNDi_So_Easy.pdf
- Schulzrinne, H. G., Schulzrinne, H. G., Rosenberg, J. D., & Rosenberg, J. D. (1998). The Session Initiation Protocol: Providing Advanced Telephony Services Across the Internet. Bell Labs Technical Journal, (December), 144-160. doi:10.1002/bltj.2133
- Schulzrinne, H., & Rosenberg, J. (1998). A Comparison of SIP and H. 323 for Internet Telephony. Proc. International Workshop on Network and Operating System

Support for Digital Audio and Video (NOSSDAV), 83-86. Recuperado a partir de http://www.cs.columbia.edu/~hgs/papers/Schu9807_Comparison.pdf

- Shim, E., Narayanan, S., & Daley, G. (2006). An architecture for peer-to-peer session initiation protocol (P2P SIP). (work in progress), IETF draft, draft-shim-sippingp2p-arch-00, 1-30. Recuperado a partir de http://www.p2psip.org/drafts/draft-shim-sipping-p2p-arch-00.txt
- Singh, K., & Schulzrinne, H. (2005). Peer-to-peer internet telephony using SIP. NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video, 63-68. doi:http://doi.acm.org/10.1145/1065983.1065999
- Srisuresh, P., & Egevang, K. (2001). RFC-3022: Traditional IP Network Address Translator (Traditional NAT), Jan. 2001. Status: Informational, 1-16. Recuperado a partir de http://tools.ietf.org/html/rfc3022
- Tsirtsis, G. (2000). RFC-2766: Network Address Translation - Protocol Translation (NAT-PT), 1-21. Recuperado a partir de http://www.rfc-editor.org/rfc/pdfrfc/rfc2766.txt.pdf
- Tunnels, D. (2003). Tinc, The Userspace VPN Daemon: Own a Private Network. Know How, (November), 50-51.
- Tyapa, E. (2010). Peer-to-Peer Session Initiation Protocol Standardisation Progress. Rhodes University. Recuperado a partir de http://www.cs.ru.ac.za/research/g10t3627/P2PSIPLiteratureReview.PDF
- Understanding SIP. (2003). Quality, 6.
- Wolinsky, D., Lee, K., Boykin, P. O., & Figueiredo, R. J. (2011). SociaIDNS: A Decentralized Naming Service for Collaborative P2P VPNs, 10.
- Zheng, X., Guo, W., Zhong, S., & Yu, Z. (2012). A secure and hierarchical architecture for P2PSIP session initiation. Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2012, 1183-1189. doi:10.1109/IPDPSW.2012.144
- D. Bryan, «Use Cases for Peer-to-Peer Session Initiation Protocol (P2P SIP): draft-bryan-p2psip-usecases-00», pp. 1-12, 2008.