"Diseño de una Plataforma de Computación Distribuida Cooperativa, Utilizando Servicios de una Red Compañero a Compañero"

Tesista:

Fernando Raúl Alfredo Bordignon

Director

Prof. Jorge Ardenghi

Trabajo final presentado para obtener el grado de Magister en Redes de Datos

Universidad Nacional de La Plata
Facultad de Informática

- Mayo 2005 -

Resumen

"Diseño de una Plataforma de Computación Distribuida Cooperativa, Utilizando Servicios de una Red Compañero a Compañero"

Fernando Raúl Alfredo Bordignon

Maestría en Redes de Datos
Universidad Nacional de La Plata
Facultad de Informática

Director Prof. Jorge Ardenghi

El modelo de arquitectura compañero a compañero (P2P), utilizado en la construcción de sistemas distribuidos, ha tomado un auge importante con la expansión de Internet. Los servicios basados en sistemas P2P permiten que computadoras de usuario final se conecten directamente formando comunidades, cuya finalidad es el compartir recursos de distinta índole.

En esta tesis se exponen, a modo de contribución original, un prototipo de middleware denominado GnutWare, que brinda soporte de comunicaciones a aplicaciones de usuario sobre una red que opera bajo el modelo de comunicaciones compañero a compañero; y un diseño de un servicio de cómputo masivo distribuido denominado P2P-Flops, que está soportado sobre la mencionada red, donde un conjunto de computadoras de usuario final donan ciclos de CPU ociosos a determinados proyectos que requieran cómputo intensivo.

INDICE

Agradecimientos

A los forjadores y defensores de la educación pública, gracias a ellos soy un ciudadano libre, ilustrado y agradecido infinitamente por la oportunidades que me brindó este bendito país.

A mi familia, especialmente mi madre y hermanos.

A mis colegas y amigos Jorge Peri y Gabriel Tolosa, por sus acertados comentarios.

A mis compañeros de trabajo y amigos, por "bancarme" y facilitarme recursos para desarrollar este de trabajo .

A mi director Jorge Ardenghi, por confiar en mi.

A mis alumnos, por ellos día a día trato de perfeccionarme.

A la Universidad Nacional de Luján por brindarme la oportunidad de desarrollarme técnica y humanamente.

Organización

En el primer capítulo se realiza una presentación del modelo de arquitectura de construcción de sistemas distribuidos denominado "compañero a compañero". Se enfoca el estudio hacía las últimas aplicaciones emergentes en Internet.

El segundo capítulo trata sobre un par de aplicaciones paradigmáticas que implementan la filosofía compañero a compañero. Se describen con sumo grado de detalle, a modo de ejemplo de aplicaciones exitosas de uso masivo, los sistemas clásicos de búsqueda y almacenamiento distribuido de archivos Napster y Gnutella.

En el tercer capítulo, como contribución original, se presenta un prototipo de middleware que brinda soporte de comunicaciones a aplicaciones de usuario final sobre una red de aplicación (overlay network) que opera bajo el modelo de comunicaciones compañero a compañero.

A continuación, en el cuarto capítulo, se presenta un relevamiento bibliográfico sobre proyectos de cómputo masivo distribuido en Internet.

En el capítulo quinto, como contribución original, se expone el diseño de un sistema de cómputo masivo distribuido denominado P2P-Flops, que opera sobre la intranet de una organización y utiliza el middleware GnutWare como soporte de comunicaciones.

Finalmente, en el sexto capítulo se presentan los trabajos futuros y conclusiones.

Capítulo I

I El modelo Compañero a Compañero

I.1 Introducción

En los últimos años han surgido una serie de aplicaciones en Internet, operando bajo el modelo de comunicaciones denominado compañero a compañero (P2P). Cada una de estas aplicaciones implementan un servicio que opera de manera distribuida, tomando y brindando recursos, en computadoras de usuario final.

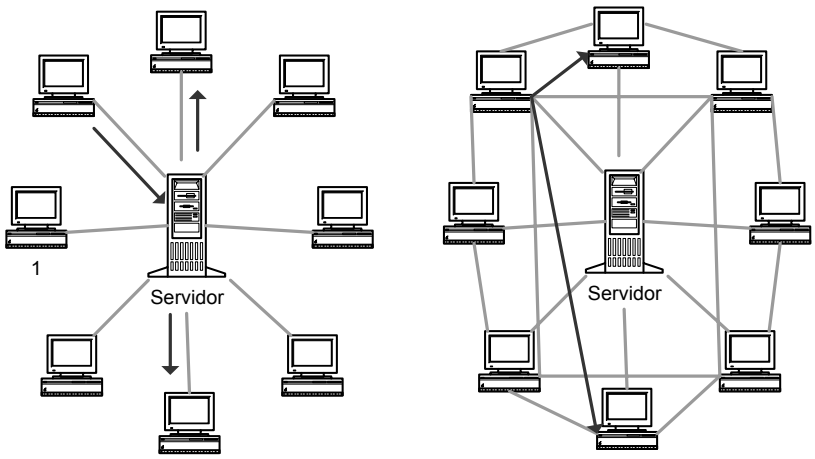
Aplicaciones P2P han ganando popularidad, programas de mensajería, tales como ICQ [ICQ] y MSN Messenger [MSN], y de intercambio de archivos, como Napster [Napster] y Gnutella [Bordignon], son las más notables en el ámbito de las aplicaciones P2P.

El concepto de compañero a compañero no es nuevo. De forma simple puede verse como la comunicación entre pares o iguales utilizando un sistema de intercambio. En telefonía se utiliza este modelo, donde se requiene una ó n centrales de conmutación a los efectos de mantener una comunicación entre usuarios finales. Actualmente, los usuarios de computadoras poseen equipamiento con buenas capacidades de procesamiento –es mucho más que un teléfono o una terminal boba- y estas características son las que utiliza plenamente la computación P2P.

Los sistemas compañero a compañero permiten que computadoras de usuario final se conecten directamente para formar comunidades, cuya finalidad sea el compartir recursos y servicios computacionales [FIPA] [Minar]. En este modelo, se toma ventaja de recursos existentes en los extremos de la red, tales como tiempo de CPU y espacio de almacenamiento. Las primeras aplicaciones emergentes se orientaban a compartir archivos y a la mensajería.

IBM plantea que el modelo P2P es el resultado natural de las tendencias de descentralización en la ingeniería de software [Sunsted], e Internet determinó que aplicaciones clásicas, de tipo monolítico, cambien a esquemas descentralizados.

Para la firma Intel [Barkai] el modelo P2P provee una alternativa a la arquitectura clásica cliente/servidor. Utilizando la infraestructura actual, compuesta de redes, servidores y clientes, P2P ofrece un modelo ortogonal al modelo cliente/servidor; dado que los dos modelos coexisten, se intersectan y se complementan. En un ambiente P2P, cada computador es un compañero que provee servicios de cliente y servidor a la vez., dependiendo de la aplicación, se determinará qué recursos tomará y qué recursos brindará a la red.



Modelo cliente/servidor *Modelo P2P*

En los gráficos anteriores se muestra lo expresado por la firma Intel, donde la insfraestructura de una red cliente/servidor puede servir de base para montar una red P2P. Nótese que la función de equipo servidor puede estar presente en una red P2P, dado que bajo ciertas circunstancias puede ser útil tener equipos con una funcionalidad distinta.

I.2 Orígenes de los sistemas P2P sobre Intenet

En el año 1969, la Agencia de Proyectos de Investigación Avanzados (ARPA) conectó cuatro computadoras de universidades estadounidenses (UCLA, Stanford Research Institute,

UCSB, y la Universidad de Utah). Donde el principal objetivo de este proyecto era desarrollar una red que pudiera resistir a ataques de un país enemigo y como objetivo secundario se propuso el desarrollo de un método que permitiera compartir poder computacional situado en computadoras remotas.

Según Arthur Bushkin, participante activo del proyecto, *"ninguna de las personas que formaban parte del grupo inicial sabía que se estaba gestando y eran incapaces de imaginarse que la red ARPAnet iba a ser precursora de Internet"* [Bushkin]. Cuando ARPA puso en línea a las computadoras mencionadas estableció una red P2P, donde cada una ofrecía y tomaba a la vez recursos de las otras.

Las primeras aplicaciones populares, tales como telnet y FTP, operaban bajo el esquema cliente/servidor. En un principio, aunque las aplicaciones fueran cliente/servidor, el uso de las computadoras donde residían tales clientes y servidores eran simétrico; debido a que casi no existían computadoras dedicadas a tareas de servidor exclusivamente y por otro lado las computadoras personales ó de usuario final no existían. Sistemas propios de Internet, como USENET y DNS, utilizaron la tecnología compañero a compañero [Minar] en su infraestructura.

I.2.1 USENET

Es un sistema distribuido de conferencias, donde a cada una se las denomina grupos de noticias y a los mensajes artículos [Palmer]. En su infraestructura, USENET implementa un modelo descentralizado de control, donde fundamentalmente permite la copia de archivos entre computadoras sin la existencia de un control central.

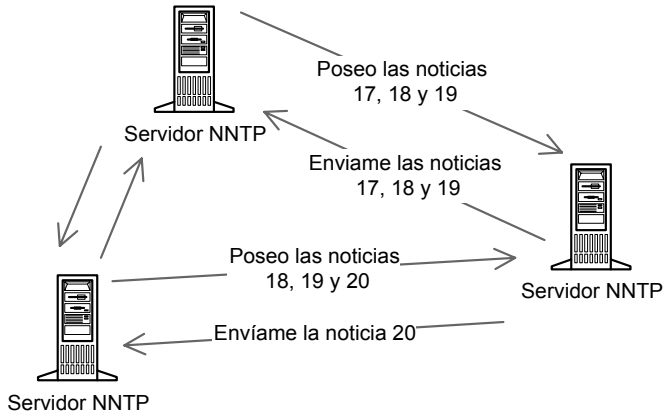
Su construcción se basó en un servicio propio del sistema operativo UNIX denominado UUCP (Unix to Unix Copy Protocol). Bajo UUCP una computadora, vía modem, discaba a otra, se conectaba e intercambiaba archivos y finalmente se desconectaba. Este mecanismo permitía en la práctica el intercambio de correo electrónico, revisiones de código, etc. Estudiantes de la Universidad de Carolina del Norte y de la Universidad Duke

implementaron, utilizando UUCP como transporte, un protocolo para el intercambio de artículos de difusión y de respuesta, que abarcaban una serie amplia de tópicos. Más instituciones se sumaron al sistema de conferencias y la red USENET creció hasta llegar a operar con cientos de miles de sitios. Este crecimiento involucró reformas en la infraestructura de la red, y se decidió utilizar el juego de protocolo TCP/IP como soporte del protocolo de transporte de noticias USENET (NNTP, Network News Transport Protocol) [RFC977] [RFC1036], el cual posibilita que dos computadoras de la red USENET puedan descubrir nuevos grupos de noticias de forma eficiente e intercambiar nuevos mensajes.

USENET es un ejemplo de un sistema de gran escala que posee control descentralizado, dado que no existen autoridades que controlan la publicación de noticias. A los efectos de agregar nuevos grupos se realiza un proceso de votación.

En toda organización de mediano porte (por ejemplo una universidad o un proveedor de acceso a Internet), generalmente, existe un servidor de noticias local al cual sus usuarios se conectan para recibir y enviar sus artículos. Sobre el mencionado servidor se implementan dos variantes del protocolo NNTP: una destinada a la comunicación entre pares (otros servidores) y otra destinada a la comunicación con sus clientes. Cada servidor puede descargar de servidores adyacentes, lo que éste quiera o lo que este disponible. Existe en cada servidor una lista de seguimiento de grupos de noticias que se utiliza a los efectos de que un servidor pueda actualizar información de los grupos que localmente atiende y no recibir noticias duplicadas (una especie de control de ciclicidad sobre la red). Cada mensaje perteneciente a cualquier grupo de noticias lleva un identificador que lo hace único, por medio del cual los servidores y los clientes pueden administrar sus descargas a los efectos de obtener siempre las noticias nuevas y de no recibir duplicadas.

La siguiente figura muestra como los nuevos artículos son reenviados de un servidor de noticias a otro. Un servidor le dice a sus adyacentes que puede él ofrecer, y cada uno consulta sus tablas de seguimiento y decide que descargará.



El servidor de noticias no maneja información alguna acerca de el estado de la lectura por parte de sus clientes. Éstos, los clientes, deben implementar mecanismos para el control de la actualización de las noticias.

El modelo propuesto por usenet, en vigencia desde 1979, ha demostrado ser útil y exitoso, permitiendo la comunicación eficaz entre grupos de personas que forman comunidades temáticas a lo largo del mundo.

I.2.2 DNS

El sistema de registro de nombres de dominio (DNS) [RFC822] combina una red P2P, conformado por el conjunto de servidores de nombres, con un modelo de información jerárquico [Minar]. Esta infraestructura permitió que DNS [RFC974] [RFC1034] [RFC1035] escale de miles de equipos (en 1883) a millones actualmente. La razón por la cual surgió DNS fué a partir de buscar un mecanismo más ágil que el intercambio del archivo "/etc/hosts", que contenía la asociación de nombres de equipos con direcciones de red. DNS fue desarrollado como una forma de distribuir datos pausibles de ser compartidos a lo largo de la red Internet.

El espacio de nombres definido en DNS es jerárquico, y los equipos servidores de nombres operan bajo una estructura simétrica, dado que actuan como clientes y servidores a la vez. Son servidores desde el punto de vista que brindan información sobre su dominio ó

eventualmente sobre otros dominios (en el caso de que esté almacenada en su memoria cache); y clientes porque envían consultas a otros pares[1], en nombre de sus clientes .

La escalabilidad, probada, del sistema DNS se basa en una serie de elementos de diseño, como ser:

a- Que los servidores DNS pueden operar como servidores y clientes, propagando consultas de terceros.

b- Que los servidores pueden almacenar y consultar su memoria cache a los efectos de satisfacer consultas sobre otros dominios de forma local.

c- Que un servidor puede consultar a otro, pero existe un camino de consulta en base a una jerarquía preestablecida o cadena de autoridad (aqui las cargas son balanceadas por medio de la jerarquía).

Analizando otros protocolos, tales como SMTP (Single Mail Transport Protocol) [RFC822], también se puede observar que el principio de simetría, clásico en entornos P2P, está presente entre el conjunto de servidores que conforman la red de correo electrónico de Internet.

I.3 Hacía una definición de sistema P2P

Tradicionalmente el modelo P2P es un tipo de red en la cual cada estación de trabajo tiene capacidades y responsabilidades equivalentes. La diferencia con la arquitectura cliente/servidor están dada por que en este caso algunas computadoras estan exclusivamente dedicadas a brindar servicio a otras.

La definición anterior está basada en la concepción tradicional de las redes P2P, donde computadoras de usuario final, generalmente en el ámbito de una red local, formaban un

[1] Cuando operan en modo de consulta recursiva.

grupo de trabajo y cada usuario podía permitir que otros compañeros accedan a sus archivos o impresoras locales.

El profesor Clay Shirky[2] [Shirky] caracteriza a la tecnología P2P como " una clase de aplicaciones que toman ventaja de los recursos –almacenamiento, ciclos de CPU, contenidos, presencia humana- disponibles en Internet", y define una prueba, que consta de dos preguntas, a los efectos de validar si una aplicación determinada puede ser definida como compañero a compañero

1. ¿Es normal que los usuarios tengan conectividad variable y se les asigne direcciones de red temporales?

2. ¿Se da a los nodos situados en los extremos de la red un importante grado de autonomía?

Donde si ambas preguntas son contestadas afirmativamente la aplicación es compañero a compañero.

La organización FIPA -Foundation for Intelligent Pyhsical Agents- [FIPA] ve los sistemas P2P como un medio para "compartir recursos y servicios computacionales por medio de intercambio directo". Las aplicaciones populares incluyen intercambio de archivos y procesamiento distribuido. Debido a que la computación P2P involucra la participación de los nodos de una red sin administración central, se deben considerar las siguientes cuestiones:

- Los nodos en una red P2P pueden ingresar y salir constantemente de forma arbitraria.

- La ubicación de los servicios y recursos es dinámica.

[2] Escritor que habitualmente trata sobre aspectos sociales y económicos de las aplicaciones P2P. Coopera con O´Reilly Network, New York Times, Wall Street Journalb y Harvard Business Review.

- Cada nodo regula su grado de participación en la red

- La red es un ambiente dinámico y heterogéneo.

Garcia-Molina [Yang] define un sistema P2P como "nodos de computación distribuida con iguales roles y capacidades de intercambio de información y servicios directamente entre ellos". La gran ventaja que aporta P2P es que los recursos de muchos usuarios y computadoras pueden unirse para producir un gran cúmulo de información y un significativo poder de cómputo.

Para la firma Intel [Barkai] la computación P2P es una forma de compartir recursos computacionales y servicios mediante intercambio directo. Un nodo puede actuar como cliente y servidor en el contexto de una aplicación dada. La interacción entre nodos (requerimientos y respuestas), su ámbito y como son procesados los mensajes son cuestiones específicas de la aplicación de usuario final. Por otro lado, la firma indicó que uno de los más grandes beneficios de P2P es el concepto de comunidad, dado que hace posible que los usuarios se organicen ellos mismos en grupos ad hoc, y puedan eficientemente y de forma segura llenar requerimientos, compartir recursos, colaborar y comunicarse.

Bajo la óptica del autor de este documento, un sistema compañero a compañero es un modelo de uso coordinado de recursos –ciclos de CPU, espacio de almacenamiento, archivos, etc- sobre la base de la comunicación directa entre nodos que los requieren y los poseen, bajo una topología de red donde todos los participantes tienen igual acceso a todos sus pares, y eventualmente pueden existir nodos con alguna capacidad especial.

Los sistemas P2P utilizan la descentralización, como una forma de incrementar la perfomance, escalabilidad y la disponibilidad de los recursos de los usuarios que conforman tal sistema.

I.4 Características de los sistemas P2P

La primera característica que define a un nodo en un sistema P2P, es que cumple tanto el rol de cliente como de servidor (en la terminología se lo denomina *servent*, palabra que deriva de la conjunción de los términos *serv*er-*client*). Este modelo basado en interacciones entre pares, puede reemplazar al paradigma cliente/servidor, donde cada nodo es capaz de generar y contestar consultas de otros nodos. Desde el punto de vista de la comunicación, las interacciones son simétricas.

- Los nodos participantes son autónomos. Cada uno regula su grado de participación en la red, a través de definir que recursos ofrece y en que cantidad.

- Ninguno de los nodos tiene una vista global del sistema.

- El comportamiento global emerge de las interacciones individuales directas.

- Todos los datos y servicios deben ser accesibles por cualquier nodo.

- Implementan un sistema de nombres alternativo al sistema DNS, que satisfaga sus propias necesidades.

- Los nodos en una red P2P pueden ingresar y salir constantemente de forma arbitraria. Como son nodos de usuario final no se encuentran permanentemente conectados a la red, es decir, su conectividad es variable.

- La ubicación de los recursos y disponibilidad de servicios es dinámica, ya que depende del estado del sistema en un momento del tiempo.

- La red es un ambiente dinámico y heterogéneo, dado que la conforman nodos con conectividad variable y de diversas plataformas de hardware y software. Toda computadora en Internet tiene la posibilidad de que sobre éste corra un nodo P2P.

La empresa Microsoft en el marco de su producto .NET [.NET] brindó una serie de características propias de cada nodo en un sistema compañero a compañero:

- Capacidad de descubrir compañeros. El nodo debe poder encontrar a sus pares a los efectos de saber quienes son y donde están. Esta información se puede obtener consultando a un servidor central que registra nodos ó a través de algoritmos de descubrimiento.

- Realizar consultas a compañeros.

- Compartir recursos con otros compañeros.

I.5 Sistemas cliente/servidor versus sistemas P2P

Las ventajas y desventajas de las redes basadas en un esquema cliente-servidor frente a una red compañero a compañero, son los derivados de la centralización de recursos:

- Un servidor dedicado ofrece una mayor capacidad de trabajo que una máquina que opera además como estación de trabajo.

- El poseer información centralizada ofrece más seguridad contra accesos no autorizados.

- Es más práctico actualizar programas y realizar copias de respaldo en un modelo de red centralizada.

- No se corre el riesgo de que por mal funcionamiento de una estación de trabajo corra peligro todo el sistema.

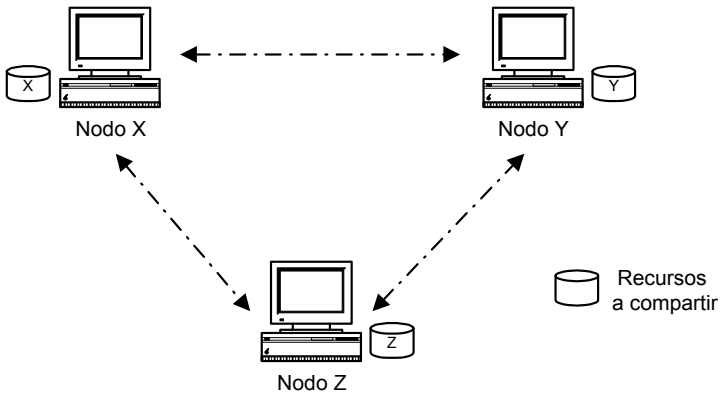
- La desventaja fundamental es que si ocurriese algún problema en el servidor todas las estaciones de trabajo podrían verse afectadas.

I.6 Tipos de redes compañero a compañero

Si un sistema P2P está íntegramente conformado por nodos que tienen iguales roles y ninguno cumple tareas administrativas o especiales, se lo considera P2P puro o completamente descentralizado [Yang]. En cambio si algún nodo posee alguna característica diferente que le hace cumplir con tareas centralizadas, entonces al sistema se lo considera híbrido.

I.6.1 Redes puras

Es una democratización total del grupo de compañeros, cada nodo participante de la red posee las mismas capacidades que los demás, cualquier nodo puede iniciar una comunicación o hasta una red [Yang]. Freenet [Clarke] y Gnutella son ejemplos de este modelo. No existen recursos centralizados, por ende no existen los riesgos propios de tales sistemas, en los cuales existen puntos críticos de falla. Tiene problemas para descubrir compañeros y localizar información.



I.6.2 Redes híbridas

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

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

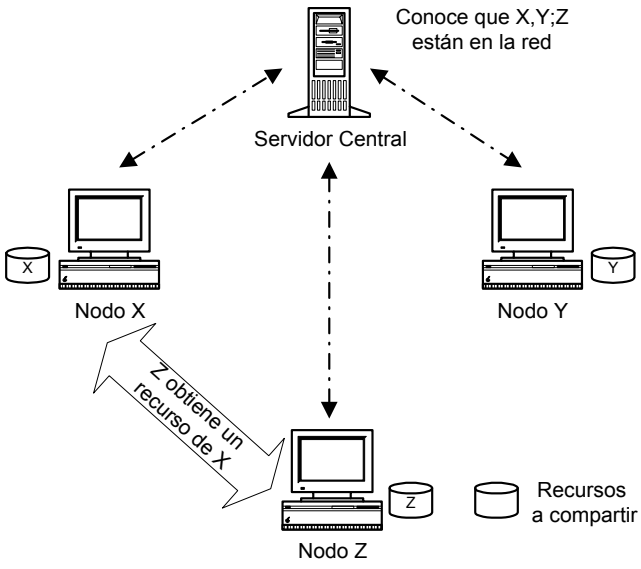
2. El nodo peticionante, en modo cliente, envía un requerimiento al nodo encaminador, éste devuelve la dirección del nodo compañero que en modalidad servidor satisfacerá tal consulta, el nodo cliente se comunica con el nodo que oficiará de servidor enviándole su requerimiento de recurso, finalmente el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

En la categoría de redes híbridas, la empresa Microsoft, presenta una serie de configuraciones en su entorno denominado .NET [.NET]:

I.6.2.1 Sistema con servidor de descubrimiento

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

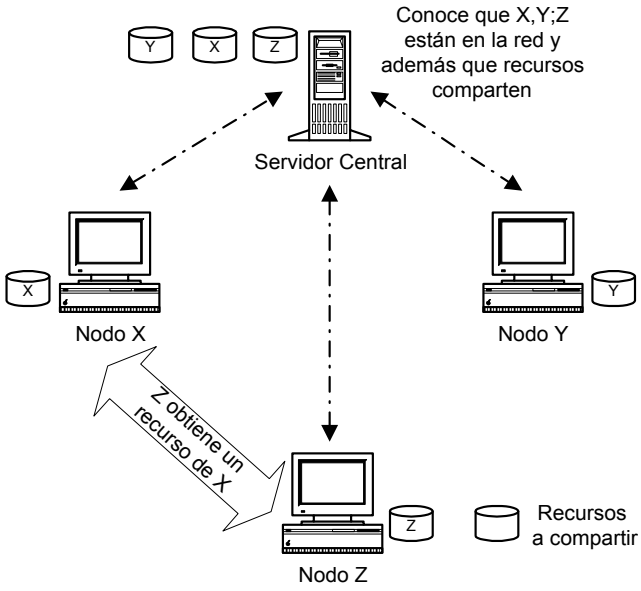
Una alternativa posible destinada a dotar de una mayor disponibilidad el servicio provisto por los nodos centrales es que el contenido pueda ser replicado automáticamente, y que los nodos compañeros puedan obtener transparentemente direcciones de los servidores con réplicas.



I.6.2.2 Sistema con servidor de descubrimiento y búsqueda

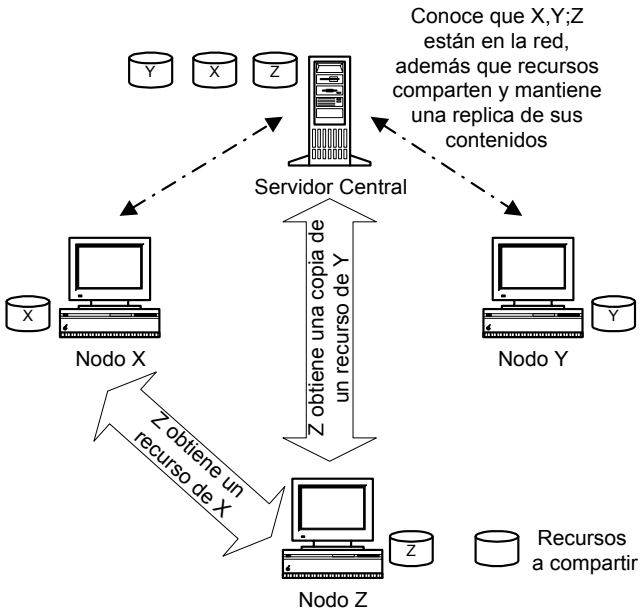
Se basa en que al sistema con servidor de descubrimiento se le adosa el servicio de base de datos y búsqueda de recursos. Para lo cual cada nodo compañero que registre su ingreso al sistema, además debe informar que recursos comparte. Cualquier nodo, además de obtener la lista de sus

compañeros, ahora puede consultar al servidor central por quién ó quienes poseen un recurso determinado.



I.6.2.3 Sistema con servidor de descubrimiento, búsqueda y contenidos

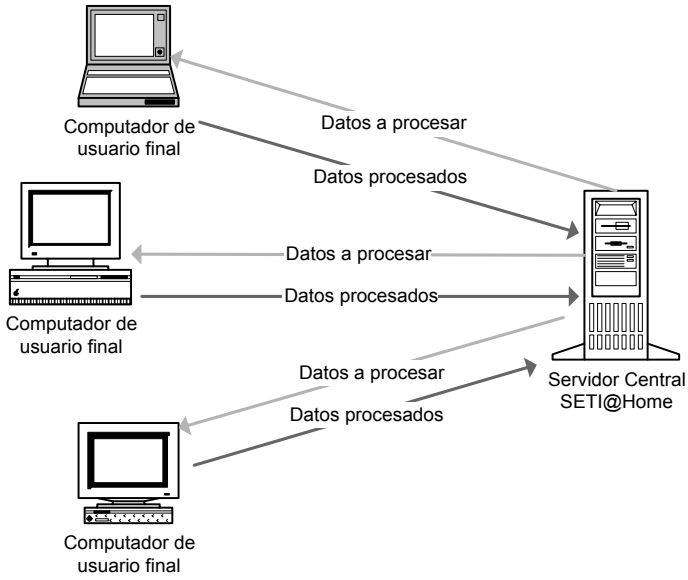
Este sistema representa un acercamiento al modelo cliente-servidor, dado que al sistema anterior se le agrega la posibilidad de que contenga réplicas de los recursos que comparten los usuarios. Donde un nodo compañero que desee un recurso puede optar por descargarlo de un nodo compañero o del servidor central.



Varios proyectos de cómputo masivo distribuido utilizan ciclos de CPU ociosos en máquinas de usuario final. Aunque muchos proyectos no sean P2P generalmente se los asocia con tales; y no son P2P debido a que no existe comunicación directa entre pares, pero se los asocia porque: utilizan los recursos de computadoras de borde (equipos de usuario final principalmente utilizados para navegación), poseen conectividad variable y a los nodos se les asigna direcciones temporales de red.

A diferencia de lo que comúnmente se proclama, el autor de este documento, entiende que el proyecto SETI@home[3] [Korpela] no pertenece a ninguna categoría de red P2P, dado que no existe comunicación directa entre pares. Su infraestructura se basa en el modelo Maestro/Esclavo, donde un nodo central se encarga de repartir unidades de trabajo entre distintas computadoras de usuario final. De esta forma, cuando cada computador cooperante detecta que tiene tiempo de CPU libre, dedica tal recurso ocioso a procesar datos, a los efectos de buscar señales indicatorias de inteligencia extraterrestre.

[3] SETI@home es un proyecto de búsqueda de inteligencia extraterrestre de la Universidad de California en el Laboratorio para Ciencias Espaciales de Berkeley



I.7 Auto-organización:

Las arquitecturas P2P generan sus propias organizaciones para aplicar a sus nodos. Algunos de los impactos de las comunicaciones P2P se relacionan con la auto-organización. Esto puede ocurrir en dos niveles:

- Con respecto a una comunidad P2P determinada, cualquier computador podría conceder el permiso de ingreso a la comunidad a un nuevo nodo. Aquí el proceso de incorporar nodos no requiere una reorganización central de la red.

- Sobre un esquema de gran escala, millones de nodos, las comunidades P2P pueden organizarse naturalmente –de acuerdo a intereses compartidos- en grupos de por ejemplo miles de nodos.

La característica de la auto-organización es una diferencia importante respecto de las formas centralizadas de organización (cliente/servidor).

I.8 Arquitectura típica de un nodo compañero

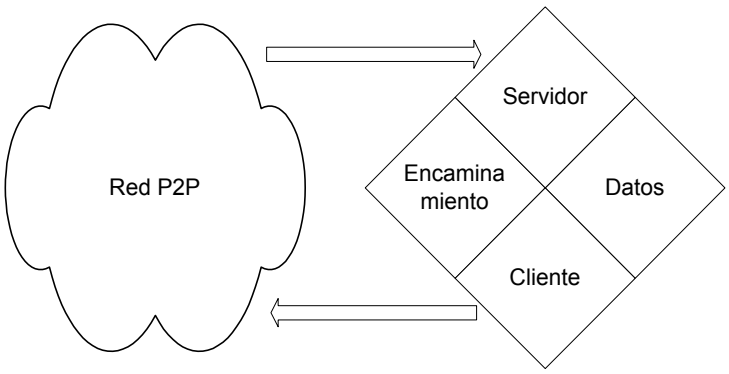
Un nodo ideal de una red compañero a compañero se forma con cuatro componentes básicos a saber:

- Módulo servidor. Parte de un nodo compañero que provee acceso a datos y recursos del compañero donde se ejecuta. Acepta conexiones entrantes.

- Módulo cliente. Parte de un nodo compañero que le permite al usuario comunicarse con otros nodos, iniciando conexiones.

- Módulo de datos. Parte de un nodo que almacena información sobre el estado del nodo y la red.

- Módulo de encaminamiento. Parte de un nodo que tiene por función principal propagar mensajes propios y de terceros.



Componentes de un nodo perteneciente a una red P2P

Funciones del componente cliente

- Proporciona la interfase de usuario.
- Permite enviar requerimientos
- Proporciona conexión con otros nodos compañeros.

Funciones del componente servidor

- Detecta, procesa y contesta requerimientos de otros nodos.
- Proporciona acceso a sus recursos (Ej. archivos, tiempo de CPU, espacio en disco).
- Acepta conexiones entrantes.

Funciones del componente datos

- Define un esquema para el almacenamiento de datos y metadatos
- Satisface requerimientos de los componentes servidor y cliente.
- Almacena información sobre el estado del nodo y la red (Ej. lista de nodos compañeros)

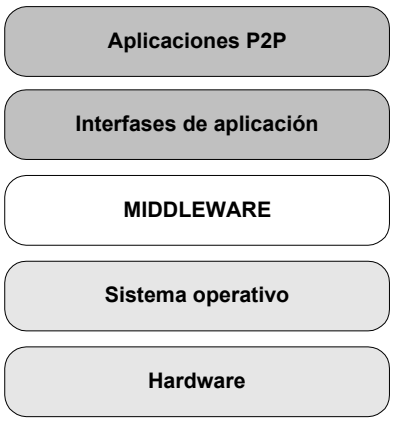
Funciones del componente ruteo

- Envía y recibe mensajes de los y a los componentes cliente y servidor
- Administra mensajes duplicados, tiempos de vida de mensajes y trata de optimizar la perfomance de la red.
- Gerencia la confidencialidad, autenticidad e integridad de las comunicaciones.
- Propaga mensajes generados localmente ó de terceros.

La propagación de mensajes en una red descentralizada generalmente se realiza utilizando campos origen y destino.

Ejemplo de una estructura de datos típica de un mensaje de red P2P

La firma Intel proporciona la idea de que debe desarrollarse un modelo de interoperabilidad entre aplicaciones P2P [Barkai], donde tal objetivo puede lograrse a través de un juego común de servicios que proporcionen la funcionalidad necesaria para operar en modo P2P, sobre las funcionalidades proporcionadas por el sistema operativo. Estos servicios comunes pueden pensarse como una capa del middleware. Uno de las ventajas principales de un middleware común es que los diseñadores de aplicaciones ya tendrán que seguir creando los mismos servicios básicos una y otra vez. Tal interoperabilidad permitirá que equipos con distintos sistemas operativos y con distintas aplicaciones, programadas en distintos lenguajes de programación, puedan comunicarse e integrarse.

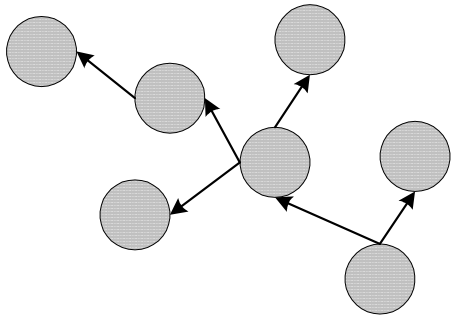


Capa de middleware P2P propuesta por Intel

I.9 Mecanismos de exploración

Las redes P2P necesitan mecanismos de exploración (discovery) que sean robustos y eficientes a los efectos que las aplicaciones puedan enviar mensajes a sus pares. Se evalúan mecanismos basados en difusión (ó inundación) y en reenvío directo (forward) de mensajes (tal como es el caso de los implementados en los proyectos Chord, Pastry y CAN). Se presentan una serie de mecanismos de descubrimiento de nodos compañeros que se implementan sobre redes descentralizadas.

I.9.1 Exploración por difusión

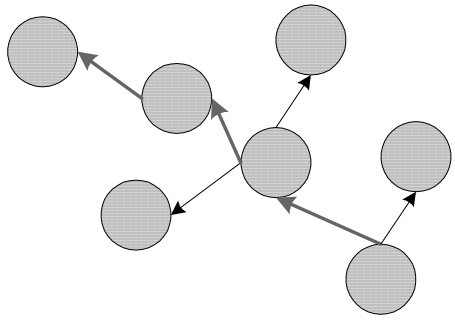


Envío por difusión

Son modelos descentralizados destinados a la tarea de descubrir vecinos. Un ejemplo es el utilizado en Gnutella. Los nodos compañeros forman una red de recubrimiento (overlay) sobre la red de transporte. A los efectos de hallar un recurso un nodo inunda la red con su requerimiento. Los nodos compañeros que posean tal recurso responderán debidamente. En la práctica, el área a alcanzar por un nodo está limitada, generalmente, a una serie de reenvíos[4].

La robustez y el buen alcance que logra este método hace que sea un atractivo recurso para los desarrolladores. Los resultados óptimos se dan sobre redes chicas, con un limitado número de nodos, dado que adolece de problemas de escalabilidad. Por otro lado, en grandes redes, este mecanismo no garantiza siempre un funcionamiento eficiente.

I.9.2 Exploración por encaminamiento



Envío por encaminamiento

[4] Similar al funcionamiento del mecanismo Time To Live del protocolo de red IP.

Otra alternativa de envío de mensajes es vía encaminamiento. Un mensaje va dirigido a un nodo en particular y los nodos intermedios, que cooperan para propagar mensajes, redireccionan un único mensaje hasta llegar a destino. Para lograr este funcionamiento, los nodos en el camino deben poseer información acerca de la topología de la red.

Dado que no existe una única topología en una red P2P (bajo ciertas circunstancias pueden existir ciclos), los nodos intermedios deben implementar un control para evitar reenvíos sucesivos. Esto se puede realizar utilizando una memoria cache que almacene los identificadores de los mensajes recientemente recibidos (durante un lapso de tiempo) a los efectos de poder determinar repeticiones de mensajes. Por otro lado, un nodo cada vez que propaga un mensaje decrementa y chequea un campo tipo TTL (time to live) a los efectos de eliminar mensajes luego que han circulado durante una serie de nodos.

I.9.2.1 Chord

El proyecto Chord [Dabek] es una parte de un proyecto más ambicioso denominado "Sistema de Archivos Distribuido y Seguro". En Chord se implementa la búsqueda y el encaminamiento sobre una red virtual representada como espacio circular unidimensional. Cada clave puede ser localizada en O (logN) saltos, donde N es el número total de nodos en el sistema. La garantía de que se hallará al nodo deseado es una de los puntos sobresalientes de Chord.

I.9.2.2 Pastry

Pastry [Rowstron] es una aplicación descentralizada destinada a implementar una infraestructura de ruteo a nivel aplicación y un sistema de localización de objetos. Cada nodo es identificado en la red de aplicación por un identificador unívoco (NodeID). Los identificadores se asignan sobre la base de una métrica de proximidad. Cada salto de un

mensaje se hace sobre la base de la selección del nodo más cercano, al nodo destino al cual debe enviarse mensaje, información extraída de la tabla de ruteo de cada nodo. Su eficiencia es O (logN). Cada nodo mantiene información sobre que nodos están operativos y sobre sus vecinos, esto lo logra por medio de mensajes de estado. Pastry es un algoritmo genérico de encaminamiento por auto-organización que puede utilizarse para almacenamiento, intercambio de datos, mensajería instantánea y sistemas de mapeo de nombres.

I.9.2.3 CAN (Content-Addressable-Networks)

CAN [Ratnasamy] es una malla de n nodos sobre un espacio virtual de d dimensiones. El espacio virtual coordinado es dinámicamente dividido en zonas y cada nodo conoce su zona de pertenencia en el espacio. Una coordenada espacial es utilizada para almacenar pares (clave, valor) tal que cada clave es de forma determinística mapeada en un punto P sobre la coordenada espacial, utilizando una función uniforme de hash. El dato (clave, valor) se almacena finalmente en el nodo, donde cuya zona está el punto P. Para recuperar un valor correspondiente a la clave k, cualquier nodo aplica la misma función de hash para mapear k sobre el punto P y luego recuperar el valor almacenado en P de forma directa o a través de nodos vecinos en O ($n^{1/d}$). De acuerdo a sus autores CAN es tolerante a fallas, robusto, con estrategias de auto-organización y provee un marco de escalabilidad mayor que la existente en los métodos de inundación.

