Universidad Politécnica de Cartagena

Departamento de Tecnologías de la Información y las Comunicaciones

**Desarrollo de sistemas y servicios de
información sobre redes *overlay*
*peer-to-peer***

Juan Pedro Muñoz Gea

2011

Universidad Politécnica de Cartagena

Departamento de Tecnologías de la Información y las Comunicaciones

Tesis Doctoral

**Desarrollo de sistemas y servicios de
información sobre redes *overlay*
*peer-to-peer***

Autor

Juan Pedro Muñoz Gea

Director

Josemaría Malgosa Sanahuja

Cartagena, marzo de 2011

Universidad Politécnica de Cartagena

Department of Information and Communication Technologies

Ph.D. Thesis

**Development of information systems
and services over peer-to-peer overlay
networks**

Author

Juan Pedro Muñoz Gea

Supervisor

Josemaría Malgosa Sanahuja

Cartagena, March 2011

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Comisión de Doctorado

AUTORIZACIÓN DE LA PRESENTACIÓN DE LA TESIS DOCTORAL
POR LA COMISIÓN ACADÉMICA RESPONSABLE DEL PROGRAMA

D. Francisco J. Ortiz Zaragoza, Coordinador de la Comisión Académica del Programa de Doctorado Tecnologías de la Información y Comunicaciones

INFORMA:

Que la Tesis Doctoral titulada “Desarrollo de sistemas y servicios de información sobre redes overlay peer-to-peer”, ha sido realizada por D. Juan Pedro Muñoz Gea, bajo la dirección y supervisión de D. Josemaría Malgosa Sanahuja y que la Comisión Académica ha dado su conformidad para que sea presentada ante la Comisión de Doctorado.

En Cartagena, a 11 de marzo de 2011

EL COORDINADOR DE LA COMISIÓN ACADÉMICA DEL PROGRAMA

Fdo.: Francisco J. Ortiz Zaragoza

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

AUTORIZACIÓN DE LA PRESENTACIÓN DE LA TESIS DOCTORAL
POR EL DIRECTOR

D. Josemaría Malgosa Sanahuja, Profesor Doctor del Área de Ingeniería Telemática en el Departamento de Tecnologías de la Información y las Comunicaciones

AUTORIZA:

La presentación de la Tesis Doctoral titulada “Desarrollo de sistemas y servicios de información sobre redes overlay peer-to-peer”, realizada por D. Juan Pedro Muñoz Gea, bajo mi dirección y supervisión, en el Departamento de Tecnologías de la Información y las Comunicaciones, y que presenta para la obtención del grado de Doctor por la Universidad Politécnica de Cartagena.

En Cartagena, a 11 de marzo de 2011

EL DIRECTOR DE TESIS

Fdo.: Josemaría Malgosa Sanahuja

A mis padres y hermanos

Resumen

Una red *overlay* es una red virtual de nodos y enlaces lógicos construida sobre la red Internet. Por lo tanto, las redes *overlay* se pueden ver como una capa intermedia situada entre los protocolos básicos de Internet y el nivel de aplicación. Para entender por qué las redes *overlay* son importantes y el papel que desempeñan, merece la pena considerar las distintas razones que han motivado su aparición. En primer lugar, las redes *overlay* pueden utilizarse para soportar los requisitos especiales de una clase de aplicaciones particulares o de una determinada comunidad de usuarios. En segundo lugar, y relacionado con lo anterior, las redes *overlay* pueden desempeñar un papel importante en la evolución dinámica de la tecnología de Internet.

La arquitectura tradicional para el desarrollo de aplicaciones distribuidas ha estado basada durante muchos años exclusivamente en el modelo cliente-servidor. Este modelo era más apropiado cuando no existían los PCs, sólo estaciones de trabajo, cuya potencia estaba mucho de los grandes y costosos ordenadores centrales. Hoy en día, los PCs empleados en hogares y empresas tienen unas prestaciones cada vez más parecidas a los servidores a un precio sensiblemente menor. Por ello, las redes P2P (*Peer-to-Peer*) surgen como alternativa para aprovechar toda la capacidad de proceso, almacenamiento y ancho de banda sobrante de los ordenadores interconectados. Básicamente, una red informática P2P o “entre iguales”, se refiere a una red que no tiene clientes y servidores fijos, sino una serie de nodos que se comportan a la vez como clientes y como servidores de los demás nodos de la red. Este modelo de red contrasta con el modelo cliente-servidor tradicionalmente empleado en las aplicaciones de Internet. Aquí, todos los nodos se comportan igual y pueden realizar el mismo tipo de operaciones, pudiendo no obstante diferir en configuración local, velocidad de proceso, ancho de banda de su conexión a la red y capacidad de almacenamiento.

Respecto a los mecanismos y protocolos en que se basan las redes P2P, podemos diferenciar claramente dos componentes necesarios: I) Mecanismos para la localización de recursos (como pueden ser usuarios, servicios o contenidos) y II) Mecanismos de acceso o comunicación con los recursos (como puede ser el establecimiento de una llamada o la descarga de un fichero). Comúnmente, las redes P2P se clasifican en función de los mecanismos para la localización de recursos en redes no-estructuradas, redes basadas en *superpeers* y redes estructuradas. Las propiedades de las redes P2P las hacen muy apropiadas para el desarrollo de nuevos sistemas y servicios de información. En este trabajo se estudian en detalle las principales características de los diferentes tipos de redes P2P

previamente presentados. Además, también se analiza su idoneidad para el desarrollo de nuevos sistemas y servicios de información sobre ellas.

Una de las características más importantes de las redes P2P no-estructuradas es que presentan una arquitectura topológica plana, donde todos los nodos pertenecen al mismo nivel. Ello las hace especialmente apropiadas para la implementación de los mecanismos de provisión de anonimato tradicionalmente utilizados en redes IP. Uno de estos mecanismos, llamado *Crowds*, se basa en la construcción de un camino *random-walk* entre el origen y el destino de la comunicación. Uno de los inconvenientes de esta metodología es que el coste asociado con la comunicación aumenta enormemente cuando el número de saltos se incrementa. Por lo tanto, limitar la longitud de los caminos es un aspecto clave en este escenario. En este trabajo, se propone un mecanismo para reducir la varianza asociada a la longitud de los caminos *random-walk*. Además, se deriva un intuitivo a la vez que riguroso modelo analítico para calcular el grado de anonimato en las redes que utilizan el mecanismo propuesto. Este modelo se ha validado mediante simulación y se ha demostrado que es posible conseguir una buena solución de compromiso entre el nivel de anonimato (el beneficio) y la longitud del camino (el coste). Además, también se ha demostrado que el rendimiento del nuevo mecanismo frente a ataques especiales, como los ataques de análisis de tráfico o los ataques del predecesor, es equivalente al rendimiento conseguido por *Crowds*.

Por otra parte, las redes P2P basadas en *superpeers* presentan una arquitectura topológica jerárquica, donde los clientes se conectan a un *superpeer* que a su vez se conecta a un conjunto de *superpeers* vecinos. Este tipo de redes *overlay* intentan mejorar algunas de las características de las redes *overlay* planas, tales como el retardo y el rendimiento del encaminamiento. Para ello, cada *superpeer* mantiene un registro de los datos de sus clientes. Cuando un cliente desea enviar una solicitud a la red, la enviará a su *superpeer*. Si el *superpeer* encuentra algún resultado lo devolverá al cliente. En caso contrario, enviará la solicitud a sus *superpeers* vecinos y devolverá las posibles respuestas al cliente. Esta arquitectura fue utilizada en [Nafaa et al., 2008] para proporcionar un servicio de distribución de vídeo bajo demanda (*Video-On-Demand*, VOD) basado en P2P. En esta arquitectura, los contenidos son almacenados en los *Set-Top-Boxes* (STB) que actúan como clientes, y las sesiones son proporcionadas mediante *streaming* multi-fuente. En este trabajo se ha extendido la funcionalidad de esta arquitectura. En primer lugar el sistema se ha evaluado en la red de un operador y se ha demostrado que los mecanismos existentes no son suficientes para aprovechar completamente los recursos de la red. A continuación, se ha evaluado el sistema en el contexto de Internet.

Respecto al escenario de la red del operador, se ha propuesto un nuevo mecanismo de traducción de popularidad en disponibilidad, con el objetivo de terminar de forma óptima el número de copias necesarias de cada título en la red. A continuación, se ha presentado un nuevo mecanismo de reparto. Para llevar a cabo la evaluación del sistema completo con los nuevos mecanismos anteriores, se ha implementado una estrategia simple de asignación de recursos. Esta estrategia ordena los STBs en función del ancho de banda de subida disponible. A partir de la evaluación anterior, se ha deducido que los mecanismos previos no son suficientes para aumentar el número de sesiones de VOD simultáneas, y se ha demostrado que una condición necesaria para conseguir ese objetivo es poder distribuir uniformemente la tasa de rechazo sobre toda la librería de contenidos, inde-

pendientemente de su popularidad. La estrategia de reasignación de recursos dinámica presentada es capaz de conseguir el objetivo anterior.

En el contexto de Internet, se ha propuesto un nuevo algoritmo de reparto de contenidos. La decisión de qué contenido se reparte a qué STB se determina mediante un algoritmo de programación no lineal. Los resultados de simulación muestran que este algoritmo puede ayudar a mitigar la carga del servidor. A continuación, se han presentado dos nuevos mecanismos para reenviar las solicitudes entre diferentes redes de operador. Se ha demostrado que ambos mecanismos son capaces de mejorar el rendimiento global del sistema. Además, también se ha implementado y evaluado la estrategia de reasignación de recursos dinámica para los dos mecanismos previos. Para el mecanismo *full requests forwarding*, se demuestra que el porcentaje de carga del servidor se distribuye de forma más uniforme entre las diferentes categorías de contenidos. Por otra parte, también se demuestra que esta estrategia no es capaz de mejorar los resultados obtenidos por el mecanismo *sub-request forwarding*.

Por otra parte, las redes P2P estructuradas fueron desarrolladas con el objetivo de proporcionar búsquedas deterministas. Estas redes ofrecen un servicio de encaminamiento capaz de enviar un mensaje con una clave asociada a un solo nodo, responsable de esa clave. Por esta razón, a este servicio se le llama *Key-Based Routing* (KBR). En general, las arquitecturas P2P descentralizadas requieren de algún mecanismo de mantenimiento de red, ya que los servidores centralizados han sido eliminados. Los mecanismos de auto-organización pueden ser una buena forma de solucionar estos problemas de control. Se trata de mecanismos que permiten que la organización de un sistema se realice de forma automática, sin necesidad de ser gestionada por una fuente externa. En este trabajo, se ha presentado un mecanismo de auto-organización para aplicaciones P2P basadas en *superpeers*, que hacen uso de la funcionalidad de encaminamiento fiable asociada a las redes P2P estructuradas. El mecanismo propuesto se ha implementado en dos aplicaciones P2P basadas en *superpeers*. La primera es una aplicación de dos niveles, donde todos los *superpeers* son miembros de un grupo ALM (*Application Level Multicast*). La segunda es una aplicación P2P jerárquica. Esta aplicación es similar a la anterior, pero en este caso los *superpeers* tienen comunicación directa entre ellos utilizando una arquitectura jerárquica de *superpeers*. Además, los mecanismos de anonimato previamente presentados se han integrado en ambas aplicaciones P2P.

Finalmente, se han presentado los sistemas DHT (*Distributed Hash Table*). Se trata de sistemas de almacenamiento de datos fiables y escalables que liberan a los programadores de muchas de las complicaciones asociadas a la construcción de los sistemas distribuidos. Las DHT almacenan bloques de datos en cientos o miles de máquinas conectadas a Internet, replican los datos para aumentar la fiabilidad, y además son capaces de localizar datos rápidamente, aunque sean ejecutados sobre enlaces de baja capacidad. Este tipo de aplicaciones se implementan sobre la interfaz KBR ofrecida por las redes P2P estructuradas. En este trabajo se ha propuesto un mecanismo para obtener automáticamente la red de suministro asociada a un producto específico utilizando la red EPCglobal, y se ha implementado en un sistema DHT. Existen otros mecanismos con el mismo objetivo, pero la principal diferencia con la propuesta de este trabajo es que en nuestro sistema el cliente no tiene que realizar ninguna operación, es decir, él solicita la red de suministro asociada a un producto específico y los resultados se obtienen directamente. Este mecanismo se ha

integrado dentro del *Discovery Service* de la red EPCglobal, que se encuentra actualmente en desarrollo. Recientemente se han presentado varias propuestas para la implementación del *Discovery Service*. En la mayoría de ellas se ha implementado como un sistema centralizado. Por ejemplo, en el proyecto BRIDGE se ha desarrollado como una base de datos centralizada basada en LDAP. En este trabajo, se ha desarrollado un prototipo basado en DHT. Las DHTs tienen varias ventajas sobre las aproximaciones centralizadas, sin embargo también tienen un inconveniente importante: no implementan ningún mecanismo de control de acceso. Por lo tanto, en nuestro prototipo de *Discovery Service* basado en DHT también se ha implementado un servicio de control de acceso.

Abstract

In the current generation of the Internet, an overlay network is a virtual network of nodes and logical links built over the Internet. Therefore, overlay networks can be viewed as a “middle layer” above the basic Internet protocols but below the application layer. To understand why overlay networks are important and the role they play, it is worth considering the different reasons which have motivated their emergence. Firstly, overlay networks may exist to support the special requirements of a particular sort of applications or a specific community of users. Second, and related to the above, overlays may play an important role in the dynamic evolution of Internet technology.

In this respect, the traditional architecture to develop distributed applications was based on the client-server model. This model was most appropriate when there were no PCs, only work-stations, whose power was far from large and expensive mainframe computers. Today, the PCs used at home and business have a similar performance to the servers at prices substantially lower. Therefore, *peer-to-peer* (P2P) emerge as an alternative to take advantage of the full processing capacity, storage and excess bandwidth of interconnected computers. Basically, a P2P network refers to a network that does not have fixed clients and servers, but a series of nodes that act both as clients and servers for other network nodes. This network model contrasts with the traditional client-server model used in Internet applications. Thus, all nodes behave in the same way and are able to perform the same type of operation, although they can differ in local configuration, processing speed, bandwidth of the connection to the Internet, and storage capacity.

Regarding to the mechanisms and protocols in which P2P networks are based, we can clearly distinguish two necessary components: I) Mechanisms for locating resources (such as users, services or content) and II) Communication mechanisms to access resources (such as the establishment of a call or the download of a file). Typically, P2P networks are classified according to the mechanisms for locating resources in unstructured networks, superpeer-based networks and structured networks. The properties of P2P networks make them very suitable for the development of new information systems and services. In this work, the main characteristics of the previously presented P2P networks are studied in depth. In addition, it is also analyzed their suitability for the development of specific information systems and services over them.

On this matter, one of the most important characteristics of unstructured P2P networks, which is that they present a flat topological architecture, where all the nodes belong to the same level, makes this kind of networks specially suitable for the implementation

of source-rewriting mechanisms, traditionally used in IP networks to provide anonymity. One of these mechanisms, called Crowds, is based on the building of a random walk path between the initiator and the destination. One of the drawbacks of this methodology is that the cost associated with the communication increases dramatically as the number of hops also increases. Therefore, limiting the length of the paths is a key aspect in this scenario. In this work, an effective mechanism to reduce the variance associated with the length of the random walk paths is proposed. In addition, an intuitive but rigorous analytical model to calculate the degree of anonymity in networks using the proposed mechanism is also derived. This model has been validated by means of simulation and it has been proved that it is possible to achieve a good trade-off between the anonymity level (the benefit) and the length of the path (the cost). Besides this, it has also been proved that the performance of the new mechanisms over special attacks, such as the traffic analysis and predecessor attacks, is equivalent to the performance achieved by Crowds.

On the other hand, superpeer-based P2P networks present a hierarchical topological architecture, in which clients are connected to a superpeer, and it is connected to a set of neighbor superpeers. This kind of overlay networks tries to improve some of the characteristics of the traditional flat overlay networks such as delay and routing performance. Every superpeer keeps an index over its clients' data. This index must hold sufficient information to answer all queries. When a client wishes to submit a query to the network, it will send the query to its superpeer. If the superpeer finds any results, it will return one response message. Otherwise, it will submit the query to its neighbors as if it were its own query, and forward any response messages it receives back to the client. This architecture was used in [Nafaa et al., 2008] to implement a P2P-based VOD service distribution architecture in a broadband operator context. In this architecture, the content is stored in the set-top boxes (STBs), which act as peers, and the sessions are provided by means of multi-source streaming. In this work, the previous architecture has been extended. First of all, the system has been evaluated in a broadband operator scenario, and it has been concluded that the existing mechanisms are not enough to fully take advantage of the network resources. After that, the previous VOD service distribution architecture has been extended to the Internet context.

Regarding to the broadband operator scenario, it has been firstly proposed a new popularity-to-availability translation model in order to optimally determine the necessary number of copies of any title in the content library. Afterwards, a new dispatching algorithm has been presented. In order to perform the evaluation of the full system with the new previous mechanisms, a very simple resource allocation strategy, which discriminates and ranks the STBs based on their available uplink bandwidth, has been implemented. From this evaluation, it is deduced that the previous presented mechanisms are not enough to achieve the goal of increasing the number of simultaneous VOD sessions. Finally, it has been demonstrated that a necessary condition to maximize the number of simultaneous VOD sessions is to be able to uniformly distribute the rejection rate over the full library of titles, independently of their respective popularity. The dynamic re-allocation of streams strategy presented in this work is able to achieve the objective stated above.

Regarding to the Internet scenario, it has been firstly proposed a new dispatching algorithm. The decision on what content is dispatched at which STB is determined by a Non-Linear Programming (NLP) formulation. Simulation results show that using the for-

mulation decision for dispatching can help in efficiently mitigate server load. Afterwards, two new mechanisms to forward the VOD requests between different ISP networks have been presented. It has been showed that both mechanisms are able to improve the global performance of the system. After that, the dynamic re-allocation of streams strategy has been implemented and evaluated for the two VOD requests forwarding strategies previously presented. When it is evaluated for the *full requests forwarding* strategy, it is shown that the percentage of server load is more equally distributed among the different categories of contents. On the other hand, it is also proved that this mechanism is not able to improve the results obtained by the *sub-requests forwarding* strategy.

On the other hand, structured P2P networks were developed to provide deterministic bounds on the data discovery. This kind of networks offer a routing service to send a message, with an associated key, to a single node, responsible for that key. For this reason, this service is called Key-Based Routing (KBR). In general, decentralized P2P architectures require some kind of network maintenance, since centralized servers have been removed. Self-organization can be a good way to solve these control issues. It is a process in which the organization of a system spontaneously increases without being managed by an outside source. In this work, it has been presented a self-organizing mechanism for superpeer-based P2P applications, which makes use of the reliable routing functionality associated to structured P2P networks. The proposed mechanism has been implemented in two superpeer-based P2P applications. The first one is a two-level P2P application, where all the superpeers are members of an ALM group. The second one is a hierarchical P2P application. This application is similar to the previous one, but in this case the superpeers have direct communication among them using a hierarchical architecture of superpeers. In addition, the anonymity mechanism previously presented has been integrated in both P2P applications.

Finally, distributed hash table (DHT) systems have been presented. They are reliable, scalable, wide-area data storage system that free programmers from many of the complications of building a distributed system. DHTs store blocks of data on hundreds or thousands of machines connected to the Internet, replicate the data for reliability, and quickly locate data despite running over high-latency, wide-area links. This kind of applications are implemented over the key-based routing (KBR) interface offered by a structured P2P network. In this work, it has been proposed a mechanism for automatically obtaining the supply network associated to a specific product using the EPCglobal Network, and it has been implemented in a DHT system. There are other mechanisms with the same objective, but the main difference with the proposal set out in this chapter is that in our proposal the client does not have to perform any operations, that is, it queries the supply network associated to a specific product and the results are obtained directly. This mechanism has been integrated within the DS component of the EPCglobal Network, which is currently being developed. Recently, there have been several proposals for the implementation of the DS. In most of them the DS has been implemented as a centralized database. As an example, in the BRIDGE Project it has been developed as a centralized database based on LDAP. In this work, we have developed a DS prototype based on DHT. DHTs have several advantages over centralized approaches, however they also have an important drawback: they do not implement any access control mechanism. Therefore, in our DHT-based DS prototype an access control service has also been implemented.

Agradecimientos

Quiero expresar mis más sincero agradecimiento a Josemaría Malgosa Sanahuja, director de la Tesis, por la labor desempeñada en el desarrollo de la misma y por demostrarme que al final todo llega.

A Pilar Manzanares López y Juan Carlos Sánchez Aarnoutse, que han aportado su colaboración en este trabajo, así como al resto de compañeros becarios y profesores del Área de Ingeniería Telemática de la Universidad Politécnica de Cartagena por todo el ánimo y la ayuda prestada.

A mis amigos y amigas, por todos los buenos momentos y el apoyo recibido durante estos años.

A mi familia, mis padres y hermanos, que están siempre ahí.

Tabla de Contenidos

1	Introducción	1
1.1	Introducción	1
1.2	Definición de red overlay	2
1.3	Importancia de las redes overlay	3
1.4	Evolución de la arquitectura cliente-servidor hacia las redes P2P	4
1.5	Definición y propiedades de las redes P2P	4
1.6	Aplicaciones de las redes P2P	6
1.7	Clasificación y taxonomía	7
1.7.1	Redes P2P no-estructuradas	7
1.7.2	Redes P2P basadas en superpeer	9
1.7.3	Redes P2P estructuradas	11
1.8	Motivaciones y objetivos	12
1.9	Estructura y contribuciones	13
2	Provisión de anonimato utilizando redes P2P no-estructuradas	15
2.1	Introducción	15
2.2	Gnutella	16
2.3	Anonimato	17
2.4	Mecanismos de reescritura de fuente	18
2.4.1	Mixes	19
2.4.1.1	Mixmaster	19
2.4.1.2	GNUnet	20
2.4.2	Onion Routing	20
2.4.2.1	Tor	21
2.4.2.2	Tarzan	22
2.4.2.3	I2P	23
2.4.3	Crowds	23

2.4.3.1	AP3	24
2.4.3.2	Hordes	25
2.5	Antecedentes	25
2.6	El algoritmo de anonimato <i>Always Down-or-Up</i> (ADU)	26
2.7	Evaluación de los caminos ADU	28
2.7.1	Modelo analítico para la longitud de los caminos AD	28
2.7.2	Modelo analítico para la longitud de los caminos ADU	30
2.7.3	Características de los caminos ADU	32
2.8	Análisis del anonimato	33
2.9	Validación del modelo	36
2.10	Rendimiento frente a los ataques de análisis de tráfico y del predecesor . .	38
2.11	Conclusiones	44
3	Provisión de VOD utilizando redes P2P basadas en superpeer	45
3.1	Introducción	45
3.2	FastTrack	46
3.3	Sistemas P2P de VOD	47
3.3.1	Estructura de red overlay para VOD P2P	48
3.3.1.1	Estructura de red basada en árbol	48
3.3.1.2	Estructura de red basada en malla	49
3.3.2	Propuestas de envío	50
3.3.2.1	Envío basado en buffer	50
3.3.2.2	Envío basado en almacenamiento	50
3.3.2.3	Envío híbrido	51
3.3.3	Búsqueda de proveedores	51
3.3.3.1	Búsqueda en envío basado en buffer	52
3.3.3.2	Búsqueda en envío basado en almacenamiento	53
3.3.4	Planificación de datos	53
3.3.4.1	Planificación en envío basado en buffer	54
3.3.4.2	Planificación en envío basado en almacenamiento . . .	54
3.4	Antecedentes	55
3.5	Arquitectura de VOD en el contexto de la red de operador	58
3.5.1	Introducción	58
3.5.2	Descripción de la arquitectura	58
3.5.3	Suposiciones de trabajo	60
3.5.4	Modelo de traducción de popularidad en disponibilidad	60
3.5.5	Algoritmo de reparto	61
3.5.6	Evaluación de prestaciones	63

TABLA DE CONTENIDOS

3.5.6.1	Tasa de rechazo	64
3.5.6.2	Número de sesiones de VOD simultáneas	64
3.5.7	Limitación de las sesiones de VOD por categoría	66
3.5.7.1	Tasa de rechazo	66
3.5.7.2	Número de sesiones de VOD simultáneas	67
3.5.8	Reasignación dinámica de flujos	67
3.5.8.1	Tasa de rechazo	69
3.5.8.2	Número de sesiones de VOD simultáneas	70
3.6	Arquitectura de VOD para Internet	71
3.6.1	Introducción	71
3.6.2	Descripción de la arquitectura	72
3.6.3	Suposiciones de trabajo	73
3.6.4	Algoritmo de reparto	73
3.6.4.1	Descripción	73
3.6.4.2	Evaluación de prestaciones	76
3.6.5	Reenvío de solicitudes de VOD	79
3.6.5.1	Reenvío de solicitudes completas	79
3.6.5.2	Reenvío de sub-solicitudes	80
3.6.6	Reasignación dinámica de flujos	82
3.6.6.1	Reenvío de solicitudes completas	83
3.6.6.2	Reenvío de sub-solicitudes	84
3.6.7	Actualización del proceso de inyección de contenidos	84
3.6.7.1	Incremento del número de STBs	85
3.6.7.2	Decremento del número de STBs	86
3.6.7.3	Cambio de la popularidad	86
3.6.7.4	Otros casos	87
3.7	Conclusiones	87
4	Diseño de mecanismos de auto-organización utilizando redes P2P estructuradas	89
4.1	Introducción	89
4.2	Chord	90
4.3	Desarrollo de soluciones de multicast sobre redes P2P estructuradas . . .	91
4.4	Simulación de redes P2P	93
4.4.1	Introducción	93
4.4.2	Entornos de simulación P2P	93
4.4.3	OMNeT++ y OverSim	94
4.4.4	Modelado de P2P	95
4.4.4.1	Caracterización de las solicitudes	96

TABLA DE CONTENIDOS

4.4.4.2	Dinamismo de los nodos	96
4.4.4.3	Caracterización de los contenidos	97
4.5	Antecedentes	97
4.6	Sistema P2P de dos niveles	98
4.6.1	Descripción del sistema	98
4.6.1.1	Inicialización y unión a la red	98
4.6.1.2	Unión a un subgrupo	99
4.6.1.3	Gestión de la jerarquía	99
4.6.1.4	Gestión de los superpeers	99
4.6.1.5	Registro de los contenidos compartidos	99
4.6.1.6	Búsqueda de contenidos	100
4.6.2	Claves del sistema	100
4.6.3	Provisión de anonimato	100
4.6.3.1	Publicación de los contenidos	100
4.6.3.2	Búsqueda de contenidos	101
4.6.3.3	Descarga de contenidos	102
4.6.3.4	Aspectos dinámicos del sistema	103
4.6.4	Simulaciones	104
4.6.4.1	Simulaciones estáticas de la red overlay	105
4.6.4.2	Simulaciones dinámicas de la red overlay	107
4.6.4.3	Escalabilidad del simulador	108
4.7	Sistema P2P jerárquico	108
4.7.1	Descripción del sistema	108
4.7.1.1	Inicialización y unión a la red	109
4.7.1.2	Unión a un subgrupo	109
4.7.1.3	Asignación de superpeer	110
4.7.1.4	Mantenimiento de los superpeers	110
4.7.1.5	Búsqueda de contenidos	111
4.7.2	Provisión de anonimato	111
4.7.3	Mecanismo de reputación	111
4.7.4	Simulaciones	112
4.8	Conclusiones	114
5	Diseño de sistemas de trazabilidad utilizando redes P2P estructuradas	115
5.1	Introducción	115
5.2	Pastry	116
5.3	Desarrollo de aplicaciones DHT sobre redes P2P estructuradas	117
5.4	Identificación por radio frecuencia	118

TABLA DE CONTENIDOS

5.4.1	Propuestas para la identificación de productos	118
5.4.2	EPCglobal Network	120
5.5	Antecedentes	123
5.6	Prototipo de servicios de descubrimiento	125
5.6.1	Servicios de descubrimiento	125
5.6.1.1	Aplicación DHT	125
5.6.1.2	Interfaz de publicación/solicitud	128
5.6.2	Interfaz EPCIS-DS	130
5.6.3	Aplicación de solicitud	132
5.7	Infraestructura de seguridad y servicio de control de acceso	132
5.7.1	Infraestructura de seguridad	132
5.7.2	Servicio de control de acceso	133
5.7.3	Otros servicios de seguridad	135
5.8	Servicio de descubrimiento de la red de suministro	135
5.8.1	Introducción	135
5.8.2	Implementación	138
5.8.3	Presentación de la red de suministro	141
5.9	Conclusiones	142
6	Conclusiones	145
6.1	Conclusiones	145
6.2	Trabajos futuros	146

Lista de Figuras

1.1	Modelo cliente-servidor.	5
1.2	Red P2P no-estructurada.	8
1.3	Red P2P basada en superpeer.	10
2.1	Proceso de búsqueda en Gnutella.	17
2.2	Arquitectura general de un nodo Mix.	19
2.3	Arquitectura de Onion Routing.	21
2.4	Arquitectura de Crowds.	24
2.5	Parámetros del algoritmo.	28
2.6	Interpretación gráfica.	29
2.7	Validación del modelo.	37
2.8	Grado medio de anonimato en función de $\bar{l}$	38
2.9	Simulación del ataque del predecesor.	40
2.10	Función de probabilidad para los diferentes valores del intervalo.	41
2.11	Simulación del nuevo ataque frente a ADU.	43
3.1	Proceso de búsqueda FastTrack.	47
3.2	Estado del esquema de patching con $T = 10$ en el instante 40.	49
3.3	Perspectiva de la arquitectura P2P basada en malla.	50
3.4	Marco general de un sistema típico P2P de VOD de envío basado en buffer.	51
3.5	Un sistema P2P de VOD de envío híbrido.	52
3.6	Distribución de los vecinos sobre los anillos para el nodo P.	53
3.7	Estructura de VMesh.	54
3.8	Estructura de BiToS.	55
3.9	Proceso de fragmentación de vídeo.	57
3.10	Vista de la arquitectura de VOD.	59
3.11	Set-top box popularity score después de inyectar todos los fragmentos en la red.	61

3.12	Número de STBs diferentes que tienen un fragmento específico de un contenido.	62
3.13	Content dispersion index.	63
3.14	Tasa de rechazo.	64
3.15	Número de sesiones de VOD.	65
3.16	Número de sesiones de VOD por categoría de contenidos.	65
3.17	Tasa de rechazo.	67
3.18	Número de sesiones de VOD.	68
3.19	Número de sesiones de VOD por categoría de contenidos.	68
3.20	Tasa de rechazo.	69
3.21	Número de sesiones de VOD por categoría de contenidos.. . . .	70
3.22	Número de sesiones de VOD.	71
3.23	Vista de la arquitectura de VOD.	72
3.24	Porcentaje de carga del servidor.	76
3.25	Número de sesiones de VOD servidas por la red P2P.	77
3.26	Ancho de banda del servidor.	78
3.27	Porcentaje de carga del servidor.	79
3.28	Número de sesiones de VOD servidas por la red P2P.	80
3.29	Ancho de banda del servidor.	81
3.30	Porcentaje de carga del servidor.	81
3.31	Ancho de banda del servidor.	82
3.32	Porcentaje de carga del servidor.	83
3.33	Número de sesiones de VOD servidas por la red P2P.	83
3.34	Ancho de banda del servidor.	84
3.35	Porcentage de carga del servidor.	85
3.36	Ancho de bada del servidor.	85
4.1	Chord: Ejemplo de búsqueda. El nodo 8 está buscando el identificador 53 (k53).	91
4.2	Operación <i>joinGroup()</i>	92
4.3	Operación <i>sendDataToGroup()</i>	93
4.4	Procedimiento para desarrollar una aplicación de red libre de errores. . . .	94
4.5	Estructura en capas de OverSim.	95
4.6	Arquitectura general del sistema.	98
4.7	Fase de búsqueda.	101
4.8	Fase de descarga.	102
4.9	Popularidad de las solicitudes.	105
4.10	Probabilidad de que el contenido esté en el superpeer.	105

LISTA DE FIGURAS

4.11	Número de superpeers consultados.	106
4.12	Número de fuentes encontradas.	106
4.13	Número de solicitudes que no se pueden llevar a cabo en una hora porque al menos un nodo en el camino de descarga ha fallado.	107
4.14	Número medio de mensajes de control en función del tiempo.	107
4.15	Arquitectura del sistema propuesto, con una jerarquía de superpeers de tres niveles.	109
4.16	Provisión de anonimato.	111
4.17	Evolución temporal del número medio de superpeers consultados hasta la localización de los contenidos.	112
4.18	Valores teóricos y de simulación del número medio de superpeers consul- tados hasta la localización de los contenidos.	113
5.1	Operación <i>insert()</i>	117
5.2	Operación <i>lookup()</i>	118
5.3	Búsqueda de información de productos con DIALOG.	119
5.4	Estructura de la etiqueta EPC.	120
5.5	Arquitectura de EPCglobal Network.	121
5.6	Tipos de eventos EPCIS y atributos.	122
5.7	Arquitectura de los servicios de descubrimiento.	125
5.8	Interfaz EPCIS-DS.	131
5.9	Servicio de control de acceso.	134
5.10	Una estructura de producto.	135
5.11	Red de suministro asociada al EPC #1.	136
5.12	Interacción entre los nodos DHT involucrados en la reconstrucción de esta red de suministro.	141
5.13	Interpretación gráfica de la red de suministro asociada al producto con EPC #1.	142
5.14	Vista de la compañía E.	142
5.15	Diagrama de flujo de la aplicación.	143

Lista de Tablas

1.1	Ejemplos de redes overlay.	2
1.2	Ejemplos de overlays no-estructuradas.	9
1.3	Overlays estructuradas por categoría.	12
2.1	Valores de los parámetros para $\bar{l}$ específicos.	32
2.2	Varianza de la longitud de los caminos.	33
4.1	Velocidad de simulación.	108
4.2	Consumo de memoria.	108
5.1	Soporte para los estándares WS-*.	129
5.2	Información necesaria para reconstruir una red de suministro.	136
5.3	Cliente web service solicitando la información asociada a un evento AS-SIGN sin componentes.	137
5.4	Parámetros para solicitar los eventos ASSIGN, SHIP y RECEIVE.	138
5.5	Algoritmo del método <i>supply_network()</i>	139
5.6	Información para reconstruir la red de suministro asociada al EPC #1.	140

List of Contents

1	Introduction	1
1.1	Introduction	1
1.2	Definition of overlay network	2
1.3	Importance of overlay networks	3
1.4	From client-server to P2P networking	4
1.5	Definition and properties of P2P networks	4
1.6	Applications of P2P networks	6
1.7	Classification and taxonomy	7
1.7.1	Unstructured P2P networks	7
1.7.2	Superpeer-based P2P networks	9
1.7.3	Structured P2P networks	11
1.8	Motivations and goals	12
1.9	Structure and contributions	13
2	Providing anonymity using unstructured P2P networks	15
2.1	Introduction	15
2.2	Gnutella	16
2.3	Anonymity	17
2.4	Source-rewriting mechanisms	18
2.4.1	Mixes	19
2.4.1.1	Mixmaster	19
2.4.1.2	GNUnet	20
2.4.2	Onion Routing	20
2.4.2.1	Tor	21
2.4.2.2	Tarzan	22
2.4.2.3	I2P	23
2.4.3	Crowds	23

2.4.3.1	AP3	24
2.4.3.2	Hordes	25
2.5	Background	25
2.6	The anonymous aware <i>Always Down-or-Up</i> (ADU) algorithm	26
2.7	Evaluation of the ADU paths	28
2.7.1	Analytical model for the length of the AD paths	28
2.7.2	Analytical model for the length of the ADU paths	30
2.7.3	Characteristics of the ADU paths	32
2.8	Analysis of anonymity	33
2.9	Model validation	36
2.10	Performance over traffic analysis and predecessor attacks	38
2.11	Conclusions	44
3	Providing VOD streaming using superpeer-based P2P networks	45
3.1	Introduction	45
3.2	FastTrack	46
3.3	P2P VOD systems	47
3.3.1	Overlay network structure for P2P VOD	48
3.3.1.1	Tree-based network structure	48
3.3.1.2	Mesh-based network structure	49
3.3.2	Forwarding approaches	50
3.3.2.1	Buffer-forwarding	50
3.3.2.2	Storage-forwarding	50
3.3.2.3	Hybrid-forwarding	51
3.3.3	Suppliers search	51
3.3.3.1	Searching in buffer-forwarding	52
3.3.3.2	Searching in storage-forwarding	53
3.3.4	Data scheduling	53
3.3.4.1	Scheduling in buffer-forwarding	54
3.3.4.2	Scheduling in storage-forwarding	54
3.4	Background	55
3.5	VOD architecture in a broadband operator context	58
3.5.1	Introduction	58
3.5.2	Architecture description	58
3.5.3	Working assumptions	60
3.5.4	Popularity-to-availability translation model	60
3.5.5	Dispatching algorithm	61
3.5.6	Performance evaluation	63

LIST OF CONTENTS

3.5.6.1	Rejection rate	64
3.5.6.2	Number of simultaneous VOD sessions	64
3.5.7	Limitation of VOD sessions per category	66
3.5.7.1	Rejection rate	66
3.5.7.2	Number of simultaneous VOD sessions	67
3.5.8	Dynamic re-allocation of streams	67
3.5.8.1	Rejection rate	69
3.5.8.2	Number of simultaneous VOD sessions	70
3.6	VOD architecture for the Internet	71
3.6.1	Introduction	71
3.6.2	Architecture description	72
3.6.3	Working assumptions	73
3.6.4	Dispatching algorithm	73
3.6.4.1	Description	73
3.6.4.2	Performance evaluation	76
3.6.5	VOD requests forwarding	79
3.6.5.1	Full requests forwarding	79
3.6.5.2	Sub-requests forwarding	80
3.6.6	Dynamic re-allocation of streams	82
3.6.6.1	Full requests forwarding	83
3.6.6.2	Sub-requests forwarding	84
3.6.7	Update of the content injection process	84
3.6.7.1	Increase of the number of STBs	85
3.6.7.2	Decrease of the number of STBs	86
3.6.7.3	Change of the popularity	86
3.6.7.4	Other cases	87
3.7	Conclusions	87
4	Design of self-organizing mechanisms using structured P2P networks	89
4.1	Introduction	89
4.2	Chord	90
4.3	Development of multicast solutions over structured P2P networks	91
4.4	Simulation of P2P networks	93
4.4.1	Introduction	93
4.4.2	P2P simulation frameworks	94
4.4.3	OMNeT++ and OverSim	95
4.4.4	P2P modeling	96
4.4.4.1	Queries characterization	96