I.10 Áreas de aplicación

Trabajo en grupo: La colaboración incrementa la productividad disminuyendo el tiempo de múltiples revisiones por parte de los participantes de un proyecto. Además, permite trabajar juntos a equipos en distribuidos geográficamente. Disminuye el tráfico en la red por la disminución de e-mail y disminuye la necesidad de almacenamiento en servidores centrales, utilizando almacenamientos locales con posibilidad de replicación automática.

Groove [Groove] es una aplicación reciente, ejemplo de ambiente colaborativo, donde los usuarios pueden crear "espacios compartidos". Los objetos y el contenido dentro de dichos

espacios aparecen en todas las máquinas del equipo, y cada usuario puede contribuir con los archivos o los programas de su elección.

Replicación de contenidos: Una red de aplicación soportada sobre una infraestructura compañero a compañero puede ayudar a distribuir contenidos a través de distintas áreas geográficas [Tolosa]. Esencialmente pueden moverse los datos más cerca del punto donde realmente son consumidos, implementando mecanismos de replicación. Las redes de distribución de contenidos (CDN, Contents Distribution Networks), que tienen por objetivo lograr que los usuarios finales accedan más rápidamente a contenidos web, ya han comenzado a incorporar el modelo P2P, tal como en el caso del sistema Swarmcast.

Computación distribuida: Necesidades de procesamiento en gran escala pueden ser provistas utilizando una red de computadoras donde sus componentes puedan aportar capacidad ociosa de CPU y espacio en disco. Este servicio permite a Universidades, institutos de investigación y empresas obtener un gran poder de cómputo, a partir de múltiples computadoras operando en paralelo (procesamiento distribuido sobre una red débilmente acoplada) para aplicar a tareas especificas. Los resultados se pueden obtener en tiempos y costos menores.

Actualmente proyectos como SETI@Home, FightAIDS@Home [Fight], Folding@Home [Folding] y Distributed.net [Dist] utilizan computación distribuida para resolver problemas de la más variada gama.

No todos los proyectos utilizan una red compañero a compañero pura. Intel, en su servicio de computación distribuida colaborativa, y SETI@home implementan servidores centralizados, sin comunicación entre pares. Sin embargo, estos clientes son participantes activos del sistema y el proceso de cómputo se basa en poseer redundancia masiva. En algunos casos, utilizan una red P2P como soporte de comunicaciones.

Intercambio de archivos: Permiten que usuarios de computadoras personales intercambien directamente entre ellos archivos de cualquier tipo. Estas aplicaciones han obtenido una gran

popularidad gracias a Napster, que en su momento acumuló 18 millones de usuarios en 18 meses de existencia. Se considera que no es un sistema P2P puro, debido a utiliza un servidor central donde los usuarios consultan para saber donde están los recursos. Actualmente estos sistemas focalizan sus contenidos en la distribución de archivos musicales, videos y fotografías.

Mensajería instantánea: La mensajería instantánea permite la comunicación en tiempo real entre personas conectadas a la red al mismo tiempo. El intercambio se realiza al escribir mensajes o al enviar archivos. Los programas más populares son AOL Instant Messenger [AIM] e ICQ. Según la empresa AOL, propietaria de ambos productos, sus productos son utilizados por alrededor de 150 millones de usuarios.

Búsquedas: Debido a la expansión de la web, los motores de búsqueda tradicionales experimentan problemas de cobertura y de actualización de contenidos. La implementación de servicios de búsquedas distribuidas basadas en redes P2P puede mejorar esta situación, permitiendo a sus usuarios la obtención de resultados de mejor calidad [Tolosa-A]. El proyecto InfraSearch [Infra] fue un punto de partida en este tipo de aplicaciones. Si bien puede ser complicado realizar búsquedas de gran cobertura del espacio web, los sistemas P2P ofrecen otra visión de cómo abordar el problema y plantean soluciones para búsquedas extendidas sobre temas específicos.

Comercio: Algunas compañías están experimentando con modelos de conexión directa entre usuarios a los fines de implementar sistemas de comercio electrónico. La idea es reemplazar un mercado electrónico centralizado por intercambios directos entre compradores y vendedores. La empresa Netrana está desarrollando herramientas y estrategias de comercio electrónico para situaciones que escapan al ámbito de los mercados electrónicos convencionales. Su objetivo es desarrollar una plataforma de comunicaciones que permita a los participantes llevar a cambio entre ellos esta negociación, sin la intervención de un intermediario. Un área propicia para las redes P2P será el comercio electrónico, debido al potencial de la tecnología de enlazar directamente vendedores y compradores. La empresa

Biz To Peer Technologies está diseñando estrategias para proporcionar a sus usuarios herramientas de informática distribuida con el fin de establecer transacciones entre particulares de manera directa.

I.11 El poder de los bordes

Las aplicaciones P2P que se ejecutan sobre Internet residen en computadoras personales[5], equipos que se los conoce como nodos de los "bordes de Internet" (Internet edges); por otro lado tales nodos no están registrados en el sistema de nombres de dominio debido a su conectividad variable y por consiguiente su asignación dinámica de dirección de red.

Una conjunción de tales características [Shirky] determinan a las aplicaciones P2P. Principalmente se aprovechan de recursos –espacio de almacenamiento, ciclos, contenidos, presencia humana-- disponible en los bordes de la red Internet. El acceso a tales recursos descentralizados significa que se debe operar en un ambiente de conectividad inestable y variable, con asignación imprevisible de direcciones de red, sin registro de recursos en el sistema DNS y donde los nodos P2P deben tener una importante autonomía[6].

Aplicaciones P2P, tales como Napster, Popular Power, Freenet e ICQ son populares debido a que utilizan recursos, hasta el momento no utilizados, disponibles en los bordes de Internet. Sus protocolos contemplan las características de comportamiento enunciadas en el párrafo anterior.

Hasta el año 1994, la red Internet en su conjunto poseía un modelo de conectividad. Se asumía que los equipos siempre estaban operativos, sus interfaces de red poseían direcciones fijas, El sistema DNS fue pensado sobre la base de estas características. A partir del establecimiento del sistema de información denominado World Wide Web ó simplemente web, un modelo alternativo al planteado comenzó a entrar en vigencia. Computadoras personales, con capacidades gráficas importantes, comenzaron a navegar a demanda por el sistema web. Fue necesario implementar centros de acceso a Internet (ISP), donde los

[5] Equipos que en modelo cliente/servidor siempre estuvieron relegados solamente al rol de cliente.
[6] Debido a que se trata de reducir y hasta evitar la presencia de servidores centrales.

usuarios utilizando el sistema telefónico obtenían una dirección de red temporal[7]. He aquí que este modelo se caracterizó por que los usuarios ingresaban y salían frecuentemente, en forma no predictiva, a la red; sin estar registrada su condición en los servidores DNS.

En el año 1996 se comenzó a utilizar el sistema de mensajería ICQ, fue la primera vez que las computadoras personales situadas en los bordes de Internet comenzaron a poder ser visibles por otros pares, dado que ICQ estableció un sistema de localización de usuarios donde se mapea la dirección temporal de red con el nombre de usuario. Esta técnica, luego utilizada por sistemas tales como Napster y Groove, permitió superar las carencias del sistema DNS sobre tales nodos. Una característica de los sistemas P2P es que permiten crear direcciones casi ilimitadamente para máquinas y otros recursos (volúmenes, personas). Aplicaciones como Freenet y MojoNation crean direcciones o nombres para máquinas y elementos que no lo son.

Dale Dougherty [Dougherty] utiliza el término PIE para describir los elementos centrales de las aplicaciones P2P. Donde PIE significa Presencia, Identidad y recursos de borde (Edge resources).

> ***Recursos de borde:*** Actualmente las aplicaciones P2P utilizan los recursos situados en los bordes de Internet, es decir que básicamente operan sobre computadoras personales hogareños y de oficina. Las computadoras personales son un recurso económico y a la vez poderoso. Dougherty plantea que un supercomputador del año 1987 es similar en perfomance a un computador personal estándar del año 2001. Los usuarios, generalmente, utilizan estos equipos en tareas que no consumen grandes recursos, como ser envío y recepción de correo y procesamiento de texto; estando ociosos la mayor parte del tiempo en que están encendidos. El problema más importante de los recursos de borde es que ellos son transitorios, sobre la base de la característica de conectividad variable.
>
> ***Identidad:*** Las redes P2P deben ser capaces de identificar unívocamente los recursos disponibles en ellas. Las aplicaciones P2P han tenido que implementar sus

[7] Reutilizable en otros periodos de tiempo por otros usuarios.

propios esquemas de servicio de nombres dado que no pueden depender del sistema DNS, cuyo diseño no le permite satisfacer las necesidades especiales de las computadoras situadas en los bordes de Internet. El sistema de correo electrónico de Internet es P2P, dado que si se lo analiza como el conjunto de servidores SMP se llega a la conclusión de que todos los nodos son compañeros, en este sistema no existe el concepto de equipo de borde y de sistema de identidad propio, dado que utiliza DNS; y por otro lado no es en tiempo real, dado que su operación es asincrónica

Presencia: Es la habilidad para comunicar que un recurso está en línea disponible. La mensajería instantánea es la aplicación con mayor grado de madurez analizada desde el punto de vista del elemento presencia. Los usuarios constantemente pueden saber si un compañero está en línea o no. La presencia permite hacer más eficiente el trabajo sobre la red, nótese el caso de la Aplicación AudioGalaxy que como parte de la respuesta a una consulta se obtiene el atributo disponibilidad que indica si un recurso existente puede ser obtenido inmediatamente ó no.

De acuerdo a Dan Bricklin [Bricklin] las redes P2P florecerán en ciertos escenarios y en otros no. Los casos donde serán bien acogidas son:

- Cuando un mismo dato existe en diferentes computadoras..

- Cuando el contenido de los archivos sea estático y la información descargada no varíe.

- Cuando la calidad de los datos pueda ser variable. El caso de los archivos de música mp3 para escuchar uno en su hogar, sin sistemas de sonido sofisticados.

- Cuando se disponga de una conexión confiable de alta velocidad.

I.12 P2P como infraestructura

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

A los efectos de consolidar el cómputo distribuido se necesitará desarrollar APIs y compiladores robustos desarrollados específicamente para tal ambiente. Compañías como Sun a través de su producto JXTA y Microsoft con Hailstorm Technologies están intentando brindar ambientes de desarrollo unificado para construir tales aplicaciones.

I.12.1 .NET

A fines del año 2001, Microsoft elaboró una estrategia de tipo participativa a los efectos popularizar la el desarrollo y utilización de servicios compañero a compañero a través de servidores web basados en XML. Donde todo equipo participante de la red .NET colabora según sus recursos disponibles en torno al potencial conjunto de la infraestructura creada.

.NET se concibió como una plataforma de servicios web basados en XML, que les permiten a las aplicaciones de usuario final comunicarse y compartir datos sobre Internet,

I.12.2 JXTA

En el año 2001 Sun Microsystems lanzó el proyecto JXTA [JXTA-A], e indicó que es una plataforma, que aumenta la habilidad de los usuarios para compartir la información de sus computadoras personales y otros dispositivos. El objetivo de JXTA es permitir a los usuarios, por medio de nuevas aplicaciones, disponer de información clave a través de numerosos puntos de acceso, como dispositivos portables, notebooks y teléfonos móviles.

Según Bill Joy, director científico de Sun, *"Esta es la tecnología que dará acceso a una Red más*

amplia y profunda", *"La meta es suministrar un método común para cualquier nodo de la Red para que acceda a cualquier dato, contenido u otro nodo"*.

Según Sun, las ventajas que presenta JXTA son las siguientes:

- **Independencia de plataforma.** JXTA es soportada por distintos sistemas operativos, tales como Linux, Solaris, Microsoft Windows y por distintos lenguajes de programación (C, Java).

- **Ubicuidad**. Es posible implementar JXTA sobre una amplia gama de dispositivos (computadoras de mano), notebooks, computadoras personales, dispositivos de red, sistemas de almacenamiento, aparatos varios, etc.

- **Interoperatividad.** Dado que JXTA ha sido diseñada para localizar y comunicar fácilmente cada nodo participante en la red

JXTA está definido por un juego de protocolos, donde cada uno determina una serie de mensajes, que se intercambian entre nodos. Los protocolos que definen a JXTA son:

Protocolo de descubrimiento de compañeros (Peer Discovery Protocol) Se utiliza a los efectos de descubrir y publicar recursos existentes en una red. Los recursos son representados como avisos (advertisements). Un recurso puede ser un compañero, un grupo de compañeros o un servicio. Cada recurso debe ser representado como un advertisement.

Protocolo de información sobre compañeros (Peer Information Protocol): Este protocolo provee un juego de mensajes destinados a obtener información acerca del estado de un nodo compañero. Utiliza un mensaje tipo ping y otro denominado PeerInfo.

Protocolo de membresía (Peer Membership Protocol) Permite que un nodo compañero pueda incorporarse a un grupo, intercambiando credenciales. Este protocolo es la base de los grupos, dado que permite que se autogestionen.

Protocolo de ruteo (Endpoint Routing Protocol) Este protocolo define un juego de mensajes consulta/respuesta que tiene por misión ayudar a un nodo compañero a rutear sus mensajes.

Protocolo de resolución de compañero (Peer Resolver Protocol) Permite habilitar a un nodo compañero a enviar una consulta genérica a otro compañero que ofrece un servicio de resolver. Cada nodo que ofrece servicios, se registra sobre un nodo especial de su grupo que actúa como *resolver*.

I.13 Seguridad

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

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

El grupo de trabajo Peer-to-Peer [TFGT] ha desarrollado una librería de seguridad destinada a desarrolladores de software P2P, la cual incluye soporte para certificados digitales, módulos de autentificación de compañeros, almacenamiento seguro, encriptación sobre la base del modelo de claves pública/privada, firmas digitales y encriptación simétrica.

La seguridad es hasta hoy el mayor problema en ambientes P2P, nótese su impacto en un ambiente de cómputo distribuido donde personas mal intencionadas pueden entregar resultados fraudulentos y estropear un proyecto. SETI@home estimó que destino el 50% de su presupuesto a tareas relacionadas con la seguridad. La utilización de firmas digitales y encriptación resolverá la mayoría de los problemas de seguridad.

I.14 Metas futuras de los sistemas P2P

- Anonimato, servicios anticensura, y hospedaje descentralizado de bloques de información (una misma información, partirla y distribuirla en n nodos). Serán principios de diseño en las nuevas aplicaciones. El derecho a la privacidad se garantizará, implementando como una cualidad estándar de cualquier nueva aplicación.

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

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

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

I.14 Consideraciones

Sistemas P2P permiten que las computadoras individualmente dialoguen entre sí, hagan ejecuciones en paralelo y administren contenidos, en lugar de que lo hagan servidores centrales.

La principal ventaja competitiva del modelo P2P sobre Internet está dada por que utiliza los recursos de las computadoras de usuarios, aquellos equipos que tradicionalmente permanecían ocultos[8] y era imposible nombrarlos y direccionarlos bajo los esquemas clásicos.

La descentralización y los modelos P2P tienen, probablemente, más aplicaciones que las que al presente es posible imaginar. En el futuro, en cualquier área donde la escalabilidad y la capacidad de cómputo masivo sea importante, se llevará a cabo utilizando sistemas descentralizados.

En las redes P2P, cada usuario construye su propio espacio virtual, elige quienes serán sus interlocutores y lo que desea compartir con ellos. Ese esquema brinda libertad, sobre la base de las posibilidades de elección, dado que los usuarios eligen a que comunidades desean pertenecer y no por que sitios multimedios navegar.

Los problemas claves que impiden la masificación de los sistemas P2P son: confianza mutua, seguridad de las aplicaciones e integridad de los datos.

La seguridad y la privacidad se perciben como importantes barreras para que tenga lugar una adopción rápida.

[8] Clay Shirky los definió como "dark matter of Internet"

Capítulo II

II. Napster y Gnutella,
los orígenes de la computación P2P sobre Internet

II.1 Introducción

Los sistemas clásicos de distribución de archivos se basaban en protocolos de propósito general ampliamente conocidos y utilizados. Por ejemplo, la distribución de un archivo de música MP3 puede realizarse por descarga de un servidor mediante el uso de los protocolos FTP o HTTP. Tales sistemas se basan en una arquitectura cliente-servidor, en la que los usuarios se conectan a servidores donde se encuentran los recursos a descargar.

El uso de la arquitectura compañero a compañero para distribuir archivos de forma masiva es una alternativa válida al modelo anterior. En esta arquitectura, son los propios usuarios los que intercambian los archivos entre ellos; aunque también pueden existir compañeros especiales, pero su objetivo no es el de almacenar contenidos, sino el de poner en contacto a los clientes entre sí.

Es importante el grado de desarrollo de sistemas de información basados en la arquitectura compañero a compañero, hoy en día su uso no solo se limita a la distribución de archivos de distinta índole. Hay propuestas y aplicaciones en marcha que operan sobre redes P2P y ofrecen servicios tales como web cache [Wang], retransmisión masiva de flujos de multimedia [Tolosa], búsqueda distribuida cooperativa [Tolosa-A], procesamiento masivo [Intel] [Aberer],etc.

II.2 Sistemas orientados a la búsqueda e intercambio de archivos

Nuevas formas de aplicaciones de red, tales como Gnutella [Bordignon] y FreeNet [Clarke], son los primeros sistemas distribuidos, de uso masivo, destinados al almacenamiento y búsqueda de información. Estos sistemas, inspirados en el modelo Napster [Napster],

permiten el acceso a la información a usuarios de todo el mundo, brindándole, en algunos casos, una importante calidad de anonimato.

Una característica que define a estos sistemas de intercambio de información es que básicamente no disponen de una infraestructura centralizada, sino que la misma se construye sobre la base de la participación voluntaria de pares que aportan sus recursos. Típicamente, el número de miembros en un sistema P2P es dinámico, y la red por medio de servicios especiales centralizados ó no, arbitra los medios para que los requerimientos de los nodos puedan ser satisfechos de forma consistente y eficiente.

Napster y Gnutella tienen objetivos similares: facilitar la ubicación y el intercambio de archivos (típicamente audio, imágenes y video) en una comunidad numerosa de usuarios independientes conectados a través del Internet. En estos sistemas, se almacenan archivos en las computadoras de los usuarios finales; luego de la localización, el intercambio se realiza de forma directa entre pares. Todos los pares en este sistema son simétricos, dado que todos tienen la habilidad de funcionar como cliente y servidor. Esta simetría distingue a los sistemas P2P de otras arquitecturas de sistemas distribuidos convencionales. Aunque el proceso de intercambio de archivos es similar en ambos sistemas, Napster y Gnutella difieren substancialmente en cómo los compañeros localizan archivos. En Gnutella mediante un esquema descentralizado cooperativo y en Napster vía el auxilio de una base de datos centralizada.

El éxito de las aplicaciones destinadas al intercambio de archivos se basó en:

- Los usuarios fueron personas sin demasiados conocimientos de informática, a los cuales se les hacía dificultosa la tarea de mantener sus propios servidores; donde por contraste la instalación, uso y mantenimiento de aplicaciones P2P es una tarea sencilla, pudiéndose realizar de forma autosuficiente.

- Aquellos usuarios que deseaban compartir sus recursos musicales veían que los métodos clásicos de búsqueda (motores de consulta) no satisfacían sus expectativas,

dado que estaban diseñados para operar con otros contenidos, tales como páginas en formato HTML e imágenes.

Desde el punto de vista de los usuarios, se presentan ventajas proporcionadas por los sistemas de intercambio de archivos:

- **Economía,** dado que el costo de almacenamiento de archivos como el ancho de banda necesario para realizar las descargas es repartido entre miles de usuarios distribuidos a lo largo del planeta.
- **Percepción de inmunidad,** debido a que los archivos disponibles se replican y se distribuyen sobre múltiples usuarios. Se les hace difícil, y hasta imposible, a los dueños de los derechos de las obras musicales o de video poder emprender acciones sobre tal cantidad de usuarios.

En la siguiente parte de este capítulo se exponen con un mayor grado de detalle las infraestructuras utilizadas por Napster y Gnutella a los efectos de mostrar las bases de la nueva generación de protocolos P2P sobre Internet.

II.3 Napster

II.3.1 Introducción

En el año 1999 el programador Sean Fanning, creador del protocolo, tuvo la idea de elaborar un sistema donde los usuarios finales tuvieran el control de la publicación de sus contenidos y que además pudieran intercambiar archivos directamente, sin necesidad de equipos intermedios. Napster descentralizó el servicio de intercambio de archivos[9] haciendo que cada computador de usuario final sea un compañero en una red de iguales; donde los usuarios que requerían recursos de terceros, a la vez ofrecían los propios.

[9] Nótese que hasta la fecha de aparición la distribución se realizaba vía FTP o por medio del protocolo HTTP

El éxito de Napster radicó en que desplazó el control del sistema del centro hacia los bordes. Napster no es un sistema P2P puro, se lo considera híbrido, dado que contempla la existencia de un servidor central destinado a mapear que contenidos hay en la red y quién los tiene. Para incorporarse al sistema, los usuarios instalan un software, de distribución gratuita, que permite consultar por recursos, realizar conferencias y compartir sus archivos con otros usuarios. Napster, a fines del año 2000 poseía alrededor de 15 millones de usuarios con un crecimiento mensual de un 20%.

Como ventajas de este sistema se puede citar que por medio de la base de datos central se localizan archivos de forma rápida y efectiva; y los pedidos alcanzan a todos los usuarios registrados que estén actualmente conectados. Las desventajas indican que el sistema solo tiene un punto de entrada, la red podría colapsar si uno de los servidores centrales estuviera incapacitado de dar atención a los pedidos.

Según Shirky [Shirky] el modelo impuesto Napster es una violación al sistema web[10] dado que los contenidos están en los bordes. Por otro lado, expresó, que una de las razones que fundamentaron tal innovación, es el hecho de que el sistema web está pensado para localizar y descargar contenidos fácilmente, pero la tarea de publicación sigue siendo extremadamente complicada y no natural para la mayoría de los usuarios. Shirky indica que el éxito de Napster reside sobre la base de tres razones:

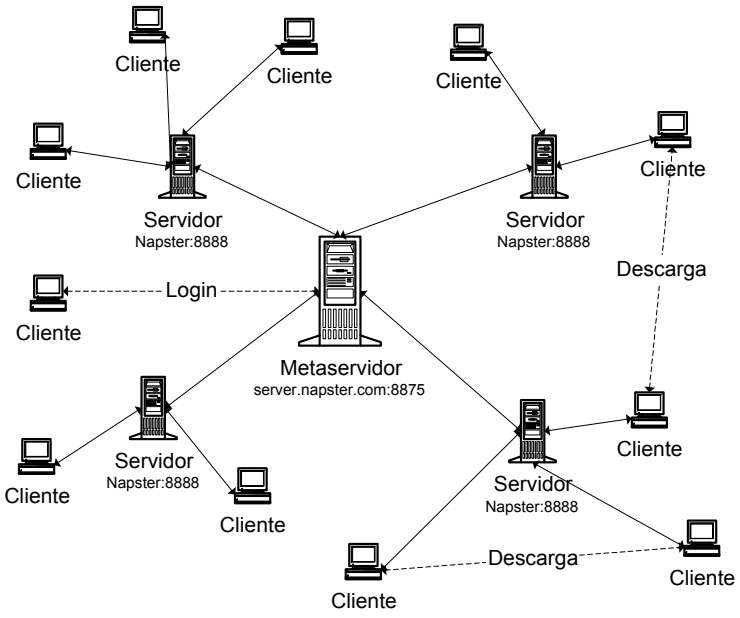
- El servidor central nada sabe acerca de las descargas que realizan los usuarios, solamente su función es la de agente de localización de archivos.
- Los equipos de usuario final que corren el programa cliente Napster no necesitan dirección de red fija y que pueden operar de forma natural con conectividad variable.
- Napster ignora al modelo cliente/servidor, dado que no hace distinción entre usuarios finales. Ellos pueden descargar y transmitir archivos indistintamente.

[10] Donde los contenidos residen en el centro

II.3.2 Arquitectura

Napster es un protocolo de capa de aplicación de red, que define la forma en que computadoras de usuario final se comunican con un servidor central. El propósito principal de dicho protocolo es el intercambio de archivos y mensajes entre usuarios. Napster utiliza servicios provistos por el juego de protocolos TCP/IP. El protocolo opera bajo el modelo cliente/servidor, dado que existe un servidor central al cual los usuarios envían sus requerimientos.

El servidor central pone en estado de apertura pasiva a los puertos TCP 8888 y 7777. Los meta servidores[11] utilizados a fin de balancear los pedidos de conexiones entre un grupo de servidores Napster, operan en el puerto TCP 8875. Nótese que el uso de servidores clonados permite aumentar la escalabilidad, uno de las críticas más fuertes al sistema.



A la unidad de intercambio de información (PDU), en el protocolo Napster se la denomina paquete, y posee la siguiente estructura:

Largo (2 byte)	Tipo (2 bytes)	Datos (largo variable)

- El campo largo [length] es un entero de dos bytes sin signo, el cual especifica el número total de bytes en el campo datos [data] del mensaje. El orden de los bits está en formato little-endian. Up to 64 Kbytes of data can be sent in a single message.

- El campo tipo [type] es un entero de dos bytes sin signo, y especifica el tipo de mensaje ó un código de comando. Este campo define como deben interpretarse los datos almacenados en el tercer campo [data]. Ejemplos:

  - Cuando el tipo se instancia con el valor 211 se indica un requerimiento exploración. En el campo datos debe adjuntarse el nombre del usuario que posee los archivos que se desean ver.

  - Cuando el valor de tipo es 2 se indica una solicitud de login por parte del usuario. El campo datos contendrá: nombre de usuario, clave, puerto de escucha, versión de cliente y tipo de enlace a Internet.

- El campo datos [data] es una secuencia de bytes cuya longitud y significado están definidos en los campos previamente enunciados.

.

Ejemplos de comandos soportados por el protocolo Napster se dan a continuación (la definición completa del protocolo se halla en el sitio de desarrollo del software alternativo OpenNap, sito en http://opennap.sourceforge.net/napster.txt). Las palabras cliente y servidor denotan quien envió el mensaje.

Campo tipo	**Descripción**
00	Mensaje de error [servidor]
02	Pedido de login [cliente]

[11] También conocidos como redirectores.

03	Aceptación de login [servidor]
04	Control de versión [cliente]
05	Actualizar versión de cliente [servidor]
100	Comparto archivo ... [cliente]
102	Dejo de compartir archivo ...[cliente]
200	Pedido de búsqueda de archivos [cliente]
201	Respuesta de búsqueda de archivos [servidor]
202	Fin de búsqueda de archivos [servidor]
203	Requerimiento de descarga [cliente]
204	ACK a requerimiento de descarga [servidor]
205	Mensaje privado a cliente [cliente]
211	Ver archivos que comparte un usuario [cliente]
212	Respuesta a ver archivos que comparte un usuario [servidor]
213	Fin de respuesta a ver archivos que comparte un usuario [servidor]
218	Aviso de comienzo de descarga de archivo [cliente]
219	Aviso de fin de de descarga de archivo [cliente]
220	Aviso de inicio de transferencia de archivo [cliente]
221	Aviso de fin de transferencia de archivo [cliente]

II.3.3 El servidor Napster

El servidor Napster opera tomando el rol de agente de información. Mantiene información acerca de que clientes están en línea y que recursos comparten (metadatos). Por otro lado es un lugar donde todos los mensajes son remitidos, dado que administra los grupos de conferencia y las hotlists de usuario. De hecho funciona como un motor de consultas, dado que los usuarios envían sus requerimientos, se resuelven con una base de datos local y se envían las repuestas. En resumen, el servidor Napster es un agente que mantiene solamente información acerca del estado de la red de aplicación y en ningún caso réplicas de recursos de usuario, dado que los mismos son siempre descargados por intercambio directo entre usuarios finales.

Las acciones a desarrollar a los efectos de descargar un archivo, utilizando Napster, son:

1. El cliente se conecta al servidor Napster.

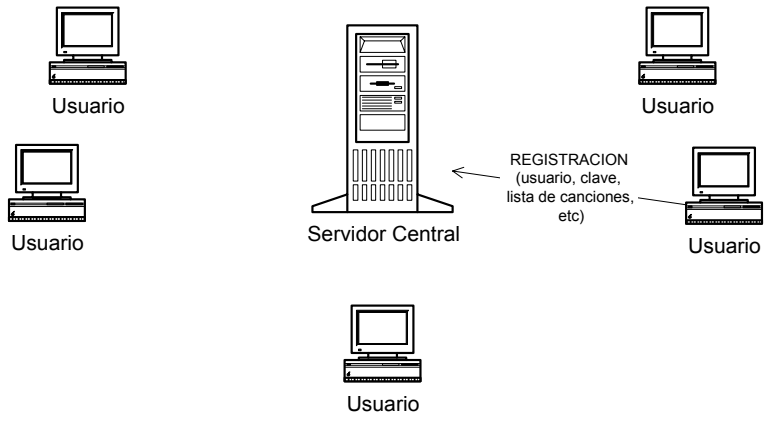
2. El cliente realiza la búsqueda del recurso deseado.
3. El servidor le envía una lista que indica cuales clientes poseen el recurso solicitado.
4. El cliente se conecta directamente a otro compañero (cliente) que haya seleccionado de la lista y descarga el recurso solicitado. Este último paso se realiza sin la intervención del servidor central.

II.3.4 Modo de operación del servicio Napster[12]

1. Inicio de conexión

° Un usuario final inicia una conexión con el servidor Napster. Envía su nombre de usuario, clave, lista de archivos a compartir y otros datos relativos a su tipo de conexión.

° El servidor central, en su base de datos, registra la disponibilidad del usuario y los recursos compartidos.



Cuando un usuario quiere conectarse al sistema Napster entabla una conexión de transporte TCP y luego envía un mensaje tipo 2 (login) de solicitud de conexión, cuya sintaxis es la siguiente:

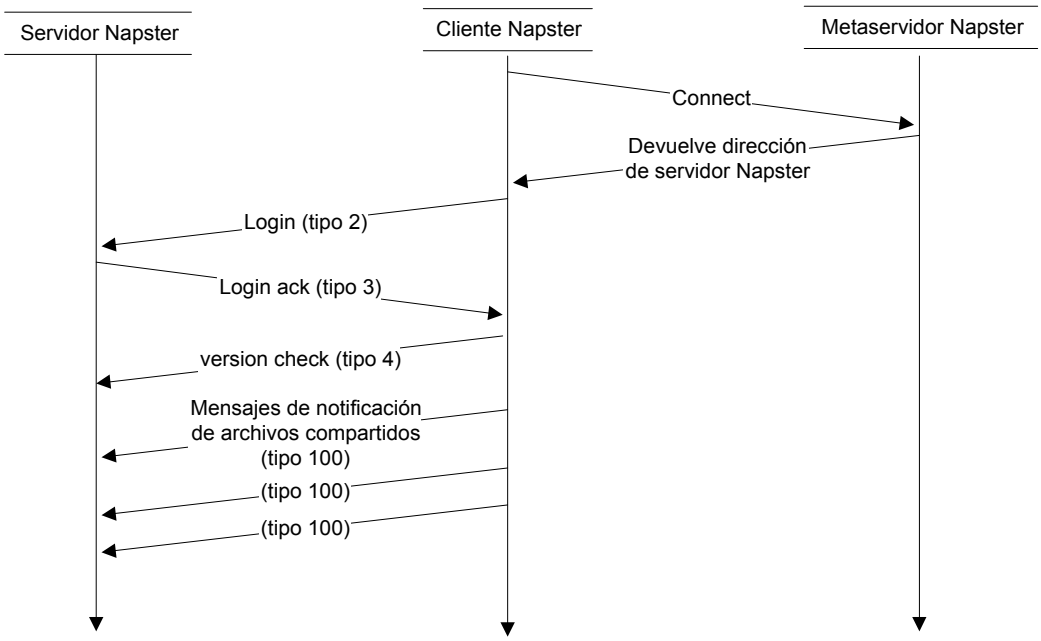
[12] Especificaciónes del protocolo tomadas del proyecto http://opennap.sourceforge.net

```
2 - login [enviado por el cliente]

        Formato: <nick> <password> <port> "<client-info>" <link-type> [ <num> ]

        <port> Puerto TCP que el cliente pone en estado de apertura pasiva a los
efectos de esperar por pedidos de transferencia. Si su valor es 0 el cliente se
halla detrás de un firewall y por lo tanto él tendrá que comenzar todas las
descargas.

        <client-info> cadena que contiene información sobre la versión del
        cliente

        <link-type> entero que indica ancho de banda del cliente. 0   no
conocido, 1 14.4 kbps, 2  28.8 kpbs, 3  33.6 kbps, 4  56.7 kbps, 5  64K ISDN, 6
128K ISDN, 7  Cable, 8  DSL, 9  T1, 10 T3 o mayor

        <num> número de versión del cliente [opcional]

        Ejemplo:

                foo badpass 6699 "nap v0.8" 3
```

El servidor central, al aceptar el pedido de conexión, envía al cliente un mensaje 3 (login ack).

```
3 - login ack [enviado por el servidor]

        Formato: <email> [<number>]

        El servidor envía al cliente la dirección de correo electrónico que
tiene registrada para el nick enviado.

        <number> posiblemente sea el número de usuario activo.
```

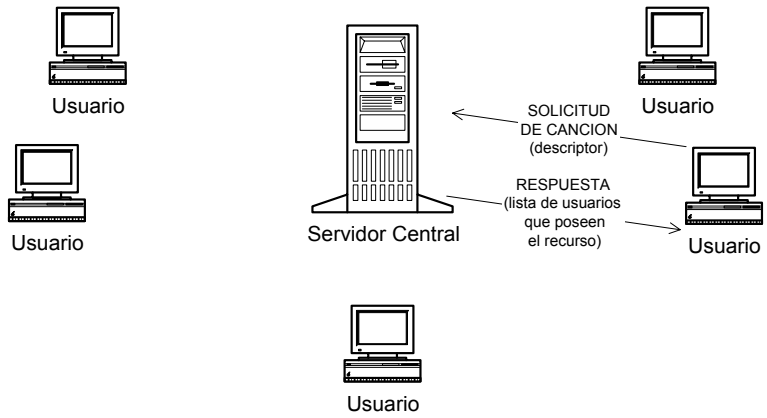
El siguiente diagrama de tiempos ilustra el inicio de una conexión por parte de un cliente contra un meta servidor Napster.



2. Solicitud de una canción

- Un usuario envía al servidor un requerimiento de búsqueda de canción.

- El servidor central consulta su base de datos y le devuelve al usuario una lista de aquellos compañeros que poseen tal recurso (identificados por su dirección de red) y están disponibles para iniciar una operación de descarga.



Los usuarios informan de los recursos que comparten utilizando el mensaje tipo 100.

```
100 – Notificación de archivo compartido [enviado por el cliente]

       Formato: "<filename>" <md5> <size> <bitrate> <frequency> <time>

       <md5> firma de integridad del archivo, bajo norma MD5.
       <size> Tamaño, en bytes
       <bitrate> Tasa de transferencia, en kbps
       <frequency> Frecuencia, en Hz
       <time> Duración, en segundos

       Ejemplo:
              "generic band - generic song.mp3" b92870e0d41bc8e698cf2f0a1ddfeac7-443008
              443332 128 44100 60
```

Una consulta por un determinado tema musical se la realiza con el mensaje tipo 200.

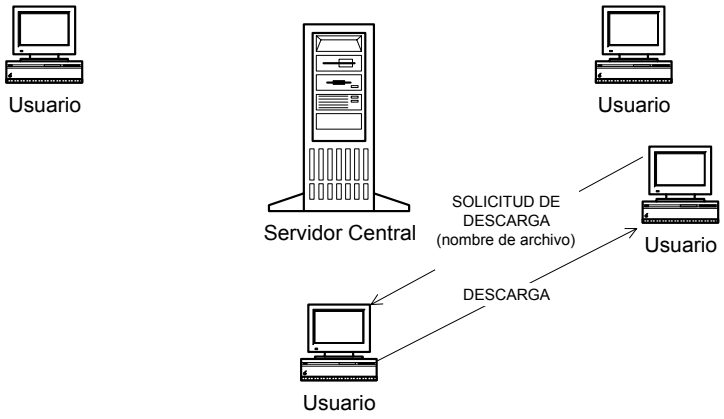
```
200 – Solicitud de búsqueda de archivos [enviado por el cliente]

      Formato:      [FILENAME CONTAINS "nombre de artista"]
                    MAX_RESULTS <max>
                    [FILENAME CONTAINS "canción"]
                    [LINESPEED <operador> <link-type>]
                    [BITRATE <operador> "<br>"]
                    [FREQ    <operador> "<freq>"]
                    [LOCAL_ONLY]

      El nombre de artista y el nombre del tema son cadenas diferentes.
      <max> número máximo de resultados a devolver (100 es el máximo).
      <operador> debe ser: "AT LEAST" "AT BEST" "EQUAL TO"

             <link-type> número entero, indica tipo de conexión. Ver mensaje
      tipo 2 para códigos a utilizar.
      <br> en kbps
      <freq> en Hz
      LOCAL_ONLY indica que el ámbito de búsqueda debe ser solamente el
      servidor al cual está conectado el usuario.

       Ejemplos:
             1. FILENAME CONTAINS "Jorge Scucimarri" MAX_RESULTS 60 FILENAME
                CONTAINS "Tango de pan" BITRATE "AT LEAST" "192"

             2. MAX_RESULTS 100 FILENAME CONTAINS "mandolina" LINESPEED "EQUAL
                TO" 5
```

Las repuestas a una consulta son generadas por el servidor central utilizando mensajes (uno por recurso hallado) de tipo 201.

```
201 – Respuesta a consulta [enviado por el servidor]

Formato:            "<filename>" <md5> <size> <bitrate> <frequency>
                              <length> <nick> <ip>  <link-type>
                              [weight]

       <md5> firma de integridad del archivo, bajo norma MD5.
       <size> Tamaño, en bytes
       <bitrate> Tasa de transferencia, en kbps
       <frequency> Frecuencia, en Hz
       <length> Duración, en segundos
       <nick> Apodo de la persona que posee el recurso

              <ip> Entero largo sin signo, que representa la dirección de red
      del usuario que posee el recurso.
      <link-type> número entero, indica tipo de conexión. Ver mensaje tipo 2
      para códigos a utilizar.
       [weight]       Factor de ponderación que permite que el cliente pueda
      ordenar los resultados. Valores positivos indican mejor coincidencia y
      valores negativos indican peor coincidencia. Por ausencia, el cliente
      debe considerarlo como 0.

       Ejemplo:

             "random band - random song.mp3"
             7d733c1e7419674744768db71bff8bcd-2557521  2558199  128  44100  159
             lefty 3437166285 4
```

3. Descarga

- El usuario selecciona un compañero de la lista e inicia una conexión con él, a los efectos de solicitarle la descarga del recurso pertinente.



Independientemente de como se localizó un recurso (vía mensaje tipo 200 ó tipo 211) a descargar, el cliente envía un mensaje al servidor central tipo 203 (get) el cual satisfactoriamente le responde con un mensaje tipo 204 (get ack) que incluye un más información acerca del recurso.

Ahora pueden contemplarse varias situaciones. Si con el mensaje tipo 204 el campo puerto posee un valor 0, significa que la transferencia la debe comenzar el dueño del recurso, para lo cual el cliente debe enviar un mensaje tipo 500 al servidor central, indicándole que haga de intermediario para avisarle al compañero que le envíe el archivo.

Si el recurso no llegara a estar detrás de un firewall (campo puerto con valor mayor a 0), el cliente iniciará una conexión TCP con el dueño del recurso sobre el puerto indicado en el mensaje tipo 204, obtenido del servidor. El compañero que posee el recurso debería aceptar la conexión y enviar el carácter ASCII `1' (ASCII 49). Una vez que obtiene tal carácter, el cliente, le envía un requerimiento de descarga con la siguiente sintaxis:

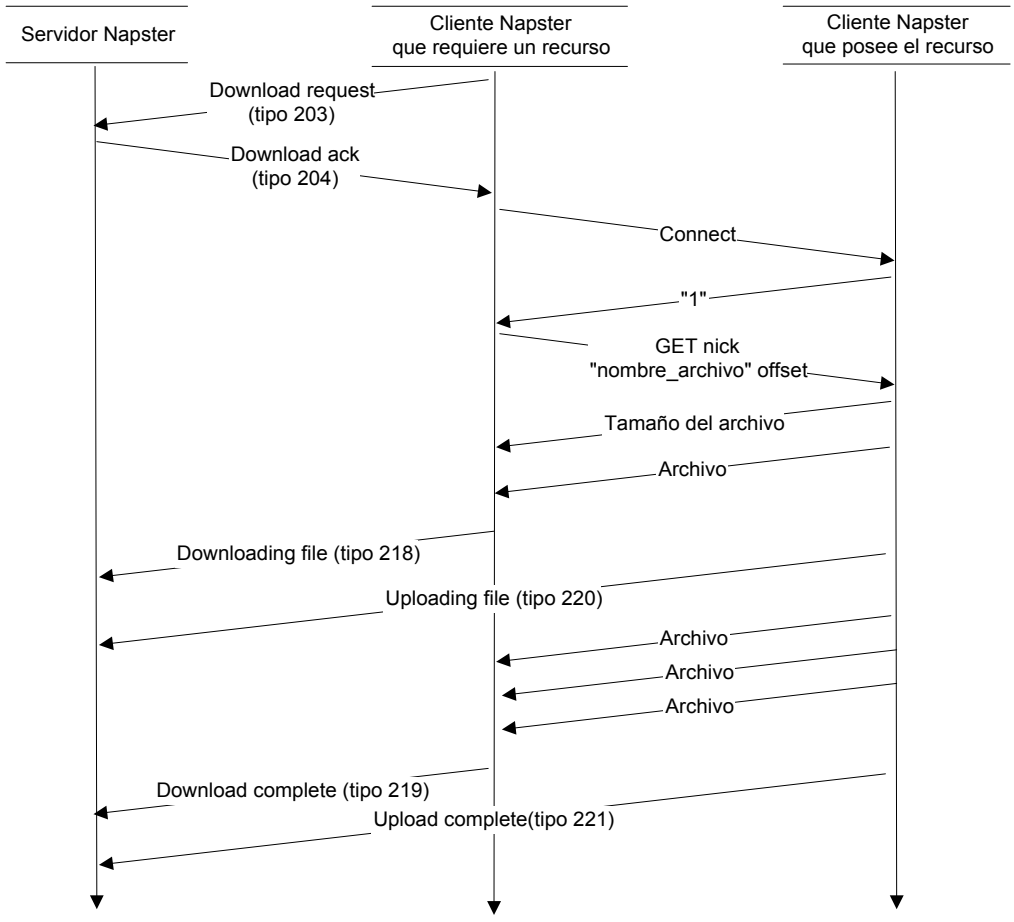
GET <mynick> "<filename>" <offset>

Donde <mynick> es el nombre del usuario que requiere el recurso, <filename> es el nombre del archivo a recuperar y <offset> es el byte a partir de donde se desea comenzar a recuperar el archivo (útil para retomar descargas inconclusas), si se instancia con el valor 0 se indica que el archivo se debe transmitir en su totalidad.

El compañero remoto retornará el tamaño del archivo, o un mensaje de error. Luego del dato que indica la longitud comienza la transmisión del archivo.

El cliente, una vez iniciada la descarga, puede informar al servidor central que ha comenzado la misma. Esto lo hace con un mensaje tipo 218. La finalización de descarga se realiza con un mensaje 219. Un cliente puede descargar múltiples archivos de forma concurrente, por cada uno debe enviar un par de mensajes tipo 218/219. De la misma forma el compañero que está enviando el archivo (uploader) debería enviar, al servidor central, un mensaje tipo 220 al inicio de la transmisión y un mensaje tipo 221 cuando finaliza.

El siguiente diagrama de tiempos ilustra la descarga de un archivo, tal como se describió en los párrafos anteriores.



Cuando el equipo que posee el recurso se halla detrás de un firewall, las descargas de archivos se realizan de la siguiente manera. El recurso necesita ser empujado (pushed) desde el

compañero que lo contiene, para lo cual el cliente que lo requiere debe enviar al servidor central un mensaje de requerimiento de descarga tipo 500. El servidor central enviará al equipo que posee el archivo un mensaje tipo 501 (similar al mensaje tipo 204 get ack), indicándole que puerto habilitó su compañero para que él comience la transmisión del recurso.

Al recibir el mensaje 501, el compañero situado detrás del firewall iniciará una conexión TCP con el usuario demandante del archivo y le enviará el carácter ASCII `1' y a continuación el comando

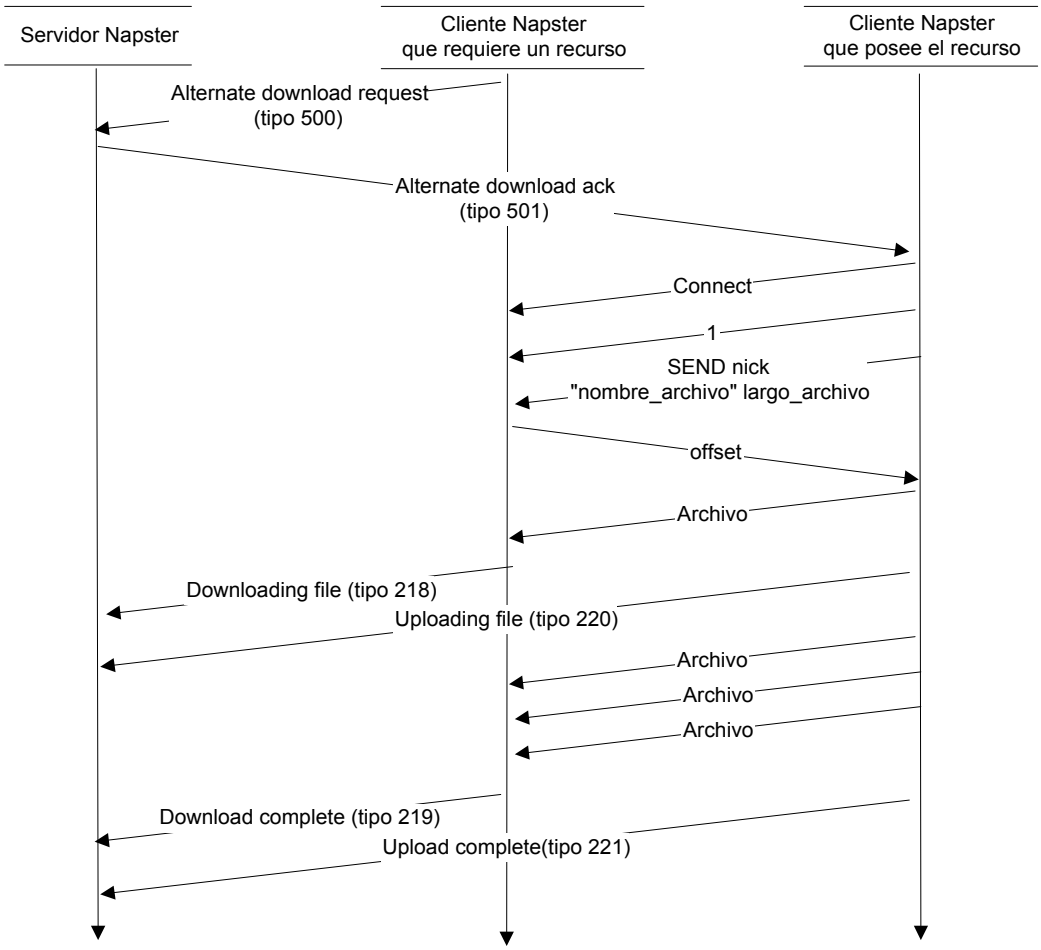
SEND <mynick> "<filename>" <size>

Donde <mynick> es el apodo del nodo poseedor del recurso, <filename> es el nombre del archivo a enviar y <size> es su tamaño en bytes.

Como respuesta, el compañero que desea recibir el recurso, enviará un mensaje indicando a partir de que byte desea comenzar a recibir el archivo (0 indica a partir del principio) ó un mensaje de error ("INVALID REQUEST").

Cada compañero debe indicar al servidor central el comienzo y fin de transmisión. Para ello el receptor enviará pares de mensaje tipo 218/219 y el transmisor pares de mensaje tipo 220/221.

El siguiente diagrama de tiempos ilustra la descarga de un archivo de un compañero situado detrás de un firewall.



Además de las funcionalidades descriptas, el protocolo Napster, contempla la conferencia pública y privada entre compañeros, como así también la posibilidad de que un usuario determinado realice actividades de "browsing" sobre otro par indicado por él.

II.3.5 Críticas

Existen dos puntos fuertes en el sistema Napster.

- La interfase es fácil de usar y los usuarios nuevos se familiarizan rápidamente con los controles.

- Las funciones de búsqueda, descarga y conferencia están integradas correctamente en el protocolo y la interfaz de usuario.

Por otro lado, se plantean dos puntos débiles.

- Protocolo propietario. Debería ser abierto.

- Comunicación de control no encriptada. Es un problema importante de seguridad, dado que los datos de usuario no viajan encriptados.

II.3.6 Aspectos sociales

Según estudios realizados por la empresa Pew Internet & American Life [Pew], el 38% de los usuarios de Internet en Estados Unidos ha descargado alguna vez canciones de la red, pero sólo un 2% ha pagado por tal privilegio. El 14% de tales usuarios ubicaron sus canciones en redes públicas, tipo Napster o Gnutella.

En el mencionado estudio se analiza el funcionamiento de Napster durante el mes de abril del año 2000 y se concluye que hay un promedio de 5.000 usuarios conectados a cualquier hora del día que comparten un promedio de 100 títulos[13]. Según datos de la compañía que gerencia el servicio Napster, en el primer semestre del año 2001 se habían alcanzado los 60 millones de usuarios.

Desde su aparición , a finales del año 1999, Napster ha recibido múltiples denuncias por parte de organizaciones de defensa de los derechos de autor y de autores individuales. El eje de tales litigios es la protección de los derechos de autor [Rojas]. El orden cronológico de los acontecimientos, indica que la historia comenzó con una denuncia de la Asociación de la Industria Discográfica de Estados Unidos (RIAA) y varias universidades estadounidenses clausurando los accesos a Napster, porque la frecuente descarga de archivos de música MP3 congestionaba sus redes y accesos a Internet. Luego se creó un movimiento estudiantil

[13] Entre 500.000 y 600.000 canciones estaban permanentemente disponibles para su descarga.

denominado 'Save our Napster', se acumularon querellas de músicos. Un fallo judicial se cumplió el 1 de julio del año 2001, clausurando el servicio, pero brindándole la posibilidad de que vuelva a funcionar en forma paga por parte de los usuarios.

II.3.7 Consideraciones

Napster es un protocolo simple, destinado principalmente a facilitar la publicación, la ubicación y el intercambio de archivos entre usuarios finales con conectividad variable. El sistema -desde el punto de vista de la utilidad- ha sido un éxito, hasta tal punto que ha sido clausurado temporalmente por parte de organizaciones de defensa de los derechos de autor.

Sus mayores problemas son de seguridad y de infraestructura, donde su escalabilidad está limitada por su topología de estrella. Aunque están implementados ciertos mecanismos de balanceo de carga, denominados meta servidores ó redirectores de conexiones.

II.4 Gnutella:
Sistema Distribuido para Almacenamiento y Búsqueda de Información.

II.4.1 Introducción

Gnutella es un protocolo y una aplicación que lo implementa. Determina una metared a nivel aplicación [Kunwadee], donde cada nodo (en adelante gnodo) es igual. La red está definida dado que cada nodo establece varias conexiones permanentes con otros pares, rutea mensajes de terceros e implementa estrategias que le permitan la supervivencia. Conceptualmente, Gnutella, es un sistema distribuido para almacenamiento y búsqueda de información.

Para comprender las características y el estado de desarrollo del protocolo, resulta necesario remitirse a sus orígenes. Gnutella surgió como un proyecto independiente de los

programadores Justin Frankel y Tom Pepper, fundadores de la empresa Nullsoft, destacada por ser dueña del producto Winamp, reproductor de archivos de sonido mp3. El día 14 de marzo del año 2000, en el sitio Slashdot (http://www.slashdot.com) se informó sobre la aparición de una versión, en fase beta, del programa Gnutella; descrito como "una herramienta de software para compartir archivos, que puede ser aún más potente que Napster". Inmediatamente la página fue puesta fuera de servicio, pero ya habían sido descargadas numerosas copias. Nullsoft, ante el hecho en cuestión, se reservó el derecho de publicación[14] aduciendo que la alta demanda colapsó el sitio.

Actualmente, existe una continuación al proyecto llevada a cabo por una comunidad de programadores. Sobre la base de técnicas de ingeniería inversa y análisis de protocolos han logrado decodificar y realizar varias implementaciones del protocolo Gnutella. Los equipos de desarrollo de las aplicaciones LimeWire (www.limewire.com) y BearShare (www.bearshare.com) son los que hoy están liderando el desarrollo del protocolo, a partir de proponer e implementar mejoras [Rohrs] [Bidson].

Es importante destacar dos características fundamentales que hacen a Gnutella merecedor de estudios por parte de grupos de investigación y empresas: su carácter descentralizado y su espíritu cooperativo. En la exposición de Mark Gordon[15] pronunciada en la conferencia P2P O´Reilly 2001, se vertieron una serie de conceptos favorables a la filosofía del protocolo Gnutella. Se expresó que la red Gnutella tiene cualidades que la hacen la mejor plataforma disponible para construir una capa información sobre Internet. Debido a que Internet proporciona un marco adecuado para el envío de mensajes de un computador a otro, pero carece de habilidad para propagar mensajes a direcciones no especificadas.

II.4.2 Características del protocolo Gnutella

Bajo Gnutella se trabaja en un modelo de ambiente distribuido. Un gnodo X ofrece abiertamente archivos que desee compartir con otros usuarios, mientras que a la vez el usuario dueño del gnodo Y puede estar recuperando, como cliente, archivos de un gnodo X.

[14] Acción que no volvio a ser realizada
[15] Director de la empresa de software Lime Wire

Normalmente las tareas de servidor y cliente están siempre activas en un nodo Gnutella, pero nada impide que alguna pueda no ser activada.

La red gnutella se compone de numerosos gnodos distribuidos a lo largo del mundo. Su topología no indica jerarquía alguna, dado que cada gnodo es igual a los demás en funcionalidad. Debido a que uno de los atributos que caracterizan a un gnodo es su ancho de banda ofrecido para descarga de archivos, el modelo sufre ciertas variaciones. Aquellos gnodos de mayor ancho de banda son preferidos a otros, formando hubs ó anillos centrales de conexión a la red.

Cada gnodo de Gnutella sólo sabe acerca de los gnodos con los que se conecta directamente Todos los otros gnodos son invisibles, a menos que ellos se anuncien contestando a un mensaje tipo ping o contestando a una consulta. Esto proporciona un interesante grado de anonimato.

De forma resumida, la red Gnutella opera bajo el modelo conocido como "propagación viral" ó broadcast. Un cliente envia un mensaje a un gnodo, y éste lo reenvía a todos los gnodos a los cuales esté conectado. De esta forma, con solo conocer la dirección de red de un gnodo ya se puede ingresar a la misma.

El ingreso a la red se realiza indicando la dirección IP y puerto TCP de cualquier gnodo que ya esté conectado. El gnodo que inicia la conexión anuncia su presencia mediante el envío de un mensaje. El gnodo al cual se conectó reenvía este mensaje a todos los gnodos a los cuales esté conectado directamente y así sucesivamente. Cada uno de estos gnodos, responde a este mensaje indicando cuántos archivos comparte y tamaño total de los mismos.

Un ejemplo del alto grado de estabilidad e independencia que presenta el modelo puede verse en el siguiente ejemplo: Suponga que un gnodo-1 se conecta un gnodo-2, todo lo que ofrezca gnodo-1 puede ser tomado por gnodo-2 y viceversa. Ahora gnodo-3 se conecta a gnodo-2 y éste le informa a qué gnodos está directamente conectado (los mensajes de gnodo-3 pueden llegar a gnodo-1 dado que los reenviará gnodo-2). A continuación gnodo-2 se pone en estado

fuera de servicio. La red sigue funcionando normalmente dado que gnodo-3 posee direcciones alternativas de gnodos pasadas anteriormente por gnodo-2, que le permitirán conectarse a gnodos alternativos para seguir vinculado a la red.

En las implementaciones del protocolo Gnutella un parámetro a definir es la cantidad de conexiones simultáneas a la red que siempre un gnodo debe tratar de mantener. Cuanto más conexiones tenga, se logrará un mayor espacio de consulta y se asegurará una alta disponibilidad de la red.

II.4.3 Estructura de mensajes

Un mensaje gnutella se transporta sobre protocolo TCP. Su cabecera siempre consta de 23 bytes y consiste de los siguientes campos: identificador de gnodo (16 bytes), función (1 byte), TTL ó tiempo de vida restante (1 byte), Hops ó saltos (1 byte) y largo de la carga en bytes (4 bytes). El valor referenciado en el campo función indica que tipo de acciones deben realizar aquellos gnodos que reciben el mensaje. La cabecera descripta es fija para cualquier tipo de función instanciada, lo que es variable en tamaño es la segunda parte ó carga del mensaje dado que depende directamente de la función. Gnutella para operar necesita cinco funciones [DSS2] que le permitirán mantenerse en la red, consultar por recursos y descargar archivos.

1. **Función Ping ó Init (código 0x00)**

   Se utiliza a los efectos de que gnodos anuncien su presencia en la red. No lleva carga de mensaje, es decir que el campo largo de la carga se instancia en cero. Cada gnodo que reciba un mensaje Ping debe responder generando mensajes Pong (pueden ser n) donde anuncien su presencia ó de gnodos que tengan noticia. Un gnodo generará un mensaje Ping para obtener información sobre nuevos puntos de acceso a la red y para recabar datos que le permitan definir su horizonte (cantidad de gnodos operativos, cantidad de archivos compartidos y megabytes de información compartida).

   Ejemplo. mensaje Ping

```
                    27 08 3F 71 49 B2 D4 11 88 23 00 80 AD 40 4E 62
                    00 07 00 00 00 00 00
Mensaje
       Identificador de gnodo: 27 08 3F 71 49 B2 D4 11 88 23 00 80 AD 40 4E 62
       Función               : 00
       TTL                   : 07
       Hops                  : 00
       Largo de la carga     : 00 00 00 00
```

2. Función Pong ó Init_Response (código 0x01)

Un gnodo envía un mensaje Pong como respuesta a un mensaje Ping. La carga de un Pong se compone de los siguientes campos: puerto (2 bytes), dirección IP (4 bytes), cantidad de archivos compartidos (4 bytes) y kbytes compartidos (4 bytes). Es interesante destacar que un gnodo al recibir datos como los descriptos puede abrir nuevas conexiones que le permitan extender su dominio de acción y, a la vez, darle una mayor estabilidad a su presencia en la red Gnutella. Un gnodo puede enviar más de un mensaje Pong como respuesta a un único Ping, dado que informa acerca de otros gnodos que tenga almacenado en su memoria cache.

Ejemplo. mensaje Pong

```
                    27 08 3F 71 49 B2 D4 11 88 23 00 80 AD 40 4E 62
                    01 07 00 0E 00 00 00 CA 18 AA D2 62 95 03 00 00
                    00 00 00 00 00
Mensaje
       Identificador de gnodo: 27 08 3F 71 49 B2 D4 11 88 23 00 80 AD 40 4E 62
       Función        : 01
       TTL                   : 07
       Hops                  : 00
       Largo de la carga     : 0E 00 00 00
 Carga
       Puerto                : CA 18
       Dirección IP          : AA D2 62 85
       Cantidad de archivos
       compartidos           : 03 00 00 00
       Kbytes compartidos    : 00 00 00 00
```

3. Función Push (código 0x40)

Es utilizada por gnodos que acceden a la red atravesando un firewall, y no pueden descargar un recurso situado en un gnodo al otro lado del mismo. El gnodo que requiere el recurso enviará un mensaje Push al gnodo que lo tiene, con los siguientes campos: identificador de gnodo (16 bytes), índice (4 bytes), dirección IP (4 bytes) y puerto (2 bytes). Una vez obtenido el mensaje, el gnodo poseedor del recurso establecerá una conexión con el gnodo peticionante y le enviará, a la dirección IP y puerto informado, el archivo referenciado por el campo índice. Una respuesta a la función Push no está definida, dado que un gnodo puede enviar el recurso ó no hacer caso de la petición.

Ejemplo. mensaje Push

```
                    27 08 3F 71 49 B2 D4 11 88 23 00 80 AD 40 4E 62
                    40 07 00 1A 00 00 00 22 11 3F 71 49 B2 D4 11 88
                    23 00 80 AD 40 22 11 02 00 00 00 AA 12 62 AA CA
                    22
Mensaje
       Identificador de gnodo: 27 08 3F 71 49 B2 D4 11 88 23 00 80 AD 40 4E 62
       Función               : 40
       TTL                   : 07
       Hops                  : 00
       Largo de la carga     : 1A 00 00 00
Carga
       Identificador de gnodo: 22 11 3F 71 49 B2 D4 11 88 23 00 80 AD 40 22 11
       Indice                : 02 00 00 00
       Dirección IP          : AA 12 62 AA
       Puerto                : CA 22
```

4. Función Query (código 0x80)

Se utiliza a los efectos de que un gnodo pueda consultar a otros gnodos por un recurso. La carga utilizada por esta función se compone de dos campos, a saber: velocidad mínima requerida para descarga (2 bytes) y un criterio de búsqueda de longitud variable. Un gnodo al recibir un mensaje Query y constatar que no hay coincidencias en su estructura de archivos no generará ningún mensaje de respuesta.

Ejemplo. mensaje Query

```
                    28 08 3F 71 49 B2 D4 11 88 23 00 80 AD 40 4E 62
                    80 07 00 06 00 00 00 00 00 2A 2E 61 00
Mensaje
       Identificador de gnodo: 28 08 3F 71 49 B2 D4 11 88 23 00 80 AD 40 4E 62
       Función               : 80
       TTL                   : 07
       Hops                  : 00
       Largo de la carga     : 06 00 00 00
 Carga
       Velocidad mínima      : 00 00
       Criterio de búsqueda  : 2A 2E 61 00 (*.a + caracter nulo)
```

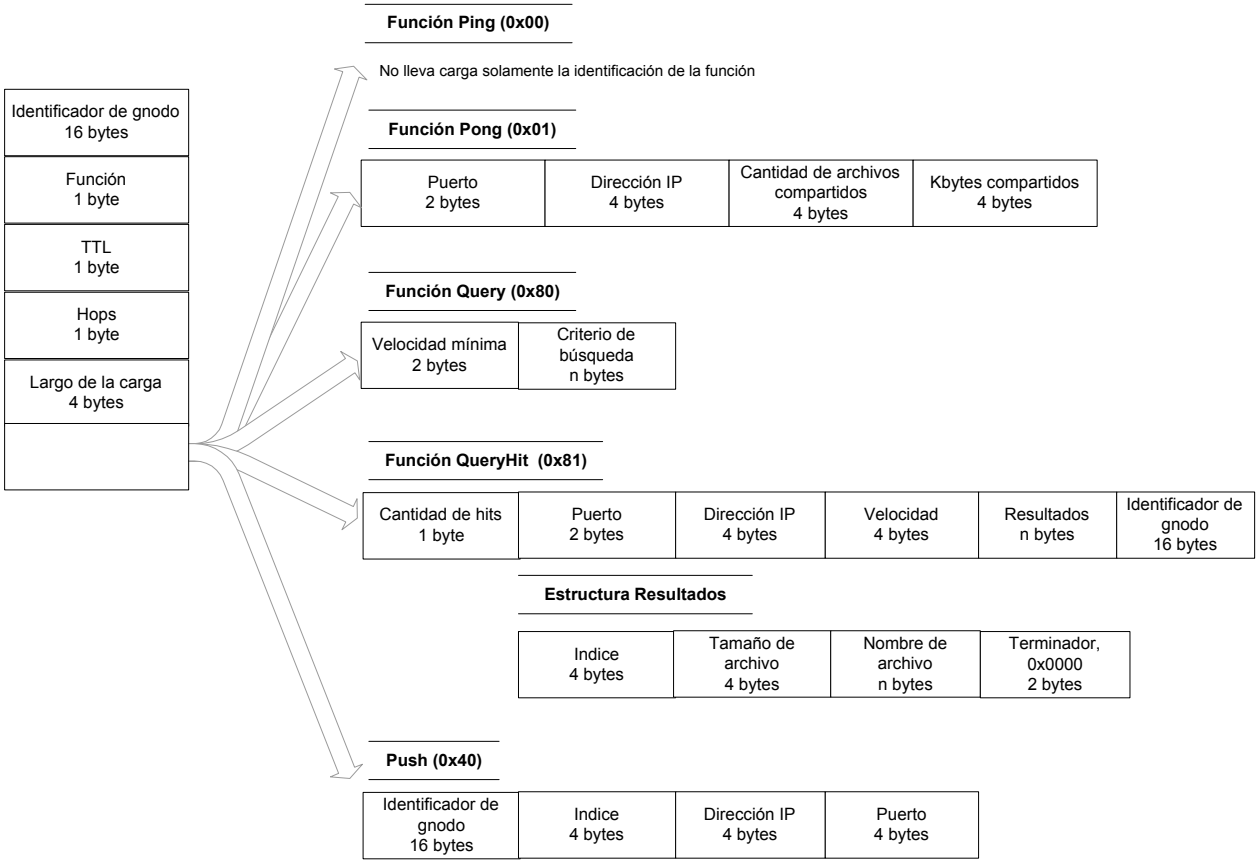
5. Función Query_Hit (código 0x81)

Se utiliza como respuesta a un mensaje Query. Los campos que componen un Query_Hit son: cantidad de hits ó coincidencias (1 byte), puerto (2 bytes) y dirección IP de descarga (4 bytes), velocidad de operación (4 bytes), resultados (longitud variable) e indentificador del gnodo que responde (16 bytes). Los resultados (que pueden ser n) se estructuran como una lista, de la siguiente forma: índice ó número de respuesta (4 bytes), tamaño del archivo (en kbytes) (4 bytes), nombre del archivo (longitud variable) y delimitador de registro (0x0000). El gnodo que responde incluye en la respuesta su identificador a los efectos de que lo pueda utilizar el gnodo peticionante, si decidiera solicitar un recurso mediante un mensaje Push.

Ejemplo. mensaje Query_Hit

```
                    28 08 3F 71 49 B2 D4 11 88 23 00 80 AD 40 4E 62
                    81 07 00 57 00 00 00 03 CA 18 AA D2 62 95 1C 00
                    00 00 00 00 00 00 70 00 00 00 31 34 39 61 72 63
                    68 69 2E 61 00 00 01 00 00 00 70 00 00 00 31 34
                    39 61 72 63 68 69 2E 62 00 00 02 00 00 00 70 00
                    00 00 31 34 39 61 72 63 68 69 2E 6D 00 00 A0 3D
                    8D 3D 7D 84 D1 11 85 8E 52 54 00 DC 37 66
Mensaje
       Identificador de gnodo: 28 08 3F 71 49 B2 D4 11 88 23 00 80 AD 40 4E 62
       Función       : 81
```

```
        TTL                   : 07
        Hops                  : 00
        Largo de la carga     : 57 00 00 00
 Carga
        Cantidad de hits      : 03
        Puerto                : CA 18
        Dirección IP          : AA D2 62 95
        Velocidad             : 1C 00 00 00
        Estructura resultados
        Identificador de gnodo: 3D 8D 3D 7D 84 D1 11 85 8E 52 54 00 DC 37 66
Estructura resultados
        1er elemento
                Indice                : 00 00 00 00
                Tamaño de archivo     : 70 00 00 00
                Nombre de archivo     : 31 34 39 61 72 63 68 69 2E 61 (149archi.a)
                Terminador            : 00 00
        2do elemento
                Indice                : 01 00 00 00
                Tamaño de archivo     : 70 00 00 00
                Nombre de archivo     : 31 34 39 61 72 63 68 69 2E 62 (149archi.b)
                Terminador            : 00 00
        3er elemento
                Indice                : 02 00 00 00
                Tamaño de archivo     : 70 00 00 00
                Nombre de archivo     : 31 34 39 61 72 63 68 69 2E 6D (149archi.m)
                Terminador            : 00 00
```

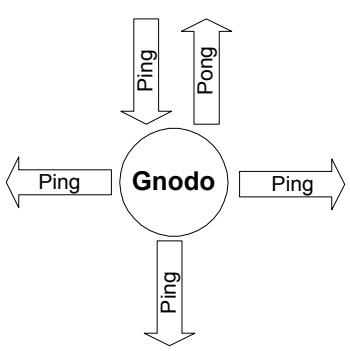


Estructuras de datos de mensajes Gnutella

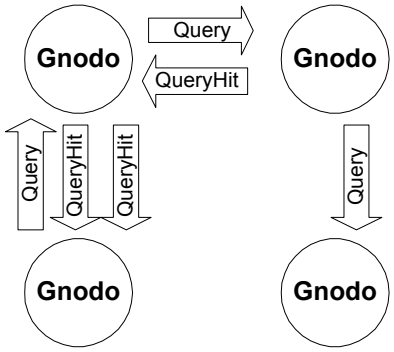
II.4.4 Reglas de Propagación

El modelo compañero a compañero implentado en Gnutella requiere que los gnodos tengan la capacidad de propagar los mensajes recibidos (funciones Ping, Pong, Query, Query_Hit y Push) a través de sus conexiones establecidas. A los efectos de hacer eficiente la operación de la red (teniendo en cuenta su topología, la no jerarquía y la existencia de loops) deben implementarse en cada gnodo, una serie de reglas de propagación [DSS2], que a continuación se detallan:

> **Regla a.** Un gnodo propagará mensajes tipo Ping y Query a todos sus gnodos conectados directamente, excepto al que entregó dicho mensaje.



Propagación de Ping



Propagación de consultas

Regla b. Los mensajes tipo Pong, Query_Hit y Push solo deben ser propagados por el mismo camino por el que viajó el mensaje inicial (Ping ó Query) asociado. Esto es posible debido a que antes de propagar un mensaje, los gnodos revisan en tablas internas por aquel que generó tal respuesta, y así obtienen la conexión por donde deben reenviarlos.

Regla c. Un gnodo decrementará en 1 el valor del campo TTL de un mensaje e incrementará el valor del campo Hops en 1 antes de propagar dicho mensaje por las conexiones pertinentes. Si al decrementar el valor de TTL el resultado obtenido es cero, debe desechar el mensaje.

Regla d. Si un gnodo recibe un mensaje con idéntico identificador de gnodo y mismo valor de función que uno recibido anteriormente (en un período breve de tiempo, no especificado formalmente) debería evitar propagarlo.

En un instante de tiempo puede haber miles gnodos en la red, pero para un gnodo en particular su dominio de acción estará restringido a la cantidad de conexiones establecidas y al alcance de sus mensajes (antes de que el valor del campo TTL llegue a cero y sea descartado). Debido al dinamismo, la topología y la existencia de usuarios temporales, es posible que las respuestas a una misma consulta varíen en el tiempo. Cada vez que un usuario ingresa en la red puede hacerlo a una parte diferente de la misma, según el estado actual de

sus gnodos vecinos. Resumiendo, los modos de propagación utilizados por el protocolo Gnutella son los siguientes:

Mensaje	*Forma de propagación*
Ping	Broadcast
Pong	Propagación directa
Query	Broadcast
Query_Hit	Propagación directa
Push	Propagación directa

II.4.5 Descarga de archivos

Una vez que un gnodo recibe un resultado (Query_Hit) a una consulta previamente realizada, puede optar por seleccionar un archivo para su descarga desde el gnodo remoto. Los archivos se recuperan directamente, sin intervención de gnodos intermedios y por lo tanto tampoco de la red Gnutella. La descarga se realiza mediante el uso del protocolo HTTP versión 1.0 [Berners]

Un gnodo al recibir un Query_Hit obtiene los siguientes datos: dirección IP y puerto donde el gnodo que posee el recurso está en modo de apertura pasiva, índice, tamaño y nombre del archivo.

A los efectos de iniciar la descarga, el gnodo que requiere el recurso abrirá una conexión TCP contra el gnodo remoto en el puerto anteriormente obtenido y, mediante la primitiva GET del protocolo HTTP, solicitará el recurso seleccionado, indicando índice y nombre de archivo.

Ejemplo

```
GET /<índice>/<nombre de archivo>/  HTTP/1.0
Connection: Keep-Alive
Range: bytes=0-
```

El gnodo remoto al recibir tal petición y validar que el recurso solicitado está disponible, transmitirá el archivo, previo envío de una cabecera estándar de HTTP con el siguiente formato:

```
HTTP 200 OK
Server: Gnutella
Content-type: application/binary
Content-length: <tamaño del archivo en bytes>
```

Es importante destacar que es posible la inclusión de la funcionalidad *resume* en la cabecera de la primitiva de protocolo GET, a los efectos de solicitar el reenvío de un archivo a partir de una posición determinada y hasta el final. Esto es útil en caso de descargas previas que hayan sido interrumpidas.

En el documento "Gnutella: Distributed System for Information Storage and Searching. Model Description" [Bordignon] se describe, a partir del análisis de una captura de datos, la interacción entre distintos gnodos.

II.4.6 Escalabilidad, un punto débil

Gene Kan[16] expresó que la primera versión del protocolo fue diseñada para que solo unos pocos cientos de usuarios se conecten a través de la red y que los programadores que la crearon, no previeron los problemas de grandes redes de usuarios con diversas velocidades de conexión.

El modelo de ruteo de Gnutella -por difusión - es ineficiente, debido a que la topología de la red responde a un grafo cíclico, y por ende un mensaje puede llegar a un único nodo más de una vez a un mismo nodo. Si se enviara un requerimiento con un valor 10 de tiempo de vida, y cada nodo que lo propague tuviera 6 conexiones, se tendrían 10^6 (1 millón de mensajes) mensajes intercambiados. Tal situación convierte a Gnutella es un ámbito propicio para que ataques de denegación de servicios sean eficaces. De hecho, en la práctica, se implementa una regla para mejorar la perfomance, que dice que si un nodo recibe un mensaje que ha sido visto anteriormente lo debe eliminar.

Por otro lado, la escalabilidad de la red de Gnutella está determinada por las conexiones de baja velocidad de los usuarios hogareños que se comunican bajo enlaces telefónicos tipo dialup. El protocolo no toma ventaja de las características particulares de un gnodo determinado, tales como ancho de banda, disponibilidad de almacenamiento, ni velocidad de CPU; es decir que todos los gnodos son tratados por iguales a la hora de retransmisión. Esto hace que algunos problemas se produzcan, y en particular congestiones, debido a la desigualdad de ancho de banda entre gnodos que disponen de conexiones permanentes (ADSL, frame relay, ATM) y gnodos con conectividad variable (v.90).

Un estudio de la empresa Clip2 DSS, realizado entre julio y agosto del año 2000, contó el número de mensajes de difusión y la cantidad de gnodos presentes, durante intervalos de tiempo. En base tales datos, se obtuvo la cantidad promedio diaria de consultas sobre la red Gnutella. Tal métrica es una medida global de utilización de la red. El siguiente gráfico muestra la evolución del uso de la red durante el periodo de tiempo aludido.

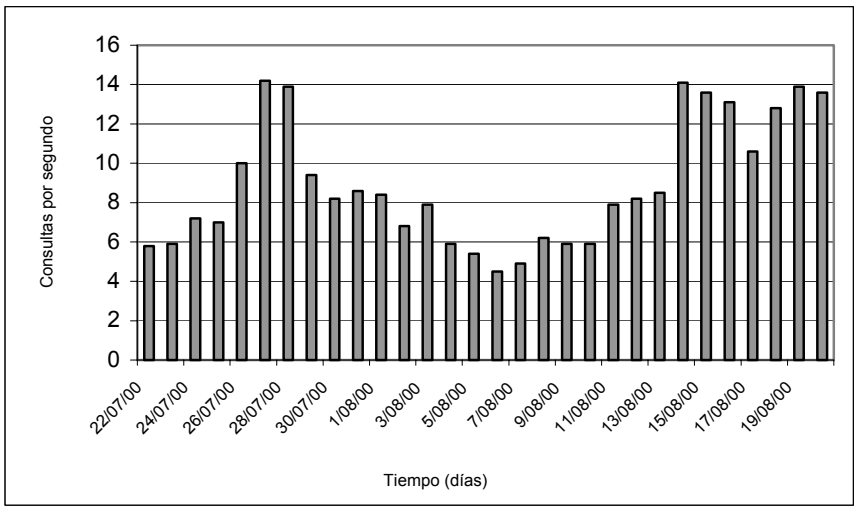


Gráfico evolución de la red Gnutella

Del análisis del gráfico, según la empresa Clip2 [Clip2], se puede inferir que:

1- Las consultas crecieron cerca de un 100% entre el 26 y el 28 de julio, periodo en el cual se anunció el inminente cierre del servidor central de Napster.

[16] Programador que lideró al primer grupo de desarrolladores del protocolo Gnutella.

2- Luego de la decisión de que Napster pueda continuar por un tiempo más, el nivel de operación de la red Gnutella volvió a valores anteriores a la situación planteada en el punto 1.

3- A partir del 6 de agosto se nota un crecimiento en el uso de la red, llegando a valores similares a los detectados en el periodo de tiempo referido en el punto 1.

A partir del día 14 de agosto, en los foros de al red, se empezaron a recibir quejas de usuarios que básicamente decían que las respuestas a sus consultas eran menores y que la cantidad de megabytes compartidos y gnodos descubiertos se redujo notablemente. Por otro lado la empresa Clip2 notó que las respuestas a mensajes tipo ping decrecieron; basándose en tales datos, el grupo de estudio, concluyó que la red posee una barrera a la escalabilidad situada en aproximadamente 10 consultas por segundo. La fuente de tal barrera son las numerosas conexiones dialup de 56 Kbps; dado que tales gnodos no poseen un ancho de banda suficiente para propagar mensajes cuando la red opera, en promedio, a más de 10 consultas por segundo.