4.4.4.2	Dynamics of peer participation	96
4.4.4.3	Files characterization	97
4.5	Background	97
4.6	Two-level P2P system	98
4.6.1	Description of the system	98
4.6.1.1	Initializing and joining the network	99
4.6.1.2	Joining a subgroup	99
4.6.1.3	Management of the hierarchy	99
4.6.1.4	Management of the superpeers	99
4.6.1.5	Registering the shared files	100
4.6.1.6	Searching for a content	100
4.6.2	Keys of the system	100
4.6.3	Providing anonymity	100
4.6.3.1	Publishing the contents	101
4.6.3.2	Searching the contents	101
4.6.3.3	Downloading the contents	102
4.6.3.4	Dynamics aspects of the system	103
4.6.4	Simulations	104
4.6.4.1	Static overlay network simulations	105
4.6.4.2	Dynamic overlay network simulations	107
4.6.4.3	Simulator scalability	108
4.7	Hierarchical P2P system	109
4.7.1	Description of the system	109
4.7.1.1	Initializing and joining the network	109
4.7.1.2	Joining a subgroup	110
4.7.1.3	Superpeer assignment	110
4.7.1.4	Superpeers maintenance	110
4.7.1.5	Searching for a content	111
4.7.2	Providing anonymity	111
4.7.3	Reputation mechanism	111
4.7.4	Simulations	112
4.8	Conclusions	114
5	Design of traceability systems using structured P2P networks	115
5.1	Introduction	115
5.2	Pastry	116
5.3	Development of DHT applications over structured P2P networks	117
5.4	Radio frequency identification	118

LIST OF CONTENTS

5.4.1	Product identifier proposals	118
5.4.2	EPCglobal Network	120
5.5	Background	122
5.6	Discovery services prototype	125
5.6.1	Discovery services	125
5.6.1.1	DHT application	125
5.6.1.2	Publish/query interface	128
5.6.2	EPCIS-DS interface	130
5.6.3	Query application	132
5.7	Security infrastructure and access control service	132
5.7.1	Security infrastructure	132
5.7.2	Access control service	133
5.7.3	Other security services	135
5.8	Supply network discovery service	135
5.8.1	Introduction	135
5.8.2	Implementation	138
5.8.3	Supply network presentation	140
5.9	Conclusions	142
6	Conclusions	145
6.1	Conclusions	145
6.2	Future works	146

List of Figures

1.1	Client-server model.	5
1.2	Unstructured P2P network.	8
1.3	Superpeer-based P2P network.	10
2.1	Search process in Gnutella.	17
2.2	General architecture of a Mix node.	19
2.3	Architecture of Onion Routing.	21
2.4	Architecture of Crowds.	24
2.5	Parameters of the algorithm.	28
2.6	Graphical interpretation.	29
2.7	Validation of the model.	37
2.8	Mean degree of anonymity as a function of $\bar{l}$	38
2.9	Simulation of the predecessor attack.	40
2.10	Probability function for the different values of the interval.	41
2.11	Simulation of the new attack against ADU.	43
3.1	Search process in FastTrack.	47
3.2	A snapshot of the patching scheme with $T = 10$ at time 40.	49
3.3	An overview of P2P mesh-based architecture.	50
3.4	General framework of a typical buffer-forwarding P2P VOD system.	51
3.5	A hybrid-forwarding P2P VOD system.	52
3.6	Neighbor distribution over rings for peer P.	53
3.7	VMesh structure.	54
3.8	BiToS peer structure.	55
3.9	Video fragmentation process.	57
3.10	High-level view of the VOD architecture.	59
3.11	Set-top box popularity score after injecting all the fragments in the network.	61

3.12	Number of different STBs containing a fragment of a specific title.	62
3.13	Content dispersion index.	63
3.14	Rejection rate.	64
3.15	Number of VOD sessions.	65
3.16	Number of VOD sessions per category of contents.	65
3.17	Rejection rate.	67
3.18	Number of VOD sessions.	68
3.19	Number of VOD sessions per category of contents.	68
3.20	Rejection rate.	69
3.21	Number of VOD sessions per category of contents.	70
3.22	Number of VOD sessions.	71
3.23	High-level view of the VOD architecture.	72
3.24	Percentage of server load.	76
3.25	Number of VOD sessions served by the P2P network.	77
3.26	Server bandwidth.	78
3.27	Percentage of server load.	79
3.28	Number of VOD sessions served by the P2P network.	80
3.29	Server bandwidth.	81
3.30	Percentage of server load.	81
3.31	Server bandwidth.	82
3.32	Percentage of server load.	83
3.33	Number of VOD sessions served by the P2P network.	83
3.34	Server bandwidth.	84
3.35	Percentage of server load.	85
3.36	Server bandwidth.	85
4.1	Chord: Lookup procedure example. Node 8 is looking for the key-identifier 53 (k53).	91
4.2	<i>joinGroup()</i> operation.	92
4.3	<i>sendDataToGroup()</i> operation.	93
4.4	Procedure to develop an error-free network application.	94
4.5	OverSim layered structure.	95
4.6	General architecture of the system.	98
4.7	Searching phase.	102
4.8	Download phase.	103
4.9	Query popularity.	105
4.10	Probability that the content is in the superpeer.	105
4.11	Number of visited superpeers.	106

LIST OF FIGURES

4.12	Number of found sources.	106
4.13	Number of request that cannot be carry out in an hour because at least one node in the download path is out.	107
4.14	Average number of control messages in function of time.	107
4.15	Proposed system architecture, with a three levels superpeer hierarchy. . .	109
4.16	Providing anonymity.	112
4.17	Time evolution of the average number of superpeers consulted until con- tent location.	113
4.18	Simulation and theoretical values of the average number of consulted su- perpeers until content location.	113
5.1	<i>insert()</i> operation.	117
5.2	<i>lookup()</i> operation.	118
5.3	Product information lookup with DIALOG.	119
5.4	EPC tag data structure.	120
5.5	EPCglobal Network Architecture.	121
5.6	EPCIS event types and attributes.	122
5.7	Discovery Services architecture.	125
5.8	EPCIS-DS interface.	131
5.9	Access Control service.	134
5.10	A product structure.	135
5.11	Supply network associated to EPC #1.	136
5.12	Interaction among the DHT nodes involved in the reconstruction of this supply network.	141
5.13	Graphical representation of the supply network associated to product with EPC #1.	142
5.14	Zoom view of company E.	142
5.15	Application flowchart.	143

List of Tables

1.1	Examples of overlay networks.	2
1.2	Examples of unstructured overlays.	9
1.3	Structured overlays by category.	12
2.1	Values of parameters for specific $\bar{l}$	32
2.2	Variance of the length of the paths.	33
4.1	Simulation speed.	108
4.2	Memory consumption.	108
5.1	Support for WS-* standards.	129
5.2	Necessary information to reconstruct a supply network.	136
5.3	Web service client asking the information associated to an ASSIGN event without components.	137
5.4	Parameters to query the ASSIGN, SHIP and RECEIVE events.	138
5.5	Algorithm of the <i>supply_network()</i> method.	139
5.6	Information to reconstruct the supply network associated to EPC #1. . . .	140

Introduction

This chapter provides an overview of the basic principles of peer-to-peer overlay networks. First of all, the concept of overlay network is presented and the importance of this kind of networks is highlighted. After that, peer-to-peer networks are introduced. Their most important properties and applications are described. It also includes a classification of existing peer-to-peer networks and the motivations and goals of this work. Finally, the structure of this document and its main contributions are summarized.

1.1 Introduction

The Internet started as a network running on top of the Public Switched Telephone Network (PSTN), i.e., it was an *overlay* that complemented the underlying infrastructure of the PSTN by adding new functionality to support the special needs of certain groups (such as universities or research centers) [Clark et al., 2005].

With the commercialization of the Internet in the 1980s and its emergence as a platform for broadband communications in the 1990s, the Internet has evolved into the principal platform for global communications. In recent years, Internet has become the main transport medium for telephony and other multimedia applications (voice, video and data). What previously was an overlay application has now become a basic infrastructure. PSTN operators have had to carry out the investments required to support and manage this new infrastructure.

The success of the Internet is mainly due to the interoperability and connectivity supported by the adoption of the TCP/IP protocols and the introduction of *end-to-end* design principles that have dominated the Internet architecture. However, the success of the Internet has also posed significant problems: the growth has brought the need to provide different services (not everyone needs or wants the same capabilities), provide new media (such as support for real-time services) and issues of complexity and size (presented by the total magnitude measured in terms of Internet traffic and connectivity).

To meet these challenges, the Internet needs to continue evolving. In a process that looks as if it was repeating in history, the Internet is now generating its own collection of overlay networks. There are many types and examples of overlay networks (see Table 1.1) which help to resolve a variety of purposes and needs. The emergence of these overlays

Type	Function/Purpose	Examples
Peer-to-Peer (P2P)	File sharing; Content distribution; IP telephony; Data synchronization	eMule, Skype
Content-delivery (CDN)	Content caching to reduce access delays and transport costs	Akamai, Digital Island
Routing	Reduce routing delays; Resilient routing overlays	Resilient Overlay Network (RON), SureRoute
Security	Enhance end-user security	Virtual private networks (VPNs)
Other	Various	Multicast (MBone, 6Bone, TRIAD, IP-NL), delay tolerant networks, etc.

Table 1.1: Examples of overlay networks.

raises interesting questions about the future of the Internet architecture and the role of the Internet as a common platform for global communications.

1.2 Definition of overlay network

In the current generation of the Internet, an overlay network is a virtual network of nodes and logical links built over the Internet. Therefore, overlay networks can be viewed as a “middle layer” above the basic Internet protocols but below the application layer. However, this model in layers only addresses one aspect of the importance of its architecture [Clark et al., 2005].

A significant reason why the overlays are interesting is that they raise interesting questions about the early end-to-end design principles that have historically driven the location of the functionality and services within the Internet architecture. Overlay networks change the location where the functionality and services can be established and what entities may operate such functionality and services.

The Internet, in its most simple concept, has two sorts of components: *end-nodes* (the computers at the edge of the network that play the roles of servers and user machines) and *routers* (which forward the packets between end-nodes). With this simplified view, one can think of the Internet as a cloud of connected routers with end-nodes connected around the edge of the cloud. End-nodes run application processes (e.g., games, web clients or servers). An application process is a program which communicates with a program in another end-node, sending packets that are forwarded by routers. The routers simply forwards packets and do not know anything about the application-level functions.

Many real applications are more complex than this simple model. In the case of email, for example, when one user sends a mail to another, it goes by way of intermediate servers with names like “SMTP server” or “pop server”. In the case of the web, there are web caches and proxies. These intermediate devices are not routers, are a form of end-nodes. However, these intermediate devices have a number of important characteristics that distinguish them from conventional end-nodes. First, they are distributed around the Internet in a way that provides an infrastructure on which applications run. From the perspective of the router, they may be an end-node but from the perspective of the application, they are infrastructure. Second, these intermediate devices tend to be provided and operated by third parties. If two users exchange mails, they depend on servers operated by others to forward those mails.

1.3. Importance of overlay networks

The introduction of these intermediate devices represents the introduction of new capabilities in new locations within the Internet architecture. The functionality and services are not provided by a conventional end-node, but they are not provided by a router. In order to distinguish these infrastructure devices from conventional end-nodes the term overlay is used. On this matter, an overlay network describes an organized set of infrastructure devices.

Another way to understand the overlay is to focus on the extra functionality that they provide. The basic functionality of the Internet is defined by the suite of core Internet protocols (IP, TCP, UDP, BGP, DNS, ...). These include the minimum set of basic protocols that any network or node must support in order to be considered as part of the Internet. The ability of basic Internet protocols to be supported on many different physical infrastructures (ATM, Frame Relay, SONET, wireless) and to support many different types of end-user applications (data, voice, video) helps to promote the connectivity and interoperability across heterogeneous infrastructures and has helped the Internet evolve into a global communications platform.

However, the basic functionality of the “best effort” service offered by the standard IP protocols is not always enough. The original Internet architecture was designed to support communications between fixed points, where the source knew the address of the destination. However, many applications have more general communication needs, such as mobility, multicast and anycast. These communication needs present new challenges that the current Internet architecture does not support.

In order to address these needs and overcome the barriers inherent of the existing infrastructure, overlay networks blur the clean distinction within the Internet architecture between packet forwarding and application processing. Overlay networks, as opposed to application-specific network solutions, are increasingly seen as the mechanism of choice for introducing functionality into the Internet. This is important because they have become the main medium to carry out the evolution of the Internet architecture.

1.3 Importance of overlay networks

To understand why overlay networks are important and the role they play, it is worth considering the different reasons which have motivated their emergence. Firstly, overlay networks may exist to support the special requirements of a particular sort of applications or a specific community of users. As the Internet becomes more pervasive across all the segments of the global economy and it becomes the general platform to support all kinds of electronic communications, it is expected that the need to address heterogeneous specialized interest increase.

Therefore, the success of the Internet as an open standard leads to the need to satisfy heterogeneous requirements, which, by themselves, provide a justification for overlay networks. It is worth noting, however, that even in this case, drawing the boundary between what functions are really general and hence should be implemented as “basic infrastructure” and which are specialized and so may be suitable as “overlays” is neither static (markets vary) nor clear (for example, are mail servers basic Internet infrastructure or part of an overlay?).

Second, and related to the above, overlays may play an important role in the dynamic evolution of Internet technology. One of the great advantages of the end-to-end architecture of the Internet is the ability to incrementally adopt innovations in end-nodes. Applications can be adopted by a large number of end-nodes without involving changes in the basic Internet. The ubiquity with which it implements TCP/IP provides a stable platform to support communications among and across heterogeneous end-nodes.

Functionality that is missing in the current Internet may be developed first in an overlay for those users who most require enhancements that may not be available yet in the general Internet. This can include such things as enhanced quality of service or security/privacy. Over time, successful innovations will be adopted as a *de facto* component of the basic Internet.

1.4 From client-server to P2P networking

The traditional architecture to develop distributed applications is based on the client-server model. This model, used on the Internet to virtualize all conventional services, consists of a number of clients accessing simultaneously to a set of servers that provide some resources or applications. Some common servers are file servers, that allow users to store and access files from one computer, and application servers, which perform tasks that directly benefit the end user. However, when you try to download large volumes of information to many clients, client-server architecture is slow, expensive and not scalable.

The client-server model was most appropriate when there were no PCs, only workstations, whose power was far from large and expensive mainframe computers. Today, the PCs used at home and business have a similar performance to the servers at prices substantially lower. Therefore, *peer-to-peer* (P2P) emerge as an alternative to take advantage of the full processing capacity, storage and excess bandwidth of interconnected computers.

Basically, a P2P network refers to a network that does not have fixed clients and servers, but a series of nodes that act both as clients and servers for other network nodes. This network model contrasts with the traditional client-server model used in Internet applications. Thus, all nodes behave in the same way and are able to perform the same type of operation, although they can differ in local configuration, processing speed, bandwidth of the connection to the Internet, and storage capacity.

These networks can grow indefinitely without increasing search time and without the need for costly centralized resource. As they use the bandwidth, storage capacity and processing power of machines interconnected, the number of these resources always increases in direct proportion with the growth of the network itself.

1.5 Definition and properties of P2P networks

P2P networks have been defined in many papers. Here there are three definitions that cover the concepts of resource sharing and interconnection [Buford and Yu, 2010]:

“A distributed network architecture may be called a peer-to-peer network, if the participants share a part of their own hardware resources (processing power, storage capacity,

1.5. Definition and properties of P2P networks

Figure 1.1: Client-server model.

network link capacity, printers). These shared resources are necessary to provide the service and content offered by the network (e.g. file sharing or shared workspaces for collaboration). They are accessible by other peers.” [Schollmeier, 2001]

“Peer-to-peer systems are distributed systems consisting of interconnected nodes able to self-organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient population of nodes while maintaining acceptable connectivity and performance, without requiring the intermediate or support of a global centralized server or authority.” [Androutsellis-Theotokis and Spinellis, 2004]

“An application layer virtual or logical network in which end points are addressable and that provides connectivity, routing, and messaging between end points. Overlay networks are frequently used as a substrate for deploying new network services, or for providing a routing topology not available from the underlying physical network. Many peer-to-peer systems are overlay networks that run on top of the Internet.” [Buford et al., 2008]

The following properties are characteristics found in most P2P networks [Buford and Yu, 2010]:

- *Resource sharing.* Each peer contributes system resources to the operation of the P2P system. Ideally this resource sharing is proportional to the peer’s use of the P2P system.
- *Networked.* All nodes are interconnected with other nodes in the P2P system, and the full set of nodes are members of a connected graph. When the graph is no longer connected, the overlay is said to be partitioned.
- *Decentralization.* The behavior of the P2P system is determined by the collective actions of peer nodes, and there is no central control point. Some systems however secure the P2P system using a central login server. The ability to manage the overlay [Buford, 2008] and monitorize its operation may require centralized elements.
- *Symmetry.* Nodes assume equal roles in the operation of the P2P system. In many

designs this property is relaxed by the use of special peer roles such as super peers or relay peers.

- *Autonomy.* Participation of the peer in the P2P system is determined locally, and there is no single administrative context for the P2P system.
- *Self-organization.* The organization of the P2P system increases over time using local knowledge and local operations at each peer, and no peer dominates the system.
- *Scalable.* This is a pre-requisite of operating P2P systems with millions of simultaneous nodes, and means that the resources used at each peer exhibit a growth rate as a function of overlay size that is less than linear. It also means that the response time does not grow more than linearly as a function of overlay size.
- *Stability.* P2P systems are able to maintain their graphs connected and to route deterministically within a practical hop-count bounds.

1.6 Applications of P2P networks

Among the most common applications of P2P networks we can include:

- *File sharing.* File sharing applications are the most famous and also the most controversial of all the possible applications of P2P technologies. A feature of P2P is that information is distributed on a completely flexible, secure and dynamic way, making the control really complicated, so it has been traditionally used for spreading copyrighted material.
- *Distribution of content.* Content distribution is different from file sharing. P2P networks are useful for content distribution, for example, VOD (Video-On-Demand) applications or video or audio streaming (such as television or radio). Again, P2P is an ideal medium to improve the efficiency and quality of the application at a lower cost.
- *Data synchronization.* Synchronization is similar to the distribution of content, with the difference that is more granular level. The home and business users get used to synchronize data between different devices, for example, contacts between the mobile and computer. Synchronization via P2P could be used, for example, for manufacturers to distribute software patches to their customers applications, or to make possible that teleworkers update their documents between their home networks and the storage servers of the company.
- *Instant messaging, IP telephony and videoconferencing.* Centralized systems used by traditional IP communications, such as MSN Messenger, are too costly for the service provider, and also the cost increases in proportion to the number of users. The result is that companies offering such services, allow very few server resources per user, and this degrades the quality of communications. Therefore, the next logical step in the revolution caused by the exchange of file through P2P technology

1.7. Classification and taxonomy

was IP telephony, and Skype was the first program to develop a telephone network of this type. Skype uses all available resources efficiently within a network thereby increasing the percentage of correct calls and quality of these, at levels slightly lower than the basic telephone system. In addition to IP telephony, Skype allows for instant messaging sessions, video conferencing and file transfer directly between users.

- *Collaboration.* Collaborative information (groupware) consists of several technologies and processes to eliminate the isolation of individual employees and build a new cooperative business model. One of the great advantages of using P2P technology for collaboration is that an application can be fully developed without relying on servers or other centrally managed systems. Thus, P2P programs are responsible to discover other nodes and securely route messages between different users.
- *Sharing of processing cycles.* Personal computers are continually improving in speed, storage space and price. However, most applications do not fully utilize the resources of computers, so that spare capacity is very important. P2P applications and protocols can use excess computing power and storage to create systems where all processing is done by peers rather than servers. The idea is to use the spare computing power to solve complex problems by dividing them into subproblems that can be solved independently by a large number of computers.

1.7 Classification and taxonomy

Regarding to the mechanisms and protocols in which P2P networks are based, we can clearly distinguish two necessary components: I) Mechanisms for locating resources (such as users, services or content), II) communication mechanisms to access resources (such as the establishment of a call or the download a file).

Typically, P2P networks are classified according to the mechanisms for locating resources. There are different solutions by which a node can query for the address of a resource or a path towards it.

1.7.1 Unstructured P2P networks

An unstructured P2P network is “an overlay in which a node relies only on its adjacent nodes for delivery of messages to other nodes in the overlay. Example message propagation strategies are flooding and random walk” [Buford et al., 2008]. The graph structure formed by unstructured overlays can be compared to that of random graphs, scale-free random graphs, graphs exhibiting small world phenomena, and other social networks. An important research focus for unstructured overlays has been the design of efficient search, including query propagation, resource placement, and query processing.

Regarding to query propagation, several alternatives have been proposed. We assume that the query contains the search criteria, such as filename or keywords. The client does not know which peers in the overlay have the resource. In the *flooding* approach it sends a query to every peer it knows. If the neighbor peers do not have the resource, they can

Figure 1.2: Unstructured P2P network.

in turn forward the request to their neighbors, and so on. A few checks are needed to prevent messages from circulating endlessly. First, in case the message loops back or is received over more than one path, each peer can keep a list of message identifiers that it has previously received. If it sees the same message again, it simply drops the duplicate message. Second, so that peers do not have to remember messages for an arbitrary time, which would require a continuously growing amount of storage, each message can have a time-to-live (TTL) value that limits its lifetime. The TTL value of a message is set by the message originator and decremented by 1 at each peer that receives the message. When the TTL value of a message reaches 0, it is no longer forwarded.

To avoid the message overhead of flooding, unstructured overlays can use some type of *random walk*. In random walk, a single query message is sent to a randomly selected neighbor. The message has a TTL value that is decremented at each hop. If the query locates a node with the desired resource, the search terminates and the resource is returned. Otherwise the query fails, as determined by a timeout or an explicit failure message returned by the last peer on the walk. The initiating peer has the choice to reissue a query along another randomly chosen path. To improve the response time, several random walk queries can be issued in parallel.

Some networks, like Freenet [Clarke et al., 2000], use a query propagation mechanism based on *hill climbing*. Both resources and peers have identifiers. Identifiers are created using the SHA-1 hash function. Peer identifiers are called *routing keys*. Each peer has a fixed-size routing table that stores links to other peers. Each entry contains the routing key of the peer. Freenet uses key-based routing (KBR) for inserting and retrieving objects in the mesh. Requests are forwarded to peers with the closes matching routing key. If a request along one hop fails, the peer will try the next closest routing key in its routing table. Depending on the organization of the links and the availability of peers, it is possible that requests could fail.

On the other hand, peers in a computer network can be seen as agents for their users. Although the context in the P2P environment is usually not identical to the offline social network, many P2P social applications form communities of interest. Thus the goal of *social overlay* design is for peers to interconnect in a way that mimics social networks and that can then be used for routing queries to neighbor peers who are likely to have

1.7. Classification and taxonomy

Type	Design	Features	References
Flooding	Gnutella 0.4	Peers use flooding of requests	http://rfc-gnutella.sourceforge.net
Random Walk	Gia	Uses techniques such as dynamic topology adaptation, active flow control, one-hop index replication, and biased random walk to improve performance	[Chawathe et al., 2003]
	LMS	Local minima search proactively replicates objects using consistent hashing of object identifiers to place objects at close node identifiers	[Morselli et al., 2005]
Hill Climbing	Freenet	Routing using hill climbing with backing, and providing security, anonymity, and deniability	[Clarke et al., 2000]
Social Overlays	Tribler	Social-based overlay in which peers exchange preference lists to exploit social affinity between peers with similar preferences	[Pouwelse et al., 2008]
Semantic Overlays	INGA	Semantic overlay in which each peer organizes a semantic index for its content, and queries are routed according to the associated topic, and evaluated using a semantic matching function	[Löser et al., 2007]

Table 1.2: Examples of unstructured overlays.

the information. Unlike other overlays, such social overlays do not rely on network layer relationships between peers. Social overlays require ways to describe peers' interests, discover peers with similar interests, order multiple peers from most similar to least similar for both queries and overlay relationships, and revise peer relationships due to changes in interests.

Finally, it is also worth introducing *semantic overlays*. In INGA [Löser et al., 2007], each peer organizes a semantic index for its local content using the Resource Description Framework (RDF) ontology representation language. Information is associated with a topic, and topics are organized in a semantic hierarchy. The semantic hierarchy and topic set is shared across all peers. Queries for information are routed according to the associated topic and evaluated using a semantic matching function evaluating the topic against topics in the local index.

Another important research focus has been the optimal graph structure for unstructured overlays, and decentralized algorithms to form and maintain these graphs structures under changing peer and object populations. Influential structured overlay designs are listed in Table 1.2.

1.7.2 Superpeer-based P2P networks

A superpeer is a node in a P2P network that operates both as a server to a set of clients, and an equal in a network of superpeers [Yang and Garcia-Molina, 2003]. Superpeer networks introduce a balance between the inherent efficiency of centralized search, and the autonomy, load balancing and robustness to attacks provided by distributed search. Furthermore, they take advantage of the heterogeneity of capabilities (e.g., bandwidth, processing power) across peers.

Clients are connected to a single superpeer only. Figure 1.3 illustrates what the topol-

Figure 1.3: Superpeer-based P2P network.

ogy of a superpeer network might look like. We call a superpeer and its clients a *cluster*, where *cluster size* is the number of nodes in the cluster, including the superpeer itself. When a superpeer receives a query from a neighbor, it will process the query on its clients' behalf, rather than forwarding the query to its clients. In order to process the query for its clients, a superpeer keeps an index over its clients' data. This index must hold sufficient information to answer all the queries. For example, if the shared data are files and queries are keyword searches over the file title, then the superpeer may keep inverted lists over the titles of files owned by its clients. If the superpeer finds any results, it will return one Response message. This Response message contains the results, and the address of each client whose collection produced a result.

If the superpeer does not find any results, then it will submit the query to its neighbors as if it were its own query, and forward any Response messages it receives back to the client. Outside of the cluster, a client's query is indistinguishable from a superpeer's query. Since clients are shielded from all query processing and traffic, weak peers can be made into clients, while the core of the system can run efficiently on a network of powerful superpeers. Hence, as mentioned earlier, superpeer networks use the heterogeneity of peer to their advantage. Also, the overhead of maintaining an index at the superpeer is small in comparison to the saving in query cost that this centralized index allows.

In order for the superpeer to maintain the content index, when a client joins the system, it will send metadata over its collection to its superpeer, and the superpeer will add this metadata to its index. When a client leaves, its superpeer will remove its metadata from the index. If a client ever updates its data (e.g., insertion, deletion or modification of an item), it will send this update to the superpeer as well.

On the other hand, a superpeer becomes a single point of failure for its cluster, and a potential bottleneck. When the superpeer fails or simply leaves, all its clients become temporarily disconnected until they can find a new superpeer to connect to. To provide reliability to the cluster and decrease the load on the superpeer, it is possible to introduce redundancy into the design of the superpeer. It is said that a superpeer is *k-redundant* if there are k nodes sharing the superpeer load, forming a single "virtual" superpeer. A k -

redundant superpeer has much greater availability and reliability than a single superpeer. If one superpeer fails, the others may continue to service clients and neighbors until a new superpeer can be found. The probability that all superpeers will fail before any failed superpeer can be replaced is much lower than the probability of a single superpeer failing.

However, superpeer redundancy comes at a cost. In order for each superpeer to have a full index with which to answer queries, a client must send metadata to each of these superpeers when it joins. Hence, the aggregate cost of a client join action is k times greater than before. Also, neighbors must be connected to each one of the superpeers, so that any superpeer may receive messages from any neighbor.

1.7.3 Structured P2P networks

A structured P2P network is: “an overlay in which nodes cooperatively maintain routing information about how to reach all nodes in the overlay” [Buford et al., 2008]. Compared to unstructured overlays, structured overlays provide a limit on the number of messages needed to find any resource in the overlay. This is particularly important when searching for infrequently occurring or low popularity objects. In order to provide deterministic routing, peers are placed into a virtualized address space, the overlay is organized into a specific geometry, and a distance function over the resource keys and peers address space is defined for the routing algorithm. Structured overlays use key-based routing in which “a set of keys is associated with addresses in the address space such that the nearest peer to an address stores the values for the associated keys, and the routing algorithm treats keys as addresses” [Buford et al., 2008]. Each peer has a local routing table which is initialized when the peer joins the overlay, using a specified bootstrap procedure. Peers periodically exchange routing table changes as part of the overlay maintenance.

Prefix routing is used in many structured overlays. PRR [Plaxton et al., 1997] uses *suffix-based routing*, which is a symmetric case of prefix routing. Suffix and prefix-based routing match increasing portions of the destination address at each hop along the path until the destination is reached. For example, if the target address is the hexadecimal address 3A9F1, in suffix-based routing matching 4 bits at a time, successive hops along the path match XXXX1, XXXF1, XX9F1, XA9F1, and 3A9F1, where X is a wildcard.

On the other hand, other structured overlays use logarithmic degree routing. Chord [Stoica et al., 2003] organizes peers on a logical ring, and peers maintain neighbor pointers spaced at logarithmic intervals around the ring. In addition, each peer has a link to its predecessor and successor peers on the ring. The Chord routing table is called *finger table*.

Unlike logarithmic-degree graphs in which each peer’s interconnections grow at an $O(\log N)$ rate, in a constant degree graph each peer has a fixed maximum number of interconnections, independent of the overlay size. This has the benefit of fixed-size routing state and potentially low overlay maintenance costs. Offsetting these benefits are complexity in routing and maintenance. The overlay address space is typically large and sparsely populated, requiring that the geometry structure be adapted. The dynamics of overlay membership require that the geometry be maintained. Because of these issues, several designs “simulate” or “approximate” constant degree graphs. Some examples of constant degree graphs are De Bruijn, butterfly, and cube-connected cycles (CCC).

Type	Design	Features	References
Prefix routing	PRR	First DOLR algorithm, used suffix based routing	[Plaxton et al., 1997]
	Tapestry	Based on PRR with an added join/leave and maintenance mechanism	[Zhao et al., 2004]
	Pastry	Prefix routing with last hop using neighbor table	[Rowstron and Druschel, 2001]
Logarithmic degree	Chord	Logarithmic spaced links to neighbors around predecessor-successor ring, consistent hashing, unidirectional requests	[Stoica et al., 2003]
	DKS(n,k,f)	Distributed k -ary search with routing region at each hop divided into k regions. $k = 2$ similar to chord	[Alima et al., 2003]
	Kademlia	XOR distance function, parallel requests	[Maymounkov and Mazières, 2002]
Fixed degree	CAN	D -torus with cartesian coordinate system	[Ratnasamy et al., 2001]
	Viceroy	Butterfly	[Malkhi et al., 2002]
	Koorde	De Bruijn, based on Chord	[Kaashoek and Karger, 2003]
O(1)-hop	EpiChord	Iterative parallel lookup with opportunistic maintenance	[Leong et al., 2006]
	OneHop	Peers organized into slices and units, requests routed through slice and unit leaders, with active maintenance	[Gupta et al., 2003]

Table 1.3: Structured overlays by category.

Though many multihop designs have demonstrated a high degree of scalability and robustness, the performance might not satisfy all applications. An O(1)-hop (or simply one-hop) overlay maintains sufficient routing information to approach constant hop routing independent of the size of the overlay. To do this, peers must have enough storage capacity to store the routing table and enough upstream and downstream bandwidth capacity to propagate the majority of join and leave events to all peers in the overlay. Table 1.3 summarizes many of the designs for structured overlays.

1.8 Motivations and goals

The properties of P2P networks make them very appropriate for the development of new information systems and services. In this work, the main characteristics of the previously presented P2P networks (unstructured, superpeer-based and structured) are studied in depth. In addition, it is also analyzed their suitability for the development of specific information systems and services over them. On this matter, unstructured P2P networks have been used to develop a new anonymity service for communications networks, and superpeer-based P2P networks have been used to design a new Video-On-Demand (VOD) service distribution architecture. On the other hand, the reliable routing functionality of structured P2P networks has been used to develop a self-organizing mechanism for superpeer-based P2P applications, and a distributed hash table (DHT) storage system developed over a structured P2P network has been used to implement the information

system component of a traceability application.

1.9 Structure and contributions

The rest of the Thesis is organized as follows:

- Chapter 2 introduces the main characteristics of unstructured P2P networks. After that, the concept of anonymity in communication networks is presented, and the most important mechanisms to obtain anonymity in IP networks are described. One of them, called Crowds, can be easily implemented over unstructured P2P networks to provide an anonymity service. This chapter proposes a mechanism that limits the length of overlay Crowds paths preserving the anonymity degree achieved by the original algorithm. This chapter is supported by the following publications [Muñoz-Gea et al., 2008d], [Muñoz-Gea et al., 2009a] and [Manzanares-Lopez et al., 2010b].
- Chapter 3 introduces the main characteristics of superpeer-based P2P networks. After that, the fundamental components of any P2P VOD system are presented. In this chapter it is extended the work presented in [Nafaa et al., 2008], which implements a VOD service distribution architecture in a superpeer-based P2P network. First of all, the system is evaluated in a broadband operator scenario, and it is presented a new mechanism which tries to dynamically re-allocate the streams. Secondly, the previous VOD system is extended to the Internet context, and it is presented a new dispatching algorithm. It is demonstrated that the global performance of the system is improved with the new proposed mechanisms. The publications related to this chapter are [Muñoz-Gea et al., 2010b] and [Muñoz-Gea et al., 2010c].
- Chapter 4 introduces the main characteristics of structured P2P networks. Although P2P is beneficial when removing a centralized server, new mechanisms are required for network maintenance. Self-organization can be a good way to solve the control issues in decentralized P2P architectures. In this chapter, it is presented a self-organizing mechanism for superpeer-based P2P applications, which makes use of the reliable routing functionality associated to structured P2P networks. The proposed mechanism is implemented in two superpeer-based P2P applications. The first one is a two-level P2P application, where all the superpeers are members of an Application Level Multicast (ALM) group. The second one is a hierarchical P2P application. This application is similar to the previous one, but in this case the superpeers have a direct communication among them using a hierarchical architecture of superpeers. The work of this chapter has been published in [Muñoz-Gea et al., 2007a], [Muñoz-Gea et al., 2007b], [Muñoz-Gea et al., 2008a], [Muñoz-Gea et al., 2008b], [Muñoz-Gea et al., 2008c], [Muñoz-Gea et al., 2008e], [Muñoz-Gea et al., 2009b] and [Manzanares-Lopez et al., 2010a].
- Chapter 5 firstly highlights the main characteristics of Distributed Hash Table (DHT) storage systems. This kind of applications are implemented over the Key-Based Routing (KBR) interface offered by a structured P2P network. Radio Frequency

Identification (RFID) technology can track the movement of products throughout an entire supply network. In this chapter, it is proposed a mechanism for automatically obtaining the supply network associated to a specific product using the EPCglobal Network. This mechanism is integrated within the Discovery Services (DS) component, which is currently being developed. Recently, there have been several proposals for the implementation of the DS based on centralized mechanisms. In this chapter, it is developed a DS prototype based on DHT and an access control service to be integrated within the DS prototype. Finally, the mechanism to reconstruct supply networks is integrated within the DS prototype. The following publications [Muñoz-Gea et al., 2010a] and [Manzanares-Lopez et al., 2011] present the outcomes of this Chapter.

- Finally, Chapter 6 presents the main conclusions drawn in this Thesis and discusses future works.

Providing anonymity using unstructured P2P networks

This chapter introduces the main characteristics of unstructured P2P networks and it describes Gnutella 0.4 as a well-known example of this kind of networks. After that, the concept of anonymity in communication services is presented, and the most important mechanisms to obtain anonymity in IP networks are described. One of them, called Crowds, can be easily implemented over unstructured P2P networks to provide an anonymity service. The main idea behind Crowds is to hide each user's communications by routing them randomly within a group of nodes. However, in this algorithm the forwarding procedure is not limited in any way. Therefore, the costs associated with the communication in the overlay network may grow excessively. This chapter proposes a mechanism that limits the length of overlay Crowds paths preserving the anonymity degree achieved by the original algorithm. It is demonstrated that it is possible to achieve a good trade-off solution between the anonymity level (the benefit) and the path length (the cost). In addition, it is also demonstrated that the performance of the new mechanism over special attacks (such as, traffic analysis and predecessor attacks) is equivalent to the performance achieved by Crowds.

2.1 Introduction

Unstructured P2P networks were the first generation of P2P networks. These networks do not impose any structure on the overlay topology. That is, they present a flat topological architecture, where all the nodes belong to the same level (see Figure 1.2). The main characteristic of unstructured P2P networks is that the location of resources is completely unrelated to overlay topology. As a result, resource search mechanism in such networks are essentially based in random search. Gnutella 0.4¹, for example, used flooding techniques, where the queries were sent to all the neighbor nodes until the corresponding item or node was found. Therefore, the routing is completely decentralized, and the routes to content providers are only established on demand. The way to discover nodes is usually

¹<http://rfc-gnutella.sourceforge.net/developer/stable/index.html>

based on a bootstrap server and also on getting the information from neighbors discovered lately through the bootstrap server.

These systems are resilient to peer dynamics (in case of peer joining and leaving, they can easily reconstruct the overlay), and support complex queries with meta information [Jin and Chan, 2010]. Peers usually check their local storage to answer a query. It makes possible that the client can use keywords or regular expressions in the queries, that lately can be applied over the name of the resources or their meta-data.

The major drawback of unstructured P2P system is the low search efficiency, especially for unpopular resources. These resources have few copies in the system, and the search for such a resource may lead to large-scale flooding. Hence, it is important to design efficient search mechanisms. Usually, with more data replications in the network, a resource can be located more quickly. It is hence a tradeoff between storage cost and search time.

2.2 Gnutella

Gnutella 0.4¹ is an unstructured P2P system for file sharing. It first appeared in 2000 and got quick development in a few years. In the basic Gnutella protocol, when a new peer joins the system, it first connects to some public peers. A new peer then sends a PING message to any peer it is connected to. The message announces the existence of the new peer. Upon receiving a PING message, a Gnutella peer returns a PONG message and propagates the PING message to its neighbors. A PONG message contains the IP address and port of the responding peer, and information of files being shared by the responding peer. In a dynamic network with frequent peer joining and leaving, a peer periodically sends PING messages to its neighbors.

Search in Gnutella is based on flooding. A search query is propagated to all neighbors from the original requesting peer. The query is replicated and forwarded by each intermediate peer to all its neighbors. Each intermediate peer also examines its local contents and responds to the query source on a match. The query responses are routed back along the opposite path towards the original requesting peer. To reduce the amount of query messages in the network, each query message contains a time-to-live (TTL) field. The TTL value will be decremented by one at each peer. When it reaches zero, the message is dropped.

Figure 2.1 shows the search process in Gnutella [Jin and Chan, 2010]. Suppose peer A wants to search for some file. It floods its search query to its neighbors, i.e., peers B and D in the figure. When peer B receives the query, it checks whether itself holds the matching file. If not, it forwards the query to its neighbors. As in the example, peer B forwards the query to its neighbor C. Suppose C holds the file that A wants. C returns a response to the peer that sends it the query, which is B in the figure. B then continues forwarding the response to the query sender A. Finally, A contacts C to download the file.

A limitation of Gnutella is its relatively low search efficiency. In flooding search, the number of query messages exponentially increases with the number of overlay hops. Then, a query may generate many messages, especially for unpopular files, where a query

¹<http://rfc-gnutella.sourceforge.net/developer/stable/index.html>

2.3. Anonymity

Figure 2.1: Search process in Gnutella.

has to go through many overlay hops and many peers before reaching a matching peer. Given the huge number of peers in the system, the traffic load for queries could be significantly high. The use of TTL can reduce the number of query messages. However, choosing an appropriate TTL is not easy. If the TTL is too high, peers unnecessarily burden the network. If the TTL is too low, a peer might not find the file even though a copy exists somewhere.