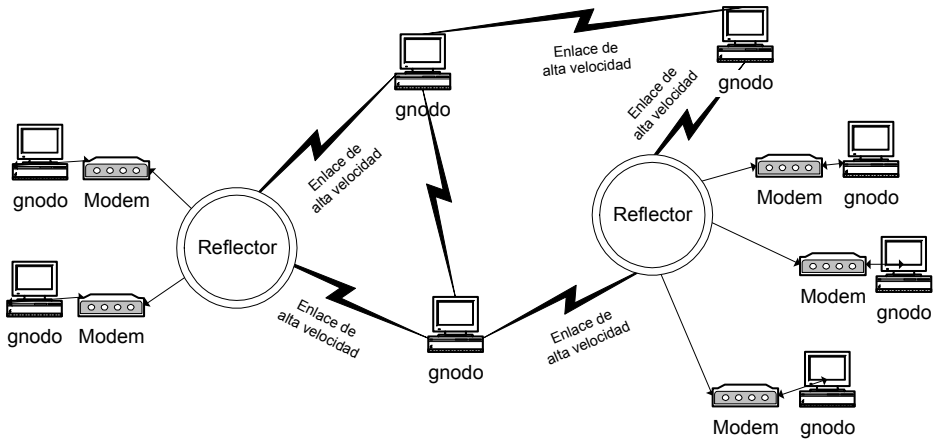
Clip2 asegura que la moda de conexiones por gnodo es 3. Un mensaje típico de consulta es de alrededor de 30 bytes; a los cuales hay que sumarle las cabeceras TCP e IP (40 bytes) lo que da un total de 70 bytes. A los efectos de hacer una estimación de ancho de banda requerido para que un gnodo típico opere se tiene que se requieren inicialmente 560 bis (70 bytes) * 3 conexiones * 10 mensajes/segundo, lo que es igual a 16.800 bits/segundo.

Por otro lado existen los mensajes de anuncio de presencia (pings y pongs), del análisis de su presencia se comprobó que cuadriplican la cantidad de consultas y poseen un tamaño similar. Por lo que la fórmula total[17] de requerimiento de ancho de banda de un gnodo sería 4 * 16.800 bps ó 67.200 bps; cifra que excede el ancho de banda disponible de una comunicación telefónica bajo norma V.90. El resultado de tal exceso, en requerimientos de velocidad, se visualiza como lo manisfestaron los usuarios en los foros.

[17] En la fórmula no se incluyeron respuestas a consultas, ni mensajes push..

II.4.7 Reflectores, un aporte a la escalabilidad

En el mes de setiembre del año 2000, cuando Napster cerró, 50.000 nuevos nodos se incorporaron a la red Gnutella [Clip2-A] y debido a sus falencias de escalabilidad se saturaron enlaces y la red se fragmentó. Algunas soluciones se propusieron a los efectos de remediar tales problemas de perfomance, se implementaron una serie de nodos reflectores [Clip2-B] que su objetivo era balancear tráfico, a partir de informar a los nodos de usuario final con conexiones lentas, donde deberían conectarse. En definitiva, los reflectores derivaban a las gnodos con conexiones de bajo ancho de banda a los bordes de la red, dejando en el centro (una especia de backbone) a un conjunto de gnodos con buen ancho de banda, capaces de propagar el tráfico creciente.



Otro aporte a la evolución del protocolo Gnutella se basa en solucionar un error de diseño del protocolo. Es la periodicidad con los que los mensajes ping son enviados a la red, demasiada frecuencia, a los efectos de recabar información sobre vecinos; estudios demostraron que tales mensajes representan el 50% de todo el tráfico y por ende esta es una de las razones de la baja perfomance y la escasa escalabilidad. La empresa de software Lime Wire propuso una nueva versión del protocolo gnutella (denominada 0.6) en donde incorpora técnicas de gestión de mensajes ping y pong a los efectos de reducir la carga de red producida por tales mensajes.

II.4.8 Seguridad

Según Gene Kan, la liberación del protocolo Gnutella y la publicación del código fuente de las aplicaciones que lo implementan contribuyen a que los usuarios puedan personalizar sus programas, desarrollar nuevas versiones y hasta proponer mejoras al protocolo. Esta apertura determina que si hay diferentes tipos de clientes hay más riesgo de fallos, pero estos fallos serán diferentes en cada programa. Esto hace a la red más robusta porque un sólo fallo no estará presente en toda la red.

Gnutella adolece del problema de no ser exhaustivo en las búsquedas y por otro lado saturarse debido a su protocolo de ruteo por amplificación, pudiendo ser víctima de ataques de denegación de servicio (bombardeo de demandas).

En el año 2001 se ha detectado un virus denominado W32/Gnuman.worm, también conocido como Mandragore. Es un archivo binario (EXE) maligno que se enmascara como un archivo de música MP3 o como un archivo de imagen el cual infecta a la computadora del usuario una vez que el programa es ejecutado.

Al realizar la infección el gusano se copia asimismo en la carpeta de inicio del sistema Microsoft Windows con el nombre de "gspot.exe" y se aplica los atributos de sistema y oculto. Mediante este primitivo sistema el virus logra permanecer oculto e iniciarse de forma automática cada vez que se inicia la computadora, de forma que permanece en la memoria como proceso activo.

II.4.9 Aspectos sociales

Es una alternativa interesante a Napster porque no existen servidores centrales que estén a merced de que órdenes emanadas de tribunales determinen su clausura. Para evitar el intercambio de música, habría que cerrar toda la red de Gnutella, algo que ningún tribunal podría hacer cumplir. Desde el punto de vista de los usuarios, un beneficio de tales redes descentralizadas es que es casi imposible clausurarlas, dado que por definición los contenidos se hallan dispersos en máquinas de usuario, y muchas veces replicados.

A pesar de las infracciones a los derechos de propiedad y la proliferación de contenidos pornográficos –generalmente relacionados con la niñez-, hay algo ideológicamente que hace que las redes P2P sean bien vistas, y es su concepto de democracia ó sitio construido por la voluntad y compromiso de los pares.

Un estudio sobre el comportamiento de los usuarios en la red Gnutella [Adar] determino la presencia de una conducta denominada "freeloading" ó "free riding". Es la práctica de descargar masivamente archivos sin ofrecer nada a cambio, nada más lejano que la idea del cooperativismo. Tal investigación, sobre 24 horas de recolección de datos, ha mostrado que esta práctica está muy difundida en redes P2P como Gnutella. En el mes de agosto del año 2000 solo un 30% de los usuarios al menos compartían un archivo y que el 50% de las respuestas eran brindadas por el 1% de los gnodos que comparten recursos. En tales 24 horas se observaron 35.352 gnodos que compartían 3.304.046 archivos.

Otro estudio [DSS1], llevado a cabo por la empresa Clip2 DSS, realizado en el mismo período utilizando otra metodología, determinó que la cantidad máxima cantidad de nodos que comparten al menos un archivo no estuvo históricamente más allá del 40 %; y en tiempos posteriores –donde a partir de la clausura de Napster la red alcanzó la máxima cantidad de compañeros- tal situación rondó un 15%. El éxito de las redes cooperativas P2P se basa en que exista un funcionamiento equilibrado por parte de los compañeros, es decir que en general lo que un nodo toma de la red sea proporcional a lo que él le aporta. Este fenómeno freeloading puede llegar a ser una causa de fracaso si no se modifica el comportamiento de los usuarios. Sistemas de incentivos tienden a contrarrestar el efecto enunciado; la aplicación MojoNation incorpora a tal efecto un modelo proporcional de créditos de descarga, calculado sobre la base de lo que el usuario ofrece a la red.

Para mayor información, en el anexo 1 "Estudio del contenido de mensajes de una red Gnutella" se muestran los resultados obtenidos a partir de analizar y clasificar mensajes capturados en la red Gnutella.

II.4.10 Consideraciones

El uso masivo del protocolo en implementaciones sobre diferentes plataformas ha demostrado que Gnutella es un punto de inicio válido para el desarrollo de servicios distribuidos basados en la filosofía compañero a compañero.

Debido a su génesis y corta existencia, Gnutella es aún un protocolo inmaduro, sometido a estudios para evaluar mejoras en eficiencia y funcionalidad (para implementar en versiones posteriores). Donde habrá que enfocar los esfuerzos en puntos tales como escalabilidad, "freeloading" (a partir de definir sistemas de incentivos) y la problemática derivada de las transferencias interrumpidas por efecto de la conectividad variable

Como característica interesante, presenta la definición de una red a nivel aplicación, orientada al cooperativismo, que brinda elementos para favorecer la heterogeneidad, la estabilidad, la redundancia y la tolerancia a fallas.

II.5 Otros Proyectos

Más allá de los protocolos expuestos en este capítulo, existen numerosas alternativas de probada eficacia, donde sobresalen aplicaciones tales como:

II.5.1 FREENET

Como Gnutella, FreeNet [Clarke] es una red destinada a que usuarios de forma anónima puedan publicar y obtener archivos. Ian Clarke[18] es el autor del protocolo Freenet cuyo objetivo es la transmisión de información anónima bajo un ambiente resistente a la censura. El programa Freenet brinda un anonimato total, es imposible saber donde está almacenado un determinado archivo, dado que está segmentado y replicado en diversas computadoras.

[18] Investigador del Laboratorio de Inteligencia Artitificial de la Universidad de Edimburgo

Por su diseño, garantiza el fracaso de cualquier intento de censura de sus contenidos. Cuando un usuario publica en un servidor de la red un documento, éste se replica en varios servidores distintos, y a la vez posee un mecanismo para frustrar cualquier intento de borrado de archivos ó de determinar su fuente. Su creador afirma además, que el hecho de que los documentos se encuentren en múltiples servidores convierte a Freenet en un sistema de distribución de información mucho más eficiente que el sistema Web.

Freenet funciona sin ningún control central, según Clarke *"es una anarquía casi perfecta*". Cuando se publica un documento, el sistema codifica el archivo y dispersa su clave de localización (un número de gran tamaño). Asimismo, incorpora un mecanismo de protección que responde a cualquier intento de ubicar una información, esparciéndola por la red.

Para su infraestructura de almacenamiento y localización de recursos, tomo ideas del método clásico de búsqueda en estructuras arborescentes binarias. Por lo cual, el sistema se vería como una comunidad de computadoras que se comportasen como un árbol binario, remitiendo las solicitudes de información hacia los nodos apropiados hasta que se encuentre la información solicitada y luego retornarla al usuario que la solicitó. El método adolece del problema que fallas en distintas computadoras de la pirámide generan problemas de distinta gravedad (no disposición de recursos), donde las fallas más cercanas a la computadora nodo raíz son las más importantes. En una red donde las computadoras que almacenan y requieren recursos son equipos de conectividad variable tal modelo puro implicaría el no funcionamiento de la red. Debido a este problema se pensó en una "jerarquía redundante" donde en vez de que cada nodo en la jerarquía sea una computadora, cada nodo sería un grupo de computadoras que compartiría la responsabilidad de ese nodo. Los nodos en la red que tengan más responsabilidad contendrían más computadoras, para que la responsabilidad de cada computadora individual esté limitada

II.5.2 GROOVE

Es una plataforma P2P destinada al trabajo en grupo en tiempo real. La transferencia de información entre usuarios es directa, sin pasar por un servidor central, con lo que los

archivos y documentos que se intercambian se encuentran en los equipos de usuario final. Groove [Suthar] permite a grupos de trabajo intercambiar archivos, de distinto tipo a través de Internet o de Intranets corporativas. También puede usarse con otros fines personales, por ejemplo, relacionado con comunidades de amigos o familiares.

Entre las herramientas actualmente disponibles, destaca el intercambio de archivos, área de discusión, conferencia, calendario, etc. Además, se dispone de una herramienta de desarrollo, denominada "Groove Developement Kit", para adaptarlo a necesidades particulares de los usuarios.

Según Ray Ozzie, su creador[19], Groove es una plataforma a la que acuden programadores en busca de herramientas e información para introducir prestaciones de colaboración entre usuarios en aplicaciones existentes. Groove distribuye de manera gratuita su software cliente, con la intención de que se transforme en una herramienta de programación estándar. Ozzie se refiere a Groove como el peerware, es decir el programa común en el que se basarán las aplicaciones de intercambio de archivos.

La idea de Groove es permitir a las empresas e individuos crear espacios de trabajo virtuales en los cuales los participantes intercambien datos directamente desde sus computadoras personales. Sin necesitar servidores *web* de por medio. Un aspecto importante es la seguridad. A pesar de utilizar la arquitectura P2P, la información y la transmisión van cifradas. Además, unos algoritmos creados expresamente actualizan los cambios que el resto de miembros de un grupo de trabajo (hasta 25) haya efectuado.

La versión inicial de Groove es gratuita, pudiéndose descargar de la página principal del proyecto. El programa a instalar en el equipo de usuario se lo denomina Transceiver, e incluye un espacio de trabajo compartido y aplicaciones de mensajería instantánea, conversaciones con voz, calendario y explorador.

II.5.3 Hotline

[19] Y además creador del sistema de colaboración Lotus Notes.

–

Hotline [Hotline] es una aplicación que opera sobre el compañero a compañero destinada a construir comunidades y compartir archivos. Consta de dos programas, Hotline cliente y Hotline servidor, que operan sobre una red tipo intranet o sobre Internet. Provee capacidades similares a FTP y al sistema de conferencias IRC. Características que lo definen son: soporta continuación diferida de descargas, conferencias en tiempo real, foros de noticias, posibilidad de transmisión de flujos de audio y video.

La aplicación está disponible desde 1997 y actualmente es soportada en plataformas Microsoft Windows y System de Macintosh.

II.5.4 Publius

Tal como Freenet, Publius [Waldman] es un sistema de distribución de contenidos cuya principal característica es la resistencia a la censura. El objetivo del programa es que cualquiera pueda publicar cualquier tipo de material escrito en Internet, sin posibilidad de ser identificado o censurado. Los archivos se encriptan y se los divide en partes que son distribuidas en varios servidores de Internet [Waldman]. Las partes, denominadas claves, están diseñadas de forma que se puedan rearmar y convertir en un mensaje completo. De esta manera, mientras que las claves pueden alojarse en decenas o cientos de servidores, el usuario sólo necesita tener acceso a algunas de esas computadoras y la suficiente información para rearmar el archivo deseado.

Los creadores de Publius afirman que incluso si un Gobierno tuviera autoridad legal para retirar un archivo de los servidores, no podría hacerlo por la cantidad misma de servidores que contienen la información.

Se podrán publicar documentos mediante este sistema de información, existiendo una limitación de tamaño, donde no se podrá sobrepasar de 100 KB. De esta manera intentan evitar que el sistema sea usado para distribuir música de manera ilegal a través de la red.

El programa emplea un complejo sistema criptográfico donde está en desarrollo el equivalente a un sistema de direcciones tipo URL en la cual se incluye la información necesaria para que la aplicación halle todas las piezas o claves ocultas y las una de nuevo. Estas direcciones no podrán ser usadas para descubrir al autor o localizar las piezas individuales del archivo para luego borrarlas.

Según sus diseñadores, el sistema tendrá una configuración por defecto que dividirá el mensaje en cuestión en 20 claves. Pero cuatro claves cualesquiera de esas 20 tendrán la suficiente información como para rearmar el documento.

II.5.5 Cuadro comparativo de sistemas de intercambio de archivos

La siguiente tabla comparativa expone características de los distintos sistemas de intercambio de archivos mencionados.

Protocolo	*Servidor central*	*Protocolo abierto*	*Opera con archivos*	*Encriptación*
NAPSTER/OPENAP	Si	Si	MP3	No
GNUTELLA	No	Si	Todo tipo	No
FREENET	No	Si	Todo tipo	Si
GROOVE	No	No	Todo tipo	Si
HOTLIN	Si (parcial)	No	Todo tipo	No
PUBLIUS	Si (Multiples)	Si	Todo tipo	Si

Características de protocolos orientados a compartir archivos

II.6 ¿Los compañeros son compañeros?

Uno de los factores que determinan el éxito de un sistema de intercambio de archivos es el grado de cooperación por parte de sus componentes. Cooperar significa poner recursos a disposición de la red, donde el recurso básico de un sistema de intercambio de archivos es el archivo mismo. Desde que muchos usuarios no creen beneficiarse directamente poniendo sus archivos a disposición de compañeros, no lo hacen; tal situación es un problema que

compromete el fututo de tales sistemas. Debido a que atenta contra la "riqueza" que se puede hallar en el sistema. Diversos estudios [Adar] [Saroiu] han demostrado que una porción importante de usuarios, llamados "free riders", utilizan la red Gnutella sin compartir archivo alguno. A los efectos de atacar tal problemática sistemas tales como AudioGalaxy han establecido mecanismos de equilibrio basados en modelos de "ratio"[20]. Los modelos de incentivos en redes P2P es una de las áreas de investigación poco exploradas, pero de vital importancia para la evolución de las aplicaciones. Estudios iniciales [Golle] tratan de modelar el comportamiento de los usuarios y definir, mediante el apoyo de la teoría de juegos, posibles modelos que incentiven la plena cooperación.

[20] Generalmente, cada n megabytes que el usuario pone a disposición de la red, se le habilitan m megabytes de crédito de descarga,

Capítulo III

III. GnutWare

III.1 Introducción

En este capítulo se describe un aporte original que consiste en el diseño y codificación de un prototipo de middleware que brinda soporte de comunicaciones a aplicaciones de usuario final sobre una red de aplicación (overlay network) que opera bajo el modelo de comunicaciones compañero a compañero [Tolosa-B].

La arquitectura propuesta está basada en una red Gnutella y un prototipo de middleware codificado en lenguaje Java que brinda acceso a dicha red a diferentes aplicaciones. A modo de ejemplo, se escribió un servicio de consulta de tiempo que opera distribuida y puede servir como soporte de un mecanismo de sincronización de relojes dentro del sistema. La arquitectura propuesta se presenta como una solución modular a los efectos de dividir la funcionalidad en lo que respecta por un lado a la red P2P y – por otro – a las aplicaciones, a través de una interfase normalizada.

Este trabajo fue realizado en conjunto con el Licenciado Gabriel Tolosa y se encuentra disponible bajo términos GNU, para su descarga, en el sitio Sourceforge http://gnutWare.sourceforge.net.

III.2 Utilización del modelo P2P para construir redes de recubrimiento

Como se mencionó anteriormente, los sistemas compañero a compañero permiten que computadoras de usuario final se conecten directamente a los efectos de formar comunidades, cuya finalidad será el compartir información y servicios computacionales [FIPA] [Minar]. En este modelo, se toma ventaja de recursos existentes en los extremos de la red, tales como ciclos de CPU y espacio de almacenamiento.

La primera característica que define a un nodo en un sistema P2P, es que cumple tanto el rol de cliente como de servidor (en la terminología se lo denomina *servent*, palabra que deriva de la conjunción de los términos *serv*er-*client*). Este modelo, basado en interacciones entre pares, puede reemplazar al paradigma cliente/servidor, donde cada nodo es capaz de generar y contestar consultas de otros nodos. Desde el punto de vista de la comunicación, las interacciones son simétricas.

Si un sistema P2P está íntegramente conformado por nodos que tienen iguales roles y ninguno cumple tareas administrativas o especiales, se lo considera P2P puro o completamente descentralizado [Yang]. En cambio si algún nodo posee alguna característica diferente que le hace cumplir con tareas centralizadas, entonces al sistema se lo considera híbrido. Según Intel, uno de los más grandes beneficios de P2P es el concepto de comunidad, dado que hace posible que los usuarios se organicen ellos mismos en grupos ad hoc, y puedan eficientemente y de forma segura llenar requerimientos, compartir recursos, colaborar y comunicarse. [Barkai]

Se plantea una división de la funcionalidad de la red de aplicación y de las aplicaciones propiamente dichas. En el primer caso, se pueden plantear diferentes estrategias para su construcción y mantenimiento, lo que permite tener prestaciones varias. En el segundo caso, la aplicación accede al middleware a través de una interfase normalizada para utilizar los servicios de propagación de mensajes. Las principales contribuciones de GnutWare son:

- Un modelo operativo de utilización de una red basada en el protocolo Gnutella [Bordignon] como infraestructura de intercambio de mensajes inespecíficos utilizando la técnica de difusión.

- Un modelo de comunicación entre aplicaciones de usuario final, a través de una red de mensajes por difusión.

- Una arquitectura sencilla y robusta para implementar servicios distribuidos sobre redes compañero a compañero.

- Un prototipo de middleware que presenta una interfaz normalizada a las aplicaciones para acceder a la red de transporte de mensajes.

III.3 Arquitectura del middleware GnutWare

En la arquitectura propuesta se definen dos niveles ó capas de software (Figura 1) para la implementación de servicios basados en redes compañero a compañero. Una capa de middleware, que toma servicio de la capa de transporte (dado que el modelo opera bajo el juego de protocolos TCP/IP), cuyo objetivo es propagar mensajes y permitir que los usuarios accedan a la red, y una capa de aplicación, que toma servicio del middleware, cuyo objetivo es implementar servicios de usuario final tales como sistemas de archivos, búsquedas ó procesamiento distribuido.

Este modelo permite separar la funcionalidad de cada capa de manera que se puedan desarrollar aplicaciones que corran sobre una red dada, o bien distintas redes de aplicación (con características propias) que se adecuen de mejor manera a la necesidad del servicio. Por ejemplo podría existir un servicio de mensajería, el cual opere sobre una red de aplicación que implementa mecanismos de protección de privacidad ó sobre otra red que implemente estrategias destinadas a garantizar anonimato. De igual manera una red de aplicación debería soportar distintos servicios, tales como mensajería o computación distribuida.

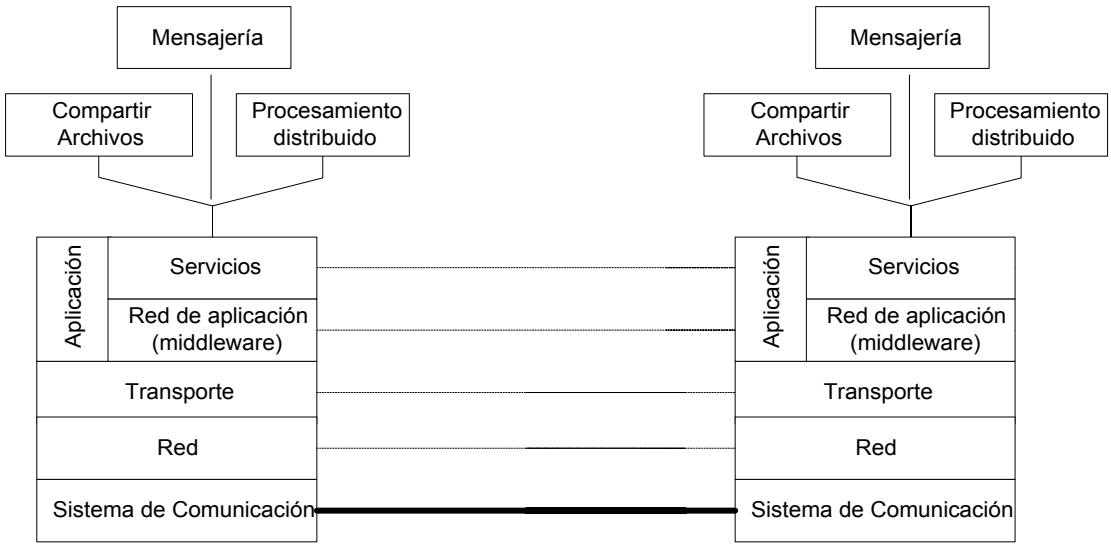


Figura 1 – División por capas de la funcionalidad de la red P2P

A partir de dividir esta funcionalidad se puede definir una capa de middleware que permita la interacción entre diferentes aplicaciones y diferentes redes P2P. Este middleware (GnutWare) brinda servicios de acceso a la red P2P a cualquier aplicación que requiera un mecanismo de envío de mensajes para operar en un ambiente distribuido a través de una interfase normalizada. Una cuestión a considerar es el tipo de servicio que se debe implementar y el mecanismo de envío de mensajes implementado por la red P2P.

En cada nodo participante de la red (nodo compañero), las dos capas de software (el middleware y la aplicación particular) deben proveer un conjunto de servicios ó funciones, de manera tal de poder asegurar el cumplimiento de los objetivos pero manteniendo el principio de división de tareas. Las funciones a implementar a nivel de middleware persiguen como objetivos:

- Permitir que la aplicación acceda a la red a través de una interfase normalizada

- Propagar mensajes de terceros (desde la red P2P hacia la aplicación y en sentido inverso).

- Tender a mantener la existencia de la red, obteniendo y brindando información de puntos alternativos de acceso a la red. El middleware puede almacenar en caché información acerca de otros nodos de la red P2P a los cuales acceder en caso de falla.

- Mantener conexiones simultáneas con otros pares. Esta función es opcional, pero puede resultar útil de acuerdo al mecanismo de reenvío de la red.

- Proporcionar a la capa de servicio (aplicación) información de puntos de acceso alternativos, es decir, de otros módulos de middleware existentes.

Para cumplir con estas funciones, el prototipo de middleware que se propone se basa en dos tipos de mensajes:

Mensajes de exploración

Tienen por finalidad el anuncio del ingreso de un nuevo nodo a la red, a los efectos de recopilar información, a través de mensajes respuesta, de otros nodos existentes.

Mensajes de servicio

Son generados y respondidos únicamente por los nodos que brindan el servicio y permiten implementar las solicitudes y respuestas de los servicios que se implementen.

Y a nivel de capa de servicio, las funciones requeridas son:

° Implementar servicios distribuidos de usuario final

- Comunicarse con los demás nodos pares

- Tender a mantener su existencia en la red (obteniendo información de puntos de acceso alternativos).

- Mantener conexiones simultáneas con uno ó más nodos pares

Toda comunicación extra necesaria (más allá de los mensajes de consulta y respuesta definidos) se realizará de manera independiente de la red de aplicación, es decir, de manera directa entre dos nodos, ya sea mediante conexiones TCP ó mensajes UDP. Esta característica responde a un criterio de optimización de la red de aplicación. Como la red es un espacio público de comunicación, solo se permiten mensajes generales y no interacciones directas particulares.

Para ejemplificar el modo de operación de la red de aplicación y su mecanismo de propagación de mensajes, se puede hacer una analogía con el funcionamiento de una red tipo Ethernet. Aquí, cualquier nodo conectado a la red puede emitir un mensaje que alcanzará a todos los destinatarios, y solo aquellos que se hagan eco del mensaje lo procesarán y eventualmente lo contestarán. Esta red de aplicación es un "pasillo virtual", cuyo ámbito de operación es una intranet ó Internet, y sus receptores todos los nodos participantes de la misma.

En el prototipo propuesto, la red de aplicación está formada por un conjunto de servents Gnutella (Gnodo), utilizando el software Gnut v.056 para Linux. Esta red P2P opera como un backbone para permitir prestar diferentes servicios distribuidos, donde una característica importante es la distribución total de las funciones, ya que no existen nodos con características especiales.

III.4 Descripción de GnutWare

El middleware GnutWare brinda servicio a las aplicaciones para acceder a la red de aplicación que sirve como transporte de mensajes genéricos. Los Gnodos se configuraron para que no posean archivos compartidos en sus sistemas locales, por lo cual se comportan simplemente como ruteadores de mensajes dentro de la red, sin prestar servicio alguno.

GnutWare se encuentra programado completamente en Java, por lo tanto es completamente portable a cualquier plataforma que soporte la ejecución de la Máquina Virtual Java. Básicamente se encuentra dividido en dos módulos principales (Figura 2):

- un módulo gnodo, que permite conectarse a un Gnodo a partir de contar la dirección IP y puerto de alguno existente en la red de aplicación

- Un módulo network_SAP, que es una interfase para las aplicaciones para posibilitar la utilización del servicio.

El módulo gnodo, implementa el protocolo Gnutella a los efectos de conectarse a la red e intercambiar mensajes de exploración (Ping/Pong) y de servicio (Query/Query_Hit), mientras que el módulo network_SAP, acepta conexiones entrantes de alguna aplicación y luego acepta mensajes de servicio y devuelve las respuestas.

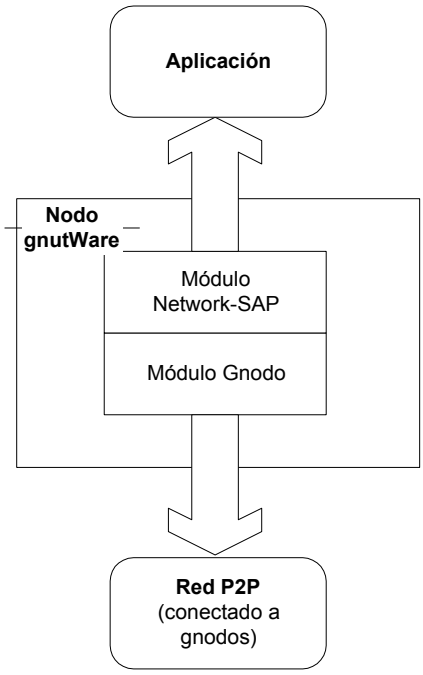


Figura 2 – Arquitectura de GnutWare.

Cuando los módulos GnuWare se conectan a algún gnodo, se forma una red de intercambio de mensajes inespecífica basada en el modelo compañero a compañero, disponible para implementar diferentes servicios distribuidos (Figura 3)

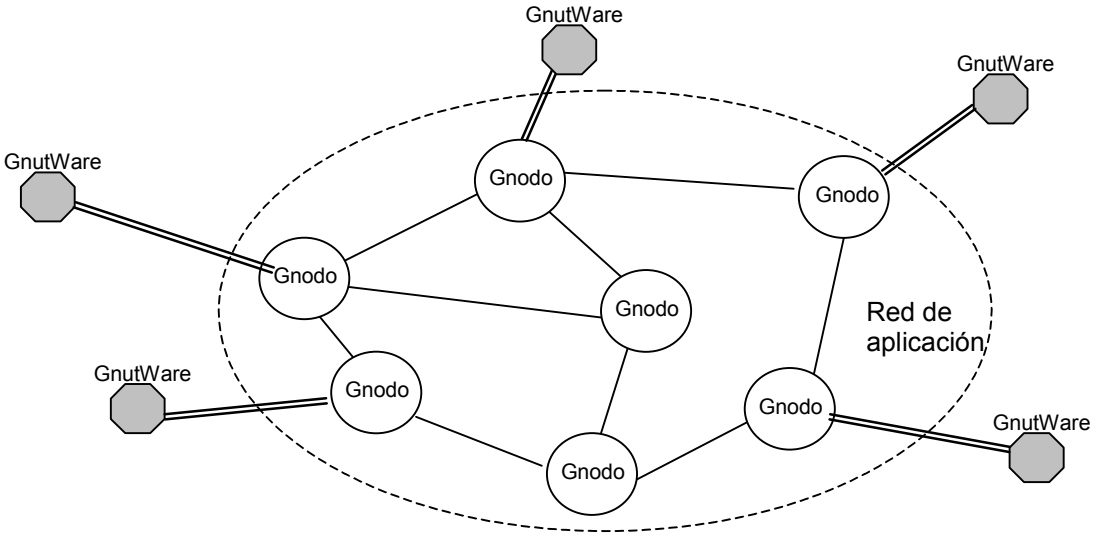


Figura 3 – Red inespecífica de intercambio de mensajes

Para acceder a dicha red, las aplicaciones debe establecer una conexión TCP contra algún módulo GnutWare y ejecutar un simple protocolo de acceso. La aplicación envía una cadena de conexión a nivel aplicación

GNUTWARE CONNECT\n\n

luego se solicita a la aplicación que identifique el código de servicio y el identificador de servicio utilizando las primitivas

GNUTWARE SERVICE CODE?\n\n

y

GNUTWARE SERVICE ID?\n\n

en este orden. La aplicación debe contestar con un entero de 2 bytes en el primer caso y con un entero de 4 bytes en el segundo. Estos datos se utilizan para permitir multiplexar a nivel de aplicación diferentes servicios y aplicaciones a los que un mismo módulo GnutWare puede brindar acceso a la red P2P. A partir de finalizar esta etapa de protocolo, la aplicación está en condiciones de enviar solicitudes de servicio y de recibir respuestas.

Cuando un módulo GnutWare recibe una solicitud de servicio, debe armar un mensaje Gnutella tipo QUERY (manteniendo la estructura de datos Gnutella) y enviarlo al gnodo con el que está vinculado, el cual se encargará de su propagación. Cuando se recibe un mensaje Gnutella tipo QUERY_HIT desde la red P2P, se pasa la respuesta a la aplicación. Las respuestas viajan dentro de la estructura de datos de una QUERY_HIT, como si fuese un mensaje Gnutella típico. Se debe respetar esta estructura ya que los Gnodos de la red de backbone solo propagan mensajes Gnutella bien formados.

A los efectos de validar el comportamiento del middleware y de la red de aplicación propuestos, se codificó una simple aplicación de solicitud de tiempo. Dicha aplicación puede servir como soporte de un mecanismo de sincronización de relojes dentro del sistema.

III.5 Un ejemplo de aplicación: Servicio cooperativo de tiempos

El servicio cooperativo de tiempos (p2pTime) pretende ser un ejemplo de una posible aplicación para correr en un ambiente distribuido soportado por GnutWare y una red P2P (Figura 4).

Una aplicación p2pTime accede a la red solicitando los tiempos (fecha y hora) de todos los demás nodos p2pTime dentro del sistema. Cuando llegan las solicitudes, todos los módulos p2pTime responden con su tiempo local al solicitante. Nótese que éstos módulos no deben preocuparse por determinar quién es el solicitante ni dónde se encuentra, ya que estas funciones son provistas por la red de aplicación subyacente. Cuando el solicitante recibe todas las respuestas, puede utilizarlas a los efectos de establecer un tiempo global aproximado del sistema ó para sincronizar relojes.

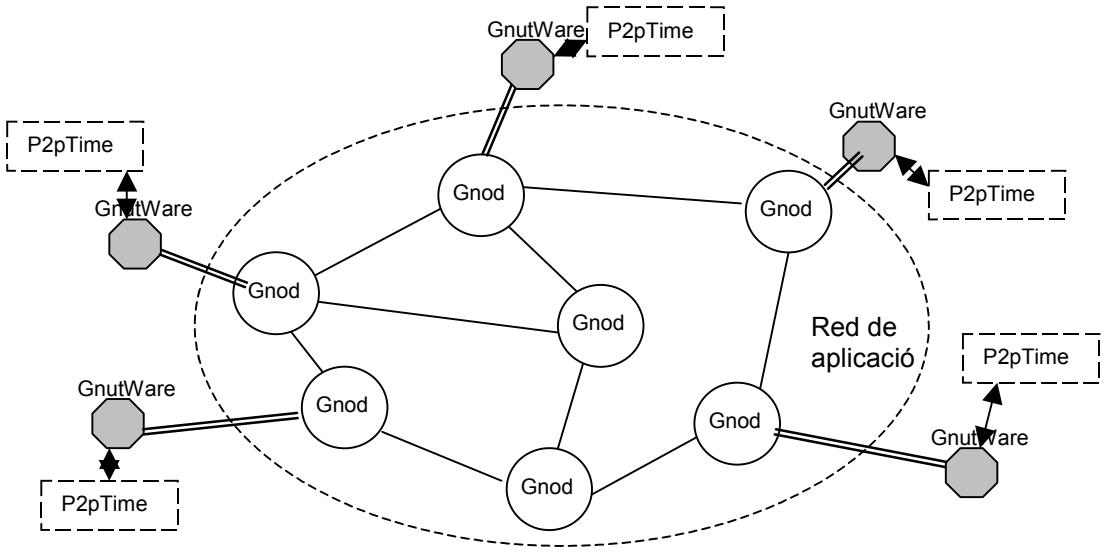


Figura 4 – Aplicaciones interactuando con el middleware

Las aplicaciones se conectan a un módulo GnutWare a través de la interfase provista por éste, es decir, estableciendo una conexión TCP a un puerto conocido. Cuando una envía una solicitud de tiempo, ésta es propagada hacia todos, los cuales responderán a la misma. Esta

estrategia de difusión es propia de la red Gnutella, por lo tanto las respuestas volverán por el mismo camino que las consultas.

El protocolo de la aplicación es sencillo. La primitiva de solicitud incluye dos caracteres que denotan que se trata de una solicitud (QY) y luego la misma:

QY:TIME, PLEASE!\n

mientras que la respuesta tiene la misma estructura: dos caracteres que denotan que se trata de una respuesta y luego la hora anunciada:

QH:WED 29-11-2002 16:02:11\n

Se pueden recibir tantas respuestas como aplicaciones p2pTime hayan recibido la consulta. Téngase en cuenta que como se trata de una red de transporte de mensajes basada en Gnutella el alcance de las consultas está limitado por el horizonte definido por el valor del campo TTL del encabezado. En la implementación de GnutWare, el TTL se inicializa en 7 pero es decrementado en uno al llegar al primer Gnodo.

Finalmente, una vez recibido todos los tiempos se puede aplicar un algoritmo de promedios a los efectos de establecer el tiempo global aproximado. Esta cuestión se puede resolver utilizando los algoritmos típicos de sincronización de relojes utilizados el área de sistemas distribuidos.

III.6 Trabajos relacionados

Nuestra aplicación se enmarca dentro del concepto de red de recubrimiento (overlay network). Tales redes son sistemas que se implementan sobre Internet a los efectos de proveer una plataforma de comunicaciones que sea eficiente, puede reconfigurarse dinámicamente en pos de un buen desempeño y posea mecanismos que la hagan tolerante a

fallas. Un ejemplo popular son las redes privadas virtuales (VPN) dado que es posible configurar una red IP restringida sobre una red pública de datos.

De forma más precisa, las redes de recubrimiento se clasifican en a) Redes de distribución de contenidos, tales como Akamai [Akamai], FFnet [FFnet], y Overcast [Janotti], b) Sistemas de ruteo, tales como Xbone [Touch] and RON [Andersen] y c) sistemas de localización de recursos, tales como Tapestry [Zhao]; como Chord [Stoica] and content-addressable networks (CAN) [Ratnasamy].

Las redes de recubrimiento son una indicación positiva de que es posible construir sistemas distribuidos de gran escala que sean simples y robustos.

III.7 Consideraciones finales

El prototipo de middleware resulta una solución válida a los efectos de brindar un punto de acceso a una red P2P a diferentes aplicaciones que requieren operar en un ambiente distribuido. En este caso, el servicio que se brinda de propagación de mensajes se hace en modo broadcast, ya que la red subyacente se basa en Gnutella.

En esta arquitectura propuesta, se estima que la red de aplicación opera de forma más eficiente, ya que los nodos de los extremos no generan mensajes de exploración. Esta tarea solo la realizan los gnodos, por lo cual la sobrecarga de la red es menor. Esta situación favorece los límites de escalabilidad encontrados a Gnutella.

Capítulo IV

IV. PROCESAMIENTO DISTRIBUIDO

IV.1.1 Introducción

Actualmente existe una significativa cantidad de aplicaciones que requieren el procesamiento de grandes volúmenes de datos en un pequeño margen de tiempo. El problema es que ni los equipos más poderosos pueden satisfacer tales requerimientos. La única solución posible es la de realizar varias operaciones a la vez, llegando a la división de tareas [Torres].

Las técnicas de procesamiento paralelo permiten llevar a cabo la ejecución de varias tareas al mismo tiempo, incrementando la potencia de cálculo. La idea principal del procesamiento paralelo es la de usar más de una unidad de procesamiento autónoma a los fines de solucionar un problema computacional, en períodos menores de tiempo. De manera similar se incluye el paralelismo en otras áreas de trabajo, como por ejemplo: dado un proyecto utilizar más trabajadores entre quienes se puedan distribuir distintas tareas.

Según Hwang, Kai y Briggs, Fayé [Hwang] el procesamiento paralelo es "*como una forma eficaz de procesamiento de información, que favorece, la explotación de los sucesos concurrentes en el proceso de computación. Concurrencia implica paralelismo, simultaneidad y solapamiento. Los sucesos paralelos son los que pueden producirse en diferentes recursos durante el mismo intervalo de tiempo; los sucesos simultáneos son los que pueden producirse en el mismo instante de tiempo; los sucesos solapados son los que pueden producirse en intervalos de tiempo superpuestos. Estos sucesos concurrentes pueden darse en un sistema computador en varios niveles de procesamiento. El procesamiento paralelo exige la ejecución concurrente en el computador de muchos programas. Ello contrasta con el procesamiento secuencial. Es un medio coste-efectivo para mejorar el rendimiento del sistema mediante la realización de actividades concurrentes en el computador*".