In addition, there are many duplicate messages in flooding search, particularly in heavily connected networks. This is because multiple copies of a query may be sent to a peer by its multiple neighbors. These duplicate queries consume extra processing power at peers and network bandwidth. Hence, it is necessary to develop some duplication detection mechanisms. However, even with duplicate suppression, the number of duplicate messages in flooding can be excessive, and the problem gets worse as the TTL increases.

2.3 Anonymity

In the common language, anonymity refers to the ability to do something without revealing one's identity (in this case, the user's), so that nobody else can tell who has done this [Pfitzmann and Hansen, 2010]. In a more technical language, anonymity is the case in which, within a certain set of possible actors (called anonymity set), an attacker -also called corrupt node- cannot associate actions to actors directly. Anonymity can be a very desirable characteristic in communication services. One example is electronic voting. In such an application anonymity is needed to prevent the vote being tied to the voter.

Anonymity in IP networks can be obtained by means of different mechanisms: source-rewriting mechanisms, broadcast protocols and Dining-Cryptographer (DC)-nets. They have been studied in [Manzanares-Lopez et al., 2010b]. Source-rewriting mechanisms provide anonymity via packet reshuffling. Examples of these systems in IP networks are Mixes, Onion Routing and Crowds. In these systems, messages are sent through the network using random paths different than the shortest path. Each node that forwards the message rewrites the source field of the packet with its own identifier. This mechanism ensures that when a particular node A receives a message from a particular node B, node A cannot be certain that B is the originator of the message. In Onion Routing, messages are routed through Onion Routers which are a dedicated set of servers. A message is encrypted into a layered data structure which contains cryptographic information for every hop on the path. The path through the network is established by the sender, who must have

a complete knowledge of the routers. The last router in the path receives the message, decrypts the last layer of the message, and forwards it to the final destination. In Crowds, the paths are determined locally and on-the-fly by the forwarders instead of a priori by the senders, and encryption is done via pair-wise key exchange and symmetric encryption between forwarders. When a Crowds member request a message from another, it makes a random choice to either forward the request to another Crowds member or submit it to the server the request is intended for. In both of these systems, returned packets follow the reverse route from the sender.

Anonymous systems based on the broadcast mechanism, such as P^5 [Sherwood et al., 2002], provide sender and receiver anonymity by transmitting encrypted packets at a constant rate to all participants. When a node has no data packets to send, it sends noise packets, which is then propagated throughout the network in the same manner as data packets. This approach provides strong anonymity, as a passive eavesdropper cannot tell which packets contain data and which packets are noise. An example might illustrate it well. Suppose two partners want to communicate anonymously with each other in an open environment. They can speak loudly in such a jargon that only they can understand the meaning of the sentences. Thus, even though other people within the sound scope can hear the talking, they cannot understand the content of the conversation. To hide the content of the message, users usually perform encryption to the messages using the receiver's public key. Thus, only desired receivers can recover the original data.

DC-nets [Chaum, 1988] or Dining-Cryptographer networks were introduced by David Chaum in 1988. This anonymity solution is based on the broadcasting of XOR combination of messages to the system in predefined time slots. As in other broadcast-based anonymity systems, the main problem with DC-nets is that bandwidth decreases quadratically with respect to the number of nodes. That is because they are not used often in practice.

DC-nets work as follows. Suppose there are three participant, Alice, Bob and Charlie, one of whom (Alice) wants to communicate a one-bit message to Charlie. Each pair of users tosses a coin in secret; call these AB , AC and BC . Bob and Charlie report the XOR of their two coin tosses (Bob reports $B = AB \oplus BC$, Charlie reports $C = AC \oplus BC$). Alice, on the other hand reports the XOR of her coin tosses along with her message; that is, she reports $A = AB \oplus AC \oplus m$. A eavesdropper or a participant can then take the XOR of all packets sent in the network, i.e. $A \oplus B \oplus C = m$, since the double-reported coin tosses cancel out to reveal m . Neither the eavesdropper nor a participant can tell that Alice originated the message since it is composed with the participation of all parties. Herbivore [Sirer et al., 2004] is an anonymous broadcasting protocol that uses DC-nets.

Source-rewriting mechanisms can be easily implemented over unstructured P2P networks to provide anonymity services. Therefore, these mechanisms are going to be explained in-depth in next section.

2.4 Source-rewriting mechanisms

Source-rewriting systems provide anonymity using indirection transmission: messages are routed from sender to receiver through intermediate nodes building a multiple-hop

2.4. Source-rewriting mechanisms

Figure 2.2: General architecture of a Mix node.

path. Examples of this type of protocols designed to offer anonymous IP communication are Mixes [Chaum, 1981], Onion Routing [Syverson et al., 1997] and Crowds [Reiter and Rubin, 1998].

2.4.1 Mixes

David Chaum introduced the concept of Mix-net [Chaum, 1981] for anonymous communication. Anonymity is achieved by using mixes, which are proxies that help in hiding the sender from the receiver, and also provide sender and receiver with unlinkability against a global eavesdropper.

This system forwards encrypted messages through a chain of nodes, where each one can only know its predecessor and successor in the chain. The basic principle followed by this method is that every node in the chain waits until it receives a group of messages and then it forwards the mix-up of messages. That is, each node collects a group of encrypted messages from the senders, decrypts them, batches, reorders them, removes the senders' name and identifying information, and forwards them to next node (see Figure 2.2). This makes it difficult for an eavesdropper to correlate the output messages with the corresponding input messages. The main drawback of this system is that each node in the chain has to wait until it gets sufficient number of messages for a proper mix-up. Some anonymous networks, such as Mixmaster and GNUnet, are based on this idea.

2.4.1.1 Mixmaster

Mixmaster [Möller et al., 2003] is an anonymous network based on the Mix-net concept. In this system, messages are encrypted using hybrid RSA and EDE 3DES encryption, while their sizes are kept constant by appending random noise at the end of the message. In version two, the integrity of the RSA encrypted header is protected by a hash, making tagging attacks on the header impossible. In version three the noise to be appended is generated using a secret shared between the network and the sender of the message.

Mixmaster allows large messages to be divided in smaller chunks and sent independently through the network. If all the parts end up at a common mix, then reconstruction happens transparently in the network, and so large emails can be sent to users without requiring a special software. It also allows messages to be sent multiple times, using different paths.

2.4.1.2 GUNet

GNUnet is a searchable, decentralized network, whose main goal is to provide an anonymous file-sharing application. This network includes a technique for efficient content encoding (ECSR - the encoding for censor-ship resistant sharing) and a new protocol for anonymous routing, GAP (GNUnet's Anonymity Protocol) [Bennett and Grothoff, 2003], which is based on Mix-net.

The content encoding technique divides a file into small blocks of 1 kb which are hashed individually. These blocks are called Data Blocks (*DBlocks*). The *DBlocks* are grouped into Indirection Blocks (*IBlocks*), and on top of these lies the Root Block (*RBlock*), which contains the description of the file and the query-hash to retrieve the file. Splitting the file into small chunks makes content distribution much easier. Further, the query for *RBlock* is sent encrypted so that the intermediaries cannot assess the information for which the user is looking for.

On the other hand, the GAP protocol has two types of messages: queries and replies. Furthermore, there exist two types of queries: search queries and download queries. A node requests for the *RBlock* of a particular file sending a search query. Once it gets the *RBlock*, it can download the entire file by sending several download queries. After receiving a download query a particular node checks its resources such as bandwidth, storage space or CPU to determine whether it can process the query or not. It drops the query if it is too busy. Then, it checks for the availability of the requested content. Finally, the different nodes in this network achieve anonymity by acting as intermediaries of messages, as in Mix-net.

2.4.2 Onion Routing

Onion routing [Syverson et al., 1997] is a technique developed by David Goldschlag, Michael Reed and Paul Syverson for anonymous communication over a network. The main goal of this protocol is to protect the identity of an initiator and responder of a message, and also to protect the content of the message while it is traversing over the network.

The concept of routing onions is the core concept used by this protocol for anonymous communication. When an initiator wants to establish anonymous communication with a responder, he sends the message to an *application proxy* (see Figure 2.3). Then, it forwards the message to an *onion proxy* which selects some routers and establishes a route, constructing an *onion*. The onion is a recursively layered data structure that contains the information about the route to be followed over a network. The routers in the onion are termed as *core onion routers*.

The next step in this process is to forward the onion to the *entry funnel*, which is an entrance for routing in the network. The entry funnel will decrypt the first layer to see the information about the next hop in the route. This process continues until the onion reaches the *exit funnel*. The exit funnel is responsible for traversing the packet to its destination.

Let us take an example for a better understanding of this protocol: User *A* wishes to send a message *m* to user *B*. Suppose the routing path is through onion routers $S_1, S_2, \dots, S_n$ (in that order), where S_i has public key P_i . It is assumed that all the parties involved use a common encryption algorithm *E*. Then, *A* sends the message $m' =$

2.4. Source-rewriting mechanisms

Figure 2.3: Architecture of Onion Routing.

$E_{P_1}(E_{P_2}(\dots(E_{P_n}(m, B), S_n)\dots, S_3), S_2)$ to S_1 . Each onion router on the path decrypts the message to find the identity of the next onion router and then forwards the encrypted payload to the next router. This process is repeated until the last router sends the message to its final destination, which is able to decrypt the original message m . If the destination responds to the message, the response message will be routed along the same path in reverse.

2.4.2.1 Tor

Tor [Dingledine et al., 2004] is an advanced version of Onion Routing. Instead of using a single onion, Tor implements an incremental path-construction in which the initiator extends the path hop by hop and negotiates session keys with each intermediate node on the path. As a benefit, the anonymous transmission is more reliable since the intermediate nodes on the path are online after the path construction. Therefore, this system is more convenient than onion routing in supporting TCP-based applications.

The algorithm to select the onion routers consists of two parts: the first part is used to select the entrance router, and the second part is used to select subsequent routers. The entrance router selection algorithm works by automatically selecting an onion router that is marked by the trusted directory servers as being “fast” and “stable”. An onion router is marked as “fast” when its bandwidth is above the median of all bandwidth advertisements. On the other hand, a stable router is defined as one that advertises an uptime that is greater than the median uptime of all other routers.

The non-entrance router selection algorithm is intended to optimize onion router selection for bandwidth and uptime, while not always choosing the very best nodes every time. Tor has a set of TCP ports that are designated as “long-lived”. If the traffic transiting a path uses one of these long-lived ports, Tor will optimize the path for stability by pruning the list of available routers to only those that are marked as stable. The next part of the algorithm optimizes the path for bandwidth. Briefly, this algorithm works as

follows: Let b_i be the bandwidth advertised by the i -th router, and let us assume that there are N routers. Then the probability that the i -th router is chosen is approximately $b_i / (\sum_{j=1}^N b_j)$. The most significant feature of the bandwidth optimization algorithm is that the more bandwidth a particular router advertises, the greater the probability that the router is chosen.

2.4.2.2 Tarzan

Tarzan [Freedman and Morris, 2002] is an anonymous network based on Onion Routing. All participant nodes run a software that: (1) discovers other participant nodes, (2) manages tunnels through chains of other participants to anonymize the packets, (3) forwards packets to implement other nodes' tunnels, and (4) operates a NAT to forward other participants' packets onto the ordinary Internet.

Tarzan routes packets through tunnels involving a randomly chosen sequence of Tarzan peers using mix-style layered encryption. Typical use of Tarzan proceeds in three stages. First, a node running an application that desired anonymity selects a set of nodes to form a path (also called tunnel) through the overlay network. Tarzan uses a simple gossip-based protocol similar to the Name-Dropper resource discovery protocol [Harchol-Balter et al., 1999] for peer discovery. The tunnel initiator clears each IP packet's source address, performs a nested encryption, and encapsulates the result in an UDP packet. Next, this source-routing node establishes a tunnel using these nodes. Finally, it routes data packets through this tunnel. That is, if the tunnel consists of a sequence of nodes $h_1, h_2, \dots, h_l, h_{pnat}$ and the public key for each node is k_{hi} , the tunnel initiator produces the block $\{\{\dots\{\{p\}_{k_{hpnat}}\}_{k_{hl}}\dots\}_{k_{h2}}\}_{k_{h1}}$ from the input packet p , and forwards it to h_1 . h_1 will decrypt one layer of encryption and forward it on to the next hop. This process continues until the packet reaches the last node in the tunnel, which strips off the innermost layer of encryption. The exit point of the tunnel is a NAT, which forwards the anonymized packets to servers that are not aware of Tarzan.

In addition, Tarzan introduces the concept of traffic mimics to implement cover traffic mechanisms. Upon joining the network, a node a asks k other nodes to exchange mimic traffic with it. The node establishes a bidirectional, time-invariant packet stream with each of the mimic nodes. Into this packet stream real data can be inserted. Tarzan uses a structured P2P lookup algorithm to discover the mimic nodes. Node a chooses its i th mimic as the peer returned by $lookup(K_i)$, where $K_i = hash_i(a.IPaddress, date)$ and $hash_i$ is a cryptographic hash function recursively applied i times. The uses of mimics slightly modifies the construction of a tunnel. Nodes only construct tunnels over links protected by cover traffic. To initiate a tunnel, node a chooses a mimic M_i^a as its first tunnel hop. This node M_i^a has its own mimics, and returns this list to a . Node a selects the next hop from the resulting set and continues by tunneling an establish request to the second hop via the first hop. It repeats this process for l mimics. Finally, a selects a random node for h_{pnat} by $lookup(random)$.

2.4. Source-rewriting mechanisms

2.4.2.3 I2P

I2P¹, also known as Invisible Internet Project, was started in 2003. I2P uses a technique called *garlic routing* for anonymous communication, in which the data is wrapped with several layers of encryption.

It is important to know the three critical concepts used by this network for a better understanding of it. First, I2P makes a distinction between the nodes participating in the network as “routers” and the endpoints, known as “destinations”. Secondly, I2P makes use of “tunnels” to traverse messages in the network. A tunnel is a path selected from the set of routers. These tunnels can be categorized as either inbound tunnels (where everything given to it goes towards the creator of the tunnel) and outbound tunnels (where the tunnel creator shoves messages away from them). When a sender wants to send a message to a certain destination, he will select one of his outbound tunnels with instructions to forward the message to one of the inbound tunnels of the destination, thereby reaching the destination. Finally, the third concept is “network database” (or “netDb”), which maintains a pair of algorithms that are used to share network metadata. The two types of existing metadata are “routerInfo” and “leaseSets”. The routerInfo gives the information that is required to contact a particular router and the leaseSet gives the necessary information required for routers to contact a destination.

As already stated, the message is wrapped with several layers of encryption so that the message looks entirely different in each hop of the tunnel. One more feature of I2P is that every participant in the network chooses the length of the tunnels and, in doing so, makes a tradeoff between anonymity, latency, and throughput according to its own needs.

2.4.3 Crowds

Crowds [Reiter and Rubin, 1998] is an anonymous protocol for web-transactions proposed by Reiter and Rubin. This protocol involves a group of users called *crowd*, each of whom wants to communicate with a corresponding web server but without revealing his identity. The idea is to randomly route each message through the crowd until one member of the crowd decides to pass it to the server.

A crowd can be thought of as a collection of user processes (also called *jondos*). When a user starts the jondo on the user’s computer, it contacts a server called *blender* to request admittance to the crowd. If admitted, the *blender* reports to this *jondo* the current membership of the crowd and information that enables this *jondo* to participate in the crowd.

The user selects this jondo as her web proxy by specifying its host name and port number in her web browser as the proxy for all services. Thus, any request coming from the browser is sent directly to the jondo. Upon receiving the first user request from the browser, the jondo initiates the establishment of a random path of jondos that carries its users’ transactions to and from their intended web servers. More precisely, the jondo picks a jondo from the crowd at random, and forwards the request to it. When this jondo receives the request, it flips a biased coin to determine whether or not to forward the request to another jondo; the coin indicates to forward with probability p_f . If the result is

¹<http://www.i2p2.de>

Figure 2.4: Architecture of Crowds.

to forward, then the jondo selects a random jondo and forwards the request to it. Otherwise the jondo submits the request to the end server for which the request was destined (see Figure 2.4).

2.4.3.1 AP3

AP3 [Mislove et al., 2004] is an anonymous protocol built on top of Pastry [Rowstron and Druschel, 2001] and based on Crowds. It relies on a network of peers to forward messages attempting to hide the originator identity.

When a node wishes to anonymously send a message, it first creates an anonymous request object comprised of the message itself and the address of the intended recipient. The request (that does not contain any information that can reveal the originator's identity) is forwarded to a node in the overlay selected by drawing a random key. Upon receiving a request, an AP3 node performs a weighted coin toss to decide whether to fulfill the request and send a message to the intended recipient, or to forward the message to another randomly selected peer. The decision to forward is made with probability p_f . This procedure provides a random path through the P2P network built from a variable number of random hops.

While the above mechanism allows nodes to send message without divulging their identity, it alone is insufficient to support a request-response communication in which the requester does not wish to divulge his identity. Since destination of a request doesn't know the identity of the sender, it is unable to reply. In order to offer this functionality, AP3 provides Anonymous Channels that allow a node to specify a return location for a message without divulging their identity.

When a node wishes to construct an anonymous channel, it first picks a channel identifier (*id*). It looks up the node closest to this *id* using Pastry, and then sends an anonymous message to that node using previously described anonymous sending mechanism. This message travels along a random path and each node along it remembers the previous node in a local table called the forwarding table.

Thus, if a node wishes to anonymously send a request and receive a response, it first creates an anonymous channel and then includes the address of the channel in the anonymously routed request. The response message is transmitted to the responsible node of the channel *id* included in the packet, and then it is forwarded to the sender through the

2.5. Background

anonymous channel. The nodes in the anonymous channel reconstruct the path back to the source node by using their forwarding tables. Using this mechanism, anonymity is preserved as no node along the channel knows if the previous node is the originator or just another intermediate node.

Additionally, when a path is established, the receiver specifies an expiration time that defines the period during which entries remain in the forwarding tables. Thus, forwarding table entries naturally expire over time. If a given channel has expired, the source node can simply create a new and different anonymous path to serve the anonymous channel.

2.4.3.2 Hordes

Hordes [Levine and Shields, 2002] provides sender anonymity by adopting the Crowds probabilistic forwarding mechanism, and achieves receiver anonymity by performing a multicast transmission.

The purpose of using a group multicast is that the participants of the group cannot be easily determined. Since multicast addresses do not work with TCP connections, the transfers that involve a multicast address require the use of a UDP. The forward connection can use TCP, and within the data of the TCP payload it encapsulates the multicast address and a random number, used to identify the return message. Each intermediate jondo forwards this information to the responding jondo. That jondo will take the response payload and send it via the multicast address. Each jondo listens to one multicast address along with other jondos. However, not all jondos listen to the same address. One reason for this is so that a few jondos on the forward path will receive the actual reply from the server. It is very important that at least two jondos listen to an address, so that there is some unlinkability between the multicast address and the actual client.

The differences between Hordes and Crowds are the following. Crowds use TCP for all communication, where Hordes use a combination of UDP and TCP. The reverse path is much more interesting. Since packets travel directly back to the client without going through the jondos on the forward path, they are not subject to the same attacks of request/response matching that Crowds is subject to. In Crowds the forward and reverse data paths follow the same set of jondos, so a timing attack between a request and response may reveal information between the two. A problem with performing an encryption/decryption operation between each jondo is that it is possible for a local eavesdropper to determine the content of all messages originating from that jondo. However, this affects both Crowds and Hordes, since their forward routing schemes are very similar.

2.5 Background

This work [Muñoz-Gea et al., 2008d, Muñoz-Gea et al., 2009a] is aimed at designing a Crowds-based mechanism to provide anonymity in overlay networks. However, in the Crowds algorithm the forwarding procedure is not limited in any way. Therefore, the costs associated with the communications (bandwidth consumption and delay) may grow excessively.

In Crowds, the initiator node creates a packet containing a random path identifier, the IP address of the destination and the data. Then, it flips a biased coin. With probability

$1-p_f$ (p_f is the probability of forwarding and it is a parameter of the system) it delivers the message directly to the destination, and with probability p_f it chooses randomly the next relay node. Each node receiving a packet with a new path identifier randomly decides, based on p_f , whether to forward it to the destination or to another (randomly chosen) relay node. With this algorithm the forwarding procedure is not limited and, as we previously pointed out, it could be a considerable problem regarding communications costs in an overlay scenario.

A possible solution is to restrict the maximum length of the paths. The system operates as the traditional scheme but, when the number of hops reaches a certain limit (called S), the path will be directed towards the destination, regardless of probability p_f . A straightforward implementation of the bounded-length random walk consists of using a time-to-live (TTL) field, initially set to S , and processing it like in IPv4 networks. However, there are multiple situations in which this implementation will immediately reveal to a “corrupt” node whether the predecessor node is the initiator or not.

An approach to solve this problem is to use high and randomly chosen (not previously known) values for the TTL field. However, the objective is to limit the forwarding procedure, and high values for the TTL represent long multi-hop paths. Therefore, the TTL would have to be small. But, in this case, the range of possible random values for the TTL is too restricted, and it results in a similar situation to the one of a well-known TTL value among all the users. In this last case, corrupt nodes can easily derive whether the predecessor node is the initiator of the intercepted message or not.

We can conclude that the TTL methodology is not appropriate to limit the length of multi-hop paths. Next section introduces a new mechanism that limits the length of overlay random walk paths without offering extra information to possible corrupt nodes.

The contribution of this work is twofold: (i) a new mechanism that limits the length of overlay random walk paths is proposed preserving the anonymity degree achieved by the Crowds algorithm. (ii) The anonymity level associated to this mechanism is also carefully characterized and evaluated analytically. It is demonstrated that it is possible to achieve a good trade-off solution between the anonymity level (the benefit) and the path length (the cost). In addition, it is also demonstrated that the performance of the new mechanism over special attacks (such as, traffic analysis and predecessor attacks) is equivalent to the performance achieved by Crowds.

2.6 The anonymous aware *Always Down-or-Up* (ADU) algorithm

The algorithm proposed in this work, like Crowds, is based on the random walk procedure. However, the variance associated with the length of the multi-hop paths is significantly smaller than in Crowds. Therefore, it can be viewed as a quasi-deterministic mechanism of a statistical TTL implementation.

Our first attempt is the *always-down* (AD) algorithm: The path initiator chooses a uniform random number (denoted by u) between 1 and a predefined parameter M ($1 \leq u \leq M$). If the value of u is equal to 1, the bound of the path has been reached and the initiator node sends the request directly to the destination. Otherwise, the node forwards

2.6. The anonymous aware *Always Down-or-Up* (ADU) algorithm

the request to a random node together with the random number u . The next node performs the same operation but replacing the upper bound M by the value of u , and imposing a condition of *lower than* in the inequality. That is, the new random number has to be in the interval $[1, u)$. The mechanism continues in a recursive way, decreasing the size of the interval $[1, u)$ in each step, until a node randomly chooses 1 and sends the request directly to the destination. Obviously, with this simple mechanism we can limit the maximum length of the paths in a statistical manner. Nevertheless, it has an important drawback: small numbers do not reveal anything about the path length, but large ones do, since they can only appear at the first steps of the algorithm. For instance, let us consider the situation in which the initiator node selects a random number $u = M$ and randomly forwards the request to a corrupt node. This node immediately establishes that the sender of the received message is the initiator of the multi-hop path. Therefore, this solution is not suitable to bound anonymous multi-hop paths.

The opposite algorithm, called *always-up* (AU) has the same benefits and drawbacks. In this case, at each step the node chooses a uniform random number between $(u, M]$. When a node randomly selects M , the random walk procedure ends and the request is directly sent to the destination. In this case, large numbers do not reveal anything about the path length, but small ones do, since they can only appear at the first steps of the algorithm. To illustrate this effect, imagine the situation in which the initiator node chooses a random number $u = 1$ and randomly forwards the request to a corrupt node. This node immediately establishes that the sender of the received message is the initiator of the multi-hop path.

Therefore, the previous algorithms are able to statistically bound the maximum length of the overlay paths, but there is a region around the end (or the start, depending on the particular algorithm) of the random numbers set where corrupt nodes can extract some information about the initiator of the paths. In order to avoid this critical drawback, we propose to mix both mechanisms as follows: The path initiator chooses a random number between 1 and M ($1 \leq u \leq M$). When this number is equal to 1 or equal to M , it represents that the bound of the path has been reached and the initiator node sends the request to the destination. If u is lower than or equal to a predefined parameter LOW_BORDER ($u \leq LOW_BORDER$) the algorithm works like AD. However, if u is greater than a predefined parameter TOP_BORDER ($u > TOP_BORDER$) the algorithm operates like AU. Finally, if u drops between LOW_BORDER and TOP_BORDER ($LOW_BORDER < u \leq TOP_BORDER$), the operation mode (AD or AU) is chosen randomly. The parameters LOW_BORDER and TOP_BORDER are not fixed; every path initiator chooses a random value for these parameters when it creates a new path. The only requirement is that these parameters have to be symmetric respect to 1 and M .

This new algorithm is called *always down-or-up* (ADU) and it is able to statistically limit the length of the path in an anonymous environment. In order to speed up the algorithm, mainly in the last steps, we introduce an additional parameter denoted by e which is used as follows: If the new chosen random number is smaller than or equal to e ($u \leq e$) or it is greater than $M - e$ ($u > M - e$), it represents that the bound of the path has been reached and the initiator node delivers the request to the destination. Thereby, now it is not necessary to reach value 1 or M to issue the request directly to the destination and,

Figure 2.5: Parameters of the algorithm.

consequently, the mean length of the multi-hop paths is reduced.

The full set of parameters used by our algorithm is: M , e , LOW_BORDER and TOP_BORDER . Figure 2.5 represents these parameters in a numerical straight line. In order to simplify the nomenclature, from now on the parameters LOW_BORDER and TOP_BORDER are called L and T respectively.

2.7 Evaluation of the ADU paths

In this section we evaluate the characteristics of the paths built using the ADU algorithm. With this objective, we deduce an analytical model for the random variable l that represents the length of the paths. In order to simplify the previous model it is presented in function of the model for the variable l in the AD algorithm.

2.7.1 Analytical model for the length of the AD paths

Using the AD algorithm, the possible values for the random variable l are

$$1 \leq l \leq M - e + 1 \quad (2.1)$$

The probability that l takes the value x in the AD algorithm, with parameter $M = i$, is defined as

$$P_{i,AD}(l = x) \quad 1 \leq x \leq M - e + 1 \quad (2.2)$$

In order to deduce a general expression for the previous probability we use as an example the simplified scenario represented in Fig. 2.6, with parameters $M = 5$ and $e = 2$. It is clear that the probability that the random variables l takes the value 1 is defined as

$$P_{M,AD}(l = 1) = \frac{e}{M} \quad (2.3)$$

On the other hand, from Fig. 2.6 it can be observed that the calculation of the rest of probabilities can be reduced in function of the possible values for the first selected random number (u_1). Next, in order to facilitate the comprehension of the presented model, we are going to concentrate in the calculation of the probability $P_{5,AD}(l = 3)$. In the first place, it is necessary to take into account that to achieve a path with length greater than one, the value of u_1 has to be always greater than e (otherwise the construction of the path finishes in the first hop). Therefore, we could think that the possible values for u_1 to calculate the probability $P_{5,AD}(l = 3)$ are 3, 4 and 5. However, note that if $u_1 = 3$ it is impossible to achieve a path with $l = 3$, since it implies that $u_2 \leq 2 = e$ (because $u_2 < u_1$), and therefore the path finishes in the second hop. As a consequence, the only

2.7. Evaluation of the ADU paths

Figure 2.6: Graphical interpretation.

possible values for u_1 are 4 and 5, and the calculation of $P_{5,AD}(l=3)$ can be reduced to the calculation of the probabilities $P_{5,AD}(l=3|u_1=4)$ and $P_{5,AD}(l=3|u_1=5)$. In the same way, the only possible value for u_1 to calculate the probability $P_{5,AD}(l=4)$ is 5, and its calculation is equivalent to the calculation of $P_{5,AD}(l=4|u_1=5)$. Generalizing this concept, it can be concluded that the possible values of u_1 , in function of e , M and the value of the random variable l , have to satisfy

$$e + (l - 1) \leq u_1 \leq M \quad (2.4)$$

It can be also be observed that the above conditioned probabilities can be calculated in a reduced scenario where $M = u_1 - 1$ and $l = x - 1$. For example, the calculation of $P_{5,AD}(l=3|u_1=4)$ is equivalent to calculate the probability that $l=2$ when $M=3$ ($P_{3,AD}(l=2)$). This property, together with the presented above, can be applied recursively until arrive to the well-known probability $P_{i,AD}(l=1)$. The following equation gives a general expression for the above discussion

$$P_{M,AD}(l=x|u_1=i) = P_{i-1,AD}(l=x-1) \quad x > 1; \quad e + (x-1) \leq i \leq M \quad (2.5)$$

Finally, applying the law of total probability to calculate $P_{M,AD}(l=x)$ when $1 < x \leq M - e + 1$

$$\begin{aligned}
 P_{M,AD}(l = x) &= \sum_{i=e+(x-1)}^M P_{M,AD}(l = x|u_1 = i) \cdot P(u_1 = i) = \\
 &= \sum_{i=e+(x-1)}^M P_{i-1,AD}(l = x - 1) \cdot P(u_1 = i) = \\
 &= \frac{1}{M} \sum_{i=e+(x-1)}^M P_{i-1,AD}(l = x - 1) \quad 1 < x \leq M - e + 1
 \end{aligned} \tag{2.6}$$

2.7.2 Analytical model for the length of the ADU paths

Next, we concentrate on the ADU algorithm. We know that

$$P_{M,ADU}(l = 1) = \frac{2 \cdot e}{M} \tag{2.7}$$

and for the values of l bigger than 1, from the conditional probability definition

$$\begin{aligned}
 P_{M,ADU}(l = x, x > 1) &= P_{M,ADU}(x > 1) \cdot P_{M,ADU}(l = x|x > 1) = \\
 &= (1 - P_{M,ADU}(l = 1)) \cdot P_{M,ADU}(l = x|x > 1)
 \end{aligned} \tag{2.8}$$

However, we are only going to use the previous equation to calculate the probabilities when the path length is greater than 1, therefore

$$P_{M,ADU}(l = x, x > 1) = P_{M,ADU}(l = x) \tag{2.9}$$

hence

$$P_{M,ADU}(l = x) = (1 - P_{M,ADU}(l = 1)) \cdot P_{M,ADU}(l = x|x > 1) \tag{2.10}$$

At this point, we need to calculate $P_{M,ADU}(l = x|x > 1)$. This problem can be reduced to three different sub-problems regarding to the value of the first random number u_1 . If it is lower than or equal to L ($u_1 \leq L$) the algorithm works like AD. If it is greater than T ($u_1 > T$) the algorithm works like AU. Finally, if it drops between L and T ($L < u_1 \leq T$), the operation mode (AD or AU) is chosen randomly (RAND). Knowing that $P(AD)$, $P(RAND)$ and $P(AU)$ are the probabilities that the first random number drops within the AD, RAND or AU operation modes respectively, and that in a general scenario it is always true that

$$P((A|B)|C) = P(A|B, C) \tag{2.11}$$

then, applying the law of total probability to calculate $P_{M,ADU}(l = x|x > 1)$

$$\begin{aligned}
 P_{M,ADU}(l = x|x > 1) &= P_{M,ADU}(l = x|x > 1, AD) \cdot P(AD) + \\
 &+ P_{M,ADU}(l = x|x > 1, RAND) \cdot P(RAND) + \\
 &+ P_{M,ADU}(l = x|x > 1, AU) \cdot P(AU)
 \end{aligned} \tag{2.12}$$

2.7. Evaluation of the ADU paths

and merging equations 2.10 and 2.12

$$\begin{aligned}
 P_{M,ADU}(l = x) = & (1 - P_{M,ADU}(l = 1)) \cdot (P_{M,ADU}(l = x|x > 1, AD) \cdot P(AD) + \\
 & + P_{M,ADU}(l = x|x > 1, RAND) \cdot P(RAND) + \\
 & + P_{M,ADU}(l = x|x > 1, AU) \cdot P(AU))
 \end{aligned} \tag{2.13}$$

The next step consists of calculating the probabilities $P_{M,ADU}(l = x|x > 1, AD)$, $P_{M,ADU}(l = x|x > 1, RAND)$ and $P_{M,ADU}(l = x|x > 1, AU)$. The scenarios to calculate $P_{M,ADU}(l = x|x > 1, AD)$ and $P_{M,ADU}(l = x|x > 1, AU)$ can be interpreted as the AD algorithm with $M = L$, simplifying the study without loss of generality. Therefore

$$\begin{aligned}
 P_{M,ADU}(l = x|x > 1, AD) &= P_{M,ADU}(l = x|x > 1, AU) = \\
 &= P_{L,AD}(l = x|x > 1) = \frac{P_{L,AD}(l = x, x > 1)}{P_{L,AD}(x > 1)} = \\
 &= \frac{P_{L,AD}(l = x)}{1 - \frac{e}{L}}
 \end{aligned} \tag{2.14}$$

The scenario to calculate $P_{M,ADU}(l = x|x > 1, RAND)$ can be also simplified, without loss of generality, as if the AD algorithm was always selected with $M = T$, but conditioned to that the first selected random number (u_1) is between L and T . Therefore

$$P_{M,ADU}(l = x|x > 1, RAND) = P_{T,AD}(l = x|L < u_1 \leq T) \tag{2.15}$$

Then, applying the distributive property of intersection over union to the right hand side of the previous equation

$$P_{T,AD}(l = x|L < u_1 \leq T) = \sum_{i=L+1}^T \frac{P_{T,AD}(l = x, u_1 = i)}{P(L < u_1 \leq T)} \tag{2.16}$$

and multiplying numerator and denominator by $P(u_1 = i)$

$$P_{T,AD}(l = x|L < u_1 \leq T) = \sum_{i=L+1}^T P_{T,AD}(l = x|u_1 = i) \cdot P(u_1 = i|L < u_1 \leq T) \tag{2.17}$$

Finally, applying the property of equation 2.5

$$\begin{aligned}
 P_{T,AD}(l = x|L < u_1 \leq T) &= \sum_{i=L+1}^T P_{i-1,AD}(l = x - 1) \cdot P(u_1 = i|L < u_1 \leq T) = \\
 &= \frac{1}{T - L} \sum_{i=L+1}^T P_{i-1,AD}(l = x - 1)
 \end{aligned} \tag{2.18}$$

where the probability $P_{i-1,AD}(l = x - 1)$ is calculated using equation 2.6.

	ADU		Crowds
$\bar{l}$	M	e	p_f
2	100	21	0.5
3	100	8	0.6667
4	100	3	0.75
5	150	2	0.80
6	350	2	0.8333

Table 2.1: Values of parameters for specific $\bar{l}$.

2.7.3 Characteristics of the ADU paths

The probability mass function deduced in equations 2.7 and 2.13 can be used to calculate the associated statistical parameters. For example, to calculate the mean value

$$E(l) = \bar{l} = \sum_{i=1}^M i \cdot P_{M,ADU}(l = i) \quad (2.19)$$

Table 2.1 presents the appropriate values for the parameters M and e in order to achieve representative values for $\bar{l}$. The table represents low values (between 2 and 6) because, as we previously said, the objective is to achieve short multi-hop paths, which implies that the cost associated with the anonymous communication (bandwidth consumption and delay) may be reduced, and this type of application can be optimally implemented in overlay scenarios.

The previous table also represents the appropriate value of p_f to achieve the same values of $\bar{l}$ in Crowds. It is known that the mean length of the multi-hop paths created using the Crowds mechanism follows the geometrical expression

$$\bar{l} = \frac{1}{1 - p_f} \quad (2.20)$$

We compare our scheme with Crowds because it is also based on random walk procedure.

Table 2.2 presents the variance of the length of the paths for ADU and Crowds. The variance of the ADU paths is calculated using

$$V(l) = E(l^2) - E(l)^2 \quad (2.21)$$

on the other hand, for Crowds it is known that

$$V(l) = \frac{p_f}{(1 - p_f)^2} \quad (2.22)$$

It is observed that the variance in ADU is always significantly smaller than in Crowds. This behavior enables to interpret the ADU algorithm like a quasi-deterministic TTL implementation. Therefore, the mechanism achieves the target goal.

$\bar{l}$	ADU	Crowds
2	1.3337	2
3	2.3559	6
4	3.4135	12
5	4.5062	20
6	5.5821	30

Table 2.2: Variance of the length of the paths.

2.8 Analysis of anonymity

In 2002, Díaz et al. [Diaz et al., 2002] proposed the use of the entropy as a measure of the anonymity level, calculating it in unlimited multiple-hop paths. In [Lu et al., 2005], the authors computed the entropy of a system with fixed length multiple-hop paths. Based on these previous works, in this paper we introduce an analytical model to calculate entropy when the multi-hop paths are built using our mechanism.

It is assumed that there are two types of nodes: corrupt and honest ones. The system consists of a total number of N nodes, where C of them are corrupt and the rest ($N - C$) honest. Corrupt nodes collaborate among them trying to find out which is the initiator of the messages, although they never send any information message.

Based on the information retrieved from the corrupt nodes, the attacker assigns a probability (p_i) of being the initiator of a particular message for each node. Let X be a discrete random variable with probability mass function $p_i = P(X = i)$. The entropy of the system ($H(X)$) can be calculated as

$$H(X) = - \sum_{i=1}^N p_i \log_2(p_i) \quad (2.23)$$

The degree of anonymity (d) of the system is defined as

$$d = \frac{H(X)}{H_{Max}} \quad (2.24)$$

where H_{Max} is the maximum entropy of the system, which is satisfied when all honest nodes have the same probability of being the initiator of the message.

If a message goes only through honest nodes the degree of anonymity will be

$$d|_{\text{honest nodes}} \triangleq d_h = 1 \quad (2.25)$$

If we assume that the message goes through at least one corrupt node with probability p_c , and it crosses only through honest nodes with probability $p_h = 1 - p_c$, the mean degree of anonymity of the system is

$$\bar{d} = p_c \cdot d|_{\text{corrupt nodes}} + p_h \cdot d_h = p_c \cdot d_c + p_h \quad (2.26)$$

Let H_i denote the event that the first corrupt node occupies the i th position along the path, and let us also define $H_{1+} = H_1 \vee H_2 \vee \dots$ which means that there is a corrupt node

on the path. Let I denote the event that the immediate predecessor of the first corrupted node on the path is the path initiator¹. Corrupt nodes hope to determine $P(I|H_{1+})$, i.e. given that a corrupt node is on the path, what is the probability that the path initiator is the first corrupt node's immediate predecessor?

In order to calculate the mean degree of anonymity of the system, it is only necessary to calculate the p_i associated with every node in the network. It is clear that $p_i = 0$ for the corrupted nodes. The probability assigned to the predecessor node of the first corrupted node is $P(I|H_{1+})$. Finally, the rest of honest nodes $(N - C - 1)$ share uniformly the rest of the probability of being the initiator of the message $(1 - P(I|H_{1+}))$. Therefore:

$$p_i = \begin{cases} 0 & i \in C \\ P(I|H_{1+}) & \text{predecessor} \\ \frac{1-P(I|H_{1+})}{N-C-1} & \text{rest} \end{cases} \quad (2.27)$$

Thus, it is enough to calculate $P(I|H_{1+})$. Since $I \rightarrow H_{1+}$ the value of this probability can be simplified as

$$P(I|H_{1+}) = \frac{P(I, H_{1+})}{P(H_{1+})} = \frac{P(I)}{P(H_{1+})} \quad (2.28)$$

Therefore, to calculate $P(I|H_{1+})$ it is necessary to previously know $P(I)$ and $P(H_{1+})$. First, we are going to calculate $P(H_{1+})$. We assume that a path of length j is composed by $j - 1$ intermediate nodes and a destination node, and any node in the path can be corrupt. Taking into account that the probability of selecting a honest node is defined as

$$q = \frac{N - C}{N} \quad (2.29)$$

then,

$$\begin{aligned} P(H_1) &= (1 - q) \\ P(H_2) &= q(1 - q) \end{aligned}$$

...

$$P(H_j) = q^{j-1}(1 - q)$$

We can simplify the previous expressions as

$$P(H_i) = q^{i-1}(1 - q) \quad 1 \leq i \leq j \quad (2.30)$$

At this point, we can resolve for $P(H_{1+})$, taking into account than in a general scenario the possible lengths of the paths are between 1 and M (i.e. $1 \leq j \leq M$)

$$P(H_{1+}) = \sum_{j=1}^M \left(P(l = j) \cdot \sum_{i=1}^j P(H_i) \right) = \sum_{j=1}^M \left(P(l = j) \cdot \sum_{i=1}^j q^{i-1}(1 - q) \right) \quad (2.31)$$

¹Note that, when the first corrupt node occupies the first position, its predecessor is surely the initiator ($H_1 \rightarrow I$), but the opposite does not hold.

2.8. Analysis of anonymity

where $P(l = j)$ has been deduced in previous section.

In order to simplify the following operations we express $P(H_{1+})$ as

$$P(H_{1+}) \triangleq \frac{1-q}{T} \quad (2.32)$$

and inspecting equation (2.31), the value of T must be

$$T = \frac{1}{\sum_{j=1}^M \left(P(l=j) \cdot \sum_{i=1}^j q^{i-1} \right)} \quad (2.33)$$

The next step consists of calculating $P(I)$. It can be derived as

$$P(I) = P(H_1)P(I|H_1) + P(H_{2+})P(I|H_{2+}) \quad (2.34)$$

In the previous equation we know that

$$P(H_1) = 1 - q \quad (2.35)$$

$$P(I|H_1) = 1 \quad (2.36)$$

$$P(H_{2+}) = P(H_{1+}) - P(H_1) \quad (2.37)$$

On the other hand, given than a node never selects itself to be the next node ($P(I|H_2) = 0$) and if the first corrupted node on the path occupies the third or higher position, then it is immediately preceded on the path by any honest peer (including the initiator) with equal likelihood, then the probability $P(I|H_{2+})$ can be calculated as

$$P(I|H_{2+}) = P(I|H_2) + P(I|H_{3+}) = P(I|H_{3+}) = \frac{1}{N-C} \quad (2.38)$$

Then, $P(I)$ can be expressed as

$$P(I) = (1-q) + [P(H_{1+}) - P(H_1)] \frac{1}{N-C} = \frac{(1-q)(1+T(N-C-1))}{T(N-C)} \quad (2.39)$$

Therefore,

$$P(I|H_{1+}) = \frac{P(I)}{P(H_{1+})} = \frac{1+T(N-C-1)}{N-C} \quad (2.40)$$

and finally

$$p_i = \begin{cases} O & i \in C \\ \frac{1+T(N-C-1)}{N-C} & \text{predecessor} \\ \frac{1-T}{N-C} & \text{rest} \end{cases} \quad (2.41)$$

Now, we can compute the entropy $H(X)$ of the system as follows

$$H(X) = - \sum_{i=1}^N p_i \log_2(p_i) = \frac{(N-C-1)(1-T)}{N-C} \log_2 \left(\frac{N-C}{1-T} \right) + \frac{1+(N-C-1)T}{N-C} \log_2 \left(\frac{N-C}{1+(N-C-1)T} \right) \quad (2.42)$$

The maximum entropy (H_{Max}), considering that the size of the anonymity set is $N-C$ and that all these nodes have the same probability of being the initiator is equal to

$$H_{Max} = - \sum_{i=1}^{N-C} \frac{1}{N-C} \log_2 \left(\frac{1}{N-C} \right) = \log_2(N-C) \quad (2.43)$$

Therefore, the degree of anonymity of the system will be

$$d_c = \frac{H(X)}{H_{Max}} = \frac{(N-C-1)(1-T)}{(N-C) \log_2(N-C)} \log_2 \left(\frac{N-C}{1-T} \right) + \frac{1+(N-C-1)T}{(N-C) \log_2(N-C)} \log_2 \left(\frac{N-C}{1+(N-C-1)T} \right) \quad (2.44)$$

where

$$T = \frac{1-q}{P(H_{1+})} \quad (2.45)$$

and $P(H_{1+})$ is obtained from equation (2.31).

Finally, the mean degree of anonymity of the system is

$$\bar{d} = p_c \cdot d_c + p_h = P(H_{1+}) \cdot d_c + (1 - P(H_{1+})) = 1 + P(H_{1+})(d_c - 1) \quad (2.46)$$

The mean degree of anonymity is a global network parameter and measures the quality of the anonymous design. However, as it was previously introduced, corrupt nodes hope to determine $P(I|H_{1+})$. This probability has been properly calculated in equation 2.40.

2.9 Model validation

All the results presented in this section correspond to a network with $N = 100$ nodes and $C = 10$ corrupt nodes. This scenario has a 10% ratio of corrupt nodes, which can be considered as a very unsafe situation. First we validate our mathematical model derived by using simulation. The simulator has been written in C language and it works as follows: Each node, after selecting a random destination, builds a limited multi-hop path towards it (this is 1 iteration). After n iterations (in our case n is set to 10,000 since for this value the accuracy of the simulation results is within the 0.1% with respect to the stable value) we measure using the following methodology (for more details see reference [Borissov, 2005]):

2.9. Model validation

Figure 2.7: Validation of the model.

Let us introduce two random variables: X , modeling the initiators ($X = 1, \dots, N - C$), and Y , modeling the predecessor observed by the first corrupt node in the path ($Y = 1, \dots, N - C, \emptyset$; where $\emptyset$ represents that there is not an attacker in the path). The probability distribution computed in equation (2.23) is really the distribution of X conditioned on a particular observation y . Therefore, the entropy metric evaluates $H(X|Y = y)$ for some y , and it can be calculated as

$$H(X|Y = y) = - \sum_x q_{x,y} \log_2(q_{x,y}) \quad (2.47)$$

The distribution $q_{x,y}$ is an estimate of the true distribution p_i . The way to calculate it is as follows: We keep a counter $C_{x,y}$ for each pair of x and y . We pick a random value x for X and simulate a request. The request is forwarded according to the algorithm until it either is sent to the destination or is intercepted by a corrupt node. In the former case, we write down the observation $y = \emptyset$, while in the latter case we set y to the identity of the predecessor node. In both cases, we increment $C_{x,y}$. Next, we can compute $q_{x,y}$ as follows

$$q_{x,y} = \frac{C_{x,y}}{K_y} \quad (2.48)$$

where $K_y = \sum_x C_{x,y}$.

On the other hand, the average entropy of the system, taking into account that a message can also go through honest nodes, can be expressed by

$$\begin{aligned} \bar{H} &= \sum_{y \geq 1} P(Y = y) H(X|Y = y) + P(Y = \emptyset) H(X|Y = \emptyset) = \\ &= \sum_y P(Y = y) H(X|Y = y) \end{aligned} \quad (2.49)$$

With these definitions we can redefine expression (2.26) as,

 Figure 2.8: Mean degree of anonymity as a function of $\bar{l}$.

$$\bar{d} = \frac{1}{H_{Max}} \sum_y P(Y = y) H(X|Y = y) \quad (2.50)$$

Figure 2.7 compares the simulation results with the analytical model (derived in equation 2.46) for the mean degree of anonymity $\bar{d}$, as a function of the mean length of the path ($\bar{l}$). The values of the parameters in this scenario are: $M = 100$ and $e = 3$. The parameters *LOW_BORDER* and *TOP_BORDER* are chosen randomly by every initiator. It can be observed that the mathematical model derived in Section 6 accurately matches the simulation results. The analytical model has been validated for other different cases (under different value ranges for the parameters) and the results achieved are equivalent to the previous scenario. That is, the model proposed matches all the simulation results. Therefore equation 2.46 is validated and from now on, it can be used to measure the anonymity level in any network that uses our methodology to limit the length of the multi-hop paths.

Next, in Figure 2.8 we compare the mean degree of anonymity ($\bar{d}$) with and without multi-hop path length limit. For the former case, the previously validated mathematical model is used, and for the latter, the results related to the Crowds network presented in [13] are used. It can be observed that the degree of anonymity achieved by the ADU algorithm perfectly matches the degree of anonymity achieved by Crowds. Therefore, this mechanism is a recommended methodology to achieve a good trade-off between cost/benefit associated with the anonymity in overlay networks.

2.10 Performance over traffic analysis and predecessor attacks

In this section, we are going to prove that the ADU algorithm does not increase the vulnerability of Crowds over the *traffic analysis* and *predecessor* attacks. Let us begin our discussion with the formers, which are well described in [Raymond, 2001]. Using these attacks, adversaries can monitor and/or compromise certain parts of the systems to be able

2.10. Performance over traffic analysis and predecessor attacks

to match a message initiator with the destination. These attacks can be classified taking into account different properties:

- Internal-External: An internal attacker controls one or several nodes that are part of the system, whereas an external attacker can only compromise communication channels (e.g., it can eavesdrop or tamper with messages).
- Passive-Active: A passive attacker can only listen to messages, whereas an active is able to add, remove and modify messages.

The most representative traffic analysis attacks are: *brute force*, *timing*, *contextual* and *message tagging* attacks [Raymond, 2001]. The brute force, timing and contextual attacks are passive and external, whereas the message tagging attack is active and internal. Unfortunately, Crowds and ADU do not protect against powerful passive external attacks. For example, in the brute force attack it is assumed that the attacker controls all the communications channels in the network and it is able to follow every message from the initiators to the corresponding destinations. Therefore, the vulnerability of ADU over this attack is equivalent to the vulnerability of Crowds, because in both cases the brute force attack can be effective. Timing and contextual attacks are proposed for other more complex anonymous networks in which the packets can be mixed by the intermediate nodes and they can follow different paths from every intermediate node. In this case, the number of possible paths that a specific message can follow grows potentially with the number of intermediate nodes and it can be difficult to follow it from its initiator to the destination.

On the other hand, an active internal attacker that has control of the first and last nodes in a message route can carry out the message tagging attack. The attacker can tag messages at the first node in the message route in such a way that the exit node in the route can spot them. Since the entry node knows the initiator and the exit node the destination, the system is broken. In this attack, the information attached in each hop (i.e. the interval) does not cause the ADU algorithm to be more vulnerable than Crowds. However, there exists another internal attack which could obtain useful information from the interval value in the ADU algorithm: the *predecessor attack* [Wright et al., 2004]. This attack was initially described as an attack against Crowds. However it can also be extended for other anonymous networks like DC-Net, Onion Routing and Mix-Net. It depends on the assumption that an initiator might choose to remain in contact with a destination for an extended period of time. In that case, the session between the initiator and the destination is subject to a number of resets due to intermediate node disconnections. When it happens, a new path is constructed between initiator and destination. The attack rests on the fact that, given that nodes are chosen with replacement, the attackers log the initiator more than any other node. This is because when an attacker receives the message from the initiator (I), the attacker logs I as the predecessor. In the remaining cases, due to the uniformity assumption, all participants (included I) are logged with equal probability. Thus, the expected number of times that I is logged by the attackers is greater than the expected number of times that any other node is logged. By the law of large numbers, as $time \rightarrow \infty$, I will appear more often than any other node.

In constructing the attack, several simplifying assumptions about protocol operation are made:

(a) *Crowds* and ADU with an average path length of 10 hops.

(b) ADU with an average path length of 4 hops.

Figure 2.9: Simulation of the predecessor attack.

- The subset of nodes that forward an initiator's messages are chosen uniformly at random.
- There is a recurrent connection (session) between some participant that initiates the sending of a message and the destination of that message.
- There is information available to the attacker in the packets that uniquely identifies this recurrent connection (e.g., the session identification).

In [Wright et al., 2004] the authors provide upper bounds on the time required for the degradation of an initiator's anonymity in *Crowds* when facing such an attack. The authors bind the time in terms of rounds (e.g., path reformation in *Crowds*), and the results are the following:

- The number of rounds to obtain a successful attack with high probability is $O(\frac{N}{C} \log N)$.
- The expectation of the number of rounds is $O(\frac{N}{C})$.

where N is the total number of nodes and C is the number of attackers. However, these are only analytic upper bounds on the number of rounds required. In [Wright et al., 2008] the same authors demonstrate by means of simulation that, although the bounds provide a useful analytic tool to compare protocols, they are loose. That is, the attackers can be successful in significantly fewer rounds. For example, attacks on *Crowds* with 1,000 nodes and 100 attackers, succeed in a time one-fifth of the rounds guaranteed by the upper bounds.

Using the simulation methodology presented in [Wright et al., 2008], we evaluate the vulnerability of the ADU algorithm over the predecessor attack. In order to compare *Crowds* and ADU the simulation scenario is equivalent to one of the scenarios evaluated for the *Crowds* mechanism in [Wright et al., 2008], that is, there are $N = 1,000$ total

Figure 2.10: Probability function for the different values of the interval.

nodes with $C = 200$ corrupted nodes, and the average path length is 10. Figure 2.9(a) shows the average number of simulations, out of 10,000, in which the attacker correctly determines the initiator for a given number of rounds (that is, the probability of attacker success) for Crowds and ADU. It can be seen that in both cases the probability of attacker success is 1 from 40 rounds. However, from 1 to 15 rounds, approximately, the probability of attacker success is higher for Crowds. Although the average path length is equivalent for both algorithms, the associated variance is higher in Crowds (see Table 2.2). Therefore, the possibility of having longer paths is also higher. The longer the path, the higher the probability that the initiator node appears in it, and as a consequence the higher is the probability of success of the predecessor attack, too. However, when the number of rounds is high enough, the probability that the initiator node appears in different paths of the same session grows enough for both algorithms, and the probability of attacker success is equivalent for both. In Figure 2.9(b) we present the simulation results for the predecessor attack in a new scenario where the average path length is 4. We can check that in this case the probability of attacker success is 1 from 60 rounds. If we compare it with the results presented in Figure 2.9(a), we can extract that the effectiveness of the predecessor attack decreases when the path length decreases. Those are good news, since the objective of the ADU algorithm is to limit the length of the random paths.

Nevertheless, in the ADU algorithm, attackers can extract useful information from the interval attached in each hop when they perform the predecessor attack. Figure 2.10 represents the probability function for the different values of the interval that a corrupt node would see in the hypothetical situation in which it was always on the first, second or third position of a random path. The value of ADU parameters in this scenario are: $M = 100$ and $e = 3$, and they correspond with paths with 4 hops in mean, as it is represented in Table 2.1.

As can be seen, the probability function associated to an attacker node located on the first position of a random path corresponds with a uniform distribution between $e + 1$ and $M - e$. For the rest of positions (second, third, etc.) extreme values are more common than central values. In our algorithm, every attacker in the path could associate the identity of the predecessor node with the interval values attached in the hops. For the nodes different

from the initiator node, corrupt nodes will more frequently see extreme values than central values because they will never be the initiators of the path.

On the other hand, corrupt nodes will always see the initiator node when they are located on the first position of the path, but also when they are located on other positions (the initiator node can appear in other positions in the path). However, most of the views of the initiator node by corrupt nodes will be when they are located on the first position of the path. This is because each time a corrupt node is in the first position of the path it sees the initiator node as its predecessor. But, when it is located on other positions, the predecessor node can be any other node in the network with equal probability. Therefore, the probability function for the interval values associated to the initiator node will be equivalent to a uniform distribution with value $1/(M - 2e)$, corresponding to the function deduced by an attacker node always located on the first position of the path.

We are going to evaluate this new attack against the ADU algorithm. The simulation methodology for the new attack is equivalent to that presented for the predecessor attack in [Wright et al., 2008], but in this case the attacker associates the interval values attached in the hops with the identity of the predecessor node. When the path related to every round is built, the attacker calculates the degree of similarity between the probability distribution of the interval values linked to each node, and a uniform distribution with value $1/(M - 2e)$.

Several indices have been suggested in the statistical literature to reflect the degree of similarity between any two probability distributions. Such indices have been called measures of probabilistic distance between two distributions. These indices have the common property of increasing as the two distributions involved “move apart”. An index with this property may also be called a measure of divergence of one distribution from another. Two of the most commonly used probabilistic distance measures are the Bhattacharyya [Kailath, 1967] and Mahalanobis [Mahalanobis, 1936] distances. Several recent works [Rahman et al., 2005][Comaniciu et al., 1999] have demonstrated that the precision of the Bhattacharyya distance is always superior to the precision of the Mahalanobis distance, and that the computational cost of the Bhattacharyya distance is lower than the computational cost of the Mahalanobis distance. On the other hand, the Mahalanobis distance is optimized to measure the probabilistic distance between two Gaussian (normal) distributions, but we want to measure the distance between two uniform distributions. Therefore, in this work we are going to use the Bhattacharyya distance.

For a domain X with distributions $p(X)$ and $q(X)$, the Bhattacharyya distance (D_B) is defined as:

$$D_B(p(X), q(X)) = -\ln(BC(p(X), q(X))) \quad (2.51)$$

where

$$BC(p(X), q(X)) = \sum_{x \in X} \sqrt{p(x)q(x)} \quad (2.52)$$

is the Bhattacharyya coefficient. It is easy to verify that

- $0 \leq BC(p(X), q(X)) \leq 1$.

Figure 2.11: Simulation of the new attack against ADU.

- $BC(p(X), q(X)) = 1$ if and only if $p(X) = q(X)$.
- $BC(p(X), q(X)) = 0$ if and only if $p(X)$ is orthogonal to $q(X)$.

On the other hand, clearly

- $0 \leq D_B(p(X), q(X)) < \infty$.
- $D_B(p(X), q(X)) = 0$ if and only if $p(X) = q(X)$.
- $D_B(p(X), q(X))$ increases when the degree of dissimilarity between the two distributions also increases.

We are going to present the mean value of the Bhattacharyya distance calculated for the initiator node over 10,000 iterations (in the same way as it was calculated for the predecessor attack). Figure 2.11 presents this value for paths with an average length of 4 hops. Two important conclusions can be extracted from this figure. First, it was previously presented that in this scenario, the predecessor attack can guess the initiator node with only 60 rounds. For the attack based on the interval value attached on the hops, the Bhattacharyya distance associated to 60 rounds is approximately 0.7 (the Bhattacharyya coefficient is 0.5). That means that the attacker cannot assure the identity of the initiator node. Anyway, the number of rounds required to obtain it using the interval value is very high, because as we can check in the figure, with 500 rounds the Bhattacharyya distance is not 0 yet and the attacker will need a higher number of rounds to ensure the identity of the initiator.

The usual predecessor attack and the attack based on the interval value attached on the hops have been evaluated in other scenarios (with different average length for the paths), and the results are equivalent to the presented one. That is, in all cases the predecessor attack can obtain the identity of the initiator node without the help of the interval value attached on the hops. In addition, we have checked that in all the scenarios the vulnerability of the ADU algorithm over the predecessor attack is equivalent to the vulnerability of Crowds. Therefore, we have proved that the ADU algorithm does not increase the vulnerability of Crowds over the predecessor attack.

2.11 Conclusions

Unstructured P2P networks were the first generation of P2P networks. These networks do not impose any structure on the overlay topology. That is, they present a flat topological architecture, where all the nodes belong to the same level. This characteristic makes this kind of networks specially suitable for the implementation of source-rewriting mechanisms, traditionally used in IP networks to provide anonymity. One of these mechanisms, called Crowds, is based on the building of a random walk path between the initiator and the destination. One of the drawbacks of this methodology is that the cost associated with the communication increases dramatically as the number of hops also increases. Therefore, limiting the length of the paths is a key aspect in this scenario.

Unfortunately, the common TTL methodology cannot be used here since corrupt nodes can employ this field to extract some information about the initiator identity. In this work, an effective mechanism to reduce the variance associated with the length of the random walk paths is proposed. In addition, an intuitive but rigorous analytical model to calculate the degree of anonymity in networks using the proposed mechanism is also derived. This model has been validated by means of simulation.

This study reveals that the variance in ADU is always smaller than in Crowds. In addition, the degree of anonymity achieved by ADU is equivalent to Crowds. It has also been proved that the ADU algorithm does not increment the vulnerability of Crowds over the traffic analysis and predecessor attacks. Thus, this mechanism is a recommended methodology to achieve a good trade-off between cost/benefit associated with the anonymity in overlay networks.

Providing VOD streaming using superpeer-based P2P networks

This chapter introduces the main characteristics of superpeer-based P2P networks and it describes FastTrack as a well-known example of this kind of networks. After that, the fundamental components of any P2P VOD system are presented: the overlay structure, the content forwarding policy (the way the multimedia content is stored and transmitted to each participating peer through the overlay), the mechanisms to find a partner with the necessary segments, and the collaboration with partners for content delivery. In this chapter we extend the work presented in [Nafaa et al., 2008], which was concentrated on the analysis of a VOD service distribution architecture in a broadband operator context. It is implemented in a superpeer-based P2P network, where the content is stored in the Set-Top Boxes (STBs), which act as peers, and the sessions are provided by means of multi-source streaming [Nafaa et al., 2009][Lamoriniere et al., 2009]. First of all, the system is evaluated in a broadband operator scenario, and it is deduced that the existing mechanisms are not enough to fully take advantage of the network resources. Therefore, it is presented a new mechanism which tries to dynamically re-allocate the necessary streams in order to find available resources for a new VOD session which cannot be established. The second part of the chapter extends the previous VOD service distribution architecture to the Internet context. It is firstly presented a new dispatching algorithm: the decision on what content is dispatched at which STB is determined by a new Non-Linear Programming (NLP) formulation. After that, two new mechanisms to forward the VOD requests between different ISP networks are presented. It is demonstrated that the global performance of the system is improved with the new proposed mechanisms.

3.1 Introduction

The second generation of P2P applications used superpeer-based networks. In this case, the network architecture present a hierarchical topological architecture. Clients are connected to a superpeer, and it is connected to a set of neighbor superpeers (see Figure 1.3). This kind of overlay networks tries to improve some of the characteristics of the

traditional flat overlay networks such as delay and routing performance. Every superpeer keeps and index over its clients' data. This index must hold sufficient information to answer all queries. When a client wishes to submit a query to the network, it will send the query to its superpeer. If the superpeer finds any results, it will return one response message. Otherwise, it will submit the query to its neighbors as if it were its own query, and forward any response messages it receives back to the client.

These systems introduce a balance between the inherent efficiency of centralized search, and the autonomy, load balancing and robustness to attacks provided by distributed search. Furthermore, they take advantage of the heterogeneity of capabilities (e.g., bandwidth, processing power) across peers [Yang and Garcia-Molina, 2003].

The major drawback of superpeer-based P2P systems is that a superpeer becomes a single point of failure for its clients, and a potential bottleneck. When the superpeer fails or simply leaves, all its clients become temporarily disconnected until they can find a new superpeer to connect to.

3.2 FastTrack

FastTrack [Liang et al., 2006] uses a superpeer architecture in which high-capacity peers are supernodes (SN) and low-capacity peers are ordinary nodes (ON). Each ON has a parent SN. When an ON launches the FastTrack application, the ON chooses a parent SN, maintains a semi-permanent TCP connection with its parent SN, and uploads to this SN the metadata for the files it is sharing.

As with most other P2P file sharing systems, FastTrack maintains a file index that maps file identifiers to the IP addresses. This file index is distributed across the SNs. In particular, each SN maintains a local index for all of its children ONs. But a SN is not a dedicated server; instead, it is typically a peer belonging to an individual user.

For each file an ON is sharing, the metadata that the ON uploads to its parent SN includes: the *file name*, the *file size*, the *ContentHash*, and the *file descriptors* (for example, artist name, album name, and text entered by users). The file descriptors are used for keyword matches during enquiring. The ContentHash plays an important role in the FastTrack architecture. FastTrack hashes every file to a hash signature, which becomes the ContentHash of the file. The ContentHash is the only identifier used to identify a file in an HTTP download request. If a download from a specific peer fails, the ContentHash enables the FastTrack client to automatically search for the specific file, without issuing a new keyword query.

When a user wants to locate files, the user's ON sends a query with keywords over the TCP connection to its parent SN. For each match in its database, the SN returns the IP address, server port number, and metadata corresponding to the match. Each SN also maintains long-lived TCP connections with other SNs, creating an overlay network among the SNs. When a SN receives a query, it may forward the query to one or more of the SNs to which it is connected. A given query will in general visit a small subset of the SNs, and hence will obtain the metadata information of a small subset of all the ONs.

In an overlay of about 3 million nodes, the number of SNs is in the range of 25,000 to 40,000. Each ON maintains a connection with one SN. The SN provides its client ON

3.3. P2P VOD systems

Figure 3.1: Search process in FastTrack.

with a list of other SNs, which the ON caches. After an ON issues a query to the SN and receives its responses, the ON disconnects from the current SN and reconnects to a new SN from its list. During reconnections to new SNs, it also receives a new SN list that it merges with the existing list [Buford et al., 2008].

An SN maintains about 40 to 50 connections with other SNs. An SN on a residential broadband connection maintains connections to about 50 to 80 ONs. SNs with greater network capacity - for example, a university campus network - maintain connections with about 100 to 160 ONs at any given time. The average lifetime of an SN-SN connection is 34 minutes and an SN-ON is 11 minutes. Approximately 33 % of the connections last less than 30 seconds.

3.3 P2P VOD systems

Video-on-Demand (VOD) has been identified as the key feature to attract consumers to IPTV (Internet Protocol Television) service [Muñoz-Gea et al., 2010b, Muñoz-Gea et al., 2010c]. VOD service allows users to watch any point of video at any time. Compared with live streaming, VOD offers more flexibility and convenience to users and truly reaches the goal of *watch whatever you want whenever you want*. It is more challenging to design a P2P VOD service than a P2P live streaming system because even though a large number of users may be watching the same video, they are asynchronous to one another (different users may watch different portions of the same video at any given moment).

A fundamental component of any P2P VOD system is its overlay network structure, related to the organization of participating peers into an overlay structure. Based on this component, P2P VOD systems can be broadly classified into *tree-based* and *mesh-based* categories. Tree-based systems have well organized overlay structures and they typically distribute video by actively pushing data from a peer to its children peers. One major drawback of tree-based streaming systems is their vulnerability to peer departure, which will temporarily disrupt video delivery to all peers in the subtree rooted at the departed peer. On the other hand, in a mesh-based P2P streaming system, peers are not confined to a static topology. Instead, peer relationships are established/terminated based on the

content and bandwidth availability on peers. This kind of system tries to achieve fast file downloading by swarming. With this method, a file is divided into small size segments, and the server disperses them to different users. The users download from their neighbors the segments that they currently do not have. Since multiple neighbors are maintained at any given moment, mesh-based video streaming systems are highly robust to peer failures. Therefore, this chapter is going to concentrate on mesh-based approaches.

An important component of any mesh-based P2P VOD system is the content forwarding policy, that is, the way the multimedia content is stored and transmitted to each participating peer through the overlay. There are two main categories for this component: the *buffer-forwarding* and the *storage-forwarding* approaches. In the buffer-forwarding approach each client caches a limited number of segments around its current “play offset”. In the storage-forwarding approach each peer stores a great number of segments at its local storage (such as hard disk).

Another key aspect in mesh-based P2P VOD systems is the design of mechanisms to find a partner with necessary segments. Due to scalability concerns these mechanisms usually use distributed searching structures, like the distributed hash table (DHT) and the skip-list. In the buffer-forwarding approaches, peers are sorted in the distributed searching structures by the play offset. On the other hand, in the storage-forwarding approaches, peers are sorted in the distributed searching structures by the segments they store. In both cases, when a client wants to play a segment, it first looks for the supplying peers of that segment and then sends requests to those peers for the service.

The collaboration with a partner for content delivery (*data scheduling*) is another very important and challenging problem in mesh-based P2P VOD systems. The scheduling scheme in mesh-based P2P VOD systems has two main components: how to send requests to adequate neighbors, and how to evaluate the requests received from many peers. For the buffer-forwarding approaches, many heuristics have been proposed to address this issue, such as round robin or smallest-delay. Some recently proposed schemes use network coding to improve the throughput. On the other hand, for the storage-forwarding approaches several modifications of the BitTorrent segment selection algorithm have been proposed. In addition, the use of rateless coding, such as LT or Raptor codes, is a new alternative to address this issue.

3.3.1 Overlay network structure for P2P VOD

As it was previously said, a key aspect of any P2P VOD system is the organization of participating peers into an overlay structure. Based on this component P2P VOD streaming systems can be broadly classified into two categories: *tree-based* and *mesh-based*.

3.3.1.1 Tree-based network structure

Tree-based P2P systems were originally designed to be used as multicast structures at the application layer without network layer support. However, in this kind of system the users are synchronous, and in a VOD service the users are asynchronous. Therefore, the use of a tree-based P2P system to provide VOD services is not direct. In the last years, several mechanisms have been proposed to solve this problem. For example, in [Guo et al., 2003]

3.3. P2P VOD systems

Figure 3.2: A snapshot of the patching scheme with $T = 10$ at time 40.

the authors designed a tree-based P2P VOD system inspired by the patching scheme [Hua et al., 1998]. Users that arrive within a predefined time interval T constitute a session, and together with the server they form a tree-based P2P system. The server starts streaming the entire video over the tree, and when a new client joins the session, it joins the tree and retrieves the stream from it. In addition, the new client must obtain a *patch*, the initial portion of the video that it has missed (from the start of the session to the time it joined the base tree). The patch is available at the server as well as other users who have already cached the patch. Figure 3.2 illustrates a snapshot of the solution above when a new user arrives at time 40. Each user is marked with its arrival time to the system. It shows two sessions, starting at time 20 and 31, respectively, with time interval T equal to 10. User E has to create a new session because its arrival time to the system (31) is out of the time interval T of the user A ($20 + 10 = 30$). One major drawback of tree-based streaming systems is their vulnerability to peer departure, which will temporarily disrupt video delivery to all peers in the subtree rooted at the departed peer.

3.3.1.2 Mesh-based network structure

Peers in mesh-based P2P systems are not confined to a static topology, they dynamically connect to a subset of random peers in the system. Figure 3.3 represents an overview of a mesh-based P2P VOD architecture. This kind of system tries to achieve fast file downloading by swarming. With this method, a file is divided into small size segments and the server disperses them to different users. The users download from their neighboring peers the segments that they currently do not have, and they have to be received before their playback time. To fully utilize users upload bandwidth and achieve the highest downloading throughput possible, the segments at different users have to be different from each other so that there is always something to exchange. This is the so-called *diversity requirement* in mesh-based P2P system. On the other hand, since multiple neighbors are maintained at any given moment, mesh-based systems are highly robust to peer failures. Therefore, this chapter is going to concentrate on mesh-based approaches. Within this kind of networks, several suppliers search and data scheduling mechanisms for both

Figure 3.3: An overview of P2P mesh-based architecture.

storage-forwarding and buffer-forwarding approaches are going to be introduced.

3.3.2 Forwarding approaches

The content forwarding policy is another important component of any P2P VOD system, that is, the way the multimedia content is stored and transmitted to each participating peer through the overlay. There are two main categories for this component: the *buffer-forwarding* and the *storage-forwarding* approaches. There also exist *hybrid-forwarding* approaches which integrate both the buffer-forwarding and the storage-forwarding.

3.3.2.1 Buffer-forwarding

In the buffer-forwarding approach, each client caches a limited number of segments around its current play offset, and it exchanges its cached segments with partners who have close play offset and can thus provide the expected data with high probability. The users need to search for such partners when joining or taking a VCR operation. Since the contents buffered in one peer are continuously changing, an additional structure is needed for the partner search. In this structure the participating peers organize themselves based on the playback progress. After locating the partners, the user collaborates with them to schedule the data transmission and exchange the content. Figure 3.4 depicts the general framework of a typical buffer-forwarding P2P VOD system. It is noticed that the users form two types of overlay networks: the *index overlay*, for suppliers search, and the *data overlay*, for media transmission. This approach has two main drawbacks: First, a peer only redistributes the video, that it is currently watching, to the peers that are watching approximately the same video scene, and it does not redistribute the content that it has stored in its storage. Second, the upload bandwidth of the peers that have not content in their buffers cannot be utilized.

3.3.2.2 Storage-forwarding

In the storage-forwarding approach each peer stores a great number of random video segments at its local storage (such as hard disk). This approach can be used when peers have a large storage capacity. For example, in VMesh [Yiu et al., 2007] each peer keeps a

3.3. P2P VOD systems

Figure 3.4: General framework of a typical buffer-forwarding P2P VOD system.

list of peers who have the previous and the next video segments. Following the list, it can quickly find the peers with the next requested segments. Furthermore, a peer also keeps a list of peers storing the same segment for load balancing purpose. If a node is loaded, it redirects some of its requests to other peers on its list. The main drawback of this kind of systems is the need to have a big storage capacity in order to achieve a good performance. In addition, the upload bandwidth of the peers has to be high enough to support several connections.

3.3.2.3 Hybrid-forwarding

A hybrid-forwarding P2P VOD architecture integrates both the buffer-forwarding and the storage-forwarding approaches. The peers watching the same video form an overlay with the buffer-forwarding approach, and each peer replicates one or multiple segments of the video it has watched before in its storage.

An interesting hybrid-forwarding P2P VOD architecture is proposed in [He et al., 2008]. In the example shown in Figure 3.5, peers 3, 7, 130, 118, 104 and 1 form video-1 buffer-forwarding overlay, and peers 134, 34, 115, 43, 22 and 1 form video-2 buffer-forwarding overlay. Some peers watching video 2, such as peer 115, can forward their stored segments to the peers watching video 1 (e.g. peer 7). The idle peers (e.g. peer 27, 12, and 75) are encouraged to contribute their stored segments to serve other peers. In the hybrid-forwarding architecture, the peers may have both buffer-forwarding and storage-forwarding outgoing links (e.g. peer 115). These two kinds of outgoing links participate in different video sessions and compete for the limited upload bandwidth. Furthermore, when the buffer-forwarding parents of peer 7 (e.g. peers 130 and 118) jump to other positions, peer 7 can still have a stream supply from its storage-forwarding parent (e.g. peer 115).

3.3.3 Suppliers search

A key aspect in mesh-based P2P VOD systems is the design of mechanisms to find partners with needed segments. It is very challenging to develop an efficient suppliers search

Figure 3.5: A hybrid-forwarding P2P VOD system.

mechanism among a large population of collaborative peers. Due to scalability concerns, the search mechanism should consist of a distributed structure with sub-linear search efficiency. Today, some distributed structures have been proposed with logarithmic search efficiency, for example, the DHT (distributed hash table) structure.

3.3.3.1 Searching in buffer-forwarding

In the buffer-forwarding approaches, peers are sorted in the distributed search structures by the play offset. Given the play offset as the sorting key, these structures support searching mechanisms in a distributed manner. However, the maintenance cost of these search structures is not trivial, since all peers need to be registered in the structure. Some buffer overlapping may occur among the different nodes, and removing the nodes whose buffer range is fully covered by other nodes does not reduce the total buffer coverage. The BAS (buffer-assisted search) structure [Chi et al., 2007] tries to maintain as few peers as possible for better search efficiency. In other words, it wants to minimize the search structure size without sacrificing the search effectiveness. This problem can be formulated as the Minimum Buffer Cover (MAC), which is a variation of the well-known NP-hard problem Minimum Set Cover.

However, there are other systems that use more novel management schemes. For example, in RINDY [Cheng et al., 2007], a video is divided into a series of timeslots which encapsulate the content of one second, and the buffer window can contain at most w timeslots. The distance between any two peers is calculated from their current playing positions. For instance, the distance from peer j to peer i is $d_{ji} = cur_j - cur_i$. If d_{ji} is negative, then the playing position of peer j is behind of that of peer i and peer j is called a *back-neighbor* of peer i ; otherwise peer j is called a *front-neighbor* of peer i . The neighbors are organized into a series of concentric, non-overlapping logical rings according to their relative distances (see Figure 3.6). The radius of the i -th ring is $w * 2^i$ (the i th ring covers the area between the two circles with radii $w * 2^{i-1}$ and $w * 2^i$). A peer's innermost ring, whose ring number is zero, is responsible for collecting some neighbors with close playing positions (this ring is called *gossip-ring*). Outer rings are mainly used to improve the speed of lookup operations (these rings are called *skip-rings*). For the *gossip-ring*, each peer keeps track of at most m neighbors, called *near-neighbors*;

Figure 3.6: Neighbor distribution over rings for peer P.

for each skip-ring, a peer tries to maintain at most k front far-neighbors and k back far-neighbors.

3.3.3.2 Searching in storage-forwarding

In the storage-forwarding approach, depending on the capacity of its local storage, each peer stores a number of segments randomly chosen from the N segments of the video. In this way, there are multiple copies of each video segment in the network. When a client wants to play a segment, it first looks for the supplying peers of that segment and then sends requests to those peers for the service. If there is no supplying peer, the requesting peer requests the media server for the target segment as the last resort. In this section the searching mechanisms of two representative storage-forwarding proposals are presented.

In VMesh [Yiu et al., 2007], each storage peer should register its stored segments in a DHT network in order to allow other peers to locate them. A new client searches for its segment of interest in the DHT network and starts playing the video when the requested data arrives. Then, it continues to request the next segment when the current one is nearly finished. In order to shorten the segment location latency, the peers also form an overlay mesh among themselves. This overlay mesh is represented in Figure 3.7. In this figure every circle represents a peer and the number inside represents the ID of the segment it holds. Each peer keeps a list of pointers (i.e., the IP addresses) pointing to some peers which store the next video segment and the previous video segment. By using the pointer list, clients could request for the locations of the next required segment. Besides, for the load balancing issue, each peer also keeps a list of pointers to some peers which are storing the same segment. If a node is loaded, it redirects some of its requests to other peers on its list.

3.3.4 Data scheduling

Data scheduling is an additional, important and challenging problem in mesh-based P2P VOD systems. The scheduling scheme in mesh-based P2P VOD systems has two main components: how to send requests to adequate neighbors, and how to evaluate the requests received from many peers.

Figure 3.7: VMesh structure.

3.3.4.1 Scheduling in buffer-forwarding

Many heuristics have been proposed to address the issue of buffer-forwarding approaches, such as round robin or smallest-delay. For example, in OCTOPUS [Liao and Jin, 2007] the system schedules packets based on partners' bandwidths. Each peer assigns an initial scheduling bandwidth to every partner ($W(x) = Sbps/N$), where $Sbps$ means the streaming rate of media file and N the number of partners and it is adjusted in data transferring according to the success ratio of scheduling. The system calculates the success ratio of scheduling for the past periods in function of the scheduling results from every partner. If success ratio is 1, it is obvious that the predefined scheduling bandwidth $W(x)$ is less than the actual bandwidth, and the scheduling bandwidth values can be enlarged to make full use of the bandwidth resource of partners. But if the success ratio is low, it is obvious that the scheduling bandwidth is too large and the service capacity of the partner is not so strong. The value of scheduling bandwidth may have to be decreased.