Inicialmente, aunque podría pensarse que los conceptos de ***procesamiento paralelo y procesamiento distribuido*** pueden usarse en forma indistinta, ya que ambas técnicas requieren de más de un elemento de procesamiento que coopera en pos de la resolución de un problema, existe una diferencia entre ambos términos, debido a que, mientras en el primero las unidades de procesamiento se encuentran agrupadas dentro de una máquina, en el segundo, cada unidad de procesamiento está conformada por una computadora que a su vez se encuentra interconectada con otras.

IV.1.2 Sistemas fuertemente acoplados y débilmente acoplados

Un ***sistema fuertemente acoplado***, llamado también multiprocesador, es un sistema donde existen dos o más procesadores trabajando en paralelo, los cuales comparten una misma memoria a la que acceden mediante un bus común. Este tipo de sistemas permiten aumentar el poder de cómputo al agregar más procesadores. Dicho poder de cómputo depende básicamente de tres parámetros:

1. Capacidad del bus compartido
2. Número de procesadores
3. Utilización de la memoria compartida por parte de los procesadores

Existen dos enfoques para la utilización de la memoria compartida:

- Los procesadores acceden la memoria común para ejecutar código y obtener datos, en este caso sólo hay una memoria.

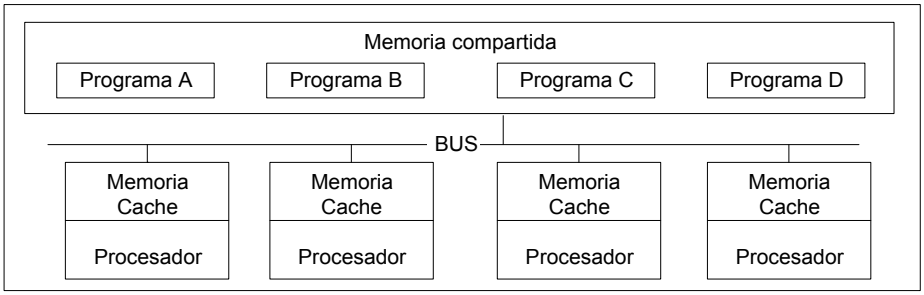
- Los procesadores poseen memoria local, y acceden a la memoria para comunicarse

En los dos casos el problema se centra en el acceso a la memoria común, debido a que se pueden producir colisiones en el acceso, esto hace que la eficiencia total del sistema

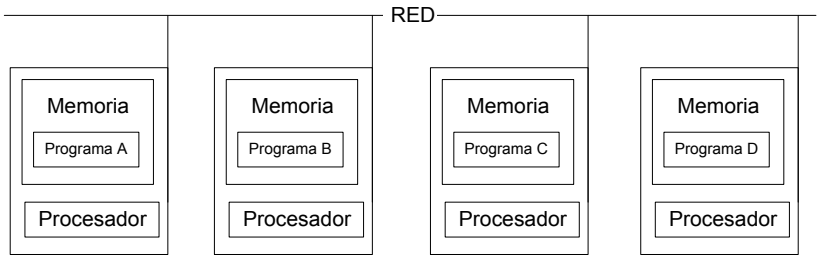
disminuya. Al aumentar el número de procesadores se incrementa la capacidad de procesamiento produciéndose al mismo tiempo un aumento en la probabilidad de colisiones en el acceso a la memoria compartida [Torres].

Por otro lado un ***sistema débilmente acoplado*** o multicomputador es un sistema donde cada procesador tiene un conjunto de dispositivos entrada/salida y una memoria local a la cual accede para obtener gran parte de sus instrucciones y datos. Los procesos que se ejecutan en los distintos procesadores se comunican en base a mensajes transmitidos por una red de datos de importante latencia.

El grado de acoplamiento en estos sistemas es débil por lo que son eficientes cuando la interacción de tareas es mínima. Un sistema débilmente acoplado es un conjunto de dos o más módulos (los módulos están compuestos por: procesador, dispositivos entrada/salida y una memoria local), los cuales se encuentran comunicados por medio de una red de datos, haciendo uso de sus dispositivos de entrada/salida.



Sistema fuertemente acoplado



Sistema débilmente acoplado

Los sistemas fuertemente acoplados se denominan también como ***sistemas paralelos*** y a los débilmente acoplados se los conoce como ***sistemas distribuidos***. A partir de lo anterior se ve que el procesamiento paralelo puede ser bajo un sistema fuertemente acoplado, siempre y cuando los procesadores compartan la memoria; porque puede darse el caso que en una misma máquina con más de un procesador, estos posean memoria local y se comuniquen a través de una red de pasajes de mensajes, en ese caso será débilmente acoplado.

IV.1.3 Sistemas distribuidos

A los efectos lograr procesamiento distribuido se necesita contar con un sistema distribuido, donde los procesos están ubicados en distintas computadoras, y el intercambio de información se realiza solamente por pasaje de mensajes sobre algún sistema de comunicación. Su propia esencia hace que al compararse con los sistemas centralizados, mono o múltiple procesador, surjan determinadas situaciones o problemas, debidos a la propia distribución: latencias o retardos, particiones de la red, fallo independiente de máquinas, seguridad, etc.

En trabajos iniciales de investigación sobre sistemas distribuidos Enslow [Enslow] los caracterizó de la siguiente forma:

- Su hardware y software son de propósito general.
- Los recursos están distribuidos geográficamente y se vinculan por una red de datos.
- Un sistema operativo unifica e integra el control de los distintos componentes.
- Hay transparencia en la distribución. Se accede a los servicios a través de sus nombres.

- Hay una autonomía coordinada que regula el funcionamiento de todos los componenetes físicos y lógicos.

Para Tanembaum [Tanenbaum] "un sistema distribuido es aquél al que sus usuarios ven como un ordinario sistema operativo centralizado; sin embargo, se ejecuta en diferentes e independientes CPUs. El concepto clave aquí es la transparencia; en otras palabras, el uso de diversos procesadores deberá ser invisible (transparente) al usuario. Otra forma de expresar esta misma idea es diciendo que el usuario verá al sistema como un uniprocesador virtual y no como una colección de máquinas diferentes".

Para Schroeder [Schoeder] las características de un sistema distribuido se pueden enuniciar de la siguiente forma:

- Existencia de varias computadoras autónomas.

- Existencia de una red de interconexión.

- Se mantiene un estado compartido entre las computadoras que pertenecen al sistema.

Actualmente, según Coulouris [Coulouris] tales conceptos siguen estando vigentes, indicando que "un **sistema distribuido** es uno en el cual los componentes localizados en una red de computadoras se **comunican y coordinan** sus acciones sólo por **pasaje de mensaje**s"

Por otro lado, el procesamiento paralelo es una técnica que se ha desarrollado rápidamente; gracias a la evolución de la tecnología computacional es posible tener hoy en día una computadora de escritorio con más de un procesador a un costo razonable. Por otro lado la tecnología de comunicaciones ha permitido comunicar múltiples computadoras uniprocesadores y tener una red de cómputo distribuido extremadamente eficiente. Debido

al avance de las comunicaciones de datos cvada vez más la distinción entre ambos tipos de procesamiento se hace más estrecha. En base a lo anterior una visión del procesamiento distribuido se da en verlo como una manera de procesamiento paralelo en un entorno especial.

IV.1.4 Sistemas distribuidos vs centralizados o paralelos

Ventajas	Desventajas
Economía: Clusters versus mainframes, siendo obviamente los primeros más económicos para un nivel inicial de trabajo.	Existe poco software para desarrollarlos.
Mayor perfomance. Proyectos de cómputo masivo distribuido, tales como SETI, han demostrado la potencia de cálculo que se puede juntar en poco tiempo.	***Fallas***. Las redes tienen congestión, errores y retardos
Distribución geográfica coherente de recursos. Se permite que los recursos se sitúen coherentemente donde deben estar. Por ejemplo, el Sistema Free-Flow de Akamai permite replicar y mover contenidos hacía donde se sitúan los usuarios, ganando así una mayor perfomance en la navegación sobre el espacio web.	***Seguridad.*** Al ser distribuidos, las computadoras que lo componen están expuestas a ataques de denegación de servicios o intrusiones por parte de hackers.
Confiabilidad. Su propia característica	

de distribuido hace que en distintos puntos geográficos puedan replicarse recursos y así atender posibles contingencias.	
Crecimiento por incrementos. Es posible, a bajo costo, incorporar nuevos recursos de hardware y software.	

Completando la caracterización anterior se presentan, según Schroeder [Schoeder], los problemas más relevantes en los sistemas distribuidos:

- **Fallos independientes**. Algunos componentes pueden desafectarse por fallas y dejar de funcionar total o parcialmente. El sistema por su naturaleza debe continuar presentando servicio.

- **Comunicación no confiable**. En la red que funciona como infraestructura del sistema distribuido pueden haber fallas. La información en tránsito puede retrasarse, perderse o duplicarse.

- **Comunicación insegura**. La red puede sufrir inrusiones donde se capturen o modifiquen los datos en trànsito.

- **Alto costo de comunicación**. Esto es contrastado contra la comunicación por bus, la cual es más econòmica y fiable.

IV.1.5 Transparencia, un concepto clave.

Es una de las características más interesantes de los sistemas distribuidos; y consiste en lograr la imagen de un único sistema, es decir; los usuarios usan el sistema como un sistema centralizado, sin saber que en realidad es un conjunto de ellos, unidos por medio de una red. En los sistemas distribuidos existen una serie de áreas en las que se debe aplicar la transparencia [Coulouris]:

- Transparencia en el acceso. Hace referencia a la posibilidad de que un usuario o un proceso obtengan recursos sin importar, ni darse cuenta, de que sean locales o remotos.

- Transparencia en respuesta. Se plantea la posibilidad de que se puedan utilizar múltiples recursos para asegurar respuesta a necesidades concretas, esta característica aumenta la confiabilidad del sistema.

- Transparencia en ubicación. Cualquiera de sus recursos puede ser accedido desde cualquier ubicación geográfica de la misma forma.

- Transparencia en concurrencia. Cualquier usuario debe poder operar al mismo tiempo sin que el trabajo de uno interfiera en otro.

- Transparencias en fallas. Debe evitarse que se pierdan recursos de software de usuarios o del sistema a pesar que fallas en computadoras individuales o en enlaces de datos sucedan.

- Transparencia en migración. Se permite el movimiento de objetos del sistema (archivos, servidores, procesadores, librerías) sin que tales cambios afecten a usuarios ni aplicaciones.

- Transparencia en rendimiento. El sistema debe ofrecer flexibilidad en cuanto a modificaciones en su rendimiento sin que sus usuarios y ni aplicaciones se vean afectados.

- Transparencia en escalabilidad. Que el sistema sobre la base de requerimientos de más recursos pueda incrementarse, y opcionalmente decrecer, sin afectar a usuarios y aplicaciones.

IV.1.6 Aplicaciones de los sistemas distribuidos

Para Singhal [Singhal] las áreas de uso de los sistemas distribuidos son: a) aquellas en la distribución es un medio para conseguir un fin y b) aquellas donde no hay otra opción.

Para el primer caso se dan los siguientes escenarios:

- Computación masiva paralela, de propósito general.
- Tolerancia a fallos.
- Respuesta a demandas bajo un esquema de tiempo real.

Y para el caso b, el escenario de trabajo determina la solución distribuida, por ejemplo: sistemas militares, bases de datos distribuidas, sistemas de control de plantas de producción, entre otros.

El procesamiento paralelo como así el distribuido, poseen extensos campos de aplicación en varias áreas del conocimiento, por ejemplo: biología, matemáticas, química, medicina, tecnología militar, inteligencia artificial, climatología, criptografía, entre otras. A continuación se brindan ejemplos de utilización del procesamiento paralelo y distribuido en cuatro categorías [Torres]:

IV.1.6.1 Modelado predictivo y simulaciones

El modelado predictivo es una tarea que requiere de importantes recursos de cálculo dado que se basa en experimentos de simulación. Los pronósticos datmosféricos son un caso ejemplo de una tarea de modelado predictivo. En general se utiliza un modelo tridimensional geográfico al cual se le añade la cuarta dimensión que representa al tiempo. Nótese que la cantidad de puntos a evaluar es importante, de allí deriva su necesidad de recursos.

El cómputo masivo tiene importancia en oceanografía, en particular se estudian los procesos de transmisión de energía hacía y desde la atmósfera, con fines de predecir fenómenos atmosféricos, mareas, migraciones de peces, erosión de costas, entre otras áreas de aplicación.

Otro campo de estudio es la investigación espacial, en particular la simulación de los fenómenos que ocurren bajo fuerzas gravitacionales.

La socioeconomía requiere de poder de cálculo cuando se analizan fenómenos econométricos, modelos de consumo o de preferencias, minería de datos, encuestas de opinión pública y censos de diversa índole.

IV.1.6.2 Diseño y automatización de proyectos de ingeniería

El área de diseño en ingeniería, que se basa en el análisis de elementos finitos, es un campo de acción donde se hace necesario tener potencia de cálculo. Diseñar y visualizar distintas construcciones u objetos (aviones, edificios, puentes, canales de agua, entre otros) son acividades de alto costo computacional.

Entre otras áreas de aplicación se encuentra la percepción remota, la cual procesa grandes volúmenes de datos (obtenidas de radares o satélites metereológicos o de evaluación de recursos naturales) y generalmente necesita obtener productos que brinden información en tiempo real.
La automatización industrial o de viviendas requiere de poder de cómputo a los eefctos de gerencias instalaciones en tiempo real.

IV.1.6.3 Exploración de recursos energéticos

La búsqueda de yacimientos gasíferos y petrolíferos son áreas que se basan en trabajar con importantes cantidades de datos adquiridas por sensores de forma automática. Las características de su procesamiento, donde existe la necesidad de contar información en tiempo real, requieren de equipos poderosos.

El estudio de sismos a partir de tareas como la evaluación de daños y la predicción de próximos desastres son áreas propicias para la tecnología de computo masivo.

IV.1.6.4 Investigación médica y militar

La ingeniería genética está brindando nuevas soluciones que tienden a mejorar la calidad de vida de las personas. La bioinformática es el área de la biología que usa la informática con el fin de organizar, analizar y distribuir información biológica a los efectos de responder preguntas complejas.

La industria militar necesita poder de cómputo a los efectos de diseñar armamento y estrategias de defensa. Los juegos de guerra, los sistemas de información geográfica, el cómputo de daños prinicpales y colaterales derivados de supuestos ataques, con ejemplos de tareas computacionalmente complejas.

Se presentan un conjunto de de direcciones de sitios web de proyectos relacionados con el procesamiento masivo distribuido y áreas de aplicación como las que han sido descriptas anteriormente:

Área	Proyecto	URL
Medicina	Investigación sobre la enfermedad gripe	http://www.popularpower.com/

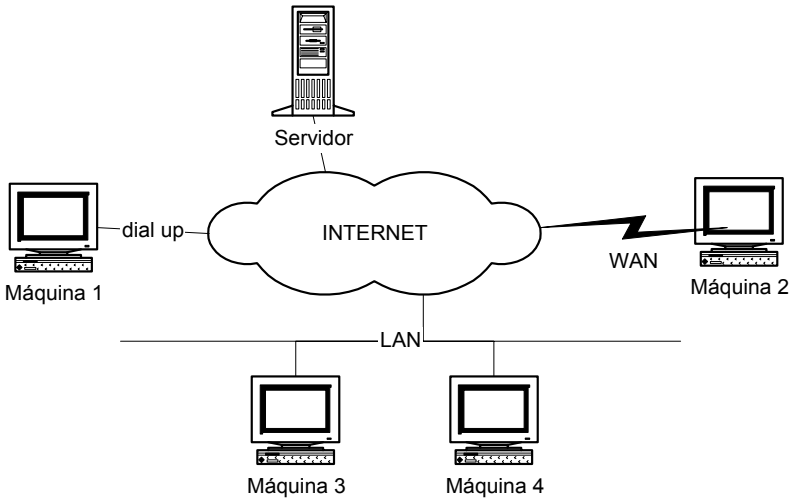
Medicina	Investigación sobre SIDA	http://www.fightaidsathome.com/
Medicina	Investigación sobre cáncer	http://www.computeagainstcancer.org/ http://www.ud.com/
Biología	Estudios moleculares y bioingeniería	http://www.stanford.edu/group/pandegroup/Cosm/
Biología	Biología evolutiva	http://www.evolutionary-research.net/
Física		http://www.jansson.net/
Matemáticas		http://www.entropia.com/
Meteorología	Predicción climática	

IV.2 Experiencias en procesamiento distribuido masivo en Internet

IV.2.1 Conceptos

El procesamiento distribuido en Internet, también llamado grid, se basa en la utilización de la infraestructura que provee tal red, haciendo uso de sus facilidades de comunicación y máquinas interconectadas; a los efectos dea distribuir trabajo de procesamiento entre distintos nodos, generalmente computadoras de usuario final, a los que se les asigna una pequeña parte de un problema mayor para contribuir a su solución.

En este esquema, es necesario la utilización de servidores unidades de trabajo que se encarguen de enviar estas pequeñas porciones del problema y recibir los resultados, para luego unirlos y obtener la solución. Esto puede verse en la siguiente figura:



Procesamiento distribuido en Internet

El procesamiento masivo distribuido es una realidad, dos proyectos en Internet han probado que tal modo de trabajo opera sumamente bien, incluso mejor de lo que muchos expertos proyectaron a principio de los años 90. El primero de estos proyectos actualmente utiliza una gran cantidad de computadoras en toda la red y fue llamado Distributed.net [UD], destinado a desencriptar claves. El segundo, SETI@Home [Korpela], que es el más exitoso y popular de los proyectos de computación distribuida, y cuenta con la mayor cantidad de voluntarios; el cual está orientado hacia la búsqueda se señales extraterrestres. Ambos proyectos demostraron que el procesamiento masivo distribuido puede acelerar los resultados disminuyendo costos asociados; siendo una alternativa claramente superior ante la utilización de equipos fuertemente acoplados de alto poder de cálculo, a la hora de lograr mayor potencia de procesamiento.

Se estima que alrededor del mundo hay más de 400 millones de computadoras conectadas en Internet [Foster], y es común que dichas computadoras muy pocas veces llegan a utilizar en plenitud todo su poder de procesamiento y la mayoría del tiempo se encuentran sin uso, a la espera de alguna tarea para ejecutar. Es esta potencia la que se encuentra disponible a la espera de ser usada en algún proyecto de procesamiento masivo distribuido.

El término grid se deriva del hecho que se apuesta por la construcción de un escenario informático ideal en el que ciclos de CPU y espacio de almacenamiento de millones de sistemas informáticos distribuidos a lo largo del mundo, funcionen como una reserva [Buyya] [Foster-A]. Tales recursos serán aprovechados por aquellso que necesiten potencia de cálculo o tal vez almacenamiento masivo, esto se parece a como las compañías de provisión de electricidad y usuarios comparten la misma red eléctrica.

Grid presenta una analogía con el nombre que se le da a la red eléctrica, dado que en Internet se pretende dar la capacidad para que los procesos obtengan recursos de procesamiento de la propia red, bajo la misma forma que un artefacto obtiene energía de la red eléctrica.

IV.2.1.1. Principios generales

Según Baker [Baker] los principios generales del procesamiento distribuido en Internet son la heterogeneidad, escalabilidad, dinamismo y adaptabilidad, deben ser tenidos en cuenta en la construcción de algún proyecto de este tipo.

- *Heterogeneidad:* un proyecto de procesamiento distribuido en Internet debe abarcar una gran variedad de recursos heterogéneos, de manera que pueda incluir la mayor cantidad de combinaciones de sistemas operativos y hardware disponibles en Internet.

- *Escalabilidad*: Un proyecto de procesamiento distribuido en Internet debe tener la capacidad de crecer desde pocas máquinas hasta millones, por lo que hay que tener en cuenta los problemas que puedan generase en las comunicaciones y en la capacidad de trabajo del servidor al aumentar la cantidad de máquinas o unidades de procesamiento.

- *Dinamismo y adaptabilidad:* Un proyecto de procesamiento distribuido en Internet debe adaptar dinámicamente su funcionamiento de acuerdo a los recursos con que cuenta, para obtener una máxima performance ya sea en la velocidad como

en la veracidad de los resultados. Estos recursos o máquinas a menudo fallan, esto hace que los resultados nunca lleguen al servidor o lleguen erróneos, ya sea por fallas en las propias máquinas, en la comunicación o por culpa de intrusos. Por lo que para solucionar estos problemas será necesario determinar un tiempo de espera para los resultados, plantearse un esquema de seguridad para controlar dichos resultados y la utilizar cálculo redundante para chequeo.

IV.2.2 Antecedentes

A principio de 1970 cuando las computadoras se empezaron a conectar en red, nació la idea de agrupar ciclos de CPU no utilizados. Los primeros experimentos con computación distribuida fueron los programas llamados Creeper y Reaper que corrían sobre la red predecesora de Internet, ARPANET.

Creeper era un programa tipo gusano, que utilizaba ciclos ociosos de procesadores disponibles en ARPANET, se replicaba y migraba a otro procesadores, borrando la instancia original. Reaper fue una especie de antigusano cuya misión era detectar a Reaper y eliminarlo. Creaper y Reaper fueron los primeros programas que actúan por infección, y son considerados como los primeros virus de la red. No hicieron ningún daño, sin embargo ayudaron a pensar las posibles bondades del procesamiento masivo distribuido.

En los laboratorios de investigación de XEROX (PARC), en 1973, se instaló la primer red de área local Ethernet y se puso en ejecución el primer esfuerzo completo de procesamiento distribuido. Este primer programa [Shoch], el cual fue una creación de los científicos de Shoch y Hupp, operaba sobre cien computadoras conectadas sobre la red local. Sus autores lo consideraban como un gusano el cual se movía de computadora a computadora utilizando los recursos desocupados para llevar a cabo algún tipo de tarea de cálculo. Dicho programa funcionaba sobre toda la red del laboratorio, se replicaba en cada memoria de cada máquina, utilizaba los recursos disponibles, se reproducía y transmitía clones a los otros nodos conectados a la red.

En otro esfuerzo en cómputo distribuido lo realizó Crandall, investigador de Apple, el cual utilizó los recursos de procesamiento disponibles en las máquinas de la red NeXT para llevar a cabo un trabajo en conjunto. Para esto Crandall instaló un software que permitía a las máquinas ejecutar cálculos y combinar los esfuerzos con otra máquina de la red siempre y cuando no estuviesen en uso. Su software llamado Zilla, en un primer momento estuvo enfocado en el hallazgo de grandes números primos, y luego siguió con pruebas de modelos de encriptación. Dicho software, en 1991, ganó el premio a la ciencia dado por Computerworld.

IV.2.3 Modo de operación

A los efectos de construir un sistema de procesamiento distribuido en Internet lo primero que se debe tener en cuenta es la descomposición de un problema inicial entre varios subproblemas independientes. Por ejemplo en los proyectos relacionados con encriptación de datos el problema principal consiste en encontrar una clave que permita acceder a los datos encriptados, para esto es necesario probar todo un conjunto de claves una por una. Cada clave puede ser probada independientemente de las demás, de manera que dichas claves pueden repartirse entre varias computadoras para chequear cual es la correcta y de esa forma disminuir el tiempo de procesamiento.

Supóngase que se tienen dos máquinas y las claves sólo pueden estar conformadas por números, repartiendo los pares a una máquina y las impares a otra es posible dividir el tiempo necesario para chequear todas las claves por dos.

Ahora para trasladar esta forma de trabajo a Internet es necesario utilizar un esquema cliente – servidor, para que el servidor sea el encargado de distribuir el trabajo y los clientes de efectuarlo, dichos clientes pueden ser computadoras personales conectadas a Internet. Por ejemplo en el proyecto de chequeo de claves RC5-56 de Distributed.net se tienen 2^{56} claves para probar, las cuales se reparten entre 2^{28} paquetes o bloques de 2^{28} claves cada uno. Una vez determinado el problema y su división en subproblemas es necesario contar con una computadora con conexión permanente a Internet, la cual hará el papel de servidor en la que

se instalará un programa que distribuirá los bloques de claves a demanda. Cada participante necesitará contar con una máquina conectada a Internet con el software cliente, el cual será el encargado de conectarse con el servidor y requerir un bloque de claves para luego chequearlas. Una vez finalizado el chequeo, éste se conectará nuevamente con el servidor para devolver el resultado, por ejemplo "he encontrado la clave" o "No he encontrado la clave", y también para requerir un nuevo bloque de claves. El servidor además de la tarea descripta anteriormente, debe mantener la información sobre los bloques chequeados y sobre aquellos bloques que han sido enviados pero que todavía no se tienen respuesta. También, en el caso de que solamente aquellas personas que estén registradas en el proyecto puedan hacer uso del cliente, es posible que el servidor también mantenga estadísticas actualizadas sobre: quien el más rápido, quien ha chequeado más claves y cuántos usuarios están participando del proyecto.

Es importante que el software cliente tenga la posibilidad de funcionar sólo cuando el usuario no está utilizando la máquina, para que no haga que los demás programas funcionen mas lentos e interfiera en el uso normal de las aplicaciones que utiliza el dueño de la computadora. También el cliente debe soportar la mayor cantidad posible de sistemas operativos ya que Internet se caracteriza por su gran heterogeneidad.

IV.2.4 Ventajas

El procesamiento masivo distribuido en Internet tiene varias ventajas que lo hacen una opción a tener en cuenta a la hora de implementar un proyecto que requiera: poder de cómputo, costos reducidos y resultados en tiempos cortos. A continuación se detallan las ventajas más importantes.

La utilización de ciclos de CPU ociosos sin uso es lo que destaca en este esquema; aún cuando una computadora esté aparentemente trabajando, en realidad la mayoría del tiempo no está aprovechado todo su poder computacional, por ejemplo, una persona al escribir, a lo sumo puede presionar 10 teclas por segundo, ésto no es mucha distracción para un procesador que puede ejecutar 100 millones de instrucciones por segundo, en este caso el

procesador gasta más tiempo preguntando que tiene que hacer dentro de un ciclo. Además son muchas las oficinas en todo el mundo donde sus computadoras con conexión a Internet permanecen encendidas toda la noche. Este es el poder que está siendo derrochado y que puede ser usado para llevar a cabo algún tipo de investigación haciendo uso del procesamiento distribuido.

Otra ventaja es que al hacer uso de Internet, las unidades de procesamiento y la comunicaciones ya existen, esto permite reducir drásticamente los costos siendo muy beneficioso frente al uso de grandes equipos de procesamiento paralelo, ya que lo único que se necesita es una máquina que funcione como servidor además del software cliente que deberán usar los dueños de las máquinas que estén conectadas a Internet.

También algo importante para destacar es que este esquema es escalar, ya que permite la incorporación de todas las máquinas que se deseen sin la necesidad de realizar ningún tipo de modificación importante. Son millones las computadoras que se encuentran conectadas actualmente a Internet, el procesamiento distribuido es una alternativa para unir todo el poder de cálculo de las mismas hacia la búsqueda de un objetivo.

IV.2.5 Restricciones

El procesamiento masivo distribuido sobre Internet tiene sus propias restricciones y obstáculos que deben ser tenidos en cuenta. Una de las principales barreras a considerar es convencer a los usuarios anfitriones a prestar los recursos de sus computadoras para ser usados en algún proyecto de procesamiento distribuido, ya que no a todas las personas les gusta que le utilicen recursos de su máquina por temor a que ésta baje el rendimiento. Una alternativa a esto es la utilización de la llamada "computación parásita" [Freeh], esto se refiere a la utilización de la máquina sin conocimiento, ni consentimiento del dueño, simplemente haciendo uso en forma directa de algún recurso.

Otro punto a tener muy en cuenta es el ancho de banda, el cual actualmente es muy limitado haciendo que la comunicación entre máquinas tenga una importante latencia, esto obliga a

que solo determinados tipos de algoritmos puedan funcionar en este esquema. Por ejemplo un algoritmo de simulación de movimiento de partículas en un campo de fuerza, como una estrella o galaxia, en donde se asigna una partícula a cada máquina y que para analizar una partícula es necesario consultar con las otras máquinas, se hace imposible su funcionamiento eficiente en Internet debido a la lentitud de la comunicación. O sea que no es recomendable utilizar en este esquema aquellos algoritmos que requieran que las unidades de procesamiento se comuniquen entre ellas, siendo imposible, sobre todo, si la comunicación necesita ser muy fluida. Los algoritmos que más se adecuan a este esquema son aquellos de granularidad gruesa en cual cuenta con un número elevado de operaciones de cómputo y como consecuencia tiene pocas operaciones de comunicación.

La capacidad de memoria es otra seria restricción. Los dueños de computadoras que deseen donar ciclos de su CPU, pueden no estar de acuerdo que alguien les ocupe gran parte de su memoria o disco de su máquina. Por lo que es recomendable que el cliente haga un uso racional de los recursos, siendo lo más conveniente hacer que funcione solamente cuando la máquina no está en uso.

Un obstáculo a considerar es la gran diversidad de máquinas y sistemas operativos a que hay en Internet, esto obliga a que el cliente abarque una amplia variedad de combinaciones, haciéndose más difícil su construcción, siempre y cuando se quiera abarcar la mayor cantidad de potenciales unidades de procesamiento.

IV.2.6 Proyectos

Son muchos los proyectos de diversa índole que hacen uso de los beneficios del procesamiento distribuido en Internet. Entre ellos se pueden nombrar, dentro de los relacionados con la medicina, el de Popular Power que en este momento trata de ayudar a encontrar vacunas contra la gripe, el proyecto FightAIDS@Home relacionado con la enfermedad del SIDA y el proyecto Compute Against Cancer que fue la primera iniciativa que utilizó cómputo distribuido en la lucha contra el cáncer. También hay proyectos sobre biología molecular y bioingeniería como Folding@Home y Genome@Home. Otros como

Evolution@Home se orientan hacia la biología evolutiva. En la rama de las ciencias del espacio el más importante es SETI@home y por el lado de la criptografía se destaca Distributed.net. En otras ramas es posible encontrar proyectos sobre matemáticas como GIMPS, en Física como Gamma Flux y sobre predicción climatográfica como Casino-21.

IV.2.6.1 SETI@home

El proyecto SETI (Search for extraterrestrial Intelligence) tiene como objetivo la búsqueda de vida inteligente externa al planeta tierra, basándose en el análisis de análisis de señales de radio extraterrestres [Korpela]. Las fuentes naturales al emitir señales de radio ocupan anchos de banda de gran tamaño, por otro lado las señales generadas por el hombre (la televisión y la radio entre otras) tienen un ancho de banda estrecho; en base a lo anterior se deduce que una señal con ancho de banda estrecho proveniente fuera del sistema solar sería un posible indicio de vida extraterrestre. La búsqueda de SETI@home asume que una civilización extraterrestre desea hacer contacto con otras especies emitiendo señales que son fácilmente detectables y distinguibles de emisiones naturales y de radio [Anderson].

El proyecto SETI@home difiere de otros en cuanto a la sensibilidad, cobertura de frecuencia y cobertura del cielo, sin embargo sus análisis de datos tienen una forma similar:

- Calcula el espectro de potencia de la señal sobre el tiempo
- Aplica patrones de reconocimiento al espectro de potencia para identificar señales candidatas
- Elimina señales candidatas que puedan ser probablemente naturales o hechas por el hombre

SETI@home usa computadoras conectadas a la red Internet para llevar a cabo los cálculos de análisis de señales. Desde su inicio en el año 1997 hasta mediados del año 2001 acumuló más de 3 millones de usuarios participantes.

En 12 meses a partir de julio del 2000, los participantes de SETI@home han procesado 178 millones de unidades de trabajo [Anderson].

Una de las características de SETI@home es que posee una tasa fija procesamiento de tareas; mientras que los datos son registrados a una tasa constante, los tiempos de CPU por unidad de datos son apenas constantes, esto hace que si las tareas de cálculo reciben poca potencia de CPU, la cola de unidades de datos a procesar crece, disminuyendo la velocidad global del sistema. Pero si hay demasiada potencia de CPU disponible el proyecto cuenta con dos elecciones: rechazar a los participantes o procesar cada unidad de datos repetidas veces a los eefctos de verificar la calidad de los resultados obtenidos, como una tarea de auditoría a los efectos de prevenir posibles fraudes o resultados provenientes de procesadores defectuosos.

Las tareas de cálculo que implementa SETI@home tienen varias propiedades que las hacen ideales para el procesamiento con recursos públicos: [DN]

- Alta proporción de cálculo de datos: cada unidad de datos toma 3.9 trillones de operaciones de punto flotante. En una Pentium II 500 el cálculo demora aproximadamente 10 horas. El proceso de recuperación y envío de datos desde el servidor involucra bajar 350Kb y subir 1Kb, a un módem de 56Kb le toma alrededor de 60 segundos, siendo un pequeño inconveniente para los participantes.

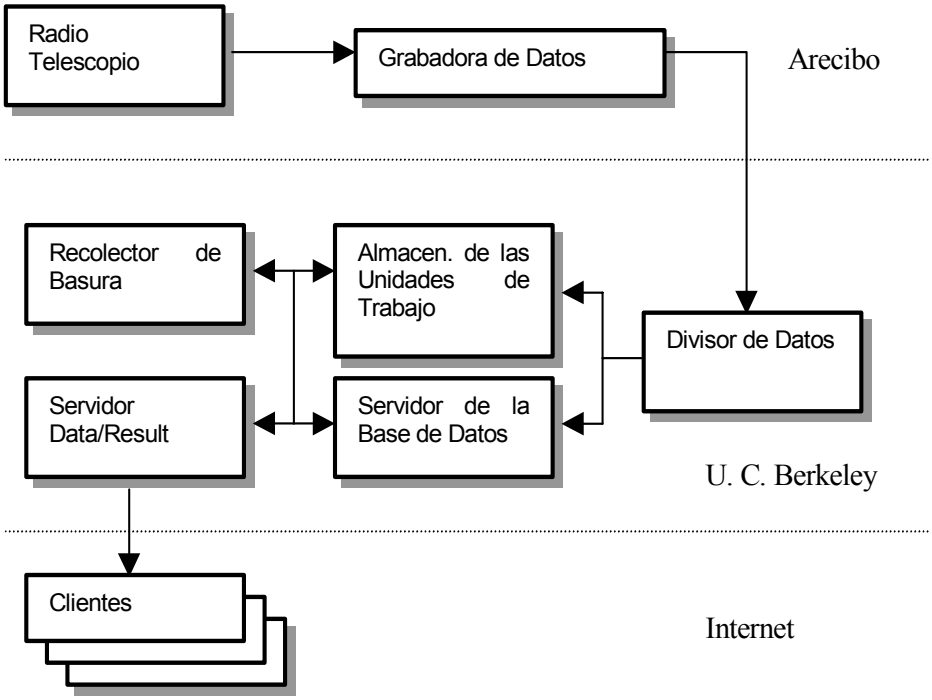
- Paralelismo independiente: las unidades de cálculo son independiente unas de otras, de esta manera no es necesario comunicación entre computadoras pares de usuario. Si una computadora falla mientras procesa una unidad de trabajo, ésta es eventualmente asignada a otra computadora.

- Tolerancia de errores y omisiones: Si una unidad de datos se analiza incorrectamente solo afecta levemente al objetivo global. La omisión es remediada la próxima vez que el telescopio rastree el mismo punto en el cielo.

Los datos se reciben utilizando el radio telescopio de Arecibo, situado en Puerto Rico. La fuente de datos de la cual hace uso SETI@home, una la banda de frecuencia de 2.5 MHz centrada en la línea de hidrógeno (1.42 GHz), es grabada digitalmente para su posterior procesamiento distribuido. Luego, en la Universidad de Berkeley, los datos son divididos en unidades de trabajo, separando las señales en 256 bandas de frecuencia, cada una alrededor de 10 KHz, por otro lado, cada segmento o banda se divide en trozos de 107 segundos, estas unidades de trabajo están superpuestss por 20 segundos, lo caul asegura que cada período de emisión éste contenido enteramente. Las unidades de trabajo poseen el tamaño aproximado de 350 Kb. Una computadora de escritorio típica puede estar procesándolas por varias horas.

Los clientes toman las unidades de trabajo de un servidor "data/result", el cual usa un protocolo basado en HTTP a fin de que los clientes detrás de sistemas de seguridad puedan establecer contacto con el servidor.

El sistema de distribución de datos se observa en el siguiente gráfico:



Sistema de distribución de datos de SETI@home

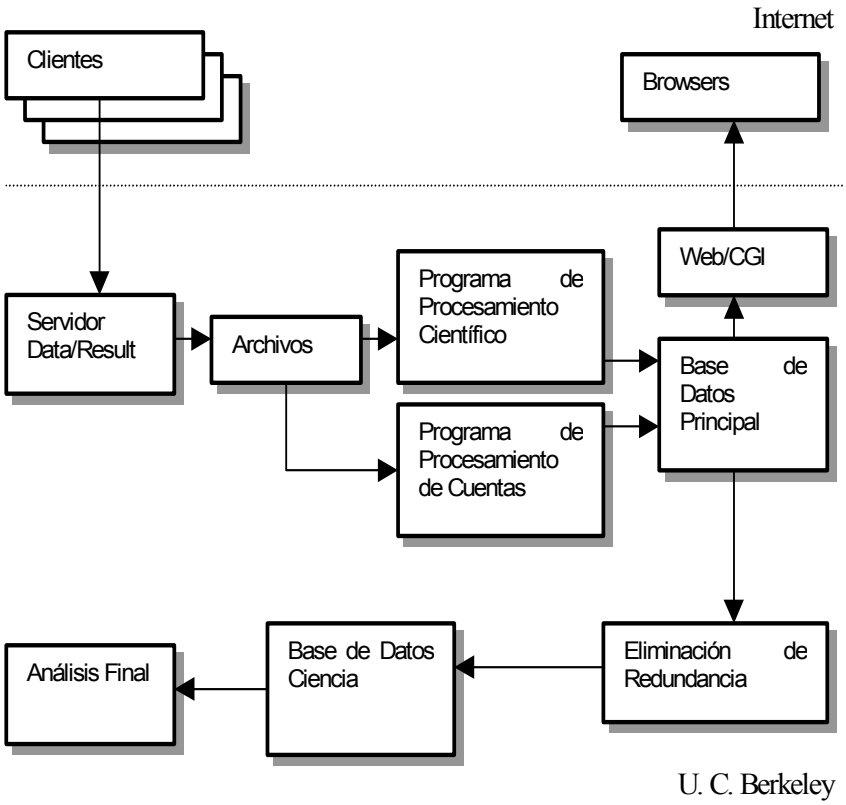
El programa cliente de SETI@home (http://setiathome.ssl.berkeley.edu) esta disponible para ser utilziado en diversas plataformas de hardware y software. En los sistemas operativos Microsoft Windows y System de Apple Macintosh, el cliente se instala por defecto como un protector de pantalla, que procesa datos solamente cuando está activo. En otras plataformas, el cliente corre en segundo plano.

El programa cliente, en forma repetitiva, busca una unidad de trabajo desde el equipo servidor data/result, la analiza y retorna el resultado que es una lista de señales candidatas. El cliente necesita una conexión a Internet solo mientras se comunica con el servidor, en la etapa de análisis el equipo no necesita tener conexión a la red. El programa periódicamente guarda su estado en un archivo en disco y lee este archivo cuando comienza a funcionar, de esta manera lleva a cabo su progreso soportando que el equipo sea puesto fuera de funcionamiento.