However, the previous schemes suffer from inefficient use of network resources in large and heterogeneous networks. Some recently proposed schemes use network coding (NC) to improve the system throughput. Wang & Liu [Wang and Liu, 2008] proposed directly applying the linear NC into VOD streaming. The original data segments are denoted as $c_1, c_2, \dots, c_w$, where w is called *coding window (CW)*. A random linear coding on $\{c_i\}$ is a vector $f_i = \sum \beta_i \times c_i$, for $i \in (1..w)$, where the coefficient vector β_i is randomly generated in a finite field F_q of size q . f_i is referred to as *combined* data segments, which also spans a range of w . If all f_i are linearly independent, once a node has a subset of f_i that spans range w , it can recover all the w original segments by solving a set of linear equations.

3.3.4.2 Scheduling in storage-forwarding

For the storage-forwarding approaches several modifications of the BitTorrent segment selection algorithm have been proposed. On the other hand, when individual peers are under common control (this is the case of residential home gateways or set-top boxes-STBs-under the control of a content provider), the use of rateless coding such as LT or Raptor codes is a new alternative to address this issue.

BitTorrent regards all segments as equally important, and it employs Tit-for-Tat (TFT)

3.4. Background

Figure 3.8: BiToS peer structure.

strategy in determining to which neighbors a peer serves content, (that is, a peer favors a neighbor who transmits data in return). However, in a VOD system the highest priority is given to the segments close to their playback time. BiToS [Vlavianos et al., 2006] supports the VOD service by modifying the segment selection algorithm of BitTorrent. A peer in BiToS has three components, as shown in Figure 3.8: The *received buffer* stores all the segments that have been received, the *high priority set* contains the segments close to their playback time which have not been downloaded yet, and the *remaining piece set* contains the segments that have not been downloaded. A segment in the *high priority set* is downloaded with probability p while one in the *remaining pieces set* is downloaded with probability $1 - p$. By setting the value of p greater than 0.5, the blocks in the *high priority set* are given preference over the ones in the *remaining pieces set* to be downloaded earlier.

In [Suh et al., 2007] the peers are set-top boxes (STBs), which are under the control of a content provider. Each video is chopped into windows of contiguous data of size W , and it is pushed to a population of STBs (supposing that there are M STBs). The authors propose two data placement schemes: full-striping and code-based. The full striping scheme stripes each window in M blocks, each of size W/M , and each one is pushed to only one box. A full window is reconstructed at a particular box by concurrently downloading $M - 1$ distinct blocks for the window from the other $M - 1$ boxes. Code-based placement tries to reduce the maximum number of simultaneous connections that a box can serve to y (for some $y < M - 1$). In order to achieve this, it applies LT coding [Luby, 2002]. It divides each window into k source symbols, and generates Ck coded symbols (C is called the expansion ratio). A viewer can reconstruct a window of a video by concurrently downloading any Cky/M distinct symbols from an arbitrary set of y boxes out of $(M - 1)$ boxes.

3.4 Background

Nowadays, a lot of telecommunications operators are offering on-demand services via IPTV. These services are normally part of a triple-play offering (Internet, telephony, television). Television and on-demand services are transmitted according to the IP protocol via the telephone networks (in DSL mode), and finally they are sent to the television via a Set-Top-Box (STB) dedicated to both services. Since they use the IP protocol, television and on-demand services distributed on DSL networks are commonly referred by the generic term “IPTV”. However, the use of the IP protocol is not specific to these networks and it may also be used in other access networks (fiber-to-the-home, mobile telephony

networks) and even in some instances of satellite distribution.

In this chapter we extend the work presented in [Nafaa et al., 2008], which was concentrated on the analysis of a VOD service distribution architecture in a IPTV context. It is based on P2P concept, where the content is stored in the STBs, which act as peers, and the sessions are provided by means of multi-source streaming [Nafaa et al., 2009][Lamoriniere et al., 2009]. The normal operation of this architecture is as follows. First, the popularity associated to each title of the library is matched to a precise number of copies that minimizes the rejection rate. In this model, the popularity of each title is calculated in terms of the average number of requests received during a fixed period of time. Once the total number of copies of all the titles has been obtained, each one of them has to be broken into a specific number of substreams (i.e., the number of layers composing each title), and injected in the different STBs using a dispatching algorithm during the off-peak hours, when bandwidth is plentiful. When a VOD request gets to the system the resource allocation strategy is responsible of allocating a given set of STBs to provide the different substreams.

The previous work [Nafaa et al., 2008] was concentrated on the first two operation mechanisms of the architecture. It proposed a popularity-to-availability translation model, and it also presented a study on the impact of the number of substreams over the rejection rate. It was expected that an increased number of substreams would increase the content availability in the network and consequently decrease the rejection rate. However, it revealed that beyond a certain point, the number of substreams has no impact on the rejection rate. A too large number of layers would actually increase the probability of a VOD session failure since the requesting STB relies on too many contributing STBs, which increase the risks of failure.

It is important to briefly describe the main characteristics of the video fragmentation process, which is central to the peer-assisted VOD streaming architecture. Readers can find more information about this topic in references [Nafaa et al., 2009] and [Lamoriniere et al., 2009]. The main goal of this process is to be able to support high definition content delivery and bypass the limited uplink capacity at the different peers (STBs). For example, ten peers with pre-cached content fragments can stream the respective content fragments at a rate of 200 Kbps to satisfy a 2 Mbps video request. In short, the content fragmentation and injection process makes it possible to aggregate the upload bandwidth of several peers and reach streaming capacities satisfying for a high definition content.

Figure 3.9 illustrates an example of the video fragmentation process. In this example, RTP (Real Time Protocol) packets of a given streaming session are divided into 5 substreams, which are stored in 5 different video files. The couple (start, step) is used to uniquely identify a given substream from a particular video content. The parameter start represents the *Sequence Number* of the first RTP packet of a substream, while step represents the stride between successive RTP sequence numbers of packets belonging to the substream. For instance, the first substream is identified by the filter $\text{start}=1/\text{step}=5$, meaning that the substream is composed of RTP packets with the *Sequence Numbers*= 1, 6, 11, 16, 21, etc.

The advantages of exploiting STBs to assist VOD service distribution on the Internet have been introduced in [Laoutaris et al., 2008]. The main advantages are the following: First of all, it requires almost no capital expenditure as it is composed of components

3.4. Background

Figure 3.9: Video fragmentation process.

which are already deployed, monitored and powered. Secondly, it solves almost all P2P's shortcomings. For example, it does not have to worry about piracy or free-riding, and it can employ near optimal overlay construction, replication and routing to offer guaranteed performance.

In [Chen et al., 2009b] and [Chen et al., 2009a] authors provide a comprehensive study about the allocation of substreams to the selected boxes. This allocation is crucial for network operators and ISPs. A good allocation can represent substantial savings when considering the network cost of the streaming sessions from their clients to the different requested substreams, i.e. the cost to transport the data from the boxes hosting the substreams to the client. On the other hand, the optimal number of substreams per video and the ideal number of boxes have been thoroughly analyzed in [Nafaa et al., 2008] and [Boufkhad et al., 2008].

Another challenge is the pricing strategy. In order to encourage viewers to make their STBs available for P2P sharing, some incentive may be given to peers who upload videos to other peers. [Chen et al., 2009c] analyzes the cost and benefit of deploying P2P technology and compares its maximum possible profit to that of a conventional service. For that, authors couple three P2P pricing models (flat-free, usage-based, and built-in) with a physical model and study their trade-offs from a service provider's perspective.

The most relevant VOD initiative, in regards to our architecture, is undoubtedly the one undertaken by VuDu [Ganesan et al., 2007] that offers a VOD service access over Internet. End-users' STBs are used as peers in a P2P video streaming architecture. In this system, an initial video sequence of each title in the library is stored in all STBs, which allows any end-user to instantly start playing the initial part of any content while the rest of the sequence is received from other STBs.

More recently, hybrid Content Delivery Networks (CDN) solutions have arisen as a viable cost-effective streaming solution where the centralized control of conventional CDN approach is combined to the scalability of P2P concept, resulting in a peer-assisted

solution able to offload the CDN servers while still providing the necessary system predictability with reasonably guaranteed QoS. Hybrid CDN architectures, such as the ones offered by Abacast [Huntington, 2005], Velocix [Twiss and Ferguson, 2007], GridNetworks¹, and BitTorrentDNA², focus on developing cost-effective video delivery systems by pushing some of the content delivery load to the clients. As a general rule, peers are put to contribution in delivering VOD session as much as possible, while the CDN servers are used as a backup to overcome the uncertainty of peers hosted at Internet end-users. However, none of the previous architectures use multi-source streaming to provide VOD sessions as in the system designed in this work.

3.5 VOD architecture in a broadband operator context

3.5.1 Introduction

In this section, it is firstly presented a performance evaluation of the full system in the IPTV scenario. The system is evaluated using a new popularity-to-availability translation model, a new dispatching algorithm and a new resource allocation strategy. This evaluation is based on simulation and it is concentrated on two different parameters: the rejection rate, that is, the percentage of requests which are rejected by the VOD architecture; and the maximum number of simultaneous VOD sessions supported by the system. From these simulation results, it is deduced that the previous presented mechanisms are not enough to fully take advantage of the network resources to increase the maximum number of VOD sessions and consequently to decrease the associated rejection rate.

After that, it is demonstrated that a necessary condition to increase the maximum number of simultaneous VOD sessions is to be able to uniformly distribute the rejection rate over the full library of titles, independently of their associated popularity. That is, when a user sends a request for a specific title to the VOD service, it will be rejected with a specific probability independently of the popularity associated to the requested title. In the last section of the paper it is presented a mechanism which is able to achieve the previous condition. It tries to dynamically re-allocate the necessary streams in order to find available resources for a new VOD session which cannot be established. It is demonstrated that this mechanism is able to get the maximum number of VOD sessions throughout the full simulation.

3.5.2 Architecture description

At the highest level of the architecture, the system comprises of three basic components:

- 1. SuperNode (SN):** this is where the main intelligence in the system lies - it allocates resources (uplink bandwidth of contributing peers) for each incoming VOD request. In order to perform this, the SN tracks two main resources: *(i)* the currently available uplink bandwidth at each peer, and *(ii)* the content stored in each peer. When a given receiving peer requests a specific VOD content, a VOD request is sent from that latter receiving

¹<http://globalmediaservices.net>

²<http://www.bittorrent.com/dna>

3.5. VOD architecture in a broadband operator context

Figure 3.10: High-level view of the VOD architecture.

peer to the SN. The SN looks up its database to determine the most appropriate set of contributing STBs that might stream complementary streams to the receiving peer. Clearly, the SN handles the VOD session initiation signalling, while the load of serving the current video streams is handled by the contributing peers and eventual caches.

2. Peer: this can be a Set-Top-Box or some other intelligence (device with computing capabilities) which typically sits close to a user's viewing capabilities (TV, projector, and monitor). The peer is assumed to have significant storage capacities materialized by a hard disc that can contain sub-streams from different video titles of a video content library; upon sending a VOD request to the SN, a requesting peer will receive a response from the SN with a list of potential contributing peers that will provide complementary VOD sub-streams. The requesting peer then initiates real-time sub-streaming sessions with the contributing peers through RTSP/RTP protocols.

3. Cache: this is a large data store. The cache can be considered as a passive peer (STB) in the sense that it can contribute to VOD sessions, but it will never request a VOD session in the network. The main role of the cache is to offset the limited uplink bandwidth capacities available at STBs - the cache can also contribute to VOD sessions with much higher data volumes. Typically, in an Internet video streaming scenario, the caches will be located in a CDN provider's network in order to ensure high availability and performance predictability. The caches can be hosted by the broadband operator if the peer-assisted VOD platform is deployed as part of an IPTV solution.

The different components of the peer-assisted VOD architecture described above are depicted in Figure 3.10.

3.5.3 Working assumptions

In the rest of the section we will analyze the performance of our P2P streaming system for a managed IP network composed of 10,000 active STBs, with a storage capacity of 100 Gbytes and an uplink capacity of 1,200 Kbps. The content library is composed of 10,000 titles with duration of 2 hours. Each single title in the content library has a storage requirement of 1,024 Mbytes and a bitrate of 3,500 Kbps. In addition, each title is broken into 10 fragments before being injected in the network. From the previous parameters, it is deduced that each STB can have a maximum of 3 simultaneous uplink connections.

Each run lasts 27 hours and it consists of generating VOD requests originating from different STBs following an exponential distribution with a mean of 7,200 seconds. In addition, the requests are generated following a popularity distribution extracted from the Netflix traces¹. The distribution of the session play times follows a normal distribution with a mean of 0.50 times the content duration and standard deviation of 0.22 times the content duration. These data have been extracted from Kasenna white paper² that reports quantitative results on typical user's behavior with VOD services.

In order to gain a thorough insight into the interaction between VOD content popularity, availability and rejection, the titles are classified into 20 different categories with the different categories accounting for exactly the same volume of VOD requests. That is, the sum of the probability of VOD request targeting any particular popularity category is equal to any other popularity category. This permits a better understanding of the system and consequently improves our abilities to improve the system performances.

3.5.4 Popularity-to-availability translation model

The number of copies of a given title available in the network plays the major role in determining whether the demand for this title would be satisfied or not. Therefore, it is essential to match a certain level of popularity with a precise number of copies that minimizes the VOD requests rejection rate. The model presented in this work is widely different from the one explored in [Nafaa et al., 2008] in the sense that this work focus on deriving an availability model that is proportional to the content popularity model.

In this new model, the ratio of the number of copies of a given title over to the total number of copies in the network (*total_number_copy*) is given by the title relative popularity in the content library. Therefore, there exists a proportional relationship between the number of copies of a given title available in the network (*number_copy(m)*) and the popularity of this title (*popularity(m)*):

$$number_copy(m) = total_number_copy \times popularity(m) \quad (3.1)$$

total_number_copy is defined as the total storage in the network (*total_storage_network*) divided by the average size of the movies (*average_size_movies*). That is,

$$total_number_copy = \frac{total_storage_network}{average_size_movies} \quad (3.2)$$

¹<http://www.netflixprize.com>

²<http://www.kasenna.com>

3.5. VOD architecture in a broadband operator context

Figure 3.11: Set-top box popularity score after injecting all the fragments in the network.

We calculate $total_storage_network$ using the number of STBs ($number_STB$) and the storage capacities of the STBs ($capacity_STB$), taking into account that all of them have similar capacities:

$$total_storage_network = number_STB \times capacity_STB \quad (3.3)$$

At the same time, we calculate $average_size_movies$, using the storage requirements ($size(m)$) of each one of the M different movies and taking into account that the percentage of copies of a given title is given by its popularity:

$$average_size_movies = \sum_{m=1}^M size(m) \times popularity(m) \quad (3.4)$$

Therefore,

$$total_number_copy = \frac{number_STB \times capacity_STB}{\sum_{m=1}^M size(m) \times popularity(m)} \quad (3.5)$$

As introduced earlier, the popularity distribution used in this work has been extracted from the publically available traces of Netflix that counts more than 10,000 titles today.

3.5.5 Dispatching algorithm

After allocating a given storage space for each title in the content library using the popularity distribution in the content library, the dispatching algorithm injects the different fragment from all the different titles. Ideally, the performance of the overall peer-assisted VOD streaming system will be optimized if the load of P2P streaming is perfectly balanced between the different active STBs. To achieve this, the service provider should ensure that the fragments of any title T are as spread as possible, to reduce the probability of having all the STBs containing a fragment of T being saturated quickly. Further, the service provider should ensure that each STB will receive the same level of solicitation to contribute to VOD sessions delivery. To this end, it is important to guarantee that the

Figure 3.12: Number of different STBs containing a fragment of a specific title.

sum of popularity of all fragments contained in a given STB is equivalent to the sum of popularity of all fragments contained in any other STB in the network. This essentially means that the different STBs will likely contribute at the same level in the process of delivering VOD services.

In the new dispatching algorithm, called popularity-weighted dispatching algorithm, the fragments are first assigned a weight that is proportional to their relative popularity (i.e., relative popularity of the title they are part of); then, each time an STB receives a fragment its STB popularity score (SPS) is incremented by the fragment's popularity weight. This algorithm takes also into account the content spreading constraint by favoring the title's fragment injection in a new STB to avoid having two fragments from the same title being stored in a given STB.

More specifically, the algorithm first considers injecting a given fragment content in the STB with the lowest SPS; if this latter already contains a fragment of that content, then the algorithm considers the STB with the second lowest SPS, and so on. After each fragment injection in a given STB, the SPS of this latter STB is augmented with the fragment popularity; also, the list of STBs, ordered from the lowest SPS to the highest, might change after each single fragment injection. Figure 3.11 illustrates the SPSs figures associated to each STB when the popularity-weighted fragment dispatching algorithm is used. Clearly, this algorithm guarantees that the sum of popularity of all fragments contained in a given STB is equivalent to the sum of popularity of all fragments contained in any other STB in the network. Therefore, it is expected the different STBs contribute at the same level in the process of delivering VOD services.

As explained earlier, the popularity-weighted dispatching algorithm also takes into account the content spreading requirement. Figure 3.12 illustrates how many different STBs contain a fragment of a specific title. However, from this figure it is very difficult to extract useful information about the spreading of the different titles. Therefore, we are going to define a new metric called Content Dispersion Index.

The Content Dispersion Index is defined as the quotient of the total number of fragments of a specific title and the number of different STBs containing a fragment of the same title (presented in Figure 3.12). This new parameter represents the average number

3.5. VOD architecture in a broadband operator context

Figure 3.13: Content dispersion index.

of fragments of a specific title stored in a STB which contains at least one fragment of the same title. This parameter is presented in Figure 3.13, and we can observe that all the titles have an associated Content Dispersion Index equal to 1. This even distribution is mainly realized by the fact that the popularity-weighted dispatching algorithm tries always to avoid injecting two fragments of the same title in any given STB. Therefore, from the previous results we can say this dispatching algorithm is able to ensure that the different titles are as spread as possible, because none of the STBs have two fragments of the same title.

3.5.6 Performance evaluation

This section presents the performance evaluation of the full system when the previous popularity-to-availability translation model and dispatching algorithm are used. However, it is also necessary to use a specific resource allocation strategy. When an incoming VOD request gets to the system this strategy is responsible of allocating a given set of STBs to provide the different substreams.

In this work we are going to use a very simple resource allocation strategy which discriminates and orders the STBs using their available uplink bandwidth. When a new VOD request is received, the SuperNode (SN) looks up its database and retrieves all the STBs that can contribute in delivering the VOD session in question. At this point, the SN will choose contributing STBs with the highest available bandwidth when the VOD request is being processed. The rationale here is to favor the usage of idle STBs that are not contributing to currently serving VOD sessions and avoid having certain STBs poorly contributing to the multi-source video streaming process. This strategy also helps balancing the utilization of all active STBs.

In the following, two different performance metrics are used to evaluate the full system: the rejection rate, which represents the percentage of VOD requests that cannot be served in the network; and the number of simultaneous VOD sessions.

Figure 3.14: Rejection rate.

3.5.6.1 Rejection rate

Figure 3.14 illustrates the total rejection rate along with the rejection rate for each popularity category, and it shows that this parameter increases with a decreasing content popularity. It is worth recalling that all the different popularity categories receive the same VOD request volume in overall.

We can observe that the rejection rate associated to the least popular titles is unacceptable for a commercial service because most of these requests are rejected by the VOD architecture. Customers could also be interested in watching less popular titles, and therefore, they should have higher guarantees to be able to watch this kind of titles than the ones that the strategies presented till the moment offer. In the next sections we will try to improve these results.

3.5.6.2 Number of simultaneous VOD sessions

Figure 3.15 represents the total number of simultaneous VOD sessions. We can observe that this parameter is around 3,000 sessions. As it was introduced earlier, each STB can have up to 3 simultaneous uplink connections. Taking into account that there are 10,000 STBs in the network, the maximum number of simultaneous uplink connections is 30,000. On the other hand, we know that each VOD session needs 10 uplink connections; therefore the theoretical maximum number of simultaneous sessions is 3,000.

However, we can observe in the presented results that it is possible to have more than 3,000 sessions. The reason is the following: before sending a VOD request to the SuperNode the STB checks if it has any of the required fragments in its local storage. If it has some of them, then the SuperNode only tries to establish connections for the rest of fragments. It means that sometimes some VOD sessions use less than 10 uplink connections in the network. Therefore, there are more uplink connections available for the new requests, and consequently the network is able to support a bit more than 3,000 simultaneous sessions.

Figure 3.16 represents the number of simultaneous VOD sessions per content category. In order to clarify the presented results we have divided the 20 categories of contents in

3.5. VOD architecture in a broadband operator context

Figure 3.15: Number of VOD sessions.

(a) Categories 0-4.

(b) Categories 5-9.

(c) Categories 10-14.

(d) Categories 15-19.

Figure 3.16: Number of VOD sessions per category of contents.

4 groups of 5 categories. The results for the 10 first categories (0-9) (Figures 3.16(a) and 3.16(b)) are very similar. We can see that the number of sessions is always between 150 and 200. However, the number of sessions for the third group of categories (Figure 3.16(c)) is lower because it is always around 150 sessions, and for the last category (Figure 3.16(d)) it is much lower, because none of the categories in this group has more than 150 sessions. In addition, this number decreases when the category increases and it is around 50 for the last category the number of sessions.

In Figure 3.15 we can check that the maximum number of VOD sessions is 3,040. Taking into account that there are 20 categories of contents, and that all of them receive the same volume of requests, the number of sessions of the different categories should be around $3,040/20=152$ sessions. However, as we can see in the first two groups of categories, the number of sessions is always much higher than 152. This situation means that the most popular titles are using more resources than the rest of titles and it makes that the least popular titles have less resources available to initiate new VOD sessions. That is, in some way, the most popular titles are consuming the resources that could be available to be used by the least popular titles to increase the number of simultaneous VOD sessions.

3.5.7 Limitation of VOD sessions per category

We have checked in Figure 3.16 that the most popular titles are able to achieve a much higher number of simultaneous VOD sessions than the rest of titles, and it makes harder for the least popular titles to initiate new sessions. In this section we propose to limit the maximum number of VOD sessions per category. That is, it was previously introduced that the maximum number of VOD sessions of the different categories of contents should be 152. Therefore, here we propose to limit the number of VOD sessions per category to 152.

With this mechanism, when the SuperNode receives a new VOD request it first checks the category of the title which is being requested, and then it finds out if the number of active sessions associated to this category is lower than the maximum. In this case, the VOD session can be established, otherwise the request would be rejected immediately.

3.5.7.1 Rejection rate

Figure 3.17 represents the total rejection rate along with the rejection rate for each popularity category. First of all, it is checked that this mechanism increases slightly the total rejection rate if we compare this figure with Figure 3.14. On the other hand, a second major difference is that this mechanism is able to distribute the rejection rate more equally among the different categories of titles, because most of them have a very similar rejection rate. This is a good result because it means that the rejection rate is independent of the popularity of the title. When a specific title is requested it does not mind if this title has been previously requested by a big or a small set of clients, customers have the same right to watch more and less popular titles. However, the least popular titles have a slightly higher rejection rate than the rest of titles. In the following subsections we will try to guess the reason of this result.

3.5. VOD architecture in a broadband operator context

Figure 3.17: Rejection rate.

3.5.7.2 Number of simultaneous VOD sessions

Figure 3.18 represents the total number of simultaneous VOD sessions. In this figure, we can clearly check that the mechanism proposed in this section does not have a positive effect over the system performance because the system is not able to get the maximum physical capacity of the network (3,000 sessions). This situation explains that the total rejection rate is a bit higher when this mechanism is used.

Figure 3.19 represents the number of VOD sessions per category. We can see that the number of sessions for all the titles is always at the maximum (152), except for the least popular titles (category 19). That is the reason why the rejection rate associated to this last category of titles is higher than for the rest of categories. On the other hand, if the number of sessions for the least popular titles is always lower than 152, and the number of sessions for the rest of categories is always 152 more or less, it makes that the total number of VOD sessions is always lower than the maximum, as we can check in Figure 3.18.

We can conclude that the mechanism proposed in this section does not have a positive effect over the system performance. However, we have learnt that there are two characteristics which can help to improve the global system performance. One of them is the distribution of the rejection rate. We have deduced that a uniform distribution of the rejection rate over the different categories of contents is a desired situation. In addition, the number of simultaneous VOD sessions per category should be equally distributed over the different categories of contents. If we can obtain these two characteristics in our system, we will be able to improve the global performance of our system, and the number of simultaneous VOD sessions could be increased towards the maximum physical capacity of the network.

3.5.8 Dynamic re-allocation of streams

In the previous section we have checked that the limitation of the VOD sessions per category of contents is not a good idea to improve the system performance. Our final goal is to try to increase the number of simultaneous VOD sessions, and if we want to get that

Figure 3.18: Number of VOD sessions.

(a) Categories 0-4.

(b) Categories 5-9.

(c) Categories 10-14.

(d) Categories 15-19.

Figure 3.19: Number of VOD sessions per category of contents.

3.5. VOD architecture in a broadband operator context

Figure 3.20: Rejection rate.

result the rejection rate and the number of VOD sessions should be equally distributed over the different categories of contents.

In this section we propose a new mechanism which tries to re-allocate the necessary connections in order to find available resources for a new VOD session which cannot be established. Each STB registers a new score called popularity of active connections (PAC) which represents the addition of the popularity of the titles which are being transmitted in the different uplink connections of the STB. When the SN cannot find a specific fragment of a VOD session in the network because all the STBs which have this fragment in their local storage do not have available uplink bandwidth, then it puts in order all the STBs with that fragment from the lowest PAC to the highest PAC. Now, the SN will try to change one of the connections of these STBs to another STB in the network, and by this way, the new VOD request will be able to establish the necessary connection for that segment.

The first candidate STB is the one with the lowest PAC. The active connections of the first candidate are put in order from the most popular to the least popular. Then, the SN finds out if it is possible to change the most popular connection to a new STB. If it is not possible, it will try to change the following connection, and it will go on by this way till it has checked all the active connections. If the SN cannot change any connection of the first candidate STB, it will try with the following candidate STB. If the SN cannot find an alternative for any of the active connections of the candidate STBs, then the new VOD request is definitely rejected.

3.5.8.1 Rejection rate

Figure 3.20 represents the total rejection rate along with the rejection rate for each popularity category. The first remarkable result is that the total rejection rate is lower than in the results presented in Figure 3.14. However, the most important conclusion extracted from this figure is that the dynamic re-allocation of streams mechanism is able to achieve one of the necessary conditions to improve the global performance of the system: It is able to equally distribute the rejection rate over the different categories of contents. In addition, in this case the category of the least popular titles (category 19) also has a rejection rate similar to the rest of categories.

(a) Categories 0-4.

(b) Categories 5-9.

(c) Categories 10-14.

(d) Categories 15-19.

Figure 3.21: Number of VOD sessions per category of contents.

3.5.8.2 Number of simultaneous VOD sessions

Figure 3.21 represents the number of VOD sessions per category of contents. We can see that the number of VOD session for all the categories of contents is always around 152. Therefore, this mechanism is also able to achieve the second of the necessary conditions to improve the global performance of the system: It equally distributes the active VOD sessions over the different categories of contents.

Finally, Figure 3.22 represents the total number of simultaneous VOD sessions. It is clearly demonstrated that the dynamic strategy is able to get the maximum number of VOD sessions throughout the full simulation, and it improves the results obtained without using this mechanism. Therefore, we can conclude that the strategy presented in this section achieves our goal: To increase the number of simultaneous VOD sessions.

Figure 3.22: Number of VOD sessions.

3.6 VOD architecture for the Internet

3.6.1 Introduction

In this section it is extended the previous VOD service distribution architecture to the Internet context. In this case, the clients are distributed over different ISP networks connected to the Internet. It is firstly presented a new dispatching algorithm. The decision on what content is dispatched at which STB is determined by a Non-Linear Programming (NLP) formulation. This formulation also implements the popularity-to-availability translation. The full system is evaluated by means of simulation using a very simple resource allocation strategy, which orders the STBs using their available uplink bandwidth. The results reported compare our optimal formulation against two other strategies: Random, by which any fragment is equally likely to be pre-seeded at any STB; and Random-Weighted, by which fragments are randomly pre-seeded to STBs weighted by the popularity of the title. Results from simulations show that using the formulation decision for dispatching can help in efficiently mitigate server load.

After that, two new mechanisms to forward the VOD requests between different ISP networks are presented. In the first one (*full requests forwarding*), when a VOD request cannot be served in the ISP network of the client, it is iteratively re-directed to other ISPs until it finds available resources to serve the VOD session. In the case that there are not resources in any ISP network, the VOD session is served by the central server. The restriction of this mechanism is that all the layers of a specific VOD session must be served by STBs of the same ISP network. The sense of this policy is to try to assure that the streaming of all the layers undergo a similar delay. It is showed that it is able to reduce the total percentage of server load. However, the percentage associated to least popular contents is unacceptable for a commercial service, because most of these sessions are provided by the central server. On the other hand, in the second mechanism (*sub-requests forwarding*) the previous restriction disappears. That is, each layer of the VOD session can be served by STBs located in different ISP networks or by the server. This mechanism reduces the percentage of server load associated to least popular contents in a high order. In addition, it also tries to distribute more equally the server load among the

Figure 3.23: High-level view of the VOD architecture.

different categories of contents.

Next, it is presented a mechanism which tries to dynamically re-allocate the necessary streams in order to find available resources for a new VOD session which cannot be established. It is demonstrated that the implementation of this mechanism in the *full requests forwarding* strategy has a very similar effect to the *sub-requests forwarding* strategy. On the other hand, it is also shown that this new mechanism is not able to improve the results provided by the *sub-requests forwarding* strategy. Finally, the last section of the paper presents an algorithm for the update of the off-line content injection process. This process takes place periodically (e.g. once or twice a week) during the off-peak hours. It is possible that between two consecutive content injections the number of clients in some ISP networks changes. In addition, it is also possible that during that time the popularity associated to the different titles has been modified. Both changes make that the optimal distribution of contents over the different STBs changes. The algorithm presented in the last section of the paper tries to reduce the data traffic to make possible the new content distribution.

3.6.2 Architecture description

The different components of the peer-assisted VOD architecture are depicted in Figure 3.23. The basic components are the same than in the previous scenario. There is one SuperNode (SN) in each ISP network, and it can be any peer located in the ISP network. Several characteristics, like the uplink bandwidth or the CPU resources, will be taken into account to select the most appropriate SN. On the other hand, each peer is a STB or some other intelligence (device with computing capabilities) which typically sits close to a user's viewing capabilities (TV, projector, and monitor).

However, in this scenario there is a central server too. In case that a specific VOD request cannot be served by the P2P network, the video is directly transmitted from this server. In order to simplify the proposed model, the streaming server of the video provider is assumed to be located in a different network, where it is not necessary that there are

3.6. VOD architecture for the Internet

peers. We assume that the video provider knows the network of the STB, i.e. the identifier of the first router that connects this STB to the Internet, and the network cost between every two routers (see [Wong et al., 2005, Augustin et al., 2006] for some techniques that may be used).

3.6.3 Working assumptions

In the rest of this section we will analyze the performance of our P2P streaming system in a scenario composed of 200 ISP networks. A router-level topology resulted from the *Network Cartographer (nec)* [Magoni and Hoerd, 2005] is used as our network topology. We randomly select 200 edge routers with degree equal to one, and we attach to each of them a number of STBs following a normal distribution $N(\mu = 110, \sigma = 10)$. Each STB has a storage capacity of 5 Gbytes and an uplink capacity of 1,000 Kbps. The content library is composed of 100 titles with duration of 2 hours. Each single title in the content library has a storage requirement of 1,024 Mbytes and a bitrate of 3,500 Kbps. In addition, each title is broken into 10 fragments before being injected in the network. From the previous parameters, it is deduced that each STB can have a maximum of 2 simultaneous uplink connections.

Each run lasts 27 hours and it consists of generating VOD requests originating from different STBs following an exponential distribution with a mean of 7,200 seconds. The identity of the titles requested are generated following the Pareto law, also known as the 20/80 rule. This principle can be reproduced using a Zipf distribution, as reported in the relevant literature. On the other hand, the distribution of the session play times follows a normal distribution with a mean of 0.50 times the content duration and standard deviation of 0.22 times the content duration. These data have been extracted from Kasenna white paper¹ that reports quantitative results on typical user's behavior with VOD services.

In order to gain a thorough insight into the interaction between VOD content popularity, availability and rejection, the titles are grouped into 4 different categories with the different categories accounting for exactly the same volume of VOD requests. That is, the sum of the probability of VOD request targeting any particular popularity category is equal to any other popularity category. This permits a better understanding of the system and consequently improves our abilities to improve the system performances.

3.6.4 Dispatching algorithm

3.6.4.1 Description

The dispatching algorithm injects the different fragments from all the different titles in the STBs. It is important to do intelligent dispatching so that we maximize the requests served by the P2P network and reduce the load on the central server. The decision on what content is dispatched at which STB is determined by a Non-Linear Programming (NLP) formulation. This formulation takes into account three important factors which have a high influence over the global performance of the system.

¹<http://www.kasenna.com>

First of all, it is necessary to take into account that the number of copies of a given title available in the network plays the major role in determining whether the demand for this title would be satisfied or not. Therefore, it is essential to match a certain level of popularity with a precise number of copies that minimizes the rate of VOD requests served by the central server. On the other hand, it is known that the performance of the overall peer-assisted VOD streaming system will be optimized if the load of P2P streaming is perfectly balanced between the different active STBs. To achieve this, the service provider should ensure that the fragments of any title T are as spread as possible, to reduce the probability of having all the STBs containing a fragment of T being saturated quickly. Further, the service provider should ensure that each STB will receive the same level of solicitation to contribute to VOD sessions delivery. To this end, it is important to guarantee that the popularity of all fragments contained in a given STB is equivalent in any other STB in the network. This essentially means that the different STBs will likely contribute at the same level in the process of delivering VOD services.

We model the setup as having N equally sized titles and M STBs. Let $i \in \{1, \dots, N\}$ represents the individual titles, $j \in \{1, \dots, 10\}$ represents the layers of each title and $k \in \{1, \dots, M\}$ represents the individual STBs. We assume that the STBs have identical capacities, each with the ability to hold C different layers (from parameters presented in Section 3.6.3, it is deduced that each STB can hold $C = 50$ different layers). We also assume that popularity of title i is given by $P(i)$.

Our decision variable is $X(i, j, k)$, which is an indicator to represent that layer j of title i is stored at node k . It is defined as an integer variable with only two possible values 0 and 1. The value 1 represents that the corresponding layer of the specific content is stored at that node, and 0 represents that it is not stored. Therefore, the formulation of the problem is defined as Mixed-Integer Non-Linear Programming (MINLP). The variable $S(i, j)$ is defined as a positive variable which represents the total number of copies in the network of the layer j of content i :

$$S(i, j) \triangleq \sum_{k=1}^M X(i, j, k) \quad (3.6)$$

The objective function has to take into account the previously presented factors. That is, it has to try that for each STB the proportion of layers of each title stored in its local storage is very similar to the popularity associated to the corresponding title. Considering the previous definitions, the problem is characterized as follows:

$$\min_{X(i,j,k)} \sum_{k=1}^M \sum_{i=1}^N \left| P(i) - \sum_{j=1}^{10} X(i, j, k) \cdot \frac{1}{C} \right| \quad (3.7)$$

$$s.t. \quad \sum_{i=1}^N \sum_{j=1}^{10} X(i, j, k) \leq C \quad (3.8)$$

$$S(i, j) - S(i, j - 1) = 0 \quad (3.9)$$

It can be checked that the objective function, Equation(2), considers the three previously presented factors: (i) the number of copies of the different titles is proportional

3.6. VOD architecture for the Internet

to the popularity distribution; (ii) the fragments of the different titles are spread over the different STBs; (iii) the proportion of contents in a given STB is equivalent for any other STB in the network. On the other hand, Constraint (3) limits the number of stored layers at a node to be C , and Constraint (4) establish that the number of copies of all the layers of a given title is the same. This constraint is defined for each pair (i, j) . The operator “—” represents a circular operator, that is, the first and last members of the set are assumed to be adjacent.

The model has been implemented in GAMS [Brook et al., 1988], a mathematical specification language specially indicated for the solution of optimization problems, and solved by using BARON (Branch-And-Reduce Optimization Navigator) [Sahinidis and Tawarmalani, 2010], a well-known MINLP solver. However, the use of integer variables increases the resolution time a lot and sometimes the solution obtained by the solver is not a global optimum of the objective function. In order to solve these problems, we decided to change the variable $X(i, j, k)$ to a continuous variable defined between 0 and 1. With this modification the new optimization problem is defined as a Discontinuous Non-Linear Programming (DNLP), because the absolute value function has discontinuous derivatives. However, the use of the DNLP is strongly discouraged. There exists an alternative to redefine the absolute value function using functions with continuous derivatives, and by this way, to translate the DNLP problem into a NLP problem. The reformulation approach is to introduce positive and negative deviation as extra variables [Drud, 1985]. The term $z = ABS(f(x))$ is replaced by $z = fplus + fminus$. $fplus$ and $fminus$ are declared as positive variables and they are defined with the identity: $f(x) = fplus - fminus$. Therefore, in our problem:

$$P(i) - \sum_{j=1}^{10} X(i, j, k) \cdot \frac{1}{C} \triangleq fplus(i, k) - fminus(i, k) \quad (3.10)$$

With this modification the problem is characterized as follows:

$$\min_{X(i,j,k)} \sum_{k=1}^M \sum_{i=1}^N fplus(i, k) + fminus(i, k) \quad (3.11)$$

$$s.t. \quad \sum_{i=1}^N \sum_{j=1}^{10} X(i, j, k) \leq C \quad (3.12)$$

$$S(i, j) - S(i, j - 1) = 0 \quad (3.13)$$

In this case, the model has been solved using CONOPT [Drud, 1985], a well-known NLP solver. The new definition of the decision variable $X(i, j, k)$ as a continuous variable makes it necessary to process the results obtained from the optimization problem to finally decide if the value assigned to each $X(i, j, k)$ value is 1 or 0. With this objective we have programmed a post-process algorithm with two stages. In the first one, the number of copies assigned to each title¹ is obtained from the results of the optimization problem. In

¹The number of copies of a specific title i is obtained from the number of copies of one of its layers j , that is, $\sum_{k=1}^M X(i, j, k)$, for a specific i and j . The result of this operation, assuming that $X(i, j, k)$ is

Figure 3.24: Percentage of server load.

the second one, for each layer j of each content i , the M values assigned to the different STBs by the optimization solver are read, and the associated STBs are ordered from the highest assigned value to the lowest one. After that, the algorithm injects the different copies of the fragment j of content i in the STBs with the highest associated $X(i, j, k)$ value until the number of copies obtained in the first stage is reached.

In addition, although the optimization problem already takes into account the content spreading constraint, the post-process algorithm also favors the title' fragment injection in a new STB to avoid having two fragments from the same title being stored in a given STB. More specifically, the algorithm first considers injecting a given fragment content in the STB with the highest $X(i, j, k)$ value; if this latter already contains a fragment of that content, then the algorithm considers the STB with the second highest $X(i, j, k)$ value, and so on.