El programa cliente está escrito en lenguaje C++ y su estructura está compuesta de cuatro partes: Una plataforma independiente para procesamiento distribuido, componentes para

cada plataforma (por ejemplo librerías gráficas), código de análisis de datos y código de visualización de estado de proceso.

Las tareas que complementan el esquema de procesamiento SETI se ven el el siguiente gráfico:



Sistema de recolección y análisis de resultados de SETI@home

El cliente, al terminar de procesar cada unidad de trabajo, envía los resultados al servidor de captura de resultados (data/result). Este graba los resultados en un archivo. Luego un programa de procesamiento lee este archivo y graba los resultados (señales halladas) para cada unidad de trabajo en una base de datos. Para cada resultado obtenido el servidor agrega al archivo de log una entrada cuya función es describir los resultados del usuario. Un programa de procesamiento de cuentas lee este archivo de auditoría y acumula en una memoria cache las actualizaciones a todos lo registros relevantes de la base de datos.

A continuación se verifica el resultado donde se examinan todos los resultados redundantes para cada señal. Estos resultados son almacenados en una base de datos separada.

Aplicando filtros las señales generadas por humanos son detectadas y borradas, a continuación las señales candidatas de frecuencia similar junto con otras que posean las mismas coordenadas del espacio pero en tiempos diferentes o que tengan suficientes méritos, son identificadas para un proceso especial en forma posterior. [Anderson]

IV.2.6.2. Distributed.net

Distributed.net [DN] es un proyecto nacido en el año 1997 y que ha logrado la colaboración de miles de personas las cuales aportan recursos de sus computadoras para asistir con potencia de cálculo al proyecto. En el año 2001 su potencia de cálculo equivalía al trabajo de 160.000 PC PII 266 Mhz., trabajando 24 horas al día, los 7 días de la semana, 365 días al año.

El proyecto Distributed.net tiene por fin proporcionar herramientas, medios de coordinación y comunicación válidos para conducir cálculos basados en el cómputo paralelo en busca de la solución a problemas de gran porte.

Los usuarios que se incorporan al proyecto necesitan descargar un programa cliente el cual será el encargado de comunicarse a la red de distributed.net y que procesará información de proyectos vigentes.

Los proyectos iniciales se orientaban a la búsqueda de claves en mensajes cifrados por algoritmos RC5 y DES. Trabajo encargado por la empresa RSA Security, la cual ha sido quien ha lanzado estos retos para probar la eficiencia de dichos algoritmos. Tal tarea se basa

en el uso de la fuerza bruta, donde las computadoras de la red de distributred.net prueban todas las claves posibles hasta encontrar la correcta.

Un vez que el cliente finalice de analizar las claves, enviará sus resultados a un servidor de captura de resultados, él cual le entregará otra unidad de trabajo a procesar. [DN-A].

Los desafios o proyectos más significativos que se hosperadron en la infraestructura Distributed.net fueron:

- Desafío Cs-Cipher: este reto fue organizado por CS Communication y Sistems. Distributed.net encontró la clave para desencriptar el mensaje, el 16 de enero de 2000 después de testear mas del 98% del total de claves posibles en menos de dos meses. Este desafío fue importante para demostrar la debilidad de la clave de 56-bits contra la fuerza bruta. [DN-B]

- DES III: Este proyecto comenzó el 13 de enero de 1999, luego de 22,5 horas con la ayuda de EFF's Deep Crack, Distributed.net finalizó exitosamente con el desafío. [12]

- DES II-1: Este desafío fue un certamen cronometrado desde los laboratorios de RSA, comenzó el 13 de enero de 1998 y concluyó el 24 de febrero de 1998 obteniéndose la clave 76 9E 8C D9 F2 2F 5D EA cuyo mensaje asociado contenía el siguiente texto: "Muchas manos hacen el trabajo". [DN-B]

- RC5-56: Finalizó el 22 de octubre de 1997 al descubrir la clave 0x532B744CC20999, obteníendose el siguiente mensaje en texto plano: "Es tiempo de migrar a una longitud de clave más larga".

Actualmente se está trabjando en el proyecto RC5-64, cuyo objetivo es descifrar la clave de de 64-bits de RSA Security Inc. Se pretenden testear 2^{64} (18.446.744.073.709.551.616) claves. El proyecto consumirá una enorme cantidad de potencia computacional para lograr su

cometido. El ritmo de procesamiento es de aproximadamente de 127,24 billones de claves testeadas por segundo. [DN-B], [DN-C]

URL del proyecto http://www.distributed .net

IV.2.6.3 Popular Power

Popular Power es una empresa fundada en enero del 2000 que tiene por objetivo gerenciar una red de procesamiento masivo distribuido destinada a cooperar en proyectos de investigación científica. Dispone de un programa cliente del que existen versiones para Microsoft Windows, Linux y Apple Macintosh, destinado a trabajar en proyectos de diversa índole comercial y no comercial. El proyecto más popular fue el relacionado con la creación de mejores vacunas contra la gripe. Otra línea de investigación utilizando Popular Power es la relacionada con el desarrollo de una vacuna contra el SIDA.

URL del proyecto http://www.popularpower.com

IV.2.6.4 Fightaids@Home

Lanzado en Septiembre del 2000, está orientado al diseño de agentes antivirales a favor de la lucha contra el virus del SIDA, es conducido por la empresa Entropia Inc. y los Laboratorios Olson. A mediados del año 2001, este proyecto contaba con la colaboración de aproximadamente 29.000 computadoras que cedieron más de 4.200.000 horas de CPU.

URL del proyecto http://www.fightaidsathome.com

IV.2.6.5 Compute Against Cancer

La empresa Parabon Computation Inc. tuvo a cargo llevar adelante el proyecto Compute Against Cancer que se basó en una insfraestructura distribuida de computadoras que aportaban recursos de cálculo a los efectos de luchar contra el cáncer. Existen distribuciones de agentes cliente para diversas plataformas de usuario común .

URL del proyecto http://www.computeagainstcancer.org/

IV.2.6.6 United Devices Inc.

Proyecto apra asistir a la lucha contra el cáncer. Fue lanzado en abril del año 2001 con respaldo de la firma Intel, la Fundación Nacional para la Investigación sobre el Cáncer, el Departamento de Química de la Universidad de Oxford en el Reino Unido y el Centro sobre Investigación Computacional de drogas terapéuticas de la Universidad deOxford.

En el proyecto se corren modelos tridimensionales de moléculas y se analizan las interacciones entre proteínas. El objetivo buscado es obtener un método o técnica que permita reducir los daños celulares.

URL del proyecto http://www.ud.com/

IV.2.6.7 Folding@Home

El proyecto ha sido puesto en marcha el septiembre de 2000, y consiste en analizar, a través de la simulación temas de la biología molecular, relacionados con las proteinas., para ello utilizan el poder de cómputo del cluster de usuarios para correr simulaciones.

El software de usuario se visualiza en la forma de un salvapantalla. Folding@home fue desarrollado por el grupo Pande del Departamento de Química de la Universidad de Stanford.

URL del proyecto http://www.stanford.edu/group/pandegroup/Cosm/

IV.2.6.8 Genome@Home

Genome@home es un proyecto de la Universidad de Stanford. Su objetivo es el diseño de nuevos genes que puedan formar proteínas funcionales en la célula. Usa un algoritmo denominado Predicción de Secuencia (Sequence Prediction Algorithm, o SPA), fue desarrollado en el Departamento de Química de la Universidad Estatal de Penn.

Como resultados del proyecto podrían obtenerse nuevos remedios contra enfermedades. En el año 2001, cerca de 20.000 usuarios participaron del proyecto donando sus recursos. El cliente de Genoma@home trabaja de forma similar a su par SETI@home, activándose en base al inicio de actividad de un protector de pantalla.

URL del proyecto http://genomeathome.stanford.edu/

IV.2.6.9 Evolution@Home

El proyecto Evolution@Home [Loewe] tiene por objetivo investigar aquellos factores que realizan la evolución de especies sobre el planeta. Hay una serie de subproyectos con diferente software de simulación a los efectos de resolver problemáticas concretas. Se investigan los efectos de las fuerzas que rigen la evolución de las especies para predecir las posibilidades de supervivencia de especies en peligro de extinción.

El proyecto Simulator005, usa algoritmos basados derivados de la teoría "trinquete de Muller" la cual estudia aquellos factores genéticos denominados mutaciones nocivas [Bornet].

Utiliza un cliente que se descarga de su sitio y se instala. El envío de resultados no está automatizado, por ello éstos deben devolverse por correo electrónico.

URL del proyecto http://www.evolutionary-research.org/)

IV.2.6.10 Casino-21

El proyecto hace uso del procesamiento distribuido en Internet para llevar a cabo la predicción y construcción de modelos que simulen el clima a mayor o menor escala [Bornet]. Organizado por investigadores de la Universidad de Oxford, el Laboratorio Rutherford Appleton, y del Instituto Tecnológico de Massachussets, en él se intentan realizar modelos climáticos que sean válidos para el siglo XXI, basándose en simulaciones de Monte Carlo de allí su nombre) a partir de los datos recogidos hasta la fecha. En las computadoras de usuario, distribuidas a lo largo del mundo, es donde se realizan los cálculos derivados del proceso de simulación.

Toma la hipótesis que hace culpable del cambio climático a los gases de efecto invernadero, se construyen modelos tridimensionales que simulen el movimiento de líquidos, gases y el intercambio térmico entre atmósfera y océanos, en los que se incorporan diferentes niveles de gases de efecto invernadero para ver las consecuencias que se deducen de tales simulaciones [Bornet].

URL del proyecto http://www.climateprediction.com

IV.2.6.11 Gamma Flux

En operación desde diciembre de 1999, el proyecto Gamma Flux tiene por finalidad mejorar la construcción de contenedores y sistemas de almacenamiento de desechos radiactivos mediante la simulación de la radiación emitida desde una fuente confinada. Se busca calcular

el flujo de radiación gamma hacia fuera del contenedor, lo que ayuda a determinar entre otros datos cuánto calor escapa del contenedor [Bornet]. Este proyecto provee versiones del programa cliente para Windows y Linux.

URL del proyecto http://www.jansson.net/

IV.2.6.12 GIMPS

El proyecto GIMPS (Great Internet Mersenne Prime Search), nació en el año 1996 y tiene como meta la búsqueda de números primos de Mersenne, que se escriben con la fórmula 2^P-1. Más de 8000 personas han contribuido con tiempo de procesador para ayudar a descubrir el récord mundial de números primos de Mersenne. A lo largo de la historia, los mayores números primos conocidos han sido normalmente primos de Mersenne.

El 1 de Junio de 1999, Nayan Hajratwala encontró el actual récord mundial de números primos, $2^{6972593}$-1. El 27 de Enero de 1998, Roland Clarkson había encontrado el anterior primo récord, $2^{3021377}$-1. El 24 de Agosto de 1997 Gordon Spencer descubrió el primo $2^{2976221}$-1. En Noviembre de 1996 Joel Armengaud halló el primo $2^{1398269}$-1. Actualmente, se ha descubierto el cuarto primo desde el inicio del proyecto, es el número $2^{6972593}$-1 y corresponde al 38º primo de Mersenne conocido.

URL del proyecto http://www.mersenne.org/

Capítulo V

V Diseño de un Servicio de Cómputo Masivo Distribuido

V.1. Introducción

En este capítulo se presenta el diseño de un ambiente de cómputo masivo distribuido basado sobre una red P2P, él cual utiliza los servicios de una red de aplicación, accedidos por medio del middleware gnutWare.

A partir de los avances presentados en el área de computación masiva distribuida y las posibilidades de aplicación se plantea definir una arquitectura completamente distribuida basada en una red compañero a compañero. Bajo este enfoque, cualquier nodo se encuentra en condiciones de aportar ciclos de su CPU a la red y – además – es potencialmente beneficiario del poder de cómputo obtenido.

Genéricamente, el sistema se encuentra soportado por una red basada en el protocolo Gnutella, a la cual se accede mediante el middleware gnutWare anteriormente descrito. De esta manera, el sistema aprovecha las ventajas del modelo P2P en cuanto a robustez, tolerancia a fallas, flexibilidad en la dinámica de la red y cambios de topología. Si bien el prototipo inicial se encuentra basado en Gnutella, la inclusión del middleware posibilita que se puedan adaptar otros protocolos que soporten el mismo servicio.

Los nodos participantes de la red de procesamiento distribuido intercambian mensajes de servicio para colaborar con el procesamiento de un determinado proyecto como así también para entregar los resultados obtenidos al nodo recolector. De esta manera, los nodos compañeros pueden participar de uno o varios proyectos cuando dispongan de tiempo de CPU libre.

El capítulo se ha dividido en tres partes. En la primera se consideran los trabajos relacionados. En la segunda parte se expone un diseño de infraestructura de sistema de cómputo masivo distribuido, denominado P2P-Flops. AL final del presente capítulo, se presenta un trabajo experimental cuyo objetivo consistió en explorar la disponibilidad de tiempo libre de CPU en una organización educativa. La actividad se realizó a los efectos de poder estimar preliminarmente el potencial total de cómputo diario existente en una organización que utilice el sistema P2P-Flops.

V.2. Trabajos relacionados

En esta sección se describen proyectos de investigación en marcha como así también infraestructuras disponibles para cómputo masivo no orientadas a una única aplicación. En particular se hace referencia a sistemas actuales tipo enrejado o grid [Foster-A] y a infraestructuras basadas en el lenguaje Java.

A los efectos de establecer cuales son las diferencias entre un cluster y un sistema grid se utilizan las siguientes definiciones:

- *Un cluster* [Morrison] es un grupo independiente de computadoras que trabajan en conjunto formando un único recurso computacional. Generalmente tales computadoras son personales, pertenecen a una misma organización y utilizan como plataforma de trabajo una versión del sistema operativo UNIX adaptada. Su ámbito de operación son redes locales dado que necesitan un canal de datos de alta velocidad

- *Un sistema grid* es una infraestructura de hardware y software heterogénea, básicamente conformado por computadoras personales de distintas organizaciones asociadas, que permite a sus usuarios acceder a grandes capacidades de recursos computacionales, naturalmente distribuidos en una red WAN, de forma confiable, consistente y económica. Una característica particular es que su control es naturalmente distribuido

V.2.1 Computación grid

A mediados de la década de 1990 un nuevo paradigma de computación distribuida fue propuesto por Ian Foster y Carl Kesselman, [Foster-A] [Foster-B] el cual se enfoca al acceso remoto a recursos computacionales. Tal paradigma no describe ninguna tecnología concreta, sino que establece las bases teóricas donde se desarrollará el computo masivo distribuido.

Foster propone una analogía con la red de energía eléctrica, en la cual el *"usuario debe tener acceso a los recursos computacionales en condiciones similares a las que tiene para utilizar la energía eléctrica: desde cualquier sitio (pervasive), con una interfase uniforme (consistent), pudiendo confiar en su funcionamiento (dependable), y a un costo racional (inexpensive)."*

En resumen, el término grid define a la infraestructura hardware y software que permite un acceso fiable, constante, extenso y económico a recursos destinados al procesamiento de datos. Esta infraestructura es naturalmente distribuida por distintos pisos de un edificio, por un país o, incluso, por continentes.

De forma distinta a como trabajan las supercomputadoras tradicionales, que generalmente se encuentran localizadas en un punto concreto, las redes grids conectan computadoras de usuario, clusters y supercomputadoras dispersas geográficamente utilizando Internet y protocolos específicos de código abierto creados por la organización internacional Globus [Globus]. Las organizaciones se conectan a una red grid para tener acceso a capacidad de proceso, capacidad de almacenamiento y ancho de banda, al igual que se conectan a la red eléctrica para tener acceso a energía.

Grid se basa en una red de gran ancho de banda, baja latencia, alta velocidad de procesamiento, y voluminosas bases de datos. Por su naturaleza, el uso de computadoras en una infraestructura grid posibilita que múltiples usuarios puedan obtener información de forma transparente en el acceso.

Existe un estándar de facto denominado Globus. Por otro lado, Toolkit Globus es un proyecto open source desarrollado por Argonne National Laboratory de la University of Southern California, define los protocolos y servicios necesarios para construir aplicaciones que operen sobre una infraestructura grid. El estandar Globus se presenta bajo una arquitectura de capas: la capa básica denominada *infraestructura* controla los recursos locales; la capa de *conectividad* se encarga de la comunicación y la seguridad; la capa *recurso* provee servicio de acceso y control de recursos, finalmente en la capa *recursos* se define la coordinación de recursos.

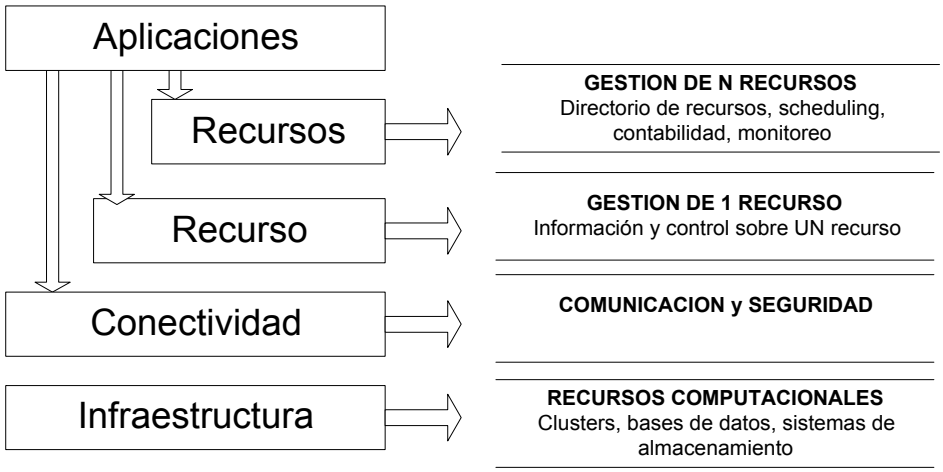


Gráfico de la arquitectura Globus

En Globus las cuestiones de seguridad se realizan por medio del protocolo Grid Security Infraestructure (GSI), existe un esquema de autenticación de usuario mediante certificados digitales PKI y X.509. La capa conectividad incluye además los protocolos habituales de Internet (IP, DNS, etc); los otros tres protocolos se vinculan a la capa recurso: Grid Resource Allocation Management (GRAM), Grid Resource Information Protocol (GRIS), y el Grid File Transfer Protocol (GridFTP).

V.2.1.1 Como funciona una red basada en grid

Un equipo central, denominado planificador, distribuye un proceso entre pares dispersos geográficamente conectados por una red de datos, donde tales pares pueden ser computadoras de escritorio o equipos de alto rendimiento (clusters, supercomputadoras, mainframes) especialmente dedicados al cómputo intensivo.

El sistema toma las capacidades disponibles de todos los equipos vinculados a la infraestructura. Cuando no están siendo utilizados plenamente por el usuario reciben tareas del equipo planificador. Todos los recursos disponibles en la red son aprovechados, independientemente de su ubicación geográfica, su arquitectura y sistema operativo. Un punto fuerte de una red grid es su espina dorsal implementada como un canal alta velocidad y baja latencia.

Cuando un usuario utiliza sólo una parte de la capacidad de su CPU, el resto se aprovecha. Por problemas derivados de fallas o de uso pleno de una computadora por parte de su usuario, la tarea se reasigna a otra máquina disponible.

A la aplicación se la accede desde cualquier equipo. A través de una interfase simple se pueden aagremar más computadoras al sistema. La conexión directa entre las computadoras (P2P) evita la sobrecarga del equipo central.

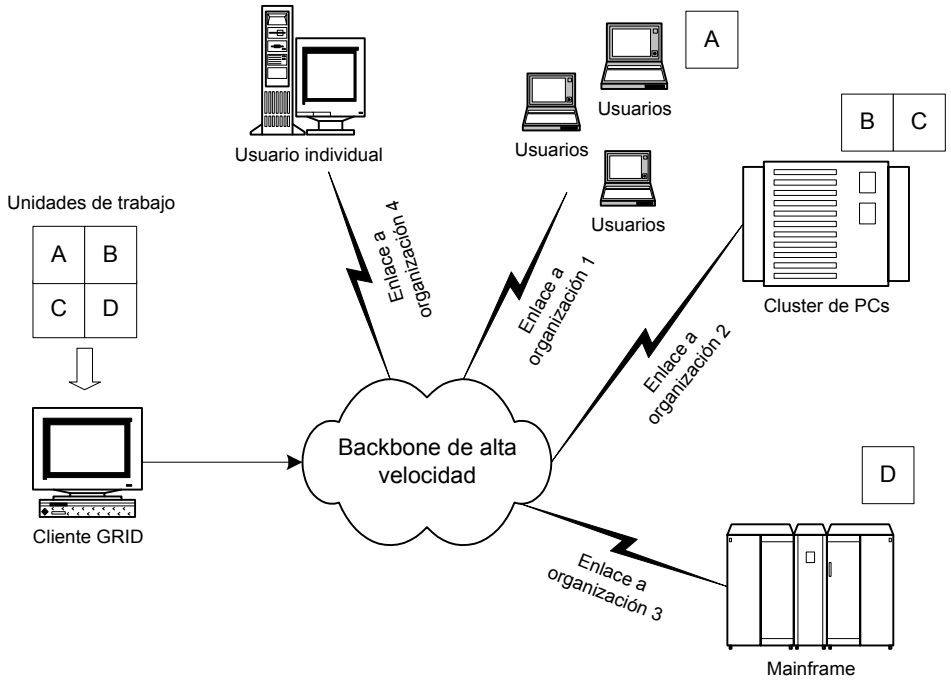


Gráfico de una red grid

Nótese en el gráfico anterior que la computadora cliente grid tiene un problema que requiere cómputo intensivo y solicitó auxilio a su planificador, el cual generó cuatro unidades de trabajo (A, B, C y D) que fueron asignadas a tres grupos de cómputo.

V.2.1.2 Beneficios

Los beneficios de la tecnología de cómputo grid se pueden resumir en los siguientes puntos:

1. Integración de equipos heterogéneos.

2. Sistemas de cómputo masivo a mejores precios.

3. Explotación de capacidades de cómputo osciosas.

4. Infraestructuras de alta disponibilidad y buena confiabilidad.

V.2.1.3 Desarrollo de la tecnología grid

Entre los primeros proyectos grid surgen Information Power Grid de la NASA [Johnston] y la iniciativa de la National Science Foundation con los centros de supercomputación de NCSA y SDC. En el área infraestructura, el proyecto americano Teragrid tiene por objetivo unir cuatro centros de supercómputo a una velocidad de 40 Gbps en su backbone. En Europa están: el proyecto de supercómputo denominado CrossGrid, EuroGrid que es una plataforma operativa destinada a procesamiento en el ámbito industrial y científico y el Reino Unido cuenta con el proyecto UK e-Science orientado a la investigación distribuida.

Desde el punto de vista de canales de alta velocidad Europa cuenta con alternativas como: GÉANT que es un backbone gigabit de 10 Gbps y REDIRIS2 [IRIS] que es un backbone español de 2.5 Gbps

Los proyectos de grid computing más conocidos son SETI@Home y la red Entropía que agregó en dos años 30.000 computadoras, logrando - por ejemplo - calcular el mayor número primo conocido.

Por otro lado, la incorporación de esta tecnología está siendo considerada como una alternativa para desarrollar negocios por empresas tales como Compaq Computer, IBM con su Arquitectura de Servicios Abiertos Grid (OGSA), Sun Microsystems con Sun Grid Engine y Oracle para su proyecto 10g.

Según la propuesta española de creación de un Programa de e-Ciencia [IRIS] los resultados que se han obtenido hasta la fecha con la tecnología grid se puede resumir en los siguientes puntos:

- *Desarrollo de servicios de directorios que permiten la búsqueda de datos en entornos ampliamente distribuidos.*

- *Empleo de técnicas de replicación como forma de mejorar las prestaciones en el acceso a los datos.*

- *Desarrollo de protocolos de acceso a datos como GridFTP.*

- *Desarrollo de una API para acceder a archivos remotos, como el servicio GASS (Global Access to Secondary Storage) que proporciona el Globus Grid Toolkit.*

Siendo los puntos pendientes por resolver:

- *Conseguir un verdadero espacio de nombres global, donde sea sencillo localizar los datos.*

- *Desarrollar técnicas de almacenamiento de altas prestaciones que permitan mejorar el acceso a los datos en un sistema grid, como por ejemplo, el empleo de técnicas de E/S paralela y distribución de datos a través de los diferentes recursos de almacenamiento*

- *Desarrollar servicios para integrar diferentes protocolos y sistemas de almacenamientos locales.*

- *Desarrollar interfaces de acceso a datos adecuadas para computación de altas prestaciones, como por ejemplo, MPI-IO para entornos Grid.*

- *Explotar las posibilidades de mecanismos de búsqueda e indexación distribuidos, que empleen la tecnologías grid para ayudar en la localización de recursos.*

- *Integrar estos procedimientos con los mecanismos de autorización, de manera que todas las interacciones sean susceptibles de personalización*

V.2.2 Proyecto GPU

Es una implementación de una plataforma de procesamiento distribuido, sobre una red compañero a compañero. El proyecto open source "Gigaglobal Proccessing Unit" [GPU]

implementa un prototipo de plataforma que permita que una aplicación utilice ciclos de CPUs de distintas computadoras de usuario final conectadas a Internet.

Para lograr este objetivo, la aplicación se conecta a una red P2P, del tipo Gnutella, y dispone de mensajes específicos para enviar, recibir y ejecutar trabajos computacionales. Dado que la red base es Gnutella, los mensajes que solicitan ejecución de trabajo remoto o devuelven resultados de procesamiento, viajan junto a mensajes de solicitud de búsqueda de archivos hasta hallar un compañero que disponga de recursos ociosos para procesar tal tarea. Una vez procesada la tarea se devuelven los resultados por la misma red P2P.

Software tipo plugin es necesario instalar en cada computadora de usuario final que coopere con algún proyecto, su objetivo es contener el código de librería al cual se refieren las tareas de cómputo cooperativo. Por ejemplo, plugins pueden computar el logaritmo discreto en aritmética modular o probar por fuerza bruta distintas combinaciones de claves con el fin de validar un algoritmo criptográfico.

V.2.3 Condor

La evolución de las redes locales de tipo académicas posibilita integrar la capacidad de todos los equipos de una institución. El sistema Condor [Tannenbaum-A] desarrollado por Miron Livny para el sistema operativo Linux se diseñó a los efectos de aprovechar el tiempo de inactividad de tales máquinas, permitiendo agregar y compartir recursos. Con el tiempo Condor se ha transformado en un sistema completo de gestión de clusters, máquinas multiprocesadores y redes grid.

Condor es un software orientado al balanceo de cargas de procesamiento entre un conjunto de computadoras cooperantes. Se basa en la migración de procesos completos a través equipos compañeros. Su operación es transparente al usuario, dado que tiene la ilusión de que sus tareas están siendo ejecutadas localmente. Posee un mecanismo de colas de trabajo, una política de planificación, y un módulo de monitoreo y administración de recursos. Los usuarios envían sus trabajos seriales o paralelos y Condor los aloja en una cola, luego decidirá cuando y quien los procesará en base a una política de trabajo. Una ventaja importante de

Condor es su mecanismo de "checkpoint", el cual almacena transparentemente el estado de una tarea en ejecución y posteriormente permite que sea ejecutada a partir del punto en que se detuvo. Eso es implementado a través de sustitución de bibliotecas del sistema UNIX por propias de Condor en el proceso de compilación de las aplicaciones.

V.2.4 NetSolve

NetSolve [Arnold] [Arnold-A] es un sistema cliente-servidor que posibilita el uso de librerías matemáticas en un ambiente distribuido. Permite que los usuarios accedan de forma transparente a recursos de hardware y software distribuidos a lo largo de una red. Donde si alguien suministra una tarea de cómputo NetSolve buscará por recursos computaciones ociosos sobre la red y utilizará una política de balanceo de cargas a los efectos de obtener resultados en menores tiempos.

Se han desarrollado interfases con lenguajes y paquetes tales como Fortran, C, Matlab, Mathematica y Octave. Un objetivo propuesto por sus diseñadores en la versatibilidad a los efectos de poder agregar, con cierta facilidad, nuevas funcionalidades de software como así también integrar nuevas tecnologías.

V.2.5 BOINC

Es un proyecto de Berkeley [Anderson-A] destinado a crear una infraestructura abierta para el cómputo distribuido. Se basa en la utilización de recursos aportados por voluntarios a los efectos de construir un supercomputador virtual.

Diferentes proyectos pueden utilizar simultaneamente la infraestructura BOINC. Los proyectos son independientes, cada uno opera con su propio servidor de planificación y su base de datos. Sin embargo, los proyectos pueden compartir recursos en el siguiente sentido: los participantes instalan un módulo central de cliente, el cual descarga y ejecuta una aplicación de cómputo intensivo específica. Los participantes deciden en que proyectos participar y que recursos aportan a cada uno. Actualmente BOINC está siendo desarrollado y probado para las plataformas Windows, Linux, UNIX, y Mac OSX.

Según sus autores BOINC posee las siguientes características: a) es un medioambiente de trabajo flexible, b) aplicaciones existentes (en C, C++, Fortran) pueden ejecutarse bajo BOINC con pocas y en algunos casos ninguna modificación, c) es seguro, dado que implementa protecciones ante ataques y encriptación de datos en tránsito, d) posee facilidades de monitoreo de trabajos por parte de los administradores y usuarios trabajadores y e) disponibilidad del código fuente de BOINC por parte de los desarrolladores.

V.2.6 Proyectos de cómputo distribuido en ambiente web basados en Java

El lenguaje Java [Barcellos] posee características que lo hacen atractivo a la hora de seleccionar una herramienta de programación en un proyecto de cómputo masivo distribuido. Sus cualidades más relevantes son: su simpleza, es de arquitectura neutra y por lo tanto portable, es orientado a objetos, soporta programación por hilos múltiples, posee métodos nativos de invocación a procedimientos remotos, es seguro y posee un interesante grado de evolución.

Con Java es posible construir infraestructuras de procesamiento masivo paralelo denominadas "casi transparentes", dado que los programadores solo deben utilizar algunas pocas funciones o sintaxis que no es parte del código base de Java, y el sistema automáticamente distribuye la aplicación sobre diversas máquinas. Hilos de programación tradicionales Java pueden ser utilizados para actividades de paralelismo y sincronización.

Gracias a su independencia de plataforma sobre la que se ejecuta, un sofisticado modelo de seguridad, RMI (Remote Method Invocation) y la capacidad de serializar objetos (empaquetamiento de objetos para su almacenamiento o transmisión y posterior restauración en la misma o en otra máquina virtual), el lenguaje Java ha sido ampliamente escogido para el desarrollo de sistemas distribuidos.

A continuación se enumeran los proyectos de infraestructura más populares basados en Java:

ParaWeb [Bretch] es un proyecto pionero (1996) en esta área, fue diseñado para operar tanto en una intranet como en Internet. Su objetivo es que los usuarios ejecuten programas de tipo serial sobre computadoras más veloces o programas construidos bajo un paradigma de programación paralela sobre una variedad de equipos heterogéneos. Tecnicamente ParaWeb aporta librerías con funciones de comunicación y sincronización de procesos al ambiente Java

JPVM [Ferrari] es una interfaz que permite a aplicaciones Java usar el software de Máquina Virtual Paralela (PVM). La máquina virtual es un componente de software que permite que una colección heterogénea de computadoras interconectadas por una red bajo protocolos TCP/IP sea utilizada como una gran máquina virtual. Tal infraestructura puede ser utilizada para resolver problemas computacionales de gran magnitud utilizando el poder agregado y memoria de muchas computadoras. El software es portable y ha sigo compilado en una gran variedad de plataformas. Los lenguajes que soporta PVM actualmente son C, C++ y Fortran. Con JPVM es posible crear interfaces de Java a programas existentes de C, C++ y Fortran y usar PVM para enviar data entre esos programas y la interfaz. Según sus autores JPVM le ofrece a los programadores una alternativa más simple y robusta a la librería de sockets de Java.

Otra plataforma es Javelin [Cappello], la cual es una infraestructura destinada al cómputo en un ambiente global. El sistema, que está basado en Java, opera distribuido sobre equipos conectados Internet. Los participantes solo deben disponer de una aplicación explorador con Java activado. En Javelin, por arquitectura, se definen tres tipos de entidades participantes: a) intermediarios, b) clientes y c) hosts. Donde un cliente es un proceso que busca recursos computacionales, un host es un nodo que ofrece recursos al sistema y un intermediario es un proceso que coordina la demanda y la oferta. Las unidades de trabajo que envían los clientes a los hosts son applets que contienen código y datos a procesarse distribuidamente.

Un sistema similar a Javelin es el denominado Knitting Factory [Baratloo], el cual también opera ejecutando applets Java en equipos distribuidos. Como características propias se puede citar que aporta: a) un servicio distribuido de nombres destinado a asistir en la tarea de

localización de otros participantes, b) un servidor web embebido a los efectos de eliminar el uso de servidores externos y c) un protocolo de comunicación directa entre applets.

Fragati [Fragati] en su trabajo final de grado de la Licenciatura en Sistemas de Información de la Universidad Nacional de Luján ha diseñado un modelo de infraestructura destinada a soportar proyectos de procesamiento distribuido, haciendo uso de la red Internet. La idea es que este sistema se implemente en sitios web que son visitados periódicamente por una importante cantidad de usuarios, los cuales permanecen durante un tiempo significativo leyendo determinadas páginas web. Un ejemplo de sitio ideal es un diario ó revista digital, dado que cumple con las características enunciadas.

Las mencionadas páginas llevan inserto un applet Java que realiza alguna tarea de cómputo cooperativo mientras el usuario lee el contenido, bajo tal operatoria el usuario está colaborando con ciclos de CPU destinados a procesar unidades de trabajo y devolver resultados a un nodo de recolección de resultados. Para validar la idea propuesta se codificó un prototipo de arquitectura de cómputo masivo distribuido basada en el sistema web como soporte de comunicaciones, applets Java y scripts en lenguaje Perl.

V.3 Arquitectura propuesta

A continuación se presenta P2P-Flops, una infraestructura destinada a brindar un punto de partida para el desarrollo de aplicaciones de procesamiento masivo distribuido, sobre una red de una organización, basadas en el modelo compañero a compañero.

V.3.1 P2P-Flops

P2P-Flops define una arquitectura para un servicio de procesamiento masivo distribuido en un ambiente cooperativo, bajo el modelo P2P. Como ya se mencionó, su arquitectura general se relaciona con una red basada en el protocolo Gnutella y el middleware gnutWare. Los objetivos de diseño fueron los siguientes:

Robustez, que permita de forma transparente y segura el ingreso y egreso de proyectos y nodos trabajadores

Simplicidad, el acceso al servicio se realiza mediante interfaces sencillas y estandarizadas.

Economía, que los beneficiarios del servicio puedan obtener resultados a costos reducidos debido a la distribución de las tareas y el aporte de recursos por parte de terceros.

Eficiencia, que las organizaciones puedan hacer uso intensivo de sus computadoras aprovechando sus tiempos de inactividad.

Transparencia, debido a que los equipos que aportan CPU cooperan en los proyectos sin necesidad de mantenimiento local y no afectan a las tareas habituales de su usuario.

En las organizaciones, en general, existen en sus redes un conjunto de nodos que normalmente disponen de tiempo de CPU libre que habitualmente se desperdicia. Bajo la arquitectura propuesta estos nodos –denominados "trabajadores"-, en sus tiempos de inactividad, pueden contribuir con ciclos de CPU a un determinado proyecto que requiera cómputo masivo.

Los clientes del sistema son aquellas entidades que presentan un proyecto a la red para que los trabajadores colaboren en su solución. Los proyectos plausibles de ser atendidos en esta arquitectura son aquellos que pueden descomponerse en subpartes individuales –unidades de trabajo- , donde la solución final se obtiene a partir de la integración de los resultados de las unidades de trabajo.

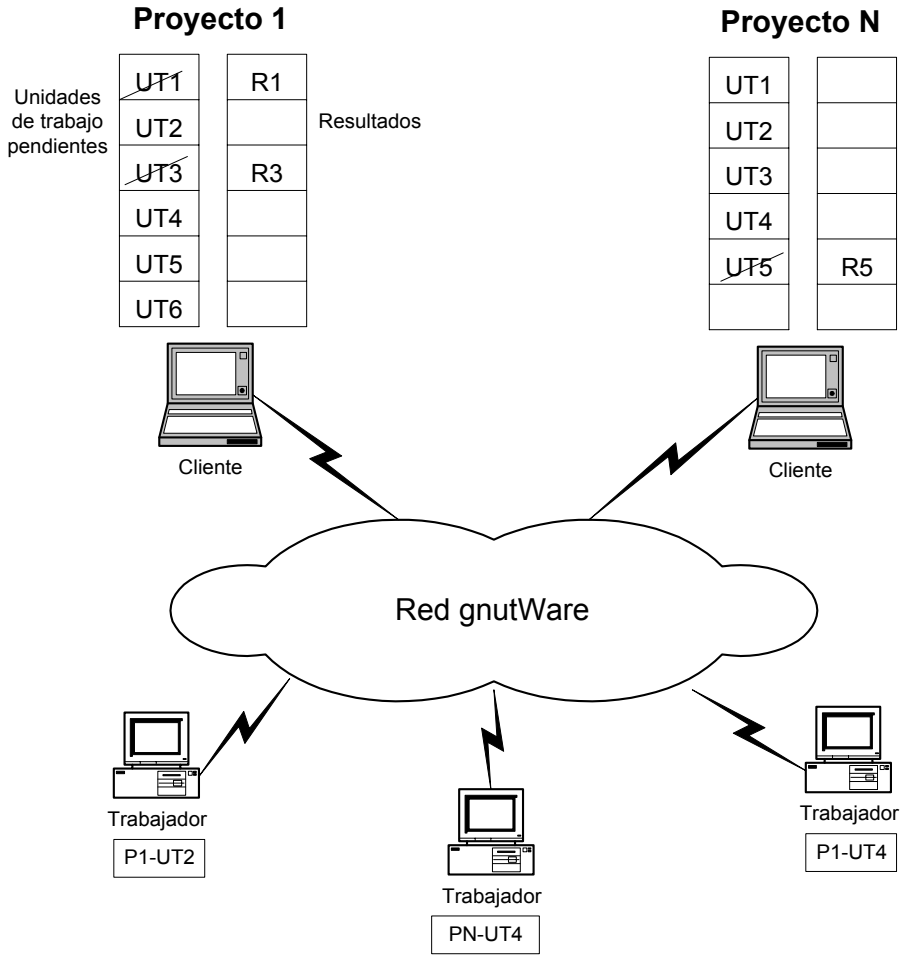


Gráfico de la infraestructura P2P-Flops

En el gráfico anterior se muestra con máxima abstracción el modelo de infraestructura P2P-Flops. Allí se pueden ver los actores involucrados, como así también la relación existente entre ellos.

El ámbito natural de operación se limita a las redes internas de una organización particular (su intranet), donde generalmente los trabajadores y clientes son conocidos, los enlaces son exclusivos de la misma y los proyectos también pertenecen a la organización. Si bien se cree que el modelo puede escalar a redes de área amplia o de varías organizaciones es necesario redefinir y estandarizar ciertas políticas de operación, como así también aspectos relativos a la seguridad.