3.6.4.2 Performance evaluation

This section presents the performance evaluation of the full system when the previous dispatching algorithm is used. However, it is also necessary to use a specific resource allocation strategy. When an incoming VOD request gets to the system this strategy is responsible of allocating a given set of STBs to provide the different substreams.

In this work we are going to use a very simple resource allocation strategy which discriminates and orders the STBs using their available uplink bandwidth. When a new VOD request is received, the SuperNode (SN) looks up its database and retrieves all the STBs within its ISP network that can contribute in delivering the VOD session in question. At this point, the SN will choose contributing STBs with the highest available bandwidth when the VOD request is being processed. The rationale here is to favor the usage of idle STBs that are not contributing to currently serving VOD sessions and avoid having certain STBs poorly contributing to the multi-source video streaming process. This strategy also helps balancing the utilization of all active STBs.

a continuous variable, is very similar to the integer case. The small differences have been solved, without loss of generality, by means of compensation.

Figure 3.25: Number of VOD sessions served by the P2P network.

In the following, three different performance metrics are used to evaluate the full system: the percentage of server load, which represents the percentage of VOD requests that are served by the central server; the number of VOD sessions served by the P2P network; and the server bandwidth, which represents the bandwidth used by the central server to stream multimedia content.

The results reported compare our optimal formulation against two other strategies: Random, by which any fragment is equally likely to be pre-seeded at any STB; and Random-Weighted, by which fragments are randomly pre-seeded to STBs weighted by the popularity of the title (e.g., if one title is twice as popular as another, it will be twice as likely to be seeded at each STB). All these strategies have been implemented in a simulator programmed in C language, which use the working assumptions presented in Section 3.6.3.

(a) Percentage of server load

Figure 3.24 illustrates the total percentage of server load along with the percentage for each popularity category for the three dispatching strategies. It is worth recalling that all the different popularity categories receive the same VOD request volume in overall. First of all, it is showed that the Random strategy performs worse than all the other strategies for the total percentage. This behavior is expected, as this strategy does not consider the title popularity at all. On the other hand, we can check in the results per category of contents that this strategy slightly decreases the percentage of server load when the category increases.

However, the other two strategies behave in a opposite way, because both increase the server load when the category increases. But it is clearly proved that the Optimization case performs better than the Random-Weighted strategy with a total percentage of server load of 24 %. Although Random Weighted considers title popularity when assigning data, it does not explicitly consider the restriction on balancing the load of P2P streaming to the different active STBs, so it is likely that certain links are overloaded because multiple

Figure 3.26: Server bandwidth.

popular titles can be allocated to the same node.

(b) Number of VOD sessions served by the P2P network

Figure 3.25(a) represents the total number of simultaneous VOD sessions served by the P2P network for the three dispatching strategies. The Optimization strategy performs better than all the other dispatching strategies with a value around 5,500 sessions. As it was introduced earlier, each STB can have up to 2 simultaneous uplink connections. Taking into account that there are 21,996 STBs in the network, the maximum number of simultaneous uplink connections is 43,992. On the other hand, we know that each VOD session needs 10 uplink connections; therefore the theoretical maximum number of simultaneous sessions is nearly 4,400.

However, we can observe in the presented results that it is possible to have around 5,500 sessions. The reason is the following: before sending a VOD request to the SuperNode the STB checks if it has any of the required fragments in its local storage. If it has some of them, then the SuperNode only tries to establish connections for the rest of fragments. It means that sometimes some VOD sessions use less than 10 uplink connections in the network. Therefore, there are more uplink connections available for the new requests, and consequently the network is able to support more than 4,400 simultaneous sessions.

Figure 3.25(b) represents the number of simultaneous VOD sessions per content category for the Optimization strategy. It is clearly shown that the number of VOD sessions increases for the most popular contents. In Figure 3.25(a) we can check that the maximum number of VOD sessions for this strategy is around 5,500. Taking into account that there are 4 categories of contents, and that all of them receive the same volume of requests, the number of sessions of the different categories should be around $5,500/4=1,375$ sessions. However, as we can see in the first two categories, the number of sessions is always much higher than 1,375. This situation means that the most popular titles are using more resources than the rest of titles and it makes that the least popular titles have less resources available to initiate new VOD sessions. That is, in some way, the most popular titles are consuming the resources that could be available to be used by the least popular titles to increase the number of simultaneous VOD sessions.

Figure 3.27: Percentage of server load.

(c) Server bandwidth

Figure 3.26 represents the server bandwidth consumption for the three dispatching strategies. As it was expected, the Optimization strategy greatly reduces this parameter. It is shown that the server bandwidth consumption obtained with the Optimization strategy is approximately half of the bandwidth consumed with the Random strategy. In addition, this consumption is roughly 2 Gbps lower than the bandwidth consumed with the Random Weighted strategy.

3.6.5 VOD requests forwarding

In this section two mechanisms to forward the VOD requests between different ISP networks are presented. When the SN located in the network of the client cannot find resources to serve the video session within this network, it forwards the requests to the SN located in the nearest network. This new SN will try to find resources in its network. If it cannot find resources neither, it will forward the request to its nearest network. This process will go on until all the networks have been checked. In the case that none of them has enough resources to serve the session, it will be finally served by the central server. The distance between two network elements is defined as the number of routers on the shortest path between them. The mechanisms presented in this section will be evaluated for the Optimization strategy.

3.6.5.1 Full requests forwarding

This mechanism considers that all the layers of a VOD session must be provided by STBs located in the same network. The sense of this policy is to try to assure that the streaming of all the layers undergo a similar delay. Therefore, when a SN receives a VOD request has to find network resources for all the layers of the video (except the layers that are located in the local storage of the client). If there are not enough resources in the network of the SN it will forward the full request to the SN located in the nearest network.

Figures 3.27, 3.28 and 3.29 represent the percentage of server load, the number of VOD sessions served by the P2P network and the server bandwidth, respectively, in order

Figure 3.28: Number of VOD sessions served by the P2P network.

to compare this forwarding mechanism with the results presented in the previous section. Regarding to the percentage of server load, if we compare Figure 3.27 with Figure 3.28, it is shown that this forwarding mechanism reduces the total percentage from 24 % to 15 %. In addition, it is also able to reduce tremendously the percentage of server load for categories 1 and 2, because now they are very near 0 %.

The reduction of the total percentage of server load is expected, as this forwarding strategy looks for available resources to serve the VOD requests in other ISP networks. On the other hand, although it also reduces the server load for category 3 (from 89 % to 72 %), the value obtained is unacceptable for a commercial service because most of these sessions are provided by the central server.

Figure 3.27 represents the total number of VOD sessions served by the P2P network. If we compare these results with the ones presented in Figure 3.25 we can check that this mechanism increases the total number of VOD sessions from 5,500 to 6,100, due to the increase of the number of VOD sessions associated to categories 1 and 2 to a value near 2,000, and the increase of the number of sessions associated to category 3 to 400 sessions. Finally, Figure 3.29 shows that this mechanism decreases 2 Gbps the bandwidth consumed by the server, and now it is around 4 Gbps.

As a conclusion, it has been proved that the forwarding of the full requests between ISP networks is able to highly improve the performance associated to the most popular contents (categories 0, 1 and 2), however it only slightly improves the results for the least popular contents (category 3).

3.6.5.2 Sub-requests forwarding

With this second forwarding mechanism it is not necessary that all the layers of the video session are served by STBs located within the same ISP network. That is, when the SN receives a VOD request it tries to find resources for the fragments that are not located in the local storage of the client. If it only finds resources for some of them, then it will

Figure 3.29: Server bandwidth.

Figure 3.30: Percentage of server load.

forward the sub-requests associated to the fragments which have not available resources in the ISP network. This process goes on until all the networks are checked. In the case that there are not available resources in all the network to serve some layers, they will be served by the central server. As a result of this mechanism, the different layers of the video sessions can be provided by STBs located in different networks and by the central server, simultaneously.

Figures 3.30 and 3.31 represent the percentage of server load and the server bandwidth, respectively. In this case it is not presented the number of VOD sessions served by the P2P network, because with this mechanism some layers of a specific VOD session can be served by the P2P network and others by the central server at the same time. Regarding to the percentage of server load, if we compare Figure 3.30 with Figure 3.27, it is shown that this forwarding mechanism reduces the total percentage from 15 % to 10 %. In addition, it is also able to reduce the percentage associated to category 3 in a high order, from 72 % to 32 %. The reduction of the total percentage is expected because this strategy eases the demand of network resources in the different ISP networks. In this case it is not strictly necessary that all the fragments are served by STBs located in the same

Figure 3.31: Server bandwidth.

ISP network.

The only drawback is that it also increases a bit the percentage associated to categories 0, 1 and 2. Although we could think that the increase of the percentage associated to categories 1 and 2 is a bad result, it is not at all. The service architecture has to try to distribute the server load more equally among the different categories of titles. That is, it should avoid that the most popular titles use more resources than the rest of titles.

On the other hand, Figure 3.31 shows that this mechanism decreases the bandwidth consumed by the server, because now it is a bit lower than 3 Gbps. As a conclusion, it has been shown that this mechanism is able to reduce the percentage of server load associated to least popular contents in a high order. In addition, it also tries to distribute more equally the percentage of server load among the different categories of contents.

3.6.6 Dynamic re-allocation of streams

In this section we propose a new mechanism which tries to re-allocate the necessary connections in order to find available resources for a new VOD session which cannot be established in an ISP network. Each STB registers a new score called popularity of active connections (PAC) which represents the addition of the popularity of the titles which are being transmitted in the different uplink connections of the STB. When the SN cannot find a specific fragment of a VOD session in the network because all the STBs which have this fragment in their local storage do not have available uplink bandwidth, then it puts in order all the STBs with that fragment from the lowest PAC to the highest PAC. Now, the SN will try to change one of the connections of these STBs to another STB in the network, and by this way, the new VOD request will be able to establish the necessary connection for that segment.

The first candidate STB is the one with the lowest PAC. The active connections of the first candidate are put in order from the most popular to the least popular. Then, the SN finds out if it is possible to change the most popular connection to a new STB. If it is not possible, it will try to change the following connection, and it will go on by this way until it has checked all the active connections. If the SN cannot change any connection of the

3.6. VOD architecture for the Internet

Figure 3.32: Percentage of server load.

(a) Total.

(b) Category of contents.

Figure 3.33: Number of VOD sessions served by the P2P network.

first candidate STB, it will try with the following candidate STB. If the SN cannot find an alternative for any of the active connections of the candidate STBs, then the new VOD request will be forwarded to a different ISP network.

This new mechanism has been implemented and evaluated for the two VOD requests forwarding strategies previously presented.

3.6.6.1 Full requests forwarding

Figures 3.31, 3.32 and 3.33 represent the percentage of server load, the number of VOD sessions served by the P2P network and the server bandwidth, respectively, for the *full requests forwarding* method using dynamic re-allocation of streams and without using this mechanism. Figure 3.32 shows that the dynamic mechanism is able to reduce the total percentage of load at the server from 15 % to 10 %. It is possible because the percentage of load associated to the least popular contents (category 3) is reduced from 72 % to 40

Figure 3.34: Server bandwidth.

%. However, this mechanism also increases the percentage associated to categories 1 and 2. These results are very similar to the results provided by the *sub-requests forwarding* strategy, presented in Section 3.6.5.2. Therefore, the dynamic mechanism is highly recommended in the case that the *full requests forwarding* strategy is implemented in the network.

Regarding to the number of VOD sessions served by the P2P network represented in Figure 3.33, the dynamic mechanism increases the total number of sessions from 6,100 to 6,600. This increase corresponds with the increase of the number of VOD sessions associated to category 3 from 400 to 900. Finally, Figure 3.34 shows that the server bandwidth is reduced from 4 Gbps to 2,5 Gbps.

3.6.6.2 Sub-requests forwarding

Figures 3.35 and 3.36 represent the percentage of server load and the server bandwidth, respectively, for the *sub-requests forwarding* method using dynamic re-allocation of streams and without using this mechanism. It is clearly shown in both figures that the dynamic mechanism cannot improve the results obtained by the forwarding mechanism. It is due to the similarity between both of them. The dynamic algorithm tries to re-allocate the necessary connections in an ISP network in order to find available resources for a new VOD session which cannot be established. The final distribution of connections after applying the algorithm can be interpreted as a possible distribution of connections in a different ISP network. The only requirement is that it is necessary to find this specific ISP network.

3.6.7 Update of the content injection process

This section presents an algorithm for the update of the off-line content injection process. This process takes place periodically (e.g. once or twice a week) during the off-peak hours. It is possible that between two consecutive content injections the number of clients in some ISP networks changes. In addition, it is also possible that during that time the popularity associated to the different titles is modified. Both changes make that the optimal distribution of contents over the different STBs changes. However, if it is assumed

3.6. VOD architecture for the Internet

Figure 3.35: Percentage of server load.

Figure 3.36: Server bandwidth.

that the changes cause slight variations with regard to the previous state, it is possible to reallocate the contents in a quasi-optimal way, without having to solve the optimization problem again. By this way, the algorithm presented in this section tries to reduce the data traffic necessary to make possible the new content distribution. We are going to distinguish three different cases: (i) an increase of the number of STBs in an ISP network; (ii) a decrease of the number of STBs; and (iii) a change of the popularity associated to the contents. However, it will be necessary to solve the full optimization problem every certain long periods of time in order to avoid big deviations from the optimal distribution.

3.6.7.1 Increase of the number of STBs

When the number of STBs in an ISP network increases, the available storage in that network also increases. This increase in the available storage should be used to increase the number of copies associated to some contents. The mechanism to inject the new copies of contents to the new STBs is as follows.

First of all, it is assumed that the old STBs are not going to suffer any change in its

local storage. Before solving the optimization problem associated to the new number of STBs, the previous solution of this problem is fixed as a constant value for the variable $X(i, j, k)$ (that is, new restrictions of the problem are considered, and therefore it is simplified in a high order). After solving the simplified optimization problem, it is possible to calculate the new number of copies of contents to distribute as the difference between the new number of copies and the previous number of copies.

Next, the post-process algorithm is applied. In this case, it is only implemented for the new STBs and the new number of copies of contents that it is necessary to distribute in the network.

3.6.7.2 Decrease of the number of STBs

In this case, the total number of copies of contents distributed in an ISP network must be lower. Therefore, we are not interested in reallocating all the contents again ($X(i, j, k)$ variable), but in properly fit the number of copies stored in the active STB. First of all, it is necessary to get the new number of copies associated to the new number of STBs. Due to the fact that the optimization problem is solved in every ISP network in an independent way, this value can be obtained from a different ISP network with the same number of STBs.

After the leaving of some STBs it is possible that the number of copies associated to some contents (previously injected in the network) is higher or lower than its new appropriate value. In the first case, it is necessary to erase the excess copies of contents. To do that, the STBs with the excess content in its local storage are randomly chosen, and the excess copies are erased. The random procedure tries to assure that the fragments of any title are as spread as possible. As a consequence of this procedure, new available storage space appears in the network.

In the second case, the space liberated in the above procedure can be used to increase the number of copies of some contents to its appropriate value. In this case, a simplified version of the optimization problem is used to allocate the copies of contents in the STBs: First of all, the fragments stored in each STB after erasing the excess copies are established as a constant value for the variable $X(i, j, k)$ of the optimization problem. It makes that the optimization solver tries to distribute contents only in the available storage. After that, the post-process algorithm is applied. In this case, it is only implemented for the available storage in the STBs and the number of copies of contents that it is necessary to distribute.

3.6.7.3 Change of the popularity

As a consequence of the change of the popularity associated to the contents distributed in an ISP network, it is possible that the number of copies of some contents has to be increased, others have to be decreased, and some contents remain with the same number of copies. The algorithm applied in this situation can be interpreted as a modification of the algorithms presented in the previous case.

First of all, the appropriate number of copies associated to the new popularity distribution is obtained. After that, the contents which excess copies erased from some STBs randomly selected. The new available storage space is used to adjust the number of copies

3.7. Conclusions

of the different contents to its appropriate value. To do that, the optimization problem is established as in the previous subsection.

3.6.7.4 Other cases

It is also possible that between two consecutive content injections the number of clients and the popularity associated to the different titles are modified simultaneously. In this general scenario, the procedure would be as in the previous case. First of all, it is necessary to get the number of copies associated to each content in the new scenario. With the new values, it is possible that it is necessary to erase some copies of contents from the STBs. That process is performed erasing the contents from randomly selected STBs. Finally, all the available storage is used to distribute the new copies of contents configuring the optimization problem as in the previous cases.

3.7 Conclusions

The second generation of P2P applications used superpeer-based networks. In this case, the network architecture presents a hierarchical topological architecture. Clients are connected to a superpeer, and it is connected to a set of neighbor superpeers. This architecture was used in [Nafaa et al., 2008] to implement a P2P-based VOD service distribution architecture in a broadband operator context. In this architecture, the content is stored in the set-top boxes (STBs), which act as peers, and the sessions are provided by means of multi-source streaming. In this chapter, the previous work has been extended. First of all, the system has been evaluated in a broadband operator scenario, and it has been deduced that the existing mechanisms are not enough to fully take advantage of the network resources. The second part of the chapter has extended the previous VOD service distribution architecture to the Internet context.

Regarding to the broadband operator scenario, it has been firstly proposed a new popularity-to-availability translation model in order to optimally determine the necessary number of copies of any title in the content library. Afterwards, a new dispatching algorithm has been presented. The performance evaluation of the full system with the previous popularity-to-availability translation model and dispatching algorithm has been presented. In order to perform this evaluation, a very simple resource allocation strategy, which discriminates and rank the STBs based on their available uplink bandwidth, has been implemented. From this evaluation, it is deduced that the previous presented mechanisms are not enough to achieve the goal of increasing the number of simultaneous VOD sessions. Finally, it has been demonstrated that a necessary condition to maximize the number of simultaneous VOD sessions is to be able to uniformly distribute the rejection rate over the full library of titles, independently of their respective popularity. The dynamic re-allocation of streams strategy presented in this chapter is able to achieve the objective stated above, while it achieves the highest number of simultaneous VOD sessions serving in the network.

Regarding to the Internet scenario, it has been firstly proposed a new dispatching algorithm. The decision on what content is dispatched at which STB is determined by a Non-Linear Programming (NLP) formulation. Simulation results show that using the

formulation decision for dispatching can help in efficiently mitigate server load. Afterwards, two new mechanisms to forward the VOD requests between different ISP networks have been presented. It has been showed that both mechanisms are able to improve the global performance of the system. In addition, the *sub-requests forwarding* mechanism distributes more equally the percentage of server load among the different categories of contents. This is a desirable property, because the service provider should avoid that the most popular titles use more resources than the rest of titles. After that, the dynamic re-allocation of streams strategy has been implemented and evaluated for the two VOD requests forwarding strategies previously presented. When it is evaluated for the *full requests forwarding* strategy, it is shown that the percentage of server load is more equally distributed among the different categories of contents. On the other hand, it is also proved that this mechanism is not able to improve the results obtained by the *sub-requests forwarding* strategy. Therefore, we could say that the dynamic mechanism is highly advisable if the *full requests forwarding* strategy is implemented, but not very useful if the architecture uses the *sub-requests forwarding* strategy.

Design of self-organizing mechanisms using structured P2P networks

This chapter introduces the main characteristics of structured P2P networks and it describes Chord as a well-known example of this kind of networks. In general, this kind of networks offer a routing service to send a message, with an associated key, to a single node, responsible for that key. For this reason, this service is called Key-Based Routing (KBR). Over this kind of networks, the services offered by an Application Level Multicast (ALM) solution can be implemented. The implementation of this service is presented in this chapter. After that, it is presented OverSim, an overlay simulation tool built over OMNeT++. P2P networks show a set of distinctive features which increase the need to simulate the new proposals. For example, they involve a massive number of nodes which are prone to unpredictable disconnections. On the other hand, although P2P is beneficial when removing a centralized server, new mechanisms are required for network maintenance. Self-organization can be a good way to solve the control issues in the decentralized P2P architecture. Self-organization is a process where the organization of a system spontaneously increases without being managed by an outside source. In this chapter, it is presented a self-organizing mechanism for superpeer-based P2P applications, which makes use of the reliable routing functionality associated to structured P2P networks. The proposed mechanism is implemented in two superpeer-based P2P applications. The first one is a two-level P2P application, where all the superpeers are members of an ALM group. The second one is a hierarchical P2P application. This application is similar to the previous one, but in this case the superpeers have direct communication among them using a hierarchical architecture of superpeers. The anonymity mechanism presented in Chapter 2 has been also implemented in both applications.

4.1 Introduction

Structured overlays are developed to provide deterministic bounds on the data discovery [Dhara et al., 2010]. These networks are based on a distributed data structure that supports deterministic behavior for data lookup. They impose restrictions on node placement in the

overlay, and support *key-based routing* (KBR) such that object identifiers are mapped to the peer identifier address space and an object request is routed to the nearest peer in the peer address space. Peer-to-peer systems using key-based routing are frequently called *distributed object location and routing* (DOLR) systems [Buford et al., 2008].

This kind of P2P networks ensures the location of a desired content if it exists. This is especially important for some kinds of applications where participating users expect a high availability, although the replication cost is low. They also scale well with an increasing number of peers.

On the negative side, this type of systems does not support complex queries; they only support exact-match lookups. In addition, they hardly scale with increasing churn rates. As peers frequently join and leave the overlay network, signaling traffic necessary to keep the overlay structure up-to-date increases.

4.2 Chord

Chord is the first structured P2P network using a ring structure. Node identifiers are situated in a circle formed by 2^m values ($id \in [0, 2^m - 1]$). Distance definition between two nodes in the ring space is clock-wise. Therefore $d(n_1, n_2) \neq d(n_2, n_1)$, that is, distance in Chord is not symmetric.

Ideally, each node is responsible for only one key identifier. However, since the number of nodes in the system (N) is smaller than 2^m , each node is in fact the responsible for a set of key identifiers. In particular, a key identifier k is assigned to the node whose identifier is equal to k , or if it does not exist, to the first node clockwise k . That node is called the *key successor*.

This ring structure allows an extremely simple routing procedure. If nodes only know their successor (the next node in the ring space), the routing can be solved hop by hop. A node n searching for a key identifier k routes the request it to its successor. If the successor is the responsible for k , the routing procedure has finished. Otherwise, the above procedure is repeated at the successor until the key identifier is found.

This routing procedure can be improved to reduce the path length if a node maintains information about other non-consecutive nodes in the system. Let us suppose that node n knows its successor and a list of nodes ordered by distance $\{n_A, n_B, \dots, n_F, n_G\}$. The routing procedure performs as follows: If the distance between n and k is bigger than the distance between n and its further known node n_G , the node directly routes the query to n_G (then, the hop by hop path toward n_G has been avoided). Otherwise, n considers the previous node in the list n_F . If the key identifier is located between n_F and n_G , the request is sent to n_F (then, the hop by hop path toward n_F has been avoided). Otherwise the previous node in the list is used, and so on.

Chord implements the improved routing mechanism explained above. A node n maintains its successor and a list of extra nodes called *finger table*. The *finger table* is composed of m entries. The i -th table entry associates the key identifier $n + 2^i$ with its successor, where $0 \leq i < m$. It can be noticed that the first entry is the successor and each entry halves the ring space recursively (see Figure 4.1).

A Chord node also knows its *predecessor* and a *successors list*. The *predecessor* is its

4.3. Development of multicast solutions over structured P2P networks

Figure 4.1: Chord: Lookup procedure example. Node 8 is looking for the key-identifier 53 (k53).

counterclockwise node in the ring. It is used when a node joins and when a node departs. As we have seen above, the routing correctness relies on the fact that each node always knows its successor. To improve system reliability, a node maintains a list of r successors, that is, a list of the following r nodes in the ring. The first node in the *successor list* is the *successor*. All r successors would have to simultaneously fail in order to disrupt the Chord ring, an event that can be very improbable with modest values of r .

Therefore, a Chord system performs a halve-based recursive routing procedure using the *finger tables*. A node n routing a message towards a key identifier k , checks if k is located between itself and its successor. If so, the successor is the searched node. Otherwise, it checks its *finger table* from bottom to top to locate the immediate predecessor of key identifier k . Then, node n requests that node to deal with the routing. This procedure is recursively done by different nodes until successor node of key identifier k is found. Due to the halve-based *finger table* property, the number of nodes that must be contacted to find a successor of a given key identifier is $\theta(\log_2 N)$, where N is the current number of active nodes.

Figure 4.1 shows a routing example in a steady-state system, where $N_{max} = 2^6$. Node 8 initiates a routing procedure to locate key identifier 53. It locates the immediate predecessor of 53, node 42, in its *finger table*. That node analyzes its own table and sends the request to node 51. Finally, node 51 determines that its successor stores the searched key identifier.

4.3 Development of multicast solutions over structured P2P networks

In general, structured P2P networks offer a routing service to send a message, with an associated key, to a single node, responsible for that key. For this reason, this service is called *key-based routing* (KBR). In [Dabek et al., 2003] the authors developed a common API with the ability needed to interact with the KBR service offered by any structured

Figure 4.2: *joinGroup()* operation.

P2P network. These are as follows: *void route(key k, msg m)*, which forwards a message, *m*, towards the responsible key node *k* (it is implemented in the structured P2P network layer); *void forward(key k, msg m, nodehandle nexthopnode)*, which is invoked from the structured P2P network layer of each node that forwards message *m* during its routing (this function is implemented in the application layer); *void deliver(key k, msg m)*, which is invoked from the structured P2P network layer of the node that is responsible for key *k* upon the arrival of message *m* (this function is also implemented in the application layer).

Structured P2P networks can be used to implement Application Level Multicast (ALM) solutions. ALM solutions are classified in tree-based and flooding-based solutions [Manzanares-Lopez et al., 2010a]. In tree-based solutions (also called tree-per-group solutions), nodes from different multicast groups join the same overlay. Then, each multicast group forms a particular multicast tree built on top of the overlay network. On the other hand, in the flooding-based solutions (also called overlay-per-group solutions), all the members of a multicast group form a particular overlay. Therefore, the multicast transmission is transformed into a broadcast transmission to all the members of the overlay.

The multicast solution provides two operations: *joinGroup(groupId)*, and *sendDataToGroup(groupId, data)*, which are used to join and to send data to a specific group, respectively. These services can be implemented over the previously introduced KBR interface. However, its implementation depends on the kind of ALM solution. In the tree-based solutions each multicast group is associated to a key (the *groupId*). Thus, the node responsible of that key becomes the root (rendez-vous point) of the multicast group tree. Nodes join the group using the *joinGroup(groupId)* function. The implementation of *joinGroup()* routes a SUBSCRIBE message containing the local node's nodehandle (*S1*) to the group's key, using *route(groupId, [SUBSCRIBE, S1], NULL)*. When the *forward()* up-call is invoked at a node, the node checks if it is a member of the group. If so, it terminates the SUBSCRIBE message; otherwise, it inserts its nodehandle (*S2*) into the message and forwards the message towards the group key's root, using *route(groupId, [SUBSCRIBE, S2], NULL)*. In either case, it adds the nodehandle of the joining node (*S1*) to its list of children in the group multicast tree. On the other hand, the *sendDataToGroup()* operation routes a MCAST message using *route(groupId, [MCAST, data], NULL)* to the group key. The group key's root, upon receiving this message, forwards the message to its children

4.4. Simulation of P2P networks

Figure 4.3: *sendDataToGroup()* operation.

in the group's tree, and so on recursively. Figures 4.2 and 4.3 represent the *joinGroup()* and *sendDataToGroup()* operations for a tree-based multicast solution.

The implementation of the previous operations in a flooding-based solution is more simple. When a node wants to join a specific multicast group, it only has to join the specific overlay network in the common way. Once a node has joined to the multicast group, it can send a multicast message to the members of this group. That is done using the *sendDataToGroup()* operation. When the overlay network receives this request, it sends a multicast message to every node in its routing table. When the rest of nodes receive the multicast message, they will perform an equivalent operation.

4.4 Simulation of P2P networks

4.4.1 Introduction

The cost associated to the implementation of a network application is too large in most cases [Muñoz-Gea et al., 2008b, Muñoz-Gea et al., 2009b]. Therefore, to avoid a costly failure, it is recommended to simulate the new algorithms and specifications before. P2P networks show a set of distinctive features which increase the need to previously simulate the new proposals. First of all, these applications involve a massive number of nodes (1,000,000 or more), and they require to operate in large-scale environments, i.e. on the Internet. Therefore, scalability is one of the most important characteristics that researchers have to prove. Secondly, P2P nodes are usually home computers. Therefore, they are prone to unpredictable disconnections and consequently, developers have to ensure that the new applications can appropriately adapt to the node failures and the continuous joining and leaving of nodes inherent to the P2P nets.

After validating the new (under study) P2P algorithm, developers have the possibility to prove a real P2P application instance (natural programming) in an emulated environment. A network emulator is a program running in a set of computers (at least one) that appears to be a network. Fig. 4.4 graphically summarizes the above discussion.

Figure 4.4: Procedure to develop an error-free network application.

4.4.2 P2P simulation frameworks

There are several open-source P2P simulation frameworks available; next, the most relevant ones in the author's opinion are briefly described.

P2PSim¹ is a discrete event simulator for structured P2P networks written in C++. It was developed in the IRIS project at the Massachusetts Institute of Technology. There are a range of different underlying network models. However, it is largely undocumented and therefore hard to extend. Additionally, it has a limited set of statistics.

PeerSim² uses query-cycle or discrete-event techniques to simulate unstructured P2P networks. It is written in Java, and it was developed in the BISON project at the University of Bologna (Italy). The components can be implemented to gather statistical data and it can simulate up to 1,000,000 nodes. However, the underlying TCP/IP network is not modeled.

PlanetSim³ is a discrete event simulator written in Java. It was developed in the Planet project at the University of Rovira i Virgili (Spain). It has a detailed tutorial and it can simulate up to 100,000 nodes. Its design is thoroughly documented. However, it has a limited simulation of the underlying TCP/IP network.

OverSim⁴ is a new simulation framework based on the well-known discrete event simulator OMNeT++. It has been developed in the ScaleNet project at the University of Karlsruhe (Germany). It can simulate up to 100,000 nodes and its design is thoroughly documented. It implements three underlying network models: In the so called Simple model, data packets are sent directly from one overlay node to another by using a global routing table. The INET underlay model includes simulation models for all network layers. Finally, in the SingleHost underlay model each OverSim instance only emulates a single host, which can be connected to other instances over an existing real network; that is, overlay nodes are simulated but the MAC, IP, TCP/UDP layers work in a real scenario.

A more comprehensive survey of P2P simulation frameworks can be found in [Naicken et al., 2007]. Among all the available tools we have chosen OverSim because of its flexibility: Its modular design and the use of the Common API [Dabek et al., 2003] facilitate its extension with new protocols. In addition, its flexible underlying network scheme can be very useful during the application development.

¹<http://pdos.csail.mit.edu/p2psim>

²<http://peersim.sourceforge.net>

³<http://planet.urv.es/trac/planetsim>

⁴<http://www.oversim.org>

4.4. Simulation of P2P networks

Figure 4.5: OverSim layered structure.

4.4.3 OMNeT++ and OverSim

OMNeT++ is a well-known open-source network simulation engine and it is one of the most used simulation frameworks for the research community. Additionally, its coding procedure is well documented and it also offers a multiplicity of network simulation models. The development of a simulation model with OMNeT++ has the following steps: Description of the system structure using the NED language; implementation of the simple modules in C++; compilation; and configuration of the simulation.

OverSim is an overlay simulation tool built over OMNeT++ as simulation engine. The layered structure of this simulator appears in Fig. 4.5 [Baumgart et al., 2007]. OverSim includes all the facilities about underlay and overlay networks (and its primitives). Users only need to code the overlay application in the application layer. The communication between the overlay layer and the application layer uses the API described and well documented in [Dabek et al., 2003]. The set of functions defined in this API are the same independently of the overlay network selected. Therefore, the same application can be easily translated from one overlay system to another with a minimum effort. It should be noted that the application layer is at the same time divided into two different sublayers (Tier 1 and Tier 2). A simple application can be programmed at Tier 1 sublayer. However, more complex applications can be programmed at Tier 2, if they use the services provided by the application implemented at Tier 1.

To build a P2P simulation using OverSim it is necessary to implement the modules in the application layer and to define the messages which will be interchanged among them. Regarding to the modules implementation in C++ it is important to emphasize two features: (1) all the overlay protocols inherit from the BaseOverlay class; (2) all the tier 1 and 2 applications inherit from the BaseApp class. To create the messages, a file of type .msg must be created. Messages have the structure *message < name > {....}*. Within this structure, the different fields of this message are pointed out with the pattern type name. The available data types are: bool, char, short, int, long and double. It is also possible to use user defined types (a class declared, struct or typedef, in a C++ header)

in a message definition. To do this, they must be declared with the structure *cplusplus* $\{\{\dots\}\};$.

4.4.4 P2P modeling

In order to perform an adequate evaluation of a P2P application, it is necessary to program a realistic model of the system. Therefore, it is necessary to characterize the queries, the node connection inter-arrival time, the distribution of the session length, the files life-time, the files arrival rate and the number of peers receiving them. Next, the most recent models for the previous parameters are going to be described.

4.4.4.1 Queries characterization

An accurate characterization of peer query behavior is absolutely indispensable. This characterization includes the time until the first query, the query interarrival time and the query popularity. In [Klemm et al., 2004a] the authors provide a model that is based on passive measurement of a peer in the Gnutella P2P system over a period of 40 days.

Specifically, the time until the first query can be modeled by a bimodal distribution composed of a Weibull distribution for the body and a log-normal distribution for the tail. In the same way, the query interarrival time is modeled by a bimodal distribution composed of a log-normal distribution for the body and a Pareto distribution for the tail.

On the other hand, the query popularity follows a Zipf-like distribution. Zipf's law states that the size of the r th largest occurrence of an event is inversely proportional to its rank r :

$$f_r \approx 1/r^\alpha (r = 1, 2, 3, \dots, N) \quad (4.1)$$

where α is close to unity, and N is the number of distinct occurrences of an event. Under a Zipf-like distribution, the probability of requesting the i th most popular document is given by:

$$p_i = \frac{1}{K \cdot i^\alpha} \text{ where, } K = \sum_{j=1}^N 1/j^\alpha \quad (4.2)$$

4.4.4.2 Dynamics of peer participation

The dynamics of peer participation, or *churn*, are an inherent and critical property of P2P systems. A peer joins the system when a user starts the application, contributes with some resources while making use of the resources provided by others, and leaves the system when the user exits the application. This join-participate-leave cycle is defined as a *session*. Thousands of independent sessions creates the collective effect called *churn*.

The two fundamental properties of churn are the following: (i) the inter-arrival time between sessions and (ii) the session length. Several simulation and analytical studies have typically assumed both distribution to be exponential, though some studies have modeled the session length distribution as Pareto. In [Stutzbach and Rejaie, 2006] authors found other distributions for inter-arrival time and session length.

4.5. Background

Exponential distributions are typically used to model behavior resulting from a large number of independent events. However, peer arrivals are not completely independent. Users are less likely to be active during certain times of day (or during certain days of the week), and a surge of arrivals may occur when a link to a file appears on a popular website. In this scenario, Weibull distributions are a more flexible alternative to exponential distributions (in fact, exponential distributions are a special case of the Weibull distributions where the shape parameter is 1). Therefore, for inter-arrival times, a Weibull distribution provides a much better fit.

In the same way, session lengths do not follow the commonly-used exponential distribution. Several studies have reported that peer sessions lengths are heavy-tailed. However, neither the exponential nor heavy-tailed distributions proved consistent with recent observations. There are other two models that might provide a good fit: log-normal distributions and Weibull distributions. The reason is that while most sessions are short (minutes), some sessions are very long (days or weeks). This differs from exponential distributions, which exhibit values closer together in length, and heavy-tailed distributions, which have more pronounced extremes (years).

4.4.4.3 Files characterization

Characterizing available files among participants peers is valuable because it reveals the properties, distribution and heterogeneity of the contributed resources (i.e. storage space and available files) by individuals users in the system. The reference [Stutzbach et al., 2007] is the first study that has explored the dynamic characteristics of stored files in P2P systems. Among them, we can find the variations in the shared files by individual peers. There are two types of changes that may occur in the list of shared files at each peer: First, the user may add new files, either by downloading them from other peers or by manually adding them to the shared folder. Second, the user may remove files, by moving (or deleting) those files from the sharing folder. The total number of added and deleted files at a single peer is defined as the *degree of change*.

On the other hand, file replication can be modeled by a Zipf-like distribution. In [Klemm et al., 2004b] authors relate query popularity and file replication by characterizing the matching probability between queries and files. They showed that this matching probability is given by a probability function composed by a summation of an exponential and a normal distributions. The probability function depends on the popularity rank of the query, but it is independent of the replication rank of the file. Thus, the matching probability between queries and files is easily computable.

4.5 Background

P2P is beneficial when removing a centralized server. On the other hand, new mechanisms are required to compensate for the server, especially for network maintenance. However, owing to the lack of a centralized server, any attempts of additional control could be difficult to achieve in the distributed P2P architecture. In contrast, self-organization could be a good way to solve the control issues in the decentralized P2P architecture.

Figure 4.6: General architecture of the system.

Self-organization is a process where the organization of a system spontaneously increases without being managed by an outside source.

In Chapter 3, it was presented that superpeer-based P2P applications have a lot advantages. One of the most important is that users can use high level metadata (human readable form) to find a desired object. Furthermore, they take advantage of the heterogeneity of capabilities (e.g., bandwidth, processing power) across peers [Yang and Garcia-Molina, 2003]. However, they also have a very important drawback: a superpeer becomes a single point of failure for its clients, and a potential bottleneck. When the superpeer fails or simply leaves, all its clients become temporarily disconnected until they can find a new superpeer to connect to.

On the other hand, one of the main advantages of structured P2P networks is that they always ensure the routing of a message towards the node responsible of a specific key. Therefore, this reliability can be used to organize the superpeers of a P2P network in a way that almost all the network management tasks can be done automatically.

4.6 Two-level P2P system

4.6.1 Description of the system

In this system all the nodes are immersed in a structured overlay network (anyone of the previously mentioned types) [Muñoz-Gea et al., 2008e]. In addition, the nodes divide automatically into different subgroups, in a more or less uniform way, surrounding a superpeer. This node has the best performances in terms of CPU, bandwidth and reliability. When searching for a content, the user will send the search parameters to its superpeer, and it will return information about who has the contents in this subgroup.

All the superpeers of the network are members of a multicast group defined within the same structured network. By this way, if the search fails, the superpeer will send the request to the rest of superpeers in a multicast way. Fig. 4.6 describes the general architecture of the system.

4.6. Two-level P2P system

4.6.1.1 Initializing and joining the network

Every node needs to obtain a *NodeID*. In this work, this identifier is obtained applying a hash function (MD5 or SHA-1) to its MAC or IP address. In the same way each node also needs to obtain a *SubgroupID* that identifies the subgroup to which the node is going to belong. It is proposed to use a well-known server to obtain this identifier. Each subgroup will have a maximum number of nodes, and the nodes will be assigned by order to each one of the subgroups until completing their maximum capacity. When the existing subgroups are completed new subgroups will be created.