Hoy en día, un cluster de computadoras es la herramienta natural de cómputo intensivo distribuido en organizaciones, presenta la restricción que debe haber una importante inversión en equipamiento dedicado, y su amortización es directamente proporcional a la tasa de uso. Contrariamente, P2P-Flops no exige una inversión especial en equipamiento sino que aprovecha el existente en la organización. La eficiencia total estará en función del tiempo ocioso de los trabajadores.

El utilizar la red basada en el protocolo gnutWare proporciona una canal tipo multicast donde es posible el envío de mensajes de solicitud de servicio a los trabajadores de manera simple y normalizada. Esta solución evita el uso de un servidor central de registro de trabajadores y proyectos, con lo cual aumenta la eficiencia del intercambio de mensajes de servicio entre clientes y trabajadores.

En base a lo expuesto anteriormente se diseñó una posible arquitectura que consta de una serie de módulos, a saber:

- **Módulo trabajador:** Es el componente que realiza el procesamiento propiamente dicho. Se instala en las computadoras personales de aquellos usuarios de una organización que donan ciclos de CPU ociosos. Gestiona la recepción y envío de unidades de trabajo y resultados

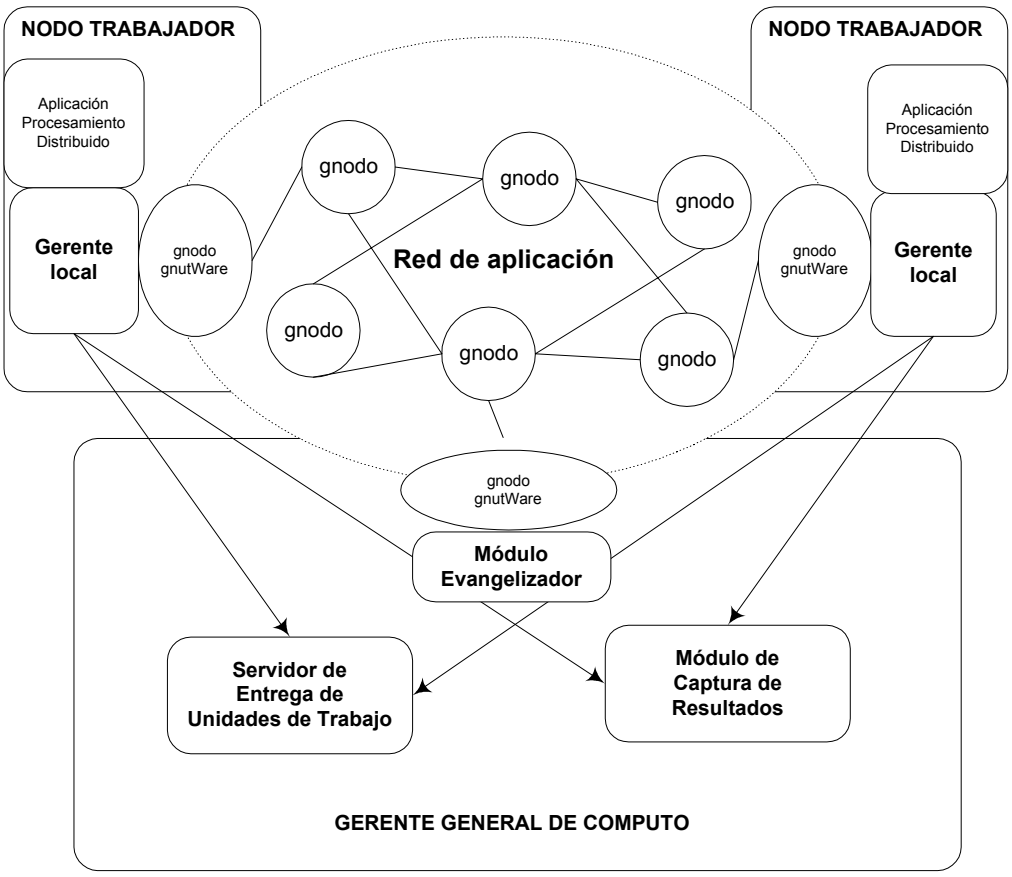
- **Módulo evangelizador:** Tiene por misión, a pedido de un proyecto de cómputo masivo, enviar periódicamente mensajes de solicitud de ciclos de CPU a los trabajadores utilizando el middleware gnutWare.

- **Módulo de captura de resultados:** Componente que recibe de los trabajadores los resultados obtenidos del procesamiento de una unidad de trabajo.

- **Servidor de entrega de unidades de trabajo:** Es el encargado de brindar nuevas unidades de trabajo a los trabajadores

- **Gerente general de cómputo:** Es el encargado de coordinar todo un proyecto de cómputo masivo distribuido bajo mandato de un cliente. Controla directamente al módulo de evangelización, al servidor de entrega de unidades de trabajo y al módulo de captura de resultados.

El gráfico siguiente ilustra la relación existente entre los componentes del sistema de cómputo masivo distribuido P2P-Flops.



V.3.2 Modo de operación

Supongase que existe un proyecto P1 que requiere que n unidades de trabajo UT sean procesadas por trabajadores. Se define que un nodo N1 opere en modalidad gerente general del proyecto P1, sobre el cual se ejecutará el servidor de entrega de unidades de trabajo. Lo primero que debe realizar N1 es comunicar a los trabajadores que necesita cooperación para el proyecto P1, para lo cual utilizará como herramienta de difusión al módulo evangelizador.

Cuando un trabajador T1 recibe un mensaje de solicitud de servicio y no existe una unidad de trabajo procesándose en éste, atenderá tal petición comunicándose con el nodo servidor de entrega de unidades de trabajo referenciado en el mensaje de evangelización, a los efectos de obtener una unidad de trabajo.

Una unidad de trabajo está caracterizada por: a) un identificador de unidad de trabajo, b) una referencia a un programa ejecutable que reside en el nodo gerente general de cómputo en una URL determinada, c) una serie de parámetros de ejecución del programa en cuestión y d) una dirección de retorno de resultados (que corresponde al módulo de captura de resultados).

El trabajador con estos datos evalúa si tiene en cache la aplicación o debe descargarla del gerente general. A continuación preparará el ambiente de ejecución a los efectos que el evento "activación de salvapantalla" (el cual marca que hay ciclos de CPU ociosos) pueda invocar a la aplicación con los parámetros correspondientes. Por otro lado está previsto que ante al desactivación del salvapantallas la aplicación pueda grabar su estado de avance a los efectos de continuar en otro momento. Es importante destacar que una unidad de trabajo se puede completar en n ejecuciones de la aplicación.

Cuando una unidad de trabajo finaliza su ejecución el modulo trabajador que la aloja debe enviar los resultados al módulo de captura pertinente. Luego de tal tarea, el nodo trabajador

se encuentra disponible para procesar una nueva unidad de trabajo (que puede pertenecer al mismo proyecto o a otro).

V.3.3 Implementación

A los efectos de obtener un prototipo operativo de la arquitectura propuesta se implementó una versión inicial de cada uno de los componentes. Los trabajadores son computadoras personales con sistema operativo de la familia Windows de Microsoft, mientras que para los restantes módulos se utilizó plataforma Linux. El lenguaje de trabajo fue Perl, excepto el módulo salvapantallas, el cual fue codificado en Visual Basic de Microsoft. Todas las comunicaciones se realizan mediante la pila de protocolos TCP/IP a través de la interfase sockets.

En esta versión prototipo no se tuvieron en cuenta aspectos relativos a la seguridad de las comunicaciones, autenticación de actores, ni tampoco al control del código que se ejecuta en los trabajadores.

V.3.3.1 Módulo trabajador

El módulo trabajador requiere de la existencia, en el mismo nodo, de una instancia del middleware GnutWare (descripto en el capítulo III), el cual permite la conectividad, por un lado, a la red de aplicación conformada por gnodos, y por otro lado a la aplicación de gerenciamiento local del trabajador de procesamiento distribuido.

La conexión con la red de gnodos se realiza mediante protocolo TCP, solicitando servicio de red de aplicación sobre el número de puerto 5634 de cualquier gnodo. La sintaxis de ejecución del módulo de GnutWare en un trabajador es la siguiente:

```
java GnutWare IP_GNODO PUERTO_GNODO SID PUERTO_LOCAL
```

Donde

IP_GNODO es la dirección de red del gnodo a conectarse

PUERTO_GNODO es el número puerto del gnodo a conectarse

SID es un entero de 2 bytes que identifica al servicio asociado sobre el puerto local

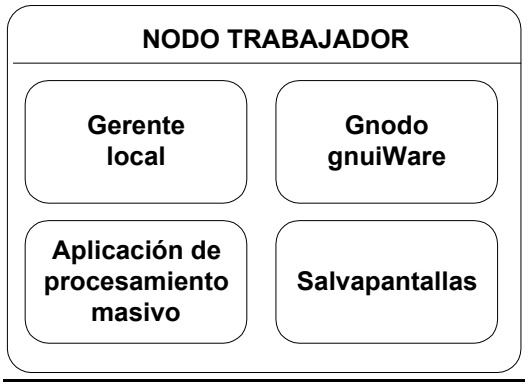
PUERTO_LOCAL número de puerto donde se conecta el demonio del trabajador local para acceder a la red de propagación de mensajes.

Ejemplo

java GnutWare 10.10.10.222 5634 55 20001

En el ejemplo anterior, en un trabajador, se corrió el software GnutWare solicitando una conexión con el gnodo residente en la dirección de red 10.10.10.222:5634. Las aplicaciones locales que requieran servicio de middleware lo obtendrán a través de una conexión TCP local sobre el puerto 20001.

Una segunda aplicación, denominada gerente local, es la encargada de administrar toda la operación de procesamiento cooperativo cuando existan tiempos ociosos de procesador central.



Dado que la aplicación prototipo actualmente corre sobre sistemas operativos Microsoft, el estado de procesador ocioso se determina cuando se activa el salvapantallas, y se sale de tal estado cuando la mencionada aplicación finaliza. En cada trabajador se instala una versión propia de software salvapantallas que al activarse genera un archivo bandera (indicación de salvapantalla activo) sobre el sistema de archivos, cuando la aplicación finaliza borra el mencionado archivo.

El gerente local de procesamiento, lo mismo que el middleware GnutWare, corre en segundo plano tal como si fuera un demonio. Al detectar el evento salvapantalla activo, por medio de la verificación de la existencia del archivo bandera, ejecuta la aplicación relacionada con la unidad de trabajo actual.

La aplicación que procesa las unidades de trabajo finaliza parcialmente su ejecución si detecta la inexistencia del archivo bandera. De esta forma simple la aplicación determina cuando debe finalizar y grabar su estado parcial de ejecución, a los efectos de continuar en otro momento.

Si por el contrario la aplicación finaliza su ejecución dado que completó la unidad de trabajo, indica tal situación al módulo de gerenciamiento local renombrando la extensión del archivo de datos por la cadena "fin".

La situación descripta en el párrafo anterior es advertida por el módulo de gerenciamiento local, el cual deberá entregar tal archivo de resultados al módulo de recolección de resultados correspondiente al proyecto.

A continuación se presenta un ejemplo de una porción de un archivo que describe una unidad de trabajo. Nótese que en el mismo se van anexando resultados parciales de ejecución a los efectos que la aplicación de procesamiento pueda retomar a partir de un estado previo.

```
DESDE=226844
HASTA=227500
IP-RETORNO=10.10.10.22
ID-UNIDAD-TRABAJO=1046519134
URL_APLICACIÓN=HTTP://10.0.0.21/aplicaciones/p3-v4.pl


INICIO=Fri Feb 28 20:34:37 2003
RESULTADOS=desde=226844*hasta=227500*226871*226901*226903*226907
DESDE=226908
FIN=Fri Feb 28 20:34:37 2003


INICIO=Fri Feb 28 20:37:30 2003
RESULTADOS=desde=226908*hasta=227500*226913*226937*226943
DESDE=226950
FIN=Fri Feb 28 20:37:30 2003


INICIO=Fri Feb 28 21:35:49 2003
RESULTADOS=desde=226950*hasta=227500*226991*227011*227027*227053*227081*227089*227093*2
27111*227113*227131*227147*227153*227159*227167*227177*227189*227191*227207*227219*227231*2
27233*227251*227257*227267*227281*227299*227303*227363*227371*227377*227387*227393*227399*2
27407*227419*227431*227453*227459*227467*227471*227473*227489*227497
DESDE=230144
FIN=Fri Feb 28 21:35:51 2003


FALLO-ENTREGA=Fri Feb 28 21:36:42 2003
FALLO-ENTREGA=Fri Feb 28 21:37:32 2003
FALLO-ENTREGA=Fri Feb 28 21:40:54 2003
```

El procesamiento que se realizó a modo de ejemplo corresponde a la determinación de números primos sobre un rango dado.

Las primeras cinco líneas corresponden a la información inicial de configuración enviada originalmente por el servidor de entregas de unidades de trabajo. Las etiquetas DESDE y

HASTA definen un rango de números enteros donde la aplicación de procesamiento distribuido buscará números primos. La etiqueta IP-RETORNO se instancia con la dirección de red del módulo de captura de resultados. ID-UNIDAD-TRABAJO es la etiqueta que indica, por medio de un entero, el número de unidad de trabajo, siendo un dato de utilidad para la generación global del proyecto. Finalmente, en URL_APLICACIÓN se explicita la localización de la aplicación a ejecutar por el trabajador (en este caso la que busca números primos). Nótese que el gerente local la descargará siempre y cuando no posea una copia local de la misma.

Los bloques delimitados por las etiquetas INICIO y FIN indican datos relativos a un intervalo de ejecución de la aplicación de procesamiento distribuido. Recordando que la aplicación es ejecutada por el módulo de gerenciamiento local cuando detecta que el proceso salvapantalla está activo.

En cada bloque se registra el tiempo de ejecución con marcas de inicio y fin, resultados parciales (en este caso números primos hallados) denotados bajo la etiqueta RESULTADOS, y se modifica la etiqueta DESDE que define el número entero inicial de búsqueda en la próxima instancia de ejecución.

En el archivo presentado como ejemplo se puede ver que la unidad de trabajo operó en tres instancias de ejecución. Al final, se puede observar la etiqueta FALLO-ENTREGA que indica que el módulo de gerenciamiento local no pudo contactarse con el servidor de captura de resultados. En el prototipo se definió una política inicial de entrega de resultados que consiste en n reintentos cada m minutos, en caso de fallo sobre todos ellos se releva de la tarea silenciosamente.

V.3.3.2 Módulo evangelizador

Tiene por misión el envío de mensajes de solicitud de donación de ciclos de CPU, utilizando la red de middleware GnutWare. Tales mensajes son recogidos por los nodos trabajadores que ejecutan el gerente local de procesamiento masivo distribuido. Si al momento de recibir

una solicitud no poseen una unidad de procesamiento en curso, se comunican con el servidor de entrega de unidades de trabajo indicado en el mensaje.

El formato del mensaje de evangelización es el siguiente:

EVANGELIZANDO=10.10.10.22,/cgi-bin/get-ut.cgi

Donde

10.10.10.22	es la dirección de red IP del Servidor de Entrega de Unidades de Trabajo.
/cgi_bin/get_ut.cgi	es el camino de la script a invocar vía protocolo HTTP sobre el servidor de referencia.

La comunicación con el servidor de entrega de unidades de trabajo se realiza vía protocolo HTTP sobre el puerto bien conocido decimal 80.

Como régimen inicial se envían cada 15 segundos un mensaje de evangelización. Este parámetro permite no congestionar la red y mantener de forma mínima el tiempo de inactividad de los nodos trabajadores.

V.3.3.3 Módulo de captura de resultados

Es un demonio que espera por conexiones entrantes basadas en el protocolo TCP. Los trabajadores al finalizar la ejecución de una unidad de trabajo deben contactarse con el nodo donde se ejecuta el módulo de captura a los efectos de enviarle sus resultados. La base de datos de resultados es accedida periódicamente por el gerente general a los efectos de controlar el avance de un proyecto.

V.3.3.4 Servidor de entrega de unidades de trabajo

Se implementó de manera simple utilizando un servidor estandar HTTP. Los trabajadores, utilizando el módulo de gerenciamiento local realizan la invocación de un script a través de la interface CGI, cuya URL fue provista por un mensaje de evangelización. Como retorno se obtiene una cadena de caracteres que corresponde al contenido de la unidad de trabajo. El siguiente es un ejemplo del resultado de solicitud de una UT:

```
DESDE=226844
HASTA=227500
IP-RETORNO=10.10.10.22
ID-UNIDAD-TRABAJO=1046519134
URL_APLICACIÓN=HTTP://10.0.0.21/aplicaciones/p3-v4.pl
```

V.3.3.5 Gerente general de cómputo

Como se vio anteriormente este módulo es el planificador general del proyecto, dado que además de entregar las UT por medio del servidor ya descripto, se encarga de coordinar con el módulo de captura de resultados y el evangelizador el avance de la tarea global. En primer lugar para un proyecto particular genera las n unidades de trabajo en que se lo divide. Tal actividad está en función del tipo de problema a resolver.

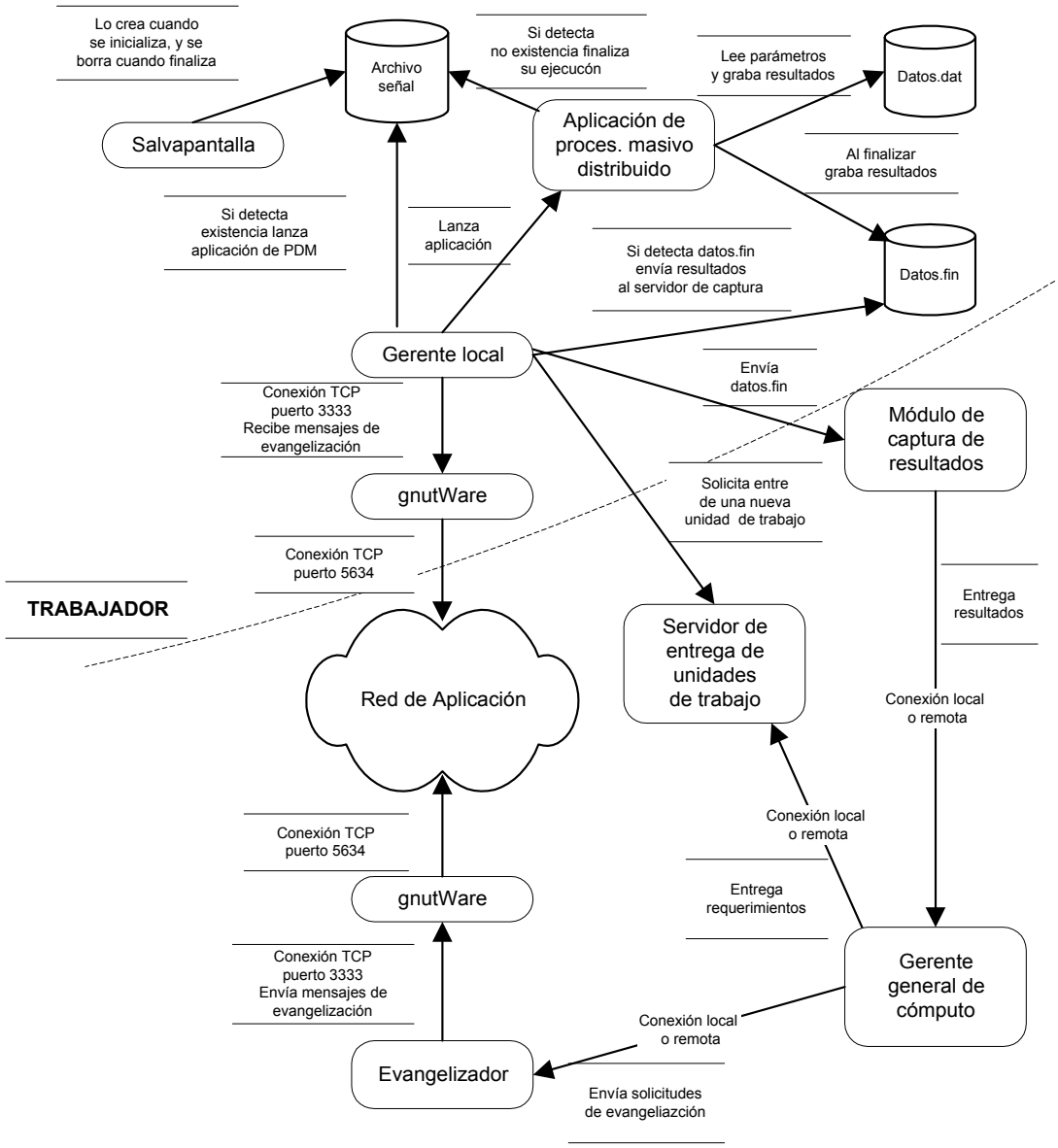
Por otro lado, registra en un servidor web local el software de aplicación correspondiente al proyecto. Se encarga de avisar al módulo evangelizador que genere mensajes de solicitud de servicio para el proyecto en cuestión.

Permanentemente, mantiene comunicación con el módulo de captura de resultados a los efectos de realizar seguimiento de las unidades de trabajo ya procesadas. Cuando el servidor de entrega de unidades de trabajo asigna una UT a un trabajador, le comunica al gerente general tal situación con el objetivo de que éste inicialice un temporizador de tiempo máximo de obtención de resultados. Vencido éste la UT se considera no procesada y vuelve a ser

colocada en una cola de UT a procesar con un identificador diferente. Por si llegasen resultados tardíos de una UT vencida se mantiene una tabla de identificadores no válidos por proyecto.

A los efectos de facilitar la especificación del proceso a realizar, el monitoreo de estado de avance del proyecto y la obtención de resultados, el módulo gerente cuenta con una interfase de usuario.

Todas las interacciones entre los componentes descriptos se resumen en el siguiente gráfico:



Vista de los distintos componentes del sistema y de sus relaciones

V.3.4 Comportamiento del sistema

En el Laboratorio de Redes de la Universidad Nacional de Luján se montó una configuración de prueba del prototipo de arquitectura del sistema de cómputo masivo propuesto. Se instalaron 8 trabajadores sobre máquinas de usuario final que poseían sistema operativo Microsoft Windows 98/2000. Todos los trabajadores cooperaron sobre un único proyecto.

En otra computadora personal, con sistema operativo Linux, se instaló el middleware GnutWare y el módulo de evangelización. Finalmente, sobre otra computadora con mismo sistema operativo se instalaron los módulos: gerente general de cómputo, entrega de unidades de trabajo y captura de resultados.

Se lanzó la aplicación distribuida, y los nodos activaron y desactivaron la aplicación salvapantalla en base a datos de comportamiento de usuario obtenidos empíricamente (ver sección V.5.4) . Durante tal período se observó que el sistema se comportó correctamente y alcanzó los objetivos funcionales propuestos.

V.4 Trabajo experimental exploratorio: Estimación de tiempo libre de CPU en una organización educativa.

El trabajo experimental tuvo como objetivo explorar la disponibilidad de tiempo libre de CPU en una organización educativa. La actividad se realizó a los efectos de poder estimar preliminarmente el potencial total de cómputo diario existente sobre un conjunto de máquinas personales de usuarios normales.

El lugar donde se llevo a cabo la experiencia fue la sede central de la Universidad Nacional de Luján. Se restringió el dominio de computadoras a evaluar solamente a equipos de usuario de carácter administrativo o académico con acceso a Internet, siendo un total de 166 computadoras. Por lo tanto, el universo en estudio no incluyó equipos de uso intensivo tales como computadoras de aula, equipos de investigación dedicados exclusivamente a tales tareas o equipos servidores de datos.

Se seleccionaron 8 usuarios voluntarios que corresponden aproximadamente al 5% del total, a las cuales se les instaló un software de captura de datos. El objetivo de tal programa es detectar durante cuánto tiempo se ejecuta la aplicación salvapantalla, por lo tanto se asume que en tal porción de tiempo la CPU se halla en estado ocioso.

El software de captura de datos es un salvapantallas construido especialmente para el proyecto y codificado en lenguaje Visual Basic. Su modo de operación es el siguiente: al activarse, luego que hay inactividad por cinco minutos, registra tal marca de tiempo. De igual manera se registra la marca correspondiente al evento de desactivación. Los datos se envían en tiempo real a un servidor de almacenamiento de datos por medio de una conexión utilizando el protocolo de transporte TCP.

Se tomaron datos durante diez días laborales. La siguiente tabla muestra cuáles fueron las fechas en cuestión como así también la cantidad de equipos de usuario que se encendieron en tales días. Este último dato se obtiene de revisar registros de actividad en equipos firewall y servidor proxy HTTP.

	Fecha	***Cantidad de equipos encendidos***
Martes	21-Oct	132
Miércoles	22-Oct	151
Jueves	23-Oct	158
Viernes	24-Oct	135
Lunes	27-Oct	152
Martes	28-Oct	156
Miércoles	29-Oct	152
Jueves	30-Oct	152
Viernes	31-Oct	142
Lunes	03-Nov	150

Tabla: cantidad de equipos encendidos por día.

La siguiente tabla muestra el tiempo ocioso diario, en minutos, de cada uno de los equipos que pertenecían a la muestra. A los efectos de este experimento se computaron como válidas aquellas activaciones de salvapantalla que fueron igual o superior al minuto. Tiempo igual a cero significa que el equipo no fue encendido tal día

	21-Oct	*22-Oct*	*23-Oct*	*24-Oct*	*27-Oct*	*28-Oct*	*29-Oct*	*30-Oct*	*31-Oct*	*03-Nov*
Usuario	***Martes***	***Miércoles***	***Jueves***	***Viernes***	***Lunes***	***Martes***	***Miércoles***	***Jueves***	***Viernes***	***Lunes***
Test1	118	425	389	453	349	169	222	332	275	170
Test2	399	245	70	755	754	715	0	203	182	233
Test3	386	230	72	251	321	187	115	147	97	169
Test4	242	0	122	223	161	490	262	581	407	438
Test5	309	170	184	185	204	214	228	392	217	218
Test6	129	156	197	171	344	163	202	183	398	156
Test7	177	89	212	148	331	276	304	0	355	327
Test8	204	164	332	199	208	178	225	128	157	210

Tabla: tiempo ocioso, en minutos.

Luego se computó el tiempo total ocioso de CPU diario en la organización para el periodo de tiempo en estudio. El análisis de realizo bajo tres criterios: a) criterio pesimista, el cual se basó en multiplicar para cada día el tiempo ocioso de CPU de la máquina que acuse menos tiempo de salvapantalla (distinto de cero) por la cantidad de equipos encendidos en tal día, b) criterio del promedio, donde el factor de multiplicación es múltiplo el promedio diario de ocio y c) criterio de la mediana, en este caso el factor es la mediana diaria de ocio. Los resultados se pueden observar –expresados en días- en la siguiente tabla:

	21-Oct	*22-Oct*	*23-Oct*	*24-Oct*	*27-Oct*	*28-Oct*	*29-Oct*	*30-Oct*	*31-Oct*	*03-Nov*
	Martes	***Miércoles***	***Jueves***	***Viernes***	***Lunes***	***Martes***	***Miércoles***	***Jueves***	***Viernes***	***Lunes***
Criterio pesimista	10,81	16,36	7,68	13,88	16,99	17,66	12,14	13,51	9,57	16,25
Criterio del promedio	22,50	19,39	21,64	27,95	35,26	32,39	20,56	25,94	25,74	25,01
Criterio de la mediana	20,44	17,51	20,90	19,78	34,41	21,72	23,59	20,37	24,26	22,29

Tabla: tiempo diario de ocio de CPU, en días.

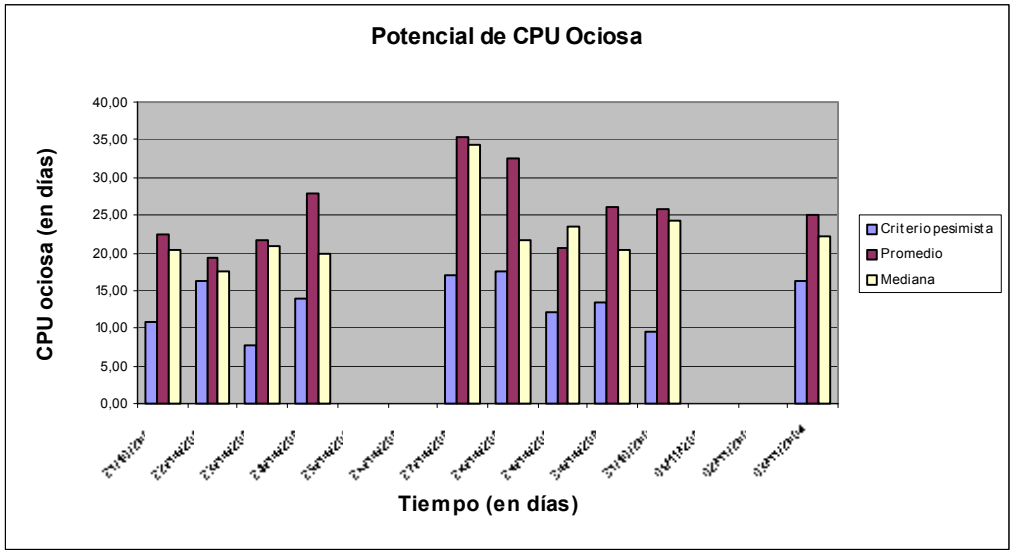


Gráfico: tiempo diario de ocio

De la tabla anterior se puede observar que es significativo el potencial disponible de cómputo en la organización estudiada, donde se puede notar claramente, que en promedio, la disponibilidad mínima de tiempo total de CPU para un día particular es equivalente a aproximadamente 20 días de 24 horas para un único equipo.

A continuación se computó la potencia horaria acumulada disponible del grupo de pruebas de computadoras, es decir aquellas a las cuales se le instaló el software salvapantalla. Para esto se evaluaron los cuatro primeros días del experimento. La idea es comparar, hora a hora, tal disponibilidad de CPU versus el tiempo de CPU brindado por una computadora totalmente dedicada.

Hora	*21 – Oct martes*	*22 – Oct miércoles*	*23 – Oct jueves*	*24 – Oct viernes*	*Una CPU dedicada*
1	**57,52**	**32,69**	73,77	103,87	60
2	152,18	**108,02**	146,08	261,12	120
3	270,13	**135,02**	285,98	475,65	180
4	469,10	**224,07**	354,47	562,30	240
5	687,27	303,31	553,47	746,73	300
6	993,22	432,31	744,54	1071,98	360

7	1289,54	548,41	811,22	1241,68	420
8	1633,27	663,76	931,78	1367,78	480
9	1841,69	753,32	1092,30	1562,20	540
10	2029,22	876,41	1247,98	1692,13	600
11	2119,99	1012,21	1357,87	1859,37	660
12	2184,99	1072,21	1443,87	2039,07	720
13	2230,99	1133,21	1503,87	2159,07	780
14	2241,62	1196,37	1519,10	2197,05	840
15		1233,86	1549,10	2344,07	

Tabla de tiempos de ocio acumulados

Nótese que en la tabla anterior se denotan en rnegrita aquellos períodos en que el grupo de computadoras ofreció menos ciclos de CPU que una computadora totalmente dedicada.

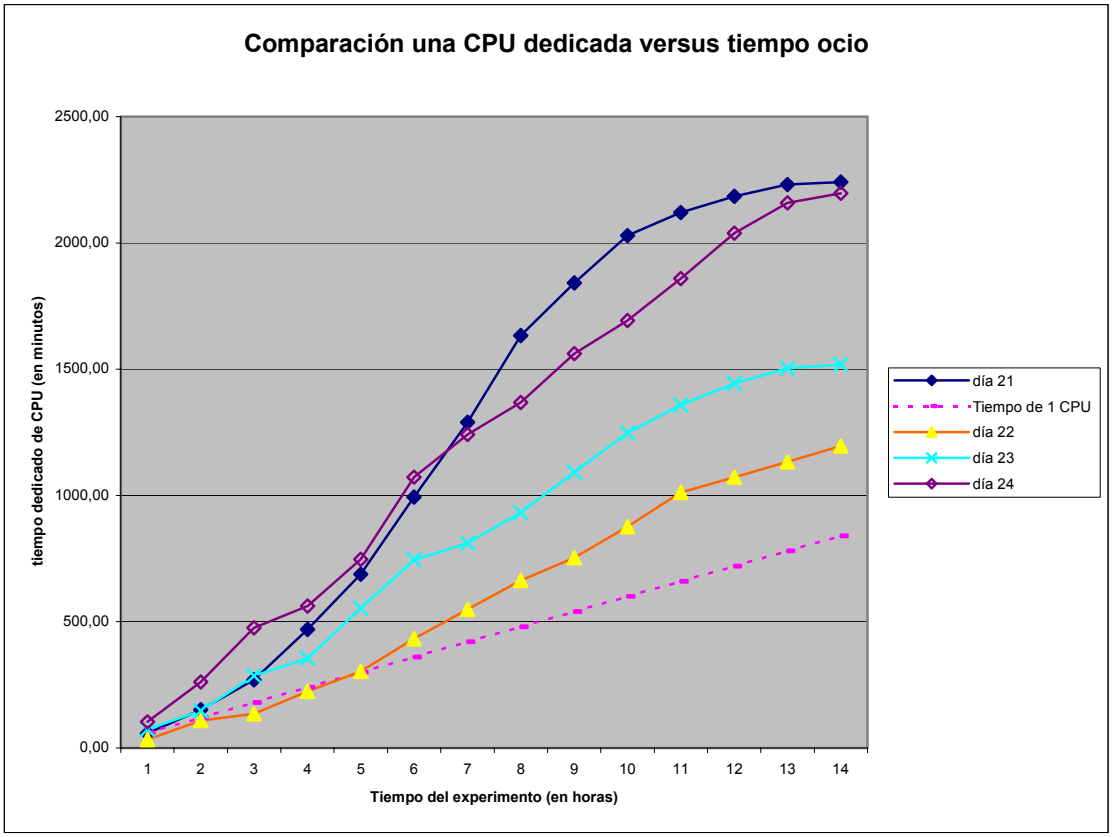


Gráfico: tiempos de ocio acumulados

La curva correspondiente a las líneas de puntos representa el tiempo de CPU aportado por una CPU totalmente dedicada. Cualquier valor de otra serie de datos que supere a la línea punteada indica que en ese instante de tiempo el tiempo de CPU brindado por el juego de computadoras de prueba es mayor, y por lo tanto se tiene una mayor potencia de cómputo.

Capítulo VI

VI Conclusiones y Trabajos Futuros

VI.1 Conclusiones

La tecnología P2P es una propuesta alternativa que permite replantearse la forma de ofrecer soluciones en el área de procesamiento distribuido. Se ha validado, a modo de prueba de concepto, la factibilidad de construcción de un sistema de tal tipo, inédito desde el punto de vista de la utilización de un canal multicast.

El middleware GnutWare presentado brinda una interfaz de acceso a una red P2P sencilla y robusta para implementar servicios distribuidos sobre redes compañero a compañero. Los nodos de la red Gnutella solamente realizan tareas de encaminamiento sin prestar servicios de usuario final, mientras que la aplicación brinda un servicio de usuario final totalmente independiente de ésta.

Se ha demostrado que en organizaciones, con ciertas características particulares, se pueden resolver problemas de cómputo intensivo a un costo reducido y en tiempos de ejecución aceptables. En una Intranet el costo es casi despreciable, si se compara con el precio de un cluster dedicado o un supercomputador.

VI.2 Trabajos futuros

En primer lugar se pretende realizar una estudio estadístico completo de disponibilidad de tiempos de CPU en distintos tipos de organizaciones, a los efectos de poder caracterizar los patrones de uso en diferentes situaciones y estimar de manera confiable el potencial de cómputo de una organización particular. De este trabajo puede surgir una metodología estándar de estimación de tiempos de CPU libre basados en características particulares de las organizaciones.

A partir del prototipo presentado, generar una versión estable de producción de los distintos módulos involucrados. Se contempla utilizar el lenguaje de programación Java debido a sus características de portabilidad y seguridad.

Una vez que se obtenga la versión estable se prevee una serie de tests benchmark que permitan comparar a P2P-Flops con infraestructuras similares. Tales resultados permitirán establecer la equivalencia en poder de cómputo entre el sistema propuesto bajo las condiciones de una organización particular contra una solución tipo cluster de n equipos. Tal actividad permitiría conocer e antemano cual es la inversión en recursos que debe realizar una organización para equiparar modelos de computación distribuida.

En lo relativo a seguridad es necesario definir políticas de autenticación de trabajadores y clientes, estrategias de validación de autenticidad de resultados y controles a nivel de entorno de ejecución del programa en los trabajadores a los efectos de chequear piezas de código maligno.

Si bien el modelo propuesto ha sido pensado para operar dentro de los límites de las redes de una organización (tipicamente su intranet), se debería estudiar cuales son las condiciones para implementar tal infrestructura a nivel de una red global.

Bibliografía Comentada

[.NET] Microsoft Corp. Proyecto .NET. Disponible en http://www.microsoft.com/net

[Aberer] Aberer, K., Cudré-Mauroux, O., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., Schmidt, R. y Wu J. "Advanced Peer-to-Peer Networking: The P-Grid System and its Applications. A ser publicado en PIK Journal". A ser publicado en PIK Journal.

Se presenta un sistema colaborativo denominado PGgrid, que opera sobre una arquitectura P2P. Se discuten varías áreas de aplicación donde puede ser efectivo, más alla del compartimiento de archivos. Se presenta una comparación de funcionamiento con el sistema FreeNet

[Adar] Adar, E. y Huberman, B. "Free Riding on Gnutella". First Monday, Vol. 5, N. 10, 2000.

Se presenta un estudio basado en la captura de mensajes de la red gnutella. Se concluye que el 70% de los usuarios no comparte archivos y que cerca del 50% de todas las respuestas son retornadas por el 1% de los nodos top que comparten recursos.

[AIM] American Online Inc. Página principal del software AIM, AOL Instant Messenger. http://www.aim.com

[Akamai] Akamai Technologies, Inc. Disponible en http://www.akamai.com/

[Andersen] Andersen, D., Balakrishnan, H., Kaashoek, M. F., and Morris, R. Resilient overlay networks. In Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01) Chateau Lake Louise, Canada, Octubre 2001.

RON es una arquitectura que permite a las aplicaciones distribuidas detectar y recobrarse de interrupciones y períodos de perfomance degradada por varios segundos. Se presentan evaluaciones de perfomance sobre aplicaciones que trabajan con el método mencionado.

[Anderson] Anderson, D., Cobb, J., Korpela, E., Lebofsky, M. y Werthimer, D. "SETI@home: An Experiment in Public-Resource Computing". Communications of the ACM, v45n11, 56, 2002.

Documento que describe el proyecto de análisis de señales capturas por un radio telescopio. Se describe su diseño e implementación, basada en una estrategia de cómputo masivo distribuido. Se valida la idea de utilización de capacidad excedente de recursos en computadoras de usuario final.

[Anderson-A] Anderson, D. Public Computing: Reconnecting People to Science. Conference on Shared Knowledge and the Web, Residencia de Estudiantes, Madrid, Spain, Nov. 2003.

Documento que explica las bondades de la computación masiva distribuida, describe aspectos sociales y presenta proyectos en curso. El autor informa brevemente acerca de su proyecto BOINC.

[Arnold] Arnold, D., Dongarra, J. "Developing an Architecture to Support the Implementation and Development of Scientific Computing Applications," to

appear in Proceedings of Working Conference 8: Software Architecture for Scientific Computing Applications, Ottawa, Canada, October 2-6, 2000.

En este artículo se describe la infraestructura NetSolve. Sistema cliente-servidor que posibilita el uso de librerías matemáticas en un ambiente distribuido. Los usuarios acceden de forma transparente a recursos de hardware y software distribuidos a lo largo de una red.