As it is usual in any structured network a node needs to know at least one address of another node in the overlay network. The previous server can also provide this information. Finally, the node will have to link to the P2P overlay network using the mechanism imposed by the structured network.

4.6.1.2 Joining a subgroup

Each node of the subgroup will be able to establish a TCP connection with its superpeer, and they will send their content lists to it. Each subgroup is identified by a *SubgroupID*. Initially, a node looks for its superpeer. To do this, it uses the structured network to locate the node whose *NodeID* fits with its *SubgroupID*. This node knows the IP address of the superpeer of its subgroup. The last step consists of transmitting this information to the requester node. Note that in this way the system builds a superpeer-based network by using an underground structured network. In addition, this last property allows to define the superpeer dynamically and to guarantee the stability of the network throughout time.

4.6.1.3 Management of the hierarchy

When the new node finds its superpeer, it notifies its resources of bandwidth and CPU. The superpeers control the nodes that are linked to their subgroups and they form an ordered list of future superpeer candidates: the longer a node remains connected (and the better resources it has), the better candidate it becomes. This list is transmitted to all the members of the subgroup, and when the superpeer fails, the first node in the list becomes its successor. Later, it must inform all subgroup members that this node is now the new superpeer. Also, it must modify this information in the node which *NodeID* fits with its *SubgroupID*.

4.6.1.4 Management of the superpeers

All the superpeers are members of a multicast group defined at the application level. When a node becomes a superpeer, it must be linked to this multicast group, in order to be able to spread the unsuccessful searches to the rest of subgroups. The proposal uses Chord-multicast. It is not necessary that the multicast process reaches all the group members before sending the searching results to the requester node. When one node responds affirmatively to a request it sends to the requester's superpeer the coincidences of the search in its database. Next, this superpeer gives back immediately the IP address and the

corresponding metadata to the requester node. Therefore, the requester node obtains the searching results as soon as possible.

4.6.1.5 Registering the shared files

In a similar way to FastTrack, when a node establishes a connection with its superpeer it sends the metadata of those files that it wants to share. This allows the superpeer to maintain a database including the identifiers of the files that all the nodes of the subgroup are sharing and the corresponding IP address of the node that contains them. The information sent by the node includes the name of the file, its size and its description.

4.6.1.6 Searching for a content

When a user wishes to make the search of certain content, his node sends a request on the TCP connection established with its superpeer. For each coincidence of the search in the database, the superpeer gives back the IP address and the corresponding metadata.

If the search fails, the user has the possibility of asking for to its superpeer that tries to contact with other superpeers. The identification of those nodes is simple, since all belong to the same application layer multicast group.

4.6.2 Keys of the system

Next, some of the contributions of the system proposed in this section are described. First, it is necessary to emphasize that all the nodes are assigned to a subgroup and not to a server. The nodes are able to automatically find the superpeer responsible for their subgroup.

It is also necessary to emphasize that this system is able to manage the heterogeneity of the network too. The most stable nodes and those with better benefits will become superpeers, which will increase the network performances.

On the other hand, the application layer multicast service provides an effective way to share information among superpeers. By this way, the maintenance of multicast group is practically made in an automatic mode. In addition, it guarantees that any content in the network can be located by any user.

Finally, the searches will be made in a simple way, similar to those made in current superpeer-based P2P file-sharing applications.

4.6.3 Providing anonymity

In this section, the anonymity mechanism proposed in Chapter 2 is integrated within the two-level P2P system [Muñoz-Gea et al., 2008c]. The solution defines three different stages. On the one hand, all peers publish their contents within their local subgroups (publishing). On the other hand, when a peer initiates a complex query, a searching phase will be carried out firstly (searching). Finally, once the requester peer knows all the matching results, the downloading phase will allow the download of the selected contents (downloading). In the following section these three stages are described in detail.

4.6. Two-level P2P system

4.6.3.1 Publishing the contents

To maintain the anonymity, our solution defines a routing table at each peer and makes use of the ADU algorithm to establish a random path from the content source to the subgroup's superpeer. When a content owner (C) wants to publish its contents, first of all it must choose randomly a connection identifier (great enough to avoid collisions). This value will be used each time this peer re-publishes a content or publishes a new one.

The *publish message* contains the connection identifier associated to C and the published content metadata. The content owner table entry associates the connection identifier and the next peer in the RW path. Each peer receiving a *publish message* follows the same procedure. It stores an entry in its table, which associates the connection identifier, the last peer in the RW path and the next node in the RW path.

Thanks to the publishing procedure, each superpeer knows the metadata of the contents that have been published into its subgroup. This publishing procedure offers sender anonymity: any peer receiving a *publish message* knows who is the previous node, but it does not know if this previous peer is the content owner or just a simple message forwarder.

The RW path distributely generated in the publishing phase will be used during the searching and downloading phases, as it will be described later. Therefore, it is necessary to keep updated the RW paths according to the non-voluntary peer departures. For this purpose, each peer maintains a *timeout timer*. When its timeout timer expires, each peer will check the RW path information stored in its table. For each entry, it will send a *ping message* to the previous and the following peer. If a peer detects a connection failure, it generates a RW failure notification that is issued peer by peer to the superpeer (or to the content owner) using the RW path. Each peer receiving a RW failure notification deletes the associate table entries (only one entry in intermediate peers and one entry for each published content in the subgroup leader peer). In addition, the content owner re-publishes all its contents. On the other hand, if an intermediate peer wants to leave the system in a voluntary way, it will get in touch with the previous and the following peer associated to each table entry in order to update their entries.

An additional verification is carried out each time a new content is going to be published. Before sending the *publish messages*, the content owner peer makes a *totalPing* (a ping sent to the subgroup's superpeer through the RW path). If the *totalPing* fails (any intermediate peer is disconnected), the content owner peer re-publishes all its contents initiating the creation of a new distributed RW path. The intermediate peers belonging to the failed RW path will update their table entries when their timeouts expire.

4.6.3.2 Searching the contents

The searching procedure to obtain requester anonymity is described in this section. Figure 4.7 describes this procedure. When a requester (R) wants to make a complex search, first of all it must select its connection identifier. If the peer already chose a connection identifier during a publish phase, this value is used. Otherwise, a connection identifier is chosen¹.

¹Each peer is associated to only a connection identifier.

Figure 4.7: Searching phase.

Then, the peer generates a new type of message called *lookup message*. A *lookup message*, that encapsulates the metadata of the searched content and the connection identifier associated to the requester peer (R), is forwarded towards the local superpeer using the RW path associated to this connection identifier (RW1). If the connection identifier is new, the RW1 path from the requester peer towards the subgroup leader is generated, distributely stored, and maintained as it was explained in the section before.

Once the *lookup message* has been received, the local subgroup's superpeer is able to locate the matching results within its subgroup. On the other hand, the system uses the ALM service to distribute the lookup messages to all the subgroup's superpeers.

Each superpeer with a positive matching result must answer the requester's superpeer with a *lookup-response message*. This message contains the connection identifier established by the requester peer, the matching content metadata, the connection identifier established by the content owner peer and the identity of the last non-superpeer peer in the RW path (RW2) towards the content owner (C).

In fact, before sending the *lookup-response message*, the subgroup leader will check the complete RW2 path towards the content owner by means of a *totalPing*. If the RW2 path fails, the superpeer will send to all the peers in the subgroup a broadcast message containing the associate connection identifier. The peer that established this connection will re-publish all its contents (that is, it initiates the creation of a new RW for the connection identifier). Peers that maintain a table entry associated to this connection identifier will delete it. The rest of peers will take no notice of the message. Consequently, the subgroup's superpeer only sends a *lookup-response message* to the requester subgroup's superpeer after checking that the associate RW2 path is active.

Finally, the *lookup-response message* will be forwarded through the requester subgroup towards the requester peer using the adequate and distributed RW1 path stored in the tables.

4.6.3.3 Downloading the contents

Once the requester peer receives the *lookup-response* messages, it is able to select which contents it wants to download. The anonymity is also important during this third phase: both the requester peer and the content owner peer should keep anonymous to the rest of peers in the system.

In the download phase (see Figure 4.8), the requester peer (R) generates a *download-request message* which initially encapsulates the required content metadata, the connec-

Figure 4.8: Download phase.

tion identifier established by itself, the connection identifier established by the content owner (C) and the identity of the last non-superpeer peer in the RW2 (in the content subgroup).

This message will be forwarded towards the superpeer using the distributely stored RW1 path. However, this message will not reach the superpeer. The last non-superpeer node in the RW1 path (the previous to the superpeer) will be the responsible for sending the *download-request message* to the last non-superpeer peer in the RW2 (in the content subgroup). Before sending the message to the content owner subgroup, this peer (the last non-leader peer in RW1) must encapsulate an additional value: its own identity. The message will be forwarded through the content owner subgroup using the distributely stored RW2 path until it reaches the content owner.

Finally, the content owner peer will start the content delivery. The *download messages* encapsulate the requited content, the connection identifier established by the requester peer, the connection identifier established by itself and the identity of the last non-leader peer in the RW1 (in the requester subgroup). These messages will be forwarded towards the subgroup leader using the adequate RW2 path. However, they will not reach the subgroup leader. The last non-superpeer node in the RW2 path will be the responsible for sending them to the node indicated in the message. Once in the requester subgroup, the messages will be forwarded using the RW1 path until they reach the requester peer.

4.6.3.4 Dynamics aspects of the system

The peer dynamism or *churn* is an inherent feature of the P2P networks. Our solution exploits this property to improve the anonymity behavior of the system. As it has been described in the previous section, each peer establishes a RW path towards its subgroup leader during the publishing or the searching phases. If there was no churn, these RW paths (which are randomly established) would not change as the time passes. However, due to the peer departures and failures, the RW paths must be continuously updated. Consequently, the RW path information that could be obtained by attackers will be obsolete as the time passes.

The random election of each RW path is another interesting feature of our proposal. The main objective of this procedure is to reach the mutual anonymity. However, there is another important consequence. The first non-leader peer (closest to the leader) in each RW path changes from a RW to another. Consequently, the remote peers involved in the download process change depending on the content owner peer and the requester

peer. As a consequence, the downloading load is distributed among all the peers. And something more important, the subgroup leaders don't participate during the download process making lighter their processing load.

As it has been described in the previous sections, the subgroup leaders have a key role during the publishing and the searching phases. Each subgroup leader maintains a list of the published contents in its subgroup that is used to solve the complex lookups. The subgroup leader management is defined by the network architecture. However, there is no doubt that the subgroup leader changes also affect the anonymous system. Each time a subgroup leader changes, all the RW paths would be obtained again. To optimize the system behavior when a subgroup leader changes we propose some actions. If a subgroup leader changes voluntarily (another peer in the subgroup has become a most suitable leader or the leader decides to leave the system), it will send to the new leader a copy of its table. On the other hand, to solve the non-voluntary leader leaving, we propose to replicate the table of the subgroup leader to the next two peers in the future leader candidate list. Consequently, when the subgroup leader fails, the new subgroup leader will be able to update its table taking into account the published contents.

4.6.4 Simulations

In this section the simulation results are presented. To better understand the performances, the results have been divided in two scenarios: In the first one, called static, the peers never die. In the second one, called dynamic, the peer birth and death processes are conveniently modeled.

The parameters of the network are: 2,500 peers, 25,000 files and 50 superpeers. The simulation only involves 2,500 peers in order to reduce the simulation time, although the simulation environment can easily reach 100,000 peers. At the end of the section, in order to characterize the model scalability, considerations about simulation time, computational and memory resource requests will be reported.

Regarding characterization of peer query behavior, we have followed the models presented in [Klemm et al., 2004a]. Therefore, the time until the first query is modeled by a bimodal distribution composed of a Weibull distribution for the body, with parameters $\alpha = 0.9821$ and $\lambda = 0.02662$ and a log-normal distribution for the tail, with parameters $\sigma = 2.359$ and $\mu = 6.301$.

In the same way, the query interarrival time is modeled by a bimodal distribution composed of a log-normal distribution for the body, with parameters $\sigma = 1.625$ and $\mu = 3.353$, and a Pareto distribution for the tail, with parameters $\alpha = 0.90411$ and $\beta = 103$.

On the other hand, the query popularity follows a Zipf-like distribution with $\alpha = 1$ and $N = 5$. This is because the available contents have been divided in 5 different popularities.

For inter-arrival times, a Weibull distribution with scale parameter $k = 0.62$ is used, as it was presented in [Stutzbach and Rejaie, 2006]. In the same way, the session length is modelled with a Weibull distribution with shape and scale parameters $k = 0.59$ and $\lambda = 41.9$. These parameters are introduced in a OverSim facility (ChurnGenerator) which manages the peers birth and death processes automatically.

4.6. Two-level P2P system

Figure 4.9: Query popularity.

Figure 4.10: Probability that the content is in the superpeer.

Finally, regarding of the dynamic characteristics of stored files, [Stutzbach et al., 2007] is the first study that has explored it. However, it has not been implemented in our simulation model. The reason is that the previous work depicts the evaluated CDF for this characteristic but it does not present an analytical model to represent it.

4.6.4.1 Static overlay network simulations

First, it is studied the P2P system performance with respect to query popularity. It is well known that the number of queries issued to request a specific file is directly connected to its popularity. The popularity assigned to every query follows the Zipf distribution, therefore it is reasonable that the number of queries issued during a simulation has somehow to follow the Zipf popularity distribution. These results are presented in Fig. 4.9, which shows (in logarithmic scale) that the number of queries performed in the simulation perfectly matches with a Zipf distribution.

Fig. 4.10 shows the probability that the requested content is in the same requester's subgroup. It is observed that this probability grows as the time increases, but converging

Figure 4.11: Number of visited superpeers.

Figure 4.12: Number of found sources.

to a value of one, which assures that our system reaches a steady state. This also indicates that with the typical (real) parameters that model the queries behavior the architecture assures that, in one month, the contents will be almost equally distributed among all the subgroups.

Fig. 4.11 reports the number of consulted superpeers for the queries that cannot be resolved in the requester's subgroup. In this case, a multicast process starts and tries to send the query to all the superpeers. However, when a superpeers has a reference to a peer which can resolve the query, it immediately stops the multicast process. As it can be seen, the number of consulted superpeers decreases when the number of queries increases, because the proposal is able to efficiently distributed the most popular contents among all the subgroups. This behavior can be viewed more precisely following the time behavior of the average value of this parameter which it is also represented.

Finally Fig. 4.12 represents, for each query, the number of found peers which have the requested content. For the first queries, the number of peers increases because the multicast procedure is frequently used. However, when the number of queries is high enough they are resolved in the requester's subgroup. The figure also represents the aver-

4.6. Two-level P2P system

Figure 4.13: Number of request that cannot be carry out in an hour because at least one node in the download path is out.

Figure 4.14: Average number of control messages in function of time.

age value, which shows that this parameters slowly increases and will stabilize around 50 peers because every subgroup is composed by this number of peers.

4.6.4.2 Dynamic overlay network simulations

In the results represented in Figure 4.13 the system does not implement the reliability mechanism. It shows the number of requests that cannot be carry out in an hour because, at least one node in the download path is down when the download is performed. This result is represented in function of time (in hours). This number fluctuates around 11,000.

Therefore, it is clear than in a real anonymous P2P network (with unpredictable node failures) it is necessary to supply a mechanism to reconstruct efficiently the paths, although it entails an extra network overhead in terms of control packets.

In the proposed architecture, a download process never fails since it is a reliable anonymous P2P network, but this feature involves an extra control traffic. Figure 4.14 represents the average number of control messages in function of time, in the previous scenario. Ini-

tially, due to simulation constraints, the *timeout timer* expires simultaneously in a lot of nodes, but in steady state the average number of control messages is under 40,000.

Number of nodes	Simul. sec. / Real sec.
500	112.861
1000	46.3345
1500	26.6125
2000	18.9614
3000	10.4441
5000	4.4518
10000	3.1518

Table 4.1: Simulation speed.

Number of nodes	Memory [Mb]
500	30.72
1000	51.2
1500	73.72
2000	96.25
3000	141.31
5000	229.37
10000	315.39

Table 4.2: Memory consumption.

Usually, control messages have a small payload (about 10 bytes) and a IP/TCP header (40 bytes). If we suppose each control message has a length of 50 bytes, control traffic supposes only a 4.4 Kbps traffic rate.

4.6.4.3 Simulator scalability

In this section it is presented the OverSim overall performance. Table 4.1 represents the ratio between the simulated seconds and the real seconds, for different number of nodes. For example, for 500 nodes in one real second OverSim is able to simulate 112.861 seconds. These results have been obtained using a Intel Core 2 architecture at 2.13 GHz with 2 GB of RAM memory. It is observed that the simulation speed decreases in a potential way when the number of nodes increases. This is due to the increment in the number of overlay connections among all the nodes in the network.

Table 4.2 represents the memory consumption as a function of the number of simulated nodes. In this case, the amount of memory increases linearly with the number of nodes.

4.7. Hierarchical P2P system

Figure 4.15: Proposed system architecture, with a three levels superpeer hierarchy.

4.7 Hierarchical P2P system

4.7.1 Description of the system

The system proposed in this section is similar to the previous one. In this case, every subgroup is managed by a level-1 superpeer, probably the node in the subgroup with the best performance in terms of CPU, bandwidth and reliability. Level-1 superpeers are also grouped into higher level subgroups. Each subgroup at level-2 is managed by level-2 superpeers. This process continues recursively up to the highest level subgroup. Fig. 4.15 represents the architecture of the proposed system [Muñoz-Gea et al., 2007b].

Each level-1 superpeer maintains a database where the content lists of all the nodes in the subgroup are stored. The superpeers of higher levels do not need to maintain this information, because it can be obtained recursively from the lower level superpeers. To search for a content, the user's node will send the search parameters to its level-1 superpeer, and then it will return the information about the nodes in that group which have the content. If the local search fails, the level-1 superpeer will send the request to the level-2 superpeer, which is responsible for the subgroup it belongs to at level-2. If it is necessary, this process will continue up to the highest level superpeer.

4.7.1.1 Initializing and joining the network

All the nodes in our system are immersed in a structured overlay network. Consequently, each node needs to obtain a node identifier *NodeId*. This value is obtained applying a well-known hash function (MD5 or SHA-1) to its MAC or IP address. Once a node knows its *NodeId*, it also needs to know an existing member in the structured overlay network who helps it to join. From now on, our nodes are supposed to use an underlying service to obtain that node identity.

The new node will use that existing node to join the structured network and also to obtain its *SubgroupId*; that is, the value that identifies the subgroup at level 1 to which the existing node is grouped into. The new node will use the *SubgroupId* value to try to join the same subgroup.

4.7.1.2 Joining a subgroup

To join a subgroup, a node must identify the subgroup's superpeer. The node will use the structured network to locate the node whose *NodeId* fits the *SubgroupId*. That node (the responsible node for the *SubgroupId*) maintains the IP address of the subgroup's superpeer (see Section 4.7.1.3). Then, the joining node will contact that superpeer and requests it to join the subgroup.

Each subgroup has a maximum number of nodes. Therefore, only if there is room, the joining node establishes a TCP connection to the superpeer and sends its content list to it. Otherwise, a new subgroup is created, as it is explained in Section 4.7.1.3.

4.7.1.3 Superpeer assignment

As it has been mentioned in Section 4.7.1.2, if a node tries to join a full subgroup, a new subgroup is created. In this case, a new (randomly generated) *SubgroupId* is assigned to the node by the superpeer, and it becomes the superpeer (and the only member) of the new subgroup. In order to be reachable by the rest of the community, the new level-1 superpeer uses the structured network to insert its own IP address in the node whose *NodeId* fits the new *SubgroupId*.

Every superpeer only generates an additional *SubgroupId*. This additional *SubgroupId* must be kept by the superpeer to answer future requests. Therefore, any joining request will be retransmitted from one subgroup to another until an incomplete subgroup (or without any additional subgroup identifier) is reached.

Just when the second level-1 subgroup is created, it also becomes necessary to create a level-2 subgroup. The superpeer of the first level-1 subgroup becomes the superpeer of the new level-2 subgroup. To maintain network integrity, the *SubgroupId* of the level-2 superpeer must be transmitted to the superpeers responsible for the other level-1 subgroups. Each level-*i* subgroup ($i > 1$) will also have a maximum number of nodes. At each level, the subgroup creation process will continue in a similar way up to the highest level of the hierarchy. Let us recall that, with this algorithm, only the set of level-1 superpeers is candidate for being a superpeer in the rest of the levels.

4.7.1.4 Superpeers maintenance

When a new node finds out its superpeer at any level, it notifies to it its available resources of bandwidth, memory and CPU. Consequently, the superpeer is able to control the nodes that are linked to its subgroup and it is able to maintain an ordered list of future superpeer candidates.

In order to supply more security to our system against centralized attacks and also to minimize the harms of spiteful superpeers, every superpeer node only plays this role for a fixed period of time. Then, it becomes a regular node and the next node in the candidate list becomes the new superpeer. The new superpeer needs to know who the superpeer of the highest level it belongs to is, in order to be able to route lookup messages. This information can be obtained if the node knows the *SubgroupId* of the highest level it belongs to. Finally, it is important to point out that, due to system dynamism, any node can become the superpeer of any level, and therefore, it must know the *SubgroupId* of any

4.7. Hierarchical P2P system

subgroup between itself and the highest superpeer. In our example, node 13 must know the *SubgroupId* of the subgroup formed by 13, 14 and 15, the *SubgroupId* of the subgroup formed by 11, 14 and 18, and the *SubgroupId* of the subgroup formed by 5, 18 and 20. In our proposal, the *SubgroupId* information is adequately propagated by using a limited broadcast mechanism.

If a node cannot contact with its superpeer, it is probably because the superpeer is turned-off. To check it, the node obtains the IP address of the superpeer again, asking to the responsible node for that *SubgroupId*. If the IP address still corresponds to the IP address of the failed superpeer, the node sends its own IP address to the node responsible for that *SubgroupId*, and becomes the new superpeer. Thereby, when the rest of the nodes detect that the previous superpeer is off, they will be able to obtain the IP address of the new superpeer. This mechanism provides reliability in front of unexpected shutdowns.

Finally, when a node becomes a level-1 superpeer it must obtain the subgroup content database. To solve level-1 superpeer failures, the content database could be send to several superpeer candidates. Thus, when a level-1 superpeer becomes a regular node, it could transfer the (minor) database changes using rsync¹ technology in a reasonable time.

4.7.1.5 Searching for a content

As it was discussed above, to search for a content, the user's node will send the search parameters to its level-1 superpeer. This superpeer will return information about who has the content. If this local search fails, the level-1 superpeer will send the request to its associated upper level superpeer. If it is necessary, this process will continue up to the highest level superpeer. The level- i ($i > 1$) superpeers do not maintain content databases. Therefore, they just send the lookup message to their subgroup's members. When a level-1 superpeer that maintains the requested content information in its database is reached, the lookup process finishes. It should be noted that in our system, any content is always reachable even if it is not a popular content. Moreover, our proposal distributes the most popular contents around the network, what provides the traditional robustness against failures.

4.7.2 Providing anonymity

In this section, the anonymity mechanism proposed in Chapter 2 is integrated within the hierarchical P2P system [Muñoz-Gea et al., 2007a]. The solution is similar to the presented one for the previous architecture. Figure 4.16 represents the searching and downloading phases.

4.7.3 Reputation mechanism

In order to provide reliability to stand up to internal attacks, the system offers a reputation mechanism able to locate and knockout the malicious nodes. Generally, this problem is addressed collecting information about the malicious nodes in a database [Xiong and

¹<http://samba.org/rsync>

Figure 4.16: Providing anonymity.

Liu, 2004]. This proposal manages this database in the same way as content databases. Therefore, the information about malicious nodes is always reachable by any node.

If a user downloads a corrupt content from a node, it informs to its corresponding level-1 superpeer about this fact, sending a special message (vote) containing the *NodeId* of that node. Every level-1 superpeer maintains a database that registers the *NodeIds* and the associated number of votes. When the number of votes exceeds a predefined threshold, the level-1 superpeer considers that *NodeId* as a malicious node.

When a user searches for a content and receives the associated *NodeIds* (more than one if chunking is enabled), it asks to its superpeer if the *NodeIds* are malicious. If the level-1 superpeer does not consider those nodes as malicious, it will ask its level-2 superpeer. This process continues up to the highest level superpeer, which can ask all the members of the system about the malicious nodes. Finally, the user chooses if the download is performed or not.

The superpeers periodically eliminate the votes registered in the previous database, and reconsider all the nodes as non malicious. Thereby, a node that has been erroneously described as a malicious node can restore its initial state. This simple solution is, in fact, a trade-off solution between protocol efficiency and complexity.

4.7.4 Simulations

The simulation scenario is formed by 12,500 different contents uniformly distributed among 6,250 nodes. The subgroups sizes are set to 50 and 5 nodes for the first and the rest of the hierarchies, respectively. Therefore, there are 125 level-1 superpeers, 25 level-2 superpeers, 5 level-3 superpeers and one level-4 superpeer.

Fig. 4.17 shows the time evolution of the average number of superpeers consulted until a content is located. It is observed that the number of consulted superpeers quickly decreases, converging to unit in a short period of time (indicating that the content is in the same requester's subgroup). Obviously, this variable is strongly related to the probability that a content is in the same requester's subgroup. When this probability is close to one, the average number of consulted superpeers is also close to one. On the other hand, if the probability is very small, then the average number of consulted superpeers is close to the higher level (4 in our case). It has been developed an analytical study that establishes the

4.7. Hierarchical P2P system

Figure 4.17: Time evolution of the average number of superpeers consulted until content location.

Figure 4.18: Simulation and theoretical values of the average number of consulted superpeers until content location.

relation between these pairs of parameters.

p is defined as the probability that a content is in the same requester's subgroup. It is supposed that the network has L different levels. It is also supposed that, irrespective of the number of nodes at the first level, there are exactly N nodes per subgroup in the rest of the levels. S is defined as the average numbers of consulted superpeers until a content is located, then

$$S = \sum_{n=1}^L nP(n) \quad (4.3)$$

where $P(n)$ is the probability of consulting n superpeers. It can be defined as,

$$P(n=1) = p \quad P(n=L) = (1-p)^x \quad (4.4)$$

$$P(n) = (1 - p)^x \sum_{j=1}^y \binom{y}{j} p^j (1 - p)^{y-j} \quad 1 < n < L \quad (4.5)$$

where x and y are defined as

$$x = N^{n-2} \quad y = (N - 1)x \quad (4.6)$$

Note that the equation 4.5 can be simplified using the binomial theorem since,

$$P(n) = \sum_{j=0}^y \binom{y}{j} p^j (1 - p)^{y-j} = 1 \quad (4.7)$$

and therefore,

$$P(n) = (1 - p)^x [1 - (1 - p)^y] \quad 1 < n < L \quad (4.8)$$

Finally, replacing in the above equations the values used in the simulations,

$$P(1) = p \quad P(4) = (1 - p)^{25} \quad (4.9)$$

$$P(2) = (1 - p)[1 - (1 - p)^4] \quad P(3) = (1 - p)^5[1 - (1 - p)^{20}] \quad (4.10)$$

Figure 4.18 represents the theoretical and the simulated values of the average number of superpeers consulted until a content is located. Observing the picture, it is clear that the theoretical study perfectly adjusts to the simulation results. It is also important to see that when p tends to unit (which is the expected value for popular contents), the number of superpeers involved in the searching process (S) also tends to unit. This result analytically demonstrates that the most popular contents are -sooner or later- widely spread around the network.

4.8 Conclusions

Structured overlays are developed to provide deterministic bounds on the data discovery. This kind of networks offer a routing service to send a message, with an associated key, to a single node, responsible for that key. For this reason, this service is called Key-Based Routing (KBR). In general, decentralized P2P architectures require some kind of network maintenance, since centralized servers have been removed. Self-organization can be a good way to solve these control issues. It is a process in which the organization of a system spontaneously increases without being managed by an outside source.

In this chapter, it has been presented a self-organizing mechanism for superpeer-based P2P applications, which makes use of the reliable routing functionality associated to structured P2P networks. The proposed mechanism has been implemented in two superpeer-based P2P applications. The first one is a two-level P2P application, where all the superpeers are members of an ALM group. The second one is a hierarchical P2P application. This application is similar to the previous one, but in this case the superpeers have direct communication among them using a hierarchical architecture of superpeers. In addition, the anonymity mechanism presented in Chapter 2 has been integrated in both P2P applications.

Design of traceability systems using structured P2P networks

This chapter firstly highlights the main characteristics of Distributed Hash Table (DHT) storage systems. This kind of applications are implemented over the KBR interface offered by a structured P2P network. Next, it is presented Pastry, as a well-known example of structured P2P network, and the implementation of the services offered by a DHT application. Radio Frequency IDentification (RFID) technology can track the movement of products throughout an entire supply network by giving a Globally Unique Product Identifier (GUPI) to every product. Suppliers, manufacturers, distributors and retailers are typically interconnected within networks and it is for this reason that the relationships among all them are represented as supply networks. Having enough information about the supply network in which an organization is involved can be useful in planning strategies or in assuring product quality. In this chapter, it is proposed a mechanism for automatically obtaining the supply network associated to a specific product using the EPCglobal Network. This mechanism is integrated within the Discovery Services (DS) component, which is currently being developed. Recently, there have been several proposals for the implementation of the DS based on centralized mechanisms. In this chapter, it is developed a DS prototype based on DHT and an access control service to be integrated within the DS prototype. Finally, the mechanism to reconstruct supply networks is integrated within the DS prototype.

5.1 Introduction

A distributed hash table (DHT) is a reliable, scalable, wide-area data storage system that frees programmers from many of the complications of building a distributed system. DHTs store blocks of data on hundreds or thousands of machines connected to the Internet, replicate the data for reliability, and quickly locate data despite running over high-latency, wide-area links. The DHT addresses problems of locating data and replicating it for reliability, which are common to many distributed systems, without additional work by the application. The DHT provides a generic interface, which makes it easy for

wide-variety of applications to adopt DHTs as a storage substrate: put stores data in the system under a key; get retrieves the data.

A DHT utility is attractive for several reasons. First, it exploits the multitude and diversity of nodes in the Internet to achieve strong persistence and high availability. This obviates the need for physical transport of storage media to protect backup and archival data; likewise, it obviates the need for explicit mirroring to ensure high availability and throughput for shared data. A global storage utility also facilitates the sharing of storage and bandwidth, thus permitting a group of nodes to jointly store or publish content that would exceed the capacity or bandwidth of any individual node.

However, DHTs alone are not designed for supporting consistently unstable content, with a high rate of modifications, as it is the case with some kind of applications (for example a wiki). In these cases, the system stores in each DHT node *operations*, more precisely the list of changes that produce the current version of the data. It is safe to consider these changes as the usual static data stored in DHTs, given that an operation is stored in a node independently of other operations, and no actual modifications are performed on it.

5.2 Pastry

The node identifiers and the routing procedure in Pastry depends on the base used to represent both node and content key identifiers. For example, if a hexadecimal based is used and the system requires five digits to represent the full node space, an example of a node identifier could be *65A1F*.

Each node maintains a routing table formed by so many rows as digits needed to represent the full node space and so many columns as possible digit values (in our example each routing table is formed by 5 rows and 16 columns). Row 0 maintains the IP addresses corresponding to nodes without common prefix with the local node identifier. Row 1 contains the IP addresses corresponding to nodes with a 1 digit-long prefix common with the local node identifier, and so on. As we have seen, a row is formed by some columns. The *i*-th column of a row stores, if exists, the IP address of a node whose identifier is determined by the associated row prefix followed by the *i* digit. Notice that there could be more than one node that satisfies the requirements of an entry. If so, the Pastry implementation will use some predefined criterion to select an entry (for example, the node with lowest physical delay).

In our example, row 0 stores the IP addresses of nodes without a common prefix with *65A1F*. The first entry at row 0 stores the IP address of a node whose identifier is *0***** (if exists), the second entry stores the IP address of a node whose identifier is *1***** (if exists), and so on. Row 1 stores the IP addresses of 16 nodes at most, with one digit-length common prefix: the first entry stores the IP address of a node whose identifier is *60**** (if exists), the second entry stores the IP address of a node whose identifier is *61****, and so on. It must be emphasized that if the local node does not know a node adequate to fill a concrete entry in the routing table, that entry is kept empty.

A node searching for a key-identifier elects from its routing table the node matching the longest prefix with the key-identifier. The *lookup* message is sent to that node, and then

5.3. Development of DHT applications over structured P2P networks

Figure 5.1: *insert()* operation.

it repeats the same algorithm until the searched node is found. During this process, a required entry in the routing table could be empty. In this case, the node uses the non-empty entry that is numerically closest. The routing table definition assures the convergence of the lookup algorithm, knowing that the responsible for a key-identifier is the node with the same key-identifier, if it exists, or the numerically closest node.

In our example, if the $65A1F$ node is searching for the key-identifier $654B2$, the longest common prefix in the routing table is $654 * *$ (that is, entry associated to row 2 and column 5). The local node uses the IP address stored in this entry (or the numerically closest node if the entry is empty) to forward the *lookup* message.

5.3 Development of DHT applications over structured P2P networks

Over the KBR interface (in the application layer), the services offered by a Distributed Hash Table (DHT) application can be implemented. The DHT abstraction provides the same functionality as a traditional hash table, by storing the mapping between a key and a value. That is, it implements a simple store and retrieve functionality, where the value is always stored at the overlay node to which the key is mapped by the KBR layer. This application provides two operations: *insert(key, value)*, and *value = lookup(key)*. A simple implementation of *insert()* routes a PUT message containing *value* and the local node's nodehandle (*S*), using *route(key, [PUT,value,S], NULL)*. The node responsible of that key, upon receiving the message, stores the (key, value) pair in its local storage. On the other hand, the lookup operation routes a GET message using *route(key, [GET,S], NULL)* to the node responsible of that key, and it returns the associated value in a single hop using *route(NULL, [value], S)*. The single hop routing option uses the *nodehandle* of the source node (*S*) to send the message directly to that node, without hoping along other nodes in the network. Figures 5.1 and 5.2 represent the *insert()* and *lookup()* operation.

Figure 5.2: *lookup()* operation.

5.4 Radio frequency identification

Radio Frequency Identification (RFID) technology can track the movement of products throughout an entire supply network by giving a Globally Unique Product Identifier (GUPI) [Främling et al., 2007] to every product. The identification is used to connect the physical object to an information service run by a host on the Internet. It is there that all the information associated to that physical object is stored. Nowadays, several approaches are used to implement this connection, such as the EPCglobal Network [Armenio et al., 2009] developed by the Auto-ID consortium, the Dialog system [Främling et al., 2006] developed at the Helsinki University of Technology and the World Wide Article Information (WWAI) system¹ proposed by Trackway. The EPCglobal Network stands out among the rest because in 2003 it was authorized as a Global Standards I (GS1). The GS1 system of standards is the most widely-used supply-network standards system in the world, the traditional barcode being its most widely used standard.

5.4.1 Product identifier proposals

The DIALOG system was developed at the Helsinki University of Technology. In this approach an ID@URI notation is used to create a GUPI, where the ID part identifies the product item located at the URI. If the URI is an URL, it is a straightforward task to link it to an information service. For an ID@URI to be a GUPI, the ID part should be unique for the corresponding URI. In the DIALOG system every product is implemented as a software agent [Nwana, 1996], and the information of each of them is accessed and updated through methods in the product agent interface [Främling et al., 2006]. These interface methods are as follows: *update()* and *getProductInformation()*, which are used to append and retrieve information, respectively, and *getCompositeInformation()*, which relates to managing component hierarchies. The information about a product item can be obtained and updated in two ways: by means of direct information or with a link to information. In the direct information option the product information is sent directly as a XML message, while the link to information option represents an URL where the

¹<http://www.trackway.eu>

5.4. Radio frequency identification

Figure 5.3: Product information lookup with DIALOG.

information can be accessed. It would normally be the address of an existing web page or service.

The World Wide Article Information (WWAI) protocol developed by Trackway (formerly known as Stockway) is based on P2P principles. The manufacturers form a network of nodes which are identified by company numbers. When a node has joined the network, it can autonomously issue identifiers for individual products, the product GUPI consisting of a concatenation of the company prefix and item-specific suffix. The service provided by the WWAI network is the mapping of the WWAI identifiers to the URL of the information provider, using a DHT. A lookup is only needed when a product identifier for a given company is seen for the first time. After that, URLs of known nodes are cached so that new node lookups do not need to be performed unless the cached address fails or changes for some reason.

The common characteristic of the two previous architectures is that all the data related to a specific product is available in only one information service. This means that every organization with information about a specific product has to publish it in the corresponding information service and then, the queries have to be addressed to a specific information service depending on the specific GUPI. On the other hand, one of the fundamental design principles for the EPCglobal Network is that each company retains control over the data that they collect or generate within their own organization, i.e. information about a specific product is decentralized across multiple organizations.

EPCglobal is a not-for-profit joint venture set up by the Uniform Code Council, Inc.¹ and EAN International², the bar code standards body in Europe, to develop and oversee the commercialization and technical standards for the EPCglobal Network. This framework is based on research conducted by the Auto-ID Center at the Massachusetts Institute of Technology (MIT). It is expected that this architecture will be a worldwide standard for supply-network management. At this respect, the European Union has recently performed

¹<http://www.gs1us.org>

²<http://www.gs1.org>

Electronic Product Code (96-bit Version)			
02	. 0000A79	. 00013D	. 000154ECD
Header	General Manager Number	Object Class	Serial Number
8 bits	28 bits	24 bits	36 bits

Figure 5.4: EPC tag data structure.

several projects, such as the BRIDGE project¹, to research, develop and implement tools to enable the deployment of EPCglobal Network applications.

The EPC² serves the role of a GUPI in the EPCglobal Network. There are two different lengths of EPC tag data that have been ratified by the EPCglobal board: 64 bits and 96 bits. Figure 5.4 shows the EPC tag data structure of one 96-bit version: Header identifies the version of EPC itself; General Manager Number identifies an organization that maintains the numbers in the field of Object Class and Serial Number; Object Class refers to a unique type of products produced by an EPC general manager; Serial Number uniquely identifies each item within an object class. The EPC serial number allows individual items tracking, which is the key feature and advantage that distinguish EPC from a bar code standard. This feature enables RFID technology to capture the information about the movement of individual products in supply networks. Next section explains the rest of components of the EPCglobal Network architecture in more depth.

5.4.2 EPCglobal Network

One of the fundamental design principles for the EPCglobal Network is that each company should be able to keep control of the data that they collect or generate about a specific product within their own organization, i.e. information is decentralized across multiple organizations. With a suitable service-oriented architecture the EPC can be used both to locate a source of information, via lookup services, as well as for extracting relevant information about a particular product from each source, by using the EPC as a lookup key within a database.

The EPCglobal Network architecture includes specifications that deal with the collection of captured data and their distribution across organizations. However, the initial standards development has focused more on the EPCglobal Network's lower levels than on data exchange across supply networks. Figure 5.5 represents the EPCglobal Network architecture framework.

The reader protocol (RP) is an API that abstracts from the underlying RFID hardware readers. RP is complemented by the reader management (RM) protocol, which facilitates the naming of readers and locations. In this context, it is necessary to take into account the fact that the enormous quantity of data generated by even a few readers can easily result in an unacceptable load. For this reason, an additional middleware layer is needed that filters the data collected from readers. In the EPCglobal Network, this task is supported by the application-level events (ALE) interface. ALE allows for the conversion of a series

¹<http://www.bridge-project.eu>

²http://www.epcglobalinc.org/standards/tds/tds_1_4-standard-20080611.pdf

5.4. Radio frequency identification

Figure 5.5: EPCglobal Network Architecture.

of tag reads into a single event message with a time interval attribute. In addition, the ALE standard also supports the declaration of logical readers that collect the data streams from multiple physical readers.

RFID readers are distributed across factories, warehouses and stores. In this context, EPCIS provide a repository of historical events and related information, and they are fed by an EPCIS capturing application, such as a management system connected to an ALE middleware. In addition, EPCIS can be queried by an EPCIS accessing application. EPCISs register four different event types. An *ObjectEvent* corresponds to the detection of one or more individual items. Because the granularity of individual EPC identification isn't always needed, a *QuantityEvent* can alternatively be used, which includes only information on the product type and the number of objects. *AggregationEvents* denote a (dis-)aggregation of a group of items, such as cases that are put on a pallet for shipping. In a similar way, a *TransactionEvent* links EPCs to a specific business transaction, such as a purchase order. To enrich events with their business context, the EPCIS repository also includes master data such as the names of business locations, process steps and transactions. Figure 5.6 represents the data associated to every event type.

To find EPC-related information, business applications use the Object Name Service (ONS) to provide an EPCIS URL when queried with a tag's EPC. The granularity of ONS resolution is currently limited to product type, i.e. an ONS is not expected to retain distinct records for two objects of the same product type. Another point to note is that ONS

		Object Event	Aggregation Event	Quantity Event	Transaction Event	
When?	eventTime	●	●	●	●	Time of event observation
	recordTime	○	○	○	○	Time of event registration
	eventTimezone	●	●	●	●	Time zone information
What?	epcList	●			●	List of observed EPCs
	parentID		○		○	Containing object ID
	childEPCs		●			Contained objects IDs
	epcClass			●		Product type
	quantity			●		Number of objects
	action	●	●		●	Phase of the EPCs
Where?	readPoint	○	○	○	○	Reader name
	bizLocation	○	○	○	○	Location name
Why?	bizStep	○	○	○	○	Business process step
	disposition	○	○	○	○	State of the objects
	bizTransactionList	○	○	○	●	Associated transactions

● = mandatory, ○ = optional

Figure 5.6: EPCIS event types and attributes.

is currently implemented using the Domain Name System (DNS) [Mockapetris, 1987], using Type 35 Naming Authority Pointer (NAPTR) [Mealling and Daniel, 2000] records to return the information. Queries to ONS are therefore performed by means of a DNS query for a hostname derived from an EPC. However, the ONS is allowed only to point to the manufacturer's EPCIS repository. For this reason, the EPCglobal Network is being extended to include Discovery Services (DS)¹. These services will allow applications to find third parties' EPCIS repositories with events related to a specific EPC. The following are some of the requirements of the DS: they will need to be designed to accept updates in near real time from multiple providers of information; they will need to be sufficiently scalable to store large volumes of data; and they will also need to provide for authentication of both information providers and those making queries and accept and enforce access control policies. A critical component of the DS is the data storage component, which stores information about the list of EPCIS instances that have information about a particular EPC. There currently exist several options to implement this component, although two in particular are worthy of note, LDAP (Lightweight Directory Access Protocol) and DHT.

5.5 Background

Suppliers, manufacturers, distributors and retailers are typically interconnected within networks and it is for this reason that the relationships among them are represented as supply networks [Wareham et al., 2005],[Phillips et al., 2006],[Poulin et al., 2006],[Li and Chandra, 2007]. In this respect, supply chains are special types of supply networks in which organizations are organized in linear chains. However, many organizations do not have sufficient information about the full supply networks in which they are involved, for example, they do not know who supplies their suppliers because they are not directly

¹<http://www.epcglobalinc.org/standards/discovery>

5.5. Background

connected to them. This information would be very useful in planning strategies or in assuring product quality. For example, in order to solve some problems of delayed shipments from suppliers, the organization might need to analyze its full supply network to identify which supplier is most to blame for the delay. Unfortunately however, many organizations do not have access to this information.

In this work, we propose a mechanism for automatically obtaining the supply network associated to a specific product using the EPCglobal Network [Muñoz-Gea et al., 2010a, Manzanares-Lopez et al., 2011]. In [Bi and Lin, 2009] the authors proposed a methodology with the same objective but the main difference with the proposal set out in this article is that in [Bi and Lin, 2009] the client has to do all the operations to reconstruct the supply network. That is, initially, it has to obtain the URLs (Uniform Resource Locator) of the information services with information about a specific product, then it has to access the corresponding information services to get the necessary information, and after that it can reconstruct the full supply network. However, in our proposal the client does not have to perform any operations, that is, it queries the supply network associated to a specific product and the results are obtained directly.

Recently, there have been several proposals for the implementation of the DS, like the DS prototype of the Bridge project¹, or the Extensible Supply-chain Discovery Service (ESDS) developed by Afiliis [Young, 2008]. In both the DS has been implemented as a centralized database, to be more specific in the Bridge Project it has been developed as a centralized database based on LDAP (Lightweight Directory Access Protocol) [Zeilenga, 2006]. In this work, we are going to develop a DS prototype based on DHT [Balakrishnan et al., 2003].

LDAP is a networking protocol for querying and modifying directory services running over TCP/IP. In this context, a directory is a set of information with similar attributes organized in a hierarchical manner. LDAP could be thought of as type of database, different from a relational database. Databases are usually designed to perform many changes to their data whereas LDAP directories are optimized to read performance, as such, it is particularly useful for storing information that needs to read from many locations. One of the most important characteristics of LDAP servers is that they can replicate some or all of their data to increase security or efficiency. Each LDAP server can hold a sub tree of the hierarchy starting from a specific entry and they can store references to other servers too. In this way a query to a server can result in a reference to another server or the server can even contact the other server and return its results to the client. On the other hand, LDAP permits secure delegation of reading and modification authority based on specific needs using ACLs (Access Control Lists); although this is not part of the LDAP protocol, many implementations offer this feature. The integration of the DS records into the data structure of the LDAP is based on the decomposition of the EPC tag into the LDAP tree. That is, fields such as Company Prefix, Item Reference or Serial Number can be used to distribute the EPC among the tree structure. In addition, security can be integrated with the fine grained access control policies of the LDAP implementation and can be used to limit the access to a particular record.

On the other hand, DHT are decentralized distributed systems that distribute the management of a key table between the participating nodes. Each node maintains a routing

¹<http://www.bridge-project.eu>

table with a list of its neighbors, so that it can route messages to the unique owner of a given key. DHT is designed to scale to a large number of nodes and is prepared to deal with continuous node arrivals and failures by constructing a structured overlay network. A 160-bit key is usually used that can be obtained from a SHA1 hash of more complex data such as filenames. The typical operations are *put(key, data)* to store data into the DHT and *get(key)* to recover data. These operations can be sent to any node of the DHT structure and it will propagate that operation to the appropriate node. One of the characteristics of this model is its decentralization, meaning that there is no central coordinator in the nodes. For this reason, the failure of a node never compromises the whole system. Another advantage is that nodes are organized in a way that a node only needs to coordinate with a few other nodes in the system (usually $\log N$, with N participants). Therefore, if a node joins or leaves the system, this does not affect the whole system. These features provide the system with high scalability and fault tolerance. The integration of DS records into the data structure of the DHT is based on the use of a SHA1 hash of the EPC as the key for the hash table. However, in these systems, security options like authentication or fine grained access control have to be implemented.

As a conclusion, DHTs might be said to have several advantages over LDAP: in LDAP there is a bottleneck in the root of the architecture, whereas in the DHT the failure of one node does not affect the whole system; LDAP is not optimized for massive update operations, while DHTs are optimized for massive search and update operations. On the other hand, DHTs have an important drawback with respect to LDAP: while in LDAP the access control is already implemented by Access Control Lists (ACLs) in DHT fine grained controls have to be implemented. Therefore, in DHT-based Discovery Service prototype proposed in this paper an access control service will be implemented. This service is based on the XACML (eXtensible Access Control Mark-up Language) language. It describes both an access control policy language and a request/response language. The policy language is used to express access control policies. The request/response language expresses queries about whether a particular access can be allowed and describes answers to those queries.

Finally, our mechanism to reconstruct supply networks and its integration within the DS prototype will be introduced. This mechanism presents the supply network in which a specific product is involved in graphical user interface. The following are the most relevant data which appear in the graphic: the identity of all the organizations which participate in the supply network; the identifiers of all the components of the product; and the amount of time for the flow of the product between two entities, or the amount of time that the product remains in an entity.

Therefore, we can conclude that the key contributions of this chapter are: (i) a DHT-based DS prototype implementation; (ii) the integration of an access control service within the DS prototype; (iii) the integration of a supply networks discovery service within the DS prototype.

5.6. Discovery services prototype

Figure 5.7: Discovery Services architecture.

5.6 Discovery services prototype

The objective of the designed DS is to return a time-ordered list of links to multiple ECPIS instances which hold information related to a specific EPC. Therefore, the DS is designed to create this list of links. The architecture of the full prototype has been divided into a set of logical components represented in Figure 5.7.

5.6.1 Discovery services

The DS implementation is composed by two components: the DHT application and the Publish/Query interface.

5.6.1.1 DHT application

(a) Introduction

There exist several free software implementations of structured P2P networks, like Chimera¹ (a C implementation of Tapestry), the Chord project² (a C implementation of Chord) and FreePastry³ (a Java implementation of Pastry). From among these FreePastry has been selected because it is implemented in Java and because it is an active project in development at the MPI-MPG⁴, although it initially started at Rice University in USA.

¹<http://current.cs.ucsb.edu/projects/chimera>

²<http://pdos.csail.mit.edu/chord>

³<http://www.freepastry.org/FreePastry/>

⁴<http://www.mpi-inf.mpg.de>

The FreePastry software implements the KBR API, and within its distribution offers a DHT application, called PAST [Druschel and Rowstron, 2001]. However, this application has some particularities. The PAST system exports the following set of operations to its clients: *fileId = insert(name, owner-credentials, k, file)* which stores a file at a user-specified number *k* of diverse nodes within the PAST network. The operation produces a 160-bit identifier (*fileId*) that can be used subsequently to identify the file; *file = lookup(fileId)* reliably retrieves a copy of the file identified by *fileId* if it exists in PAST and if one of the *k* nodes that store the file is reachable via the Internet. The file is normally retrieved from a live node “near” the PAST node issuing the lookup (in terms of the proximity metric), among the nodes that store the file.

During an insert operation, PAST stores the file on the *k* PAST nodes whose *nodeIds* are numerically closest to the 128 most significant bits of the file’s *fileId*. This invariant is maintained over the lifetime of a file, despite the arrival, failure and recovery of PAST nodes. The number *k* is chosen to meet the availability needs of a file, relative to the expected failure rates of individual nodes. However, popular files may need to be maintained at many more nodes in order to meet and balance the query load for the file and to minimize latency and network traffic. PAST adapts to query load by caching additional copies of files in the unused portions of PAST node’s local disks. Unlike the *k* primary replicas of a file, such cached copies may be discarded by a node at any time.

Therefore, PAST replicates files with the objective of putting up with the failure of nodes and it also caches additional copies of files in order to balance the query load. However, in our DHT application we assume that nodes do not suffer from frequent failures, therefore we do not need the replication functionality. On the other hand, we think that the caching functionality is not very useful in our application, because it consumes additional storage resources in the DHT nodes. For these reasons, we decided to implement our own DHT application over the FreePastry software implementation.

(b)Implementation

The public methods (*insert()* and *lookup()*) of every DHT node in our implementation can be accessed by a remote application using Java RMI¹, which is a Java API which performs the object-oriented equivalent of remote procedure calls (RPC). The prototype of the previous methods is as follows: *public Boolean insert(MessageDigest key, String url)* and *public String[] lookup(MessageDigest key)*. The insert method has two parameters: *String url* represents the URL of the EPCIS query interface with information about a specific EPC; *MessageDigest key* represents the SHA1 hash [Eastlake and Jones, 2001] of the related EPC. This method returns a *Boolean* which indicates if the insert operation has been properly finished. On the other hand, the lookup method has a single parameter: the SHA1 hash of an EPC, and it returns an array with all the associated URLs.

The *insert()* method of the DHT node creates a content object with the *key*, the URL and the its own nodehandle. After that, it inserts the previous content in a message with the following fields: the identifier of the origin node, the identifier of the destination node (which corresponds with the *key* of the message), the message type (for an INSERT message corresponds type 0) and the previously created content. Once the message has been created, the insert method calls the *route()* function with the previously created message.

¹<http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp>

5.6. Discovery services prototype

The previous message is routed to the node responsible of the message's key. Then the FreePastry software will call the *deliver()* implementation of our application. This method extracts the type of the message, and if it is an INSERT message it extracts the *key* (the SHA1 hash of the EPC) and the URL and it inserts both of them in a MySQL¹ database. The communication between our Java application and the database is possible thanks to the JDBC² driver. Our database only has one table, with two columns: one for the key (SHA1 hash of the EPC) and the other for the URL. Multiple insertions of the same key, each one of them with a different URL, are allowed. By this way, the database stores the different URLs of the EPCISs with information about a specific EPC.

After that, the application creates a content object with the inserted key and a *Boolean* indicating if the insertion of the EPC's key and the associated URL in the previous database has been correct or if there has been any mistake. This content is inserted in a message similar to the INSERT message but changing the type message for a 1 which represents an INSERT_RESPONSE message. Then, the application uses the single hop option of the *route()* function in the FreePastry software. This means that it directly sends the message to the node which called the INSERT operation, using its NodeHandle included in the INSERT message. The requester DHT node receives the response message in its *deliver()* method. After that, it extracts the Boolean value indicating if the INSERT operation has been correct and returns this value to the requester of the *insert()* method.

The operation of the *lookup()* method is equivalent to the *insert()* operation, but in this case the destination node extracts from the database all the URLs associated to the key of a specific EPC. An important characteristic of the implementation of the two previous methods is the use of the *continuations* functionality offered by FreePastry. A continuation is similar to a callback, or in java, a Listener, but it is typically used only once. The primary use of these in FreePastry is to handle network latency, or other types of IO, similar to regular function calls, that may block or take a significant amount of time. If the network call takes time to execute due to latency, then the program will block until it completes the task. *Continuations* allow you to continue processing when the result arrives, and in the meantime issue other requests.

An important advantage of the DHT-based implementation for the DS is that the DHT nodes can be distributed in different physical locations. In this way, temporary problems which can occur in the DS environment, such as power cuts, will only affect some of the DHT nodes, and with the proper mechanisms, the DS will be able to continue working without problems. However, these nodes cannot be located in arbitrarily because they have very valuable data. We suppose that the companies which use the DS service are within a specific company consortium, which trusts all its members. In this way, DHT nodes can be physically located in any company of the consortium.

In addition, another important characteristic of the DHT implementation is the use of the Secure Sockets Layer (SSL) [Dierks and Rescorla, 2008] functionality of FreePastry. We use SSL to implement confidentiality, integrity and authentication. Confidentiality means that the data transmitted during communications is not available to any other party. It is implemented with information encryption; that is, the data is not available because it is unreadable. Integrity is the property that the data has not been modified by unauthorized

¹<http://www.mysql.com>

²<http://dev.mysql.com/downloads/connector/j/5.1.html>

entities (or simply lost). It means that data are not modified or deleted during the communication by an unauthorized party. Data integrity can be achieved by signing the digests associated with the transmitted message or the stored record. Finally, authentication is the verification that an identity is really the entity it claims to be. This functionality requires that every node has a KeyStore and a TrustStore. A KeyStore is a database of private keys and their associated X.509 certificate [Housley et al., 2002] chains authenticating the corresponding public keys. On the other hand, a TrustStore is a database of trusted entities and their associated X.509 certificate chains authenticating the corresponding public keys. The characteristics of these databases will be explained in depth in Section 5.7.

5.6.1.2 Publish/query interface

(a) Introduction

This interface has been implemented as a Web Service¹ using two methods available through SOAP² operations: *publish()* and *query()*, and it has been deployed in a Glassfish application server³. A web service is defined by the W3C as "a software system designed to support interoperable machine-to-machine interaction over a network". It has an interface described in a machine-processable format called Web Services Description Language (WSDL)⁴. Other systems interact with the web service in the way prescribed by its description using SOAP messages, typically represented using HTTP with an XML serialization in conjunction with other web-related standards.

There are currently several frameworks to program Java Web Services, but the two most important are Apache Axis2⁵ and Sun JAX-WS⁶. Axis2 is the follow-up of the popular Axis1 framework, and it is based on a completely new architecture. An Axis 2 runtime is a web application which can be installed on every Java Enterprise Edition Application Server⁷, and it is also a container for Web Services. Web services are packed into an own file format with the file extension *aar*, and they can be installed and configured by a simple user interface at run time. Modules providing additional functionality, for example WS-* standard support, can also be installed during runtime. However, writing an Axis2 project requires a great deal of knowledge about the build tool Ant⁸.

The Java API for XML based Web Services (JAX-WS) is the successor of the JAX-RPC specification. Its configuration is managed by annotations⁹, therefore Java 5 or higher is required. With JAX-WS it is relatively easy to write and consume web services. The default values of numerous parameters are comfortable from the point of view of the programmer and simple methods declared with a `@WebService` annotation can be used as a service. A suitable WSDL document can also be generated from the class.

¹<http://www.w3.org/standards/webofservices/>

²<http://www.w3.org/TR/soap/>

³<https://glassfish.dev.java.net/>

⁴<http://www.w3.org/TR/wsdl20-primer/>

⁵<http://ws.apache.org/axis2/>

⁶<https://jax-ws.dev.java.net/>

⁷<http://java.sun.com/javaee/index.jsp>

⁸<http://ant.apache.org>

⁹<http://java.sun.com/j2se/1.5.0/docs/guide/language/annotations.html>

5.6. Discovery services prototype

Standards	Axis2	JAX-WS
WS-Addressing	X	X
WS-Coordination	X(2)	X
WS-MetadataExchange		X
WS-Policy	X	X
WS-ReliableMessaging	X(3)	X
Web Services Security	X(1)	X
WS-SecureConversation	X(1)	X
WS-SecurityPolicy		X
WS-Transaction	X(2)	X
WS-Trust	X	X
WS-Federation		

Table 5.1: Support for WS-* standards.

(1) Supported by the additional module Apache Rampart. (2) Supported by the additional module Apache Kandula2. (3) Supported by the additional module Apache Sandesha2.

The support of the WS-Standard family can also be decisive for the selection of a web service framework. For example, messages sent to services can be secured with signatures as described in the Web Service Security standard (WSS). Table 5.6.1.2 shows the support for WS*-standards of the frameworks.

From the data represented in Table 5.6.1.2, we can conclude that the best option to code against the standard is the JAX-WS implementation. Another important advantage of JAX-WS over Axis 2 is that with JAX-WS it is relatively easy to write and consume web services. In this respect, the enterprise pack of the NetBeans¹ development environment supports JAX-WS very well. Only a few clicks are necessary in order to build a server or to call a web service. In our application, we want to integrate some security mechanisms of the WS-* standard. This task is very easy using NetBeans. Therefore we decided to use JAX-WS and the NetBeans IDE in our development process. Alternatively, web applications containing Web Services which have been carried out with JAX-WS are executable in the Glassfish Application Server without problems. To make them executable in other application servers, two libraries (JAX-WS and JAXB) have to be installed. Therefore, we decided to use Glassfish in our development process, in order to assure the operation of the web service.

(b) Implementation

The *publish()* method of the web service implementation has two parameters: the SHA1 hash of the EPC and the URL where the information related to the EPC is available. The web service has previously been configured with the URL where the RMI stubs of all the DHT nodes are available. Therefore, the web service selects one of the DHT nodes of the network, it obtains its RMI stub and, after that, it calls the *insert()* method of the DHT application. After calling the *insert()* method, the web service receives if the insert

¹<http://netbeans.org>

operation has been performed in a correct way or not. Therefore, this interface is only a proxy between the EPCIS repository of the query application and the DHT nodes.

The *query()* method is similar to that previously mentioned, but in this case it only accepts one parameter: the SHA1 hash of an EPC. This method selects one of the DHT nodes and it calls the RMI *lookup()* method of the DHT application. After that, the web service receives the URLs associated to all the EPCISs with information about that EPC. All this information will be returned to the application which called the *query()* method.

Now, we want to integrate some security mechanism to the web service interface. NetBeans provides a set of security profiles¹ for configuring security on service endpoint and client. These profiles allow a developer to select the confidentiality, integrity and authentication of a message. When a user selects a security profile web services policy elements are then added to the WSDL, and during runtime all the security related tasks specified in the WSDL are performed.

The criteria for selecting a security profile take into account the following characteristics: the level of security (transport or message level) and the type of client credentials (username/password, X509 certificates, etc.). Among all the security profiles we have selected the *mutual certificates security* mechanism. This mechanism adds security through authentication and message protection, which ensures integrity and confidentiality. When using *mutual certificates*, a KeyStore and a TrustStore file must be configured for both the client and server sides of the application. In the same way, we also integrate SSL in the RMI interface between the DHT nodes and the Publish/Query web service to cryptographically secure the traffic. In order to do that the RMI server needs a KeyStore and the RMI client needs a TrustStore. The characteristics of the KeyStore and TrustStore will be explained in depth in Section 5.7.

5.6.2 EPCIS-DS interface

In the first place, we have deployed the Fosstrak EPCIS software² in a host with a Glassfish application server. The Fosstrak EPCIS software is a complete implementation of the EPCIS standard specification (Version 1.0.1 of September 21, 2007) and it allows users to deploy an EPCIS repository. In addition, Fosstrak developers also provide an interactive EPCIS capture application which allows users to fill the EPCIS repository with EPC data using a graphical user interface. In our prototype, we have used this graphical tool to insert events into the EPCIS Repository.

The current EPCIS standard does not include a specific communication mechanism between an EPCIS and a Discovery Service. For this reason, we have developed a module to be integrated into the Fosstrak software. Its goal is to send the association between an EPC, with information registered in that EPCIS repository, and the URL to query that information to the DS, but only once. That is, it is possible that the EPCIS registers several events associated to the same EPC because it is possible that it has several associated RFID readers. Therefore, our module has to assure the DS stores only one association between that EPC and the public URL of the EPCIS.

¹https://wsit-docs.dev.java.net/releases/m6/WSIT_Security4.html

²<http://www.fosstrak.org>

5.6. Discovery services prototype

Figure 5.8: EPCIS-DS interface.

We have studied the EPCIS software implementation and we have detected that it has two independent modules: the EPCIS capture interface and the repository, implemented in a MySQL database. The communication among them is by means of the TCP protocol, and the SQL queries are sent to the 3306 TCP port of the localhost. SQL queries are sent in plain text; therefore, we decided to develop a traffic sniffer to detect the SQL queries sent to the MySQL database.

The sniffer module has been developed in Java language, using the jNetPcap¹ software development kit. The basic function of jNetPcap is to provide a java wrapper to popular libpcap library² for capturing network packets. Our module captures the packets in the loopback interface, because the TCP communication between the EPCIS capture interface and the MySQL database is in the localhost. But it does not capture all the traffic; we have implemented a filter in order to detect the SQL queries with EPCs. In the first place, we have studied the available tables in the Fosstrak database for the different kind of events (Aggregation, Object, Transaction and Quantity events). For the Agregation, Object and Transaction events we have detected that the database has the following tables: event_AggregationEvent_EPCs, event_ObjectEvent_EPCs and event_TransactionEvent_EPCs, which associate the identifier of the event with the corresponding EPC. Therefore, our filter is able to detect when a SQL query inserts an EPC in the previous tables, that is, it detects the text "insert into event_AggregationEvent_EPCs" or "insert into event_ObjectEvent_EPCs" or "insert into event_TransactionEvent_EPCs". For the quantity events, the filter detects the text "insert into event_QuantityEvent", which associates a parent EPC (the EPC assigned to all the EPCs) which a specific event.

After the filter detects an SQL query with the previous text in a specific packet it will be redirected to a special method to process it. This method inspects the full SQL query to obtain the associated EPC. After detecting the EPC, our program checks if the detected EPC has been observed previously. In order to do that, we have created a database where the program stores the detected EPCs, and it is able to assure whether it has been previously inserted. In the case that it has not been detected, that is, it is the first time that information associated to that EPC is registered in the EPCIS repository, the program

¹<http://jnetpcap.com>

²<http://www.tcpdump.org>

calls the *publish()* method of the publish/query web service interface. This method has two parameters: a key, which corresponds to the SHA-1 hash of the EPC; and a URL, which corresponds to the URL of the query interface of the corresponding EPCIS (similar to *http://localhost:8080/epcis-respository/query*). It is worth pointing out that the query interface of the Fosstrak EPCIS software is also accessible by means of a web service client.

5.6.3 Query application

This application is a very simple program which allows a user to make real queries to the DS deployed. It accepts an EPC number and returns the URLs associated to all the EPCISs with information about that EPC. In order to do that, the application implements a web service client which interacts with the *query()* method of the publish/query web service interface. It is worth mentioning that this application also implements the security mechanisms presented in Section 5.6.1.2.

5.7 Security infrastructure and access control service

5.7.1 Security infrastructure

In the previous section, we introduced that we use SSL with client authentication to control the membership of the FreePastry network and cryptographically secure the traffic. This functionality requires that every node has a KeyStore and a TrustStore. In the same way, we also integrate SSL in the RMI interface between the DHT nodes and the Publish/Query web service. In this case, the RMI server needs a KeyStore and the RMI client needs a TrustStore. In addition, we have selected the *mutual certificates security* mechanism to add security to the web service interface. A KeyStore and a TrustStore file must be configured for both the client and server sides of the web service application.

A KeyStore is a database of private keys and their associated X.509 certificate chains authenticating the corresponding public keys. On the other hand, a TrustStore is a database of trusted entities and their associated X.509 certificate chains authenticating the corresponding public keys. In order to obtain the previous X.509 certificates we need the help of a Public Key Infrastructure or simply PKI [Weise, 2001].

A PKI is mainly composed of three elements: the Certification Authority (CA), the Registration Authority (RA) and the Certificate Depositories (CD). The CA is, according to the IETF, “an authority trusted by one or more users to create and assign public key certificates”. The CA’s primary purpose is to support the generation, management, storage and revocation of public key certificates (also known as digital certificates). A public key certificate establishes the binding of an end entity’s identity (person or a computer identified by its IP address) and its public key. That is, it contains enough information for someone to be able to validate and verify the identity of the owner of the certificate. The certificate should include the name of both, the entity and the CA, the expiration period of the certificate and the most important field: the entity’s public key. The most widely accepted format for digital certificates is that based on the IETF X.509v3 standards. The CA also has its own digital certificate, which is announced and published on the Internet.

5.7. Security infrastructure and access control service

When the CA issues a certificate to user, it appends at the end of the digital document the result to sign with its private key the hash calculated over the certificate (using the MD5 hash function). In this way, any intermediate entity can verify the authenticity of the certificate by repeating the above procedure and comparing the obtained result with the unsigned hash, which is calculated with the help of the widely known CA's public key.

The RA, although optional, is a common component of a PKI. The CA entity authorizes the RA to verify an end entity's identity and determine whether an end entity is entitled to have a public key certificate issued on behalf of the CA. The CD, sometimes referred to as a certificate directory, is also an optional but common component of a PKI. A CD may be an efficient solution for closed systems in which the security functions frequently require the public keys (or equivalent, public certificates) of the participants. The CD is in charge of the storage and the publishing of the CA's public key and the certificate of any element playing a role in the party. Additionally, it also stores the Certificate Revocation List (CRL). Sometimes the CA can invalidate a digital certificate even if it is not out-of-date. This situation normally arises when the certificate's owner has compromised its associated private key. The CRL stores the invalidated certifies and is used to verify that a certificate is currently valid and it must be consulted every time that a certificate is used. It is clear that the CD is in fact no more than just a database. However, due to the fact that it is mainly used to read data (i.e., proportionally, there are few write operations) it is normally implemented using a LDAP service.

At this point, a fundamental question arises: who is responsible for building and maintaining the PKI? Technically speaking, the OpenSSL¹ free-software tool might mean that the consortium would assume this responsibility. However, due to the costs related to the PKI management and maintenance, it is recommendable that this service is subcontracted to a specialized service supplier. In this way, the consortium would only assume small costs (certification generation, management and revocation). In addition, if considered convenient, the consortium may also play the role of a Registration Authority and Certificates Depositories, whenever they reach an agreement with the PKI service provider.

5.7.2 Access control service

In our prototype, the access control service is used to determine who can read the resources stored in the DS and also who can write. The most promising technology to develop access control procedures is XACML². The XACML (eXtensible Access Control Markup Language) is an XML-based language for access control that has been standardized in OASIS (Organization for the Advancement of Structured Information Standards³, a non-for-profit consortium that drives the development, convergence and adoption of open standards for the global information society). XACML describes both an access control policy language and a request/response language. The policy language is used to express access control policies (who can do what when). The request/response language expresses queries about whether a particular access should be allowed (requests) and describes answers to those queries (responses).

¹<http://www.openssl.org>

²<http://www.oasis-open.org/committees/xacml/>

³<http://www.oasis-open.org>

Figure 5.9: Access Control service.

XACML is composed by three entities: the Policy Enforcement Point (PEP), the Policy Decision Point (PDP) and the Policy Administration Point (PAP). The PEP is the point which intercepts user's access request to a resource and enforces PDP's decision. The PDP is the point which evaluates and issues authorization decisions. Finally, the PAP is the point which manages the policies. In our architecture, there is one PEP in every DHT node. There is also one PDP and one PAP, located both in the publish/query interface. This functionality has been implemented as a web service using XACMLight¹. It is an Axis2 web service that implements a PDP and a PAP. This implementation covers all functions that are defined by XAMCL 2.0 (including optional), all mandatory elements and almost all optional features. The PDP implements the *getDecision()* method which gets an authorization decision. On the other hand, the PAP implements other methods like *addPolicy()*, *deletePolicy()*, *getPolicy()* or *setPolicy()*, which adds a policy with, deletes a policy, returns a policy, and overrides a policy with a given policy ID, respectively. At the moment, we are working in a JAX-WS web service to implement PDP and PAP.

Our access control mechanism implements one policy for the read operation implemented in the *lookup()* method of the DHT application. The default behavior is to deny the access to the information except for those clients defined by the companies of the consortium as partners. The PEP located in the DHT node (responsible for storing the URLs associated to the EPC included in the *lookup()* method) calls the *getDecision()* method of the PDP situated in the publish/query interface in order to get a write authorization decision. If it is allowed, the DHT application obtains the corresponding URL from the database. Otherwise, the DHT application returns an error message. This is possible because we have added a new field to the LOOKUP message: the identity of the query application which calls the *query()* web service method. On the other hand, the write operation implemented in the *insert()* is always allowed if it comes from an authenticated EPCIS. Figure 5.9 shows the architecture of the proposed access control system.

¹<http://sourceforge.net/projects/xacmllight>

5.8. Supply network discovery service

Figure 5.10: A product structure.

5.7.3 Other security services

The implementation of confidentiality, integrity, authentication, and access control security services has been previously introduced in this document. In this section, we are going to present the implementation of the non-repudiation, accountability and availability security services.

The non-repudiation facility is accomplished automatically in a PKI environment if integrity is enabled. In order to strengthen the non-repudiation service, it is convenient to record all the calls to the *query()* and *publish()* methods of the publish/query interface, at least during a long period of time. In this case, it is necessary to provide enough storage space to the publish/query interface (this is not a trouble due to the current cost reduction of any type of storage facility). On the other hand, the accountability is usually made by creating and maintaining a set of log files using the *syslog* facility available in all operating systems.

Finally, the availability is provided by fine tuning the operating system. Each DHT node must have perfectly configured a Firewall (to restrict the incoming communications), probes to listen, react and prevent different kind of attacks (like Denial of Service), a continuously updated antivirus, a daily backup procedure, etc.

5.8 Supply network discovery service

5.8.1 Introduction

In this section the mechanism to reconstruct supply networks and its integration within the previously designed DS prototype will be introduced. A concrete example (extracted from [Bi and Lin, 2009]) is used to demonstrate how a supply network can be mapped using our methodology. Suppose that a retailer A sells a product that is assembled by a manufacturer B. Every product item and each of its components are given an EPC tag. The structure of the product with five EPC tags is shown in Figure 5.10. Note that component #2 is a sub-assembly that consists of two smaller components.

After receiving a several requests to repair component #4, retailer A decides to map its supply network for this product in order to redirect the requests for repairs to the company that produces component #4. Retailer A is also interested in calculating the amount of

Figure 5.11: Supply network associated to EPC #1.

time taken for products to flow between entities in this supply network. The full supply network that retailer A wants to reconstruct is represented in Figure 5.11.

Now, the necessary information to reconstruct the previous supply network needs to be deduced. Firstly, we concentrate on the events that assign an EPC to a new product (ASSIGN events, they are represented by () in the figure). These events can be divided in two different sub-types. The first of which corresponds to the assignment of EPCs #3, #4, and #5 to the corresponding products. They are not all composed of other products, that is, they do not all have smaller components. The necessary information to represent the previous events is the following: #EPC, Company, Event=ASSIGN and Date. On the other hand, the assignment of EPCs #1 and #2 is different from the previous one, because in this case the new products are composed of several components with an assigned EPC. In this case, the necessary information to represent these events is similar to the previous one with the difference that the identity of the components of the product is added. That is, it will be: #EPC, Company, Event=ASSIGN, Date and #EPC of Components.

Next, we concentrate on the flow of products. To represent this we need to deduce the amount of time for the flow of products from one company to another company and the direction of the flow. The previous information can be deduced if we extract the exact date on which the product is sent from the company of origin (SHIP event) to a specific destination company, and the exact date on which the product is received by the destination company (RECEIVE event). In order to represent the SHIP event we need the following information: #EPC, Company, Event=SHIP, Date and Destination. On the other hand, to represent the RECEIVE event we only need: #EPC, Company, Event=RECEIVE and Date. In conclusion, in order to reconstruct a full supply network we need a time-ordered list of ASSIGN, SHIP and RECEIVE events with the necessary information to represent them all. This information is represented in Table 5.2, where not all the columns are necessary for every event.

EPC	EVENT	COMPANY	DATE	TO	CONTAINS

Table 5.2: Necessary information to reconstruct a supply network.

In order to obtain the previous information, it is necessary to send a query to the query interface of the Foostrak EPCISs with information about a specific EPC, asking for some information which depends on the event (ASSIGN, SHIP or RECEIVE) which we want to identify. For an ASSIGN event without components, it is necessary to ask

5.8. Supply network discovery service

```
@WebServiceRef(wsdlLocation = "WEB-INF/wsdl/localhost_8080/epcis-repository-0.4.2/
query.wsdl")
QueryOperationsWebServiceService service;
org.fosstrak.epcis.repository.query.EPCISServicePortType port =
    service.getQueryOperationsWebServicePort();

private List<QueryParam> queryParams = new ArrayList<QueryParam>();

QueryParam param1 = new QueryParam();
param.setName("eventType");
param.setValue("ObjectEvent");
queryParams.add(param1);

QueryParam param2 = new QueryParam();
param.setName("EQ_action");
param.setValue("ADD");
queryParams.add(param2);

QueryParam param3 = new QueryParam();
param.setName("MATCH_epc");
param.setValue("urn:epc:id:sgtin:0034000.987650.2686");
queryParams.add(param3);

epcglobal.epcis_query.xsd._1.Poll poll = new epcglobal.epcis_query.xsd._1.Poll();
poll.setQueryName("SimpleEventQuery");
poll.setParams(queryParams);
epcglobal.epcis_query.xsd._1.QueryResults result = port.poll(poll);
```

Table 5.3: Web service client asking the information associated to an ASSIGN event without components.

for the *date* of an *ObjectEvent* with *action=ADD* associated to a specific EPC. For an ASSIGN event with components, it is necessary to ask for the *date* and the *childEPCs* of an *AggregationEvent* with *action=ADD* associated to a specific EPC. For a SHIP event it is necessary to ask for the *date* and the *purchase order* associated to the *business transaction* of an *ObjectEvent* with *action=OBSERVE* and *businessStep=SHIPPING* associated to a specific EPC. In order to guess the destination company of a SHIP event we are going to use the purchase order associated to the business transaction. This purchase order includes the identities of the buyer and the supplier, the product related to the purchase order and the quantity of the ordered product. Finally, for a RECEIVE event, it is necessary to ask for the *date* of an *ObjectEvent* with *action=OBSERVE* and *businessStep=RECEIVING* associated to a specific EPC.

The query interface of the Foostrak EPCISs is provided by means of a web service; therefore, it is necessary to program a client web service to ask for the previous information. Table 5.3 shows as an example the way in which the information associated to an ASSIGN event without components is requested.

From the previous table, we can conclude that the queries sent to the EPCIS software are built using a set of parameters, which have an associated pair (name, value). Table 5.4 represents the parameters to query the previous ASSIGN, SHIP and RECEIVE events.

The *poll()* method returns an array with the following information: Event, occurred, recorded, Parent ID, Quantity, EPCs, Action, Business step, Disposition, ReadpointId, Business location, Business transaction. Therefore, we only have to go to the corresponding column to obtain the necessary information.

Event	eventType	EQ_action	3rd Parameter
ASSIGN	ObjectEvent	ADD	MATCH_epc = urn:epc:id:sgtin:0034000.987650.2686
ASSIGN with comp.	AggregationEvent	ADD	MATCH_parentID = urn:epc:id:sscc:0614141.1234567890
SHIP	ObjectEvent	OBSERVE	EQ_bizStep = urn:epcglobal:cbv:bizstep:shipping
RECEIVE	ObjectEvent	OBSERVE	EQ_bizStep = urn:epcglobal:cbv:bizstep:receiving

Table 5.4: Parameters to query the ASSIGN, SHIP and RECEIVE events.

5.8.2 Implementation

In this section, we are going to present the integration of a supply networks discovery mechanism within our DHT-based DS prototype. We have added a new public method to our DHT application, called *supply_network()*, which can also be accessed by a remote application using Java RMI. The prototype of the previous method is the following: *public Object[][] supply_network(MessageDigest key)*. This method has a single parameter, the SHA1 hash of an EPC, and it returns a bi-dimensional array of *Objects*. The Java *Object* class sits at the top of the class hierarchy tree in the Java development environment, that is to say, every class in the Java system is a descendent of the *Object* class. The bi-dimensional *Object* array has all the necessary information to represent the supply network. That is, the columns have these values: #EPC, Company, Event (ASSIGN, RECEIVE, SHIP), Data, Destination Company (for SHIP events) and Components (for ASSIGN events with different sub-products). In addition, every row represents an event (ASSIGN, RECEIVE or SHIP) associated to a specific EPC.

We have also added a new method to the web service publish/query interface called *network()*. The *network()* method is similar to the *query()* method. It only accepts one parameter: the SHA-1 hash of an EPC. This method selects one of the DHT nodes and it calls the RMI *supply_network()* method of the DHT application. After that, the web service receives the previous *Object* bi-dimensional array, with all the necessary information to represent the supply network. All this information will be returned to the application which called the *network()* method.

The *supply_network()* method