[Arnold-A] Arnold, D., Casanova, H., Dongarra, J. "Innovations of the NetSolve Grid Computing System," Concurrency: Practice and Experience, Volume 14, no. 13-15, pp. 1457-1479, 2002.

Artículo que describe el estado del proyecto y las últimas características de la infraestructura NetSolve

[Baker] Baker, M., Buyya, M. y Laforenza, D.. "The Grid: International Efforts in Global Computing". SSGRR 2000 The Computer & eBusiness Conference, l`Aquila, Italia, Julio. 2000

Documento que presenta el estado del arte en computación bajo el modelo grid, adicionalmente se exponen varias aplicaciones que lo implementan.

[Baratloo] Baratloo, A.; Kraul, M.; Karl, H. y Kedem Z. An Infrastructure for Network Computing with Java Applets

*Artículo en el cual sus autores presentan al proyecto s Knitting*Factory, *como una infraestructura destinada a facilitar la computación basada en ambientes web. Se describe la técnica de comunicación applet a applet, la cual reemplaza la presencia de agentes intermediarios de comunicación.*

[Barcellos] Barcellos M. y Da Costa, C. Java como Ferramenta para Programação Paralela e Distribuída. Book Chapter & Tutorial, ERI/SC 2003 - Escola Regional de Informática/SC, SBC, 2003, Chapter 2, pp.41-63, Lages, 29-31, 2003.

Capítulo de libro donde se tratan las ventajas proporcionadas por el lenguaje Java en ambientes de cómputo cooperativo distribuido. Se describen, técnicamente, cuales son las herramientas que ofrece el lenguaje a los programadores de aplicación.

[Barkay] Barkay, D. "An introduction to Peer-to-Peer Computing". Intel Developer UpdateMagazine. Febrero, 2000.

Documento de divulgación que explica los fundamentos del modelo compañero a compañero.

[Berners] Berners-Lee, T.; Fielding, R. y Frystyk, H. RFC 1945. Hypertext Transfer Protocol – HTTP/1.0. Mayo 1996.

Documento oficial del Network Working Group de Internet que especifica el protocolo de transferencia de hipertexto HTTP v1.0.

[Bidson] Bildson, G. y Rohrs, C. "An Extensible Handshaking Protocol for the Gnutella Network," Reporte Técnico. Empresa Lime Wire LLC

Propuesta de modificaciones al protocolo gnutella en el establecimiento de una conexión.

[Bordignon] Bordignon, F. y Tolosa, G. "Gnutella: Distributed System for Information

Storage and Searching. Model Description". JIT- Journal of Internet Technology. Taipei (Taiwan) Vol. 2, No. 5. 2001. Disponible en http://www.tyr.unlu.edu.ar/TYR-publica/paper-final-gnutella-english-v2.pdf

Descripción técnica del protocolo Gnutella. Tal trabajo constituye un documento pionero, dado que en el momento de su publicación solo existía información fragmentada acerca de su funcionamiento.

[Bornet] Bornet, 2001. "Varios proyectos científicos permiten una colaboración pública voluntaria gracias a Internet". Bornet, Revista de divulgación científica. Disponible en http://www.bornet.es/news/ Biologia_y_Biotecnologia/260601194425.shtml

Artículo perteneciente a una revista de divulgación científica que trata sobre proyectos de cómputo masivo colaborativo sobre Internet.

[Bretch] Brecht, T.; Sandhu, H.; Shan, M. y Talbot, J. ParaWeb: Towards World-Wide Supercomputing. Proceedings of the Seventh ACM SIGOPS European Workshop on System Support for Worldwide Applications, 1996.

Artículo donde se describe a ParaWeb, proyecto en el área de cómputo distribuido que fue diseñado para operar tanto en una intranet como en Internet. Su objetivo es que los usuarios ejecuten programas programas de tipo serial sobre computadoras más veloces o programas construidos bajo un paradigma de programación paralela sobre una variedad de equipos heterogéneos.

[Bricklin] Bricklin, D. "Thoughts on Peer-to-Peer". 2000. Disponible en http://www.bricklin.com/p2p.htm.

Artículo de divulgación que explica las propiedades que distinguen a los nodos compañeros del sistema Napster, y donde puede ser exitosa la implementación de aplicaciones P2P.

[Buyya] Buyya, R. (editor), High Performance Cluster Computing: Architectures and Systems, Vol. 1 y 2, Prentice Hall, NJ, USA, 1999

Libro que contiene una serie de artículos de investigación que aborda el tema de las infraestructuras destinadas a cómputo masivo.

[Cappello] Cappello, P., Christiansen, B. O., Ionescu, M. F.,, Neary, M. O., Schauser, K. E., Wu, D.: Javelin: Internet-Based Parallel Computing Using Java. ACM Workshop on Java for Science and Engineering Computation. (June 1997)

Javelin es una infraestructura destinada al cómputo en un ambiente global. Se definen tres tipos de entidades participantes: intermediarios, clientes y hosts. Donde un cliente es un proceso que busca recursos computacionales, un host es un nodo que ofrece recursos al sistema y un intermediario es un proceso que coordina la demanda y la oferta.

[Clarke] Clarke, I., Sandberg, O., Wiley B., y Hong. T. "Freenet: A Distributed Anonymous In-formation Storage and Retrieval System". In Hannes Federrath, editor, Proceedings of the-Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, 2000.

Descripción técnica de Freenet, sistema de búsqueda y almacenamiento de contenidos distribuidos, tolerante a la censura.

[Clip2] Clip2.com Inc. "Bandwidth Barriers to Gnutella Network Scalability". Disponible en http://www.clip2.com/dss_barrier.html,

Documento que trata el problema de la escalabilidad de la red gnutella. En base a capturas de mensajes se presentan estadísticas de uso.

[Clip2-A] Clip2.com Inc. "Clip2 Statistics". Disponible en http://www.clip2.com

Estadísticas acerca del uso de la red gnutella. Se presentan una serie de mapas de topología que muestran como está constituida la red.

[Clip2-B] Clip2.com Inc. "Clip2 Reflector Overview". Disponible en http://dss.clip2.com/reflector.html

Presentación técnica de una solución al protocolo gnutella, tendiente a distribuir de forma balanceada las conexiones y el tráfico de mensajes de usuarios.

[Coulouris] Coulouris, G., Dollimere, J. y Kimberg, T. "Sistemas Distribuidos. Conceptos y Diseño". Addison Wesley, 3ra ed. 2001

Libro orientado a estudiantes de grado que aborda la temática de los sistemas distribuidos.

[Cyrus] Cyrus Patel, "Architecture Distributed.net". Distributed.net. Correo electrónico recibido el 31 de octubre de 2001.

Información técnica acerca del proyecto Distributed.net.

[Dabek] Dabek, F., Brunskill, E., Kaashoek, M., Karger, D., Morris, R., Stoica, I. y Balakrishnan, H. "Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service". Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), Schloss Elmau, Germany, May 2001.

Se presenta a Chord, un servicio de consulta distribuida, altamente escalable, basado en tablas de dispersión.

[Dist] Proyecto Distributed.Net. Disponible en: http://www.distributed.net

[DN] Distributed.net, 2001. "La Organización". Distributed.net. Disponible en http://www.distributed.net/

[DN-A] Distributed.net, 2001. "Como Ayudar". Distributed.net. Disponible en http://www.distributed.net/howto.html.es

[DN-B] Distributed.net, 2001. "Projects". Distributed.net. Disponible en http://www.distributed.net/projects.html.en

[DN-C] Faq distributed.net, 2000. "What is the RC5-64 Project?". Distributed.net. Disponible en http://n0cgi.distributed.net/faq/cache/28.html

[DN-D] Distributed.net, 2000. "Proyecto OGR". Distributed.net. Disponible en Disponible en http://n0cgi.distributed.net/OGR/index.html

[DN-E] Faq distributed.net, 2000. “How exactly does all this work?”. Distributed.net. Disponible en http://n0cgi.distributed.net/faq/cache/51.html

[Dougherty] Dougherty, D. “All The Pieces of PIE”, libro "2001 P2P Networking Overview", capítulo 1, Ed. O´Reilly, 2001

Primer capítulo de un libro que trata sobre el resurgimiento de la arquitectura P2P, especialmente en Internet. Se hace referencia al poder existente en las computadoras de usuario y la forma en que las aplicaciones P2P hacen uso de él.

[DSS1] Distributed Search Services. "Gnutella: To the Bandwidth Barrier and Beyond." 2000. Disponible en http://dss.clip2.com.

[DSS2] Distributed Search Services. "The Gnutella Protocol Specification v0.4". 2000. Disponible en http://dss.clip2.com.

Especificación técnica del protocolo gnutella.

[Enslow] Enslow, P.H. "What is a `distributed' data processing system?" Computer, 11(1):13-21, 1978.

[Ferrari] Ferrari, A. JPVM: Network parallel computing in Java. Technical Report CS-97-29, Department of Computer Science, University of Virginia, 1997.

Documento que presenta a JPVM, una interfaz que permite a aplicaciones Java usar el software de Máquina Virtual Paralela (PVM). Tal infraestructura puede ser utilizada para resolver problemas computacionales de gran magnitud utilizando el poder agregado y memoria de muchas computadoras.

[FIPA] FIPA. "FIPA Peer-to-Peer Positioning Paper". Technical Report F-OUT-00076, Foundation for Intelligent Pyhsical Agents, Diciembre 2000.

Reporte técnico donde se describen que son las redes compañero a compañero, sus componentes y sus aplicaciones.

[Fragati] Fragati, H.; Tolosa, G. Y Bordignon, F. Experiencia en Cómputo Masivo Distribuido Utilizando Applets Java. Poster presentado en Jornadas de la Ciencia y Tecnología. Universidad Nacional de Luján. 2002.

Poster en el cual se presenta una infraestructura de sistema de cómputo masivo en el ámbito del espacio web. Se basa en la inserción de applets Java en páginas solicitadas por navegantes, a los efectos que se ejecute un cierto proceso mientras se lee tal página.

[JXTA] Sun Microsystems Projecto JXTA. Disponible en. http://www.jxta.org.

[JXTA-A] JXTA v2.0 Protocols Specification. Project JXTA. The Internet Socity.

Especificaciones técnicas de los protocolos involucrados en el sistema JXTA.

[INTEL] INTEL. "Peer-to-Peer-Enabled Distributed Computing: making the financial serviecs enterprise more productive". Intel White Paper. 2001

Documento que explica como aplicaciones P2P pueden igualar y hasta superar el poder cómputo de una supercomputadora. Se presenta un caso de estudio basado en una aplicación denominada LiveCluster que posibilita implementar un sistema de cómputo masivo distribuido.

[IRIS] Red IRIS. Propuesta de estructuración de un programa de e-ciencia.. http://irisgrid.rediris.es/irisgrid/doc/Propuesta.pdf

Documento en el cual se plasma una propuesta de creación de un programa de apoyo a la ciencia española. Se basa en la creación y puesta en marcha de una infraestructura grid en el territorio español.

[Korpela] Korpela, E., Werthimer, D., Anderson, D., Cobb, J. y Lebofsky. M. "SETI@HOME Massively Distributed Computing for SETI". Computing in Science and Engineering, 3(1):78-83, 2001.

Se expone el proyecto de búsqueda de señales inteligentes extraterrestres SETI. El cual está basado en obtener un fuerte poder de cómputo tomando ciclos de CPU ociosos de computadoras de usuario final.

[FFnet]. FastForward Networks' broadcast overlay architecture, 2000. Disponible en www.ffnet.com/pdfs/BOA-whitepaperv6.PDF.

[Fight] Proyecto FightAIDS@Home. Disponible en http://www.fightaidsathome.com

[FIPA] FIPA. "FIPA Peer-to-Peer Positioning Paper". Technical Report F-OUT-00076, Foundation for Intelligent Pyhsical Agents, Diciembre 2000.

[Folding] Proyecto Folding@Home.
Disponible en http://www.stanford.edu/group/pandegroup/Cosm/

[Foster] Foster, I. 2000. "Internet Computing and the Emerging Grid". Nature, diciembre 7, 2000.

[Foster-A] Foster, I. and Kesselman, C. (editors), The Grid: blueprint for a future computing infrastructure, Morgan Kaufmann Publishers, USA, 1999.

Es el primer libro en describir la tecnología grid. Se considera literatura fundamental a la hora de emprender un proyecto.

[Foster-B] Foster, I.; Kesselman, C. y Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Lecture Notes in Computer Science, vol. 2150, 2001.

Artículo que define la tecnología grid. Presenta los nuevos desafíos para su evolución. Se presenta un modelo de arquitectura y su paralelo con el juego de protocolos TCP/IP. Se expone la necesidad de contar próximamente con protocolos "intergrids".

[Freeh] Freeh, V. "Anatomy of a Parasitic Computer ". Dr. Dobb's Journal, (332):63-67, January 2002.

[Globus] Globus: Sitio oficial del proyecto open source "Toolkit Grid". http://www.globus.org

[Golle] Golle, P., Leyton-Brown, K., Mironov, I. y Lillibridge, M. "Incentives for Sharing in Peer-to-Peer Networks"

[GPU] http://gpu.sourceforge.net/ Sitio oficial del proyecto Gigaglobal Proccessing Unit

El proyecto open source Gigaglobal Proccessing Unit es una plataforma que permite que una aplicación utilice ciclos de CPUs de distintas computadoras de usuario final conectadas a Internet. Para lograr este objetivo, la aplicación se conecta a una red P2P, del tipo Gnutella, y dispone de mensajes específicos para enviar, recibir y ejecutar trabajos computacionales.

[Groove] Página principal del proyecto groove http://www.groove.net

[Hayes] Hayes, Brian, Marzo-abril 1998. "Collective Wisdom". American Scientist.. Disponible en http://www.amsci.org/amsci/issues/Comsci98/compsci1998-03.html

[Hotline] Hotline Communications Ltd. Online. Disponible en http://www2.bigredh.com/hotline3/

[Hwang] Hwang, K., Briggs y Fayé A. "Arquitectura de Computadoras y Procesamiento Paralelo", pp. 5-9, 44-54, Editorial Mcgraw-Hill, ISNN 968-422-344-7. 1990

[ICQ] ICQ Inc. Página principal del software ICQ. Disponible en http://www.icq.com

[Janotti]. Jannotti, J., Gifford, D. K., Johnson, K. L., Kaashoek, M. F., y Jr., J. W. O. "Overcast: Reliable multicasting with an overlay network". En *Proc. 4th USENIX OSDI.* Octubre 2000, pp. 197-212.

[Johnston] W.Johnston, D.Gannon & W.Nitzberg, "Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid", Proc. 8th Symposium on HPDC, IEEE Computer Society Press (1999)

Documento que describe el proyecto "Information Power Grid" de la NASA. Su objetivo fue construir un medioambiente de cómputo y almacenamiento de datos totalmente distribuido.

[Korpela] Korpela, E., Werthimer, D., Anderson, D., Cobb, J. y Lebofsky. M. SETI@HOME Massively Distributed Computing for SETI. Computing in Science and Engineering, 3(1):78-83, 2001.

[Kunwadee] Sripanidkulchai, K. "The popularity of Gnutella queries and its implications on scalability". , O'Reilly's Network www.openp2p.com . 2001. Disponible en http://www.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html

[Loewe] Loewe, L. "evolution@home: Experiences with work units that span more than 7 orders of magnitude in computational complexity", Bal et al. (eds) Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2002), pp. 425-431. Berlin, Germany

[Minar] Minar, N. y Hedlund, M. "A Network of Peers. Peer-to-Peer Models Through the History of the Internet",. A. Oram, editor, ", "Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology" capítulo 1. O'Reilly & Associates, Marzo 2001.

[Morrison] Morrison, R. Cluster Computing Theory: Architectures, Operating Systems, Parallel *Processing &* Programming Languages. Documento distribuido bajo licencia GNU. 2003.

Libro digital que describe la tecnología de clusters. Muestra experimentos sobre distintas configuraciones. Describe como debe emprenderse un proyectos de establecimiento de un centro de cálculo masivo.

[MSN] Microsoft. Proyecto de software MSN Messenger. Disponible en http://messenger.msn.com/

[Napster] Napster Inc. Página principal del software Napster http://www.napster.com

[Palmer] Palmer, J. Libro: Electronic Mail, Ed. Artech House Publishers, ISBN: 0-89006-802-X. 1995

[Pew] Pew Internet & American Life. Disponible en [http://www.pewinternet.org/reports/toc.asp?Report=16]

[Pointera] Página principal del proyecto Pointera http://www.pointera.com

[Postel] Postel. J. "Simple Mail Transfer Protocol". RFC 821. 1982

Documento oficial del Network Working Group de Internet que describe el protocolo de transferencia de correo electrónico (SMTP) en Internet.

[Ratnasamy] Ratnasamy, S., Francis, P., Handley, M., Karp, R. y Schenker, S. "A Scalable Content-Addressable Network," Proc. ACM SIGCOMM, San Diego, CA, Agosto 2001.

En este documento se presenta a CAN, él cual es un espacio virtual de n dimensiones donde nodos de una red conocen su zona de pertenencia. De este modo es posible, eficientemente, recuperar recursos compartidos. El sistema es tolerante a fallas, robusto, con estrategias de auto-organización y altamente escalable.

[RFC882] Mockapetris, P. "Domain Names - Concepts and Facilities". RFC 882. 1983.

Documento oficial del Network Working Group de Internet que trata sobre conceptos básicos que definen el servicio de nombres de Internet (DNS). Se presentan casos de uso, detalles de implementación y una especificación completa del protocolo.

[RFC974] Partridge, C., "Mail Routing and the Domain System", RFC 974, 1986.

Documento oficial del Network Working Group de Internet que describe como los sistemas de correo electrónico rutean mensajes de usuarios basados en información del sistema de nombres de dominio DNS.

[RFC977] Kantor, B. y P. Lapsley, P. "Network News Transfer Protocol" . RFC 977. 1986

Documento oficial del Network Working Group de Internet que explica el protocolo de transferencia de noticias Internet NNTP. Se describe su envío, distribución, recuperación y posting. Desde el punto de vista de los servidores de noticias se trata la problemática de la indexación, de las referencias cruzadas y el vencimiento de las noticias.

[RFC1034] Mockapetris, P., "Domain Names Concepts and Facilities", RFC 1034, USC/Information Sciences Institute, 1987.

Documento oficial del Network Working Group de Internet que actualiza al RFC 882.

[RFC1035] Mockapetris, P., "Domain Names Implementation and Specification", RFC 1035, USC/Information Sciences Institute, November 1987.

Documento oficial del Network Working Group de Internet que actualiza al RFC 882.

[RFC1036] Horton, M. y Adams, R. " Standard for Interchange of USENET Messages". RFC 1036. 1987.

Documento oficial del Network Working Group de Internet que describe como se realiza la transferencia de mensajes de noticias USENET.

[Rohrs] Rohrs C. "LimeWire: Ping Pong Scheme". Lime Peer Technologies LLC (LimeWire), Disponible en http://www.limewire.com/index.jsp/pingpong

Documento técnico donde se trata la revisión de la técnica de mensajes Ping/Pong sobre el protocolo gnutella. Se propone una variante, basada en la utilización de una memoria cache, orientada a una reducción en el uso de ancho de banda.

[Rojas] Ossa Rojas, C. "El fenómeno del MP3 y el caso Napster". Revista Electrónica del Derecho Informático. N. 32, marzo 2001. Disponible en http://publicaciones.derecho.org/redi/No._32_-_Marzo_del_2001/4

Artículo que trata sobre las implicaciones legales, de la distribución de contenidos protegidos con leyes de copyright y derechos de autor.

[Rowstron] Rowstron, A. y Druschel, P. "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001

Según sus autores, Pastry es un medioambiente genérico, escalable y eficiente para el desarrollo de aplicaciones P2P. Los se auto-organizan sobre una red de recubrimiento tolerante a fallas que opera sobre Internet. Este documento aborda detalles técnicos sobre la arquitectura de Pastry.

[Saroiu] Saroiu, S., Gummadi, P. y Gribble, S. "Measurement study of peer-to-peer file sharing systems". Tech Report UW-CSE-01-06-02, University of Washington,2001.

Se presenta un estudio de los sistemas Napster y Gnutella, caracterizando su población de usuarios. Se concluye que existe una alta hetereogenidad y una baja cooperación entre participantes de las comunidades.

[Schoeder] Schroeder ,M. "A state-of-the-art distributed system: Computing with BOB", Mullender editor, "Distributed Systems", capítulo 1, 1995

En el documento se exploran las características, fortalezas y debilidades de los sistemas distribuidos. Se describe un modelo de sistema distribuido que puede realizar un mejor trabajo que los sistemas de uso general existentes.

[Shirky] Shirky, C. "Listening to Napster", A. Oram, editor "Peer to Peer. Harnessing the power of disruptive technologies", capítulo 2, O'Reilly & Associates. 2001

Shirky aborda el tema de los recursos disponibles en máquinas de usuarios de Internet, los califica como la materia negra o el poder de los bordes. Afirma que los sistemas P2P hacen uso de tales recursos de una forma eficaz, opuesta a los sistemas cliente-servidor.

[Shoch] Shoch, J. y Hupp, J. ``The 'Worm' Programs - Early Experience with a Distributed Computation", CACM pp172-180, March, 1982.

Documento pionero que presenta una serie de experimentos, en el laboratorio Xerox PARC, sobre programas móviles (worns) y sus posibles aplicaciones en cómputo distribuido. Se explora la utilización de ciclos de CPU libres.

[Singhal] Singhal, M. y. Casavant, T. "Distributed computing systems". Computer, 24(8):12-14, 1991.

Es una columna de editor invitado donde se presentan tendencias sobre cómputo distribuido.

[Stoica] Stoica, I., Morris, R. Karger, D., Kaashoek, M. y Balakrishnan, H. "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," Proc. SIGCOMM, San Diego, CA, Agosto 2001.

Chord es un servicio escalable de búqueda implementado sobre un conjunto de nodos compañeros, basado en tablas de dispersión distribuidas. Este documento presenta su arquitectura.

[Sunsted] Sunsted, T. "The practice of peer-to-peer computing: Introduction and history". IBM DeveloperWorks. Disponible en http://www-106.ibm.com/developerworks/java/library/j-p2p/

Documento que presenta algunos datos sobre los orígenes de las redes compañero a compañero.

[Suthar] Suthar,P. y Ozzie, J. "The Groove Platform Architecture".. 2000. Disponible en http://dev-zone.groove.net/library/Presentations/ GrooveApplicationArchitecture.ppt.

Presentación de la plataforma Groove, destinada a la creación de entonos colaborativos basados en una red P2P.

[Swarmcast] Página principal del software Swarmcast http://www.swarmcast.com

Swarmcast es un sistema que utiliza una red P2P a los efectos de distribuir rápida y eficiente, contenidos de gran tamaño.

[Tanenbaum] Tanenbaum, A.S y Van Renesse, R. "Distributed Operating Systems". ACM Computing Surveys, Vol. 17, No. 4. 1985.

Libro dirigido a estudiantes de grado que trata sobre sistemas operativos distribuidos.

[Tannenbaum-A] Tannenbaum, T.; Wright, D.; Miller, K. y Livny, M. "Condor - A Distributed Job Scheduler", in Thomas Sterling, editor, Beowulf Cluster Computing with Linux, The MIT Press, 2002. ISBN: 0-262-69274-0

Se describe la aplicación Condor. La cual es un planificador de procesamiento distribuido de tareas de cómputo. Una característica particular del proeycto, es que los recursos de CPU destinados a una tarea son ociosos con respecto a su usuario.

[Tolosa] Tolosa, G., Bordignon, F. y Rodriguez, C. "Una Arquitectura Cooperativa Destinada a la Distribución de un Servicio de Transmisión de Radio por Internet". Memorias de Informática 2002 - I Congreso Internacional de

Tecnologías y Contenidos Multimediales en Ambientes Web. La Habana, Cuba, 2002.

Se define una propuesta de arquitectura de red a nivel aplicación que sirva de base a un servicio de retransmisión de un flujo de radio por Internet. Se propone una red de propagación de mensajes y respuestas, conformada por todos aquellos usuarios que son afines a escuchar flujos de audio.

[Tolosa-A] Tolosa G. y Bordignon, F. "Propuesta de Arquitectura Cooperativa Destinada a un Servicio de Búsqueda Distribuida en el Espacio Web. Una Alternativa a los Motores de Búsqueda Tradicionales". Poster Jornadas de la Ciencia y Tecnología. Universidad Nacional de Luján. Mayo 2001. Disponible en http://www.tyr.unlu.edu.ar/TYR-publica/poster-2001-bd.doc

Poster que presenta una arquitectura que tiene por finalidad implementar un servicio de búsqueda web cooperativa, sobre un conjunto de motores de consulta que están conectados a una red P2P.

[Tolosa-B] Tolosa G. y Bordignon, F. "Red a Nivel Aplicación como Middleware P2P para Soportar Servicios Distribuidos sobre Internet". CACIC 2002. Poster VIII Congreso Argentino de Ciencias de la Computación. 2002.

Poster que presenta el proyecto middleware P2P GnutWare. Se destaca su arquitetura y funcionalidad de sus componentes principales.

[Torres] Torres Jiménez, J. y Rodríguez Tello E. "Conceptos de Cómputo Paralelo", pp. 1-23, 67-83, Editorial Trillas, ISBN 968-24-6222-3 Preedición mayo 2000

Libro de estudio, diseñado para alumnos de grado, que trata sobre tópicos relacionados con el cómputo paralelo. Se utilizó en la sección de introducción al cómputo distribuido.

[Touch]. Touch, J., y Hotz, S. The X-Bone. In Proc. Third Global Internet Mini-Conference in conjunction with Glo becom '98, Sydney, Australia, Nov. 1998.

X-Bone es un sistema que permite un rápido desarrollo automatizado y una facil administración de redes de recubrimiento..En el documento se describe la arquitectura de X-Bone.

[UD] United Devices, 2001. "What is distributed computing?". United Devices. Disponible en: http://www.ud.com/company/dc/

Documento que explica el concepto de computación distribuida por parte de la empresa United Devices, la cual ha implementado varios proyectos comerciales de cómputo masivo distribuido.

[Waldman] Waldman, M., Rubin, A. y Cranor, L. "Publius: A robust, tamper-evident, censorship-resistant, web publishing system". Proc. 9th USENIX Security Symposium, pp. 59--72. 2000.

Publius es un sistema de publicación de contenidos en el espacio web, que tiene la característica de estar dotado de mecanismos que lo hacen resistente a la censura o a la modificación de contenidos.

[Wang] Wang, X., Siong Ng, W., Chin Ooi, B., Lee Tan K., y Zhou A. "BuddyWeb: A P2P-based Collaborative Web Caching System (Position Paper)". Peer to Peer Computing Workshop (Networking 2002). 2002

En este documento se describe un sistema de cache web colaborativo basado en el modelo P2P. Todos los recursos que esten en la memoria cache de las computadoras de usuario, se comparten con el resto de la comunidad al cual pertenecen. Se presenta una nueva estrategia de ruteo de mensajes.

[Yang] Yang, B. y García-Molina, H. "Comparing Hybrid Peer-to-Peer Systems". Proceedings of the 27th VLDB Conference, Roma, Italia, 2001.

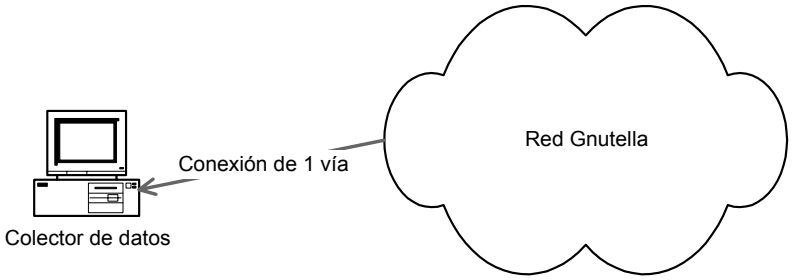
Se estudian tópicos relativos al diseño de sistemas escalables basados en arquitectura P2P. Se enfoca el tratamiento hacía los de construcción híbrida. Se realizan análisis cuantitativos de perfomance.

[Zhao] Zhao, B., Kubiatowicz, J. y Joseph, A. "Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing". UC Berkeley Computer Science Division, Report No. UCB/CSD 01/1141, Abril 2001.

Se presenta una infraestructura denominada Tapestry que tiene por misión brindar servicios de localización de recursos y ruteo de nemsajes sobre una red de recubrimiento.

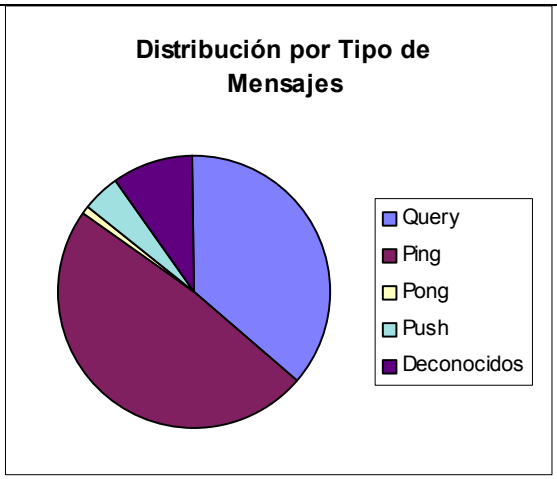
Anexo 1 - Estudio del contenido de mensajes de una red Gnutella

A los efectos de analizar el uso de la red Gnutella por parte de sus usuarios, se diseñó un experimento. El día 10 de diciembre del año 2000, por la tarde, se hizo funcionar durante cuatro horas un programa cuya función, solamente, era recibir un flujo de datos de un gnodo remoto[21]. Se clasificaron y analizaron todos los mensajes recibidos.



Clasificación de los mensajes recibidos:

Query:	5500
Ping:	7347
Pong:	172
Push:	671
Desconocidos:	1482
Total de mensajes:	15172



Nota: En esta experiencia no se descriminaron los mensajes query_hits, por lo que están incluidos en la clase desconocidos.

[21] El programa solo operaba de modo pasivo y no enviaba a la red ningún mensaje propio o de terceros.

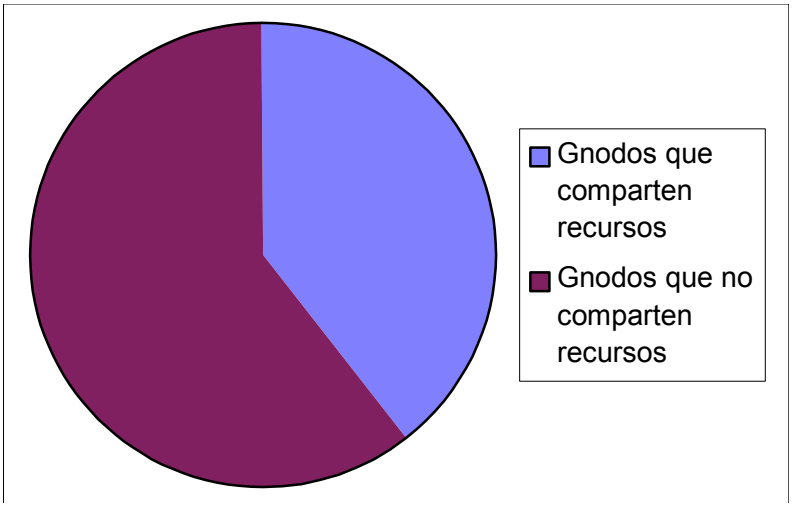
Recursos Ofrecidos

La tabla que se muestra a continuación representa solo los mensajes Pong (eliminándose las repeticiones) de aquellos gnodos que tienen recursos para compartir.

Dirección IP	Puerto	Kbytes ofrecidos	Archivos ofrecidos
213.120.145.168	6346	2458928	34194
128.211.244.26	6346	5554280	1237
144.92.240.190	5634	4609859	821
212.62.1.116	5634	3620021	302
24.219.45.99	6346	3515891	734
24.18.253.30	6355	3201020	788
24.132.166.255	6346	2970361	591
172.16.1.53	6346	2742033	551
131.230.208.167	6346	2448983	428
204.210.32.37	6346	1954304	495
63.200.125.102	6346	1878241	492
129.81.247.57	6346	1508059	352
192.168.0.3	6346	1452591	22391
147.4.228.71	6346	1302047	397
129.1.221.202	6346	1040295	182
64.0.252.52	6346	839404	166
213.99.194.126	6346	796226	178
198.82.109.192	6346	777313	61
193.203.144.172	13536	590263	88
192.168.1.100	6347	466967	62
134.130.195.44	6346	456614	1
64.39.198.9	6355	451397	49
150.214.178.31	6346	411529	117
130.83.236.245	6346	381998	73
141.166.236.45	6346	368134	97
49.0.1.1	14384	345675	3584
199.83.16.52	6346	291810	60
128.138.162.136	6346	268707	2988
24.222.107.240	6346	238156	968

194.104.245.21	6346	225067	58
141.30.220.195	6346	224176	40
24.218.76.181	6346	216369	52
10.151.2.2	6346	193650	27
192.168.0.2	6346	128831	40
163.205.7.38	6346	124978	31
192.168.1.101	6346	116951	14
64.230.146.123	6346	114562	16
128.40.40.74	6346	111060	8
128.125.243.220	6346	110144	20
24.226.61.35	6346	98412	24
24.218.16.54	80	92983	14
144.118.146.57	6346	86573	6
24.164.236.142	6346	77496	7
24.191.154.70	6346	77314	11
99.101.0.4	14	76223	11
18.212.0.100	6355	65060	10
24.14.132.214	6346	54112	3
120.1.8.36	6346	42370	16
63.102.154.165	6346	21634	353
192.168.0.1	6355	12556	8
207.210.88.73	6346	11046	2
137.204.202.68	6346	7291	3
193.137.96.46	6346	4233	6
169.254.88.1	6346	3575	3
10.2.1.26	6346	2710	10
192.109.135.69	6346	692	5
216.66.155.33	6346	448	6
169.254.83.23	6346	382	1
137.28.94.64	6346	1	2
TOTALES		***49242005***	***73254***

De la tabla anterior se puede concluir que el tamaño promedio de archivo ofrecido es de 672 Kbytes. Por otro lado, 59 gnodos (39,3%), de los 150 que respondieron con mensajes pong, ofrecen recursos para compartir.



Consultas de los usuarios

Se analizaron los mensajes Query, que en total sumaban 5500; se eliminaron mensajes con descripciones pertenecientes a juegos de caracteres no occidentales, con lo cual quedaron 4299 (78.16%) mensajes. Se ordenaron, los contenidos de las consultas, por frecuencia de aparición quedando la siguiente tabla, que muestra los primeros 200 contenidos (que representan el 51.5% de los querys en caracteres occidentales).

mpeg	49	orgy	13	next	10	y.asf	8
mpg	31	r kelly	13	papa roach	10	barenaked ladies	7
mp3	31	robbie wiliams mp3	13	rolling stones	10	boyz ii men	7
divx	29	sting	13	supergrass	10	celine dion	7
matchbox twenty	26	three 6 mafia	13	toby keith	10	christmas	7
teen	25	mpeg	13	vince gill	10	c.asf	7
.mpg	22	.avi	13	alanis morissette	9	da brat	7
moby	21	aaliyah	12	billy gilman	9	dandy warhols	7
christina	20	alliyah	12	bon jovi	9	destinys child	7
godsmack	20	brooks and dunn	12	brooks & dunn	9	devon	7
vertical horizon	20	destiny's child	12	creed	9	eminem	7
metallica	19	eiffel 65	12	deftones	9	e.asf	7
santana	19	enrique iglesias	12	faith hill	9	jagged edge	7
'nsync	18	everclear	12	frank sinatra	9	l.asf	7
orgasm	18	foo fighters	12	f.asf	9	macy gray	7
richards	18	lolita	12	green dag green day	9	madonna don\'t tell me	7

toni braxton	18	mandy moore	12	lonestar	9	martina mcbride	7
dido	17	marc anthony	12	loveandjoy	9	ruff endz	7
lil kim	17	red hot chili peppers	12	mya	9	three doors down	7
profyle	17	r. kelly	12	phil collins	9	w.asf	7
zip	17	xxx	12	pink	9	z.asf	7
britney mpg	16	a perfect circle	11	p.asf	9	*.zip	7
common	16	adobe	11	radiohead	9	amateur	6
fuck	16	anal	11	rape	9	b.mp3	6
jennifer lopez	16	dave matthews band	11	sr-71	9	craig david	6
kid rock	16	janet jackson	11	t.asf	9	erykah badu	6
porn	16	jessica simpson	11	112	9	fuck fuck	6
son by four	16	jimi hendrix	11	*.mpg	9	hip hop hits 4	6
tim mcgraw	16	j.asf	11	aaron tippin	8	h.asf	6
beatles	15	leann rimes	11	avi	8	jay z	6
dr. dre	15	lil' kim	11	backstreet boys	8	jenna	6
elvis	15	megadeth	11	dmx	8	jenna jameson	6
garth brooks	15	r.asf	11	elvis presley	8	lfo	6
sex	15	sammie	11	fuel	8	madison avenue	6
tyrese	15	third eye blind	11	fuoco nel fuoco	8	movies	6
u.asf	15	whitney houston	11	george strait	8	mpg mpg	6
asf	14	*.asf	11	g.asf	8	nsync	6
eve 6	14	'n sync	10	joan osborne	8	sister hazel	6
korn	14	alan jackson	10	k.asf	8	souldecision	6
marcy playground	14	a.asf	10	mystikal	8	spice girls	6
nelly	14	bloodhound	10	n.asf	8	v.asf	6
nine days	14	boyz 2 men	10	o.asf	8	x.asf	6
pink floid	14	busta rhymes	10	o.mp3	8	y.mp3	6
q.asf	14	cypress hill	10	paula abdul	8	3 doors down	6
98 degrees	14	de la soul	10	savage garden	8	*girl*	6
billy ray cyrus	13	eric clapton	10	sisqo	8	*.avi	6
david bowie	13	jpg	10	snoop dogg	8	*.mp3	6
don henley	13	lara fabian	10	s.asf	8	.zip	6
goo goo dolls	13	madonna	10	u2	8	anime	5
limp	13	m.asf	10	young	8	babysitter	5

Realizando una somera lectura acerca de la información buscada por los usuarios de una red Gnutella, puede verse claramente, que recursos relacionados con la imagen, música y

animación están en los primeros lugares. Temáticamente música pop y rock es la preferida y recursos relacionados con el sexo son frecuentemente requeridos.

Código fuente utilizado para la captura de mensjes.

```
#!/usr/bin/perl

    open(TEMPO ,">gnut-22.cap");
    my $port = 2222;

    my $PF_INET     = 2;
    my $AF_NS       = 6;
    my $SOCK_STREAM = 1;

    ($nombre, $alias, $prototipo)              = getprotobyname("tcp");
    my $thisaddr = pack("C4", 170, 210, 96, 253);
    my $thataddr = pack("C4",  35,  10,  25, 114);
    $this = pack('S n a4 x8', $PF_INET, 0, $thisaddr);
    $that = pack('S n a4 x8', $PF_INET, 6346, $thataddr);

    $socket = socket(S, $PF_INET, $SOCK_STREAM, $prototipo) || die $!;
    bind(S, $this) || die $!;
    connect(S, $that) || die $!;

    select(S); $| = 1; select(STDOUT);

    print S "GNUTELLA CONNECT/0.4\n\n\n";
    my $buf = "";
    while (<S>) {
       $buf    = $_;
       $lar    = length($buf);
       if (ord(substr($buf,$lar-1,1)) == 10) {
              $buf     = substr($buf,0,$lar-1);
       }
       print TEMPO  $buf;
    }
    close S;

close TEMPO;
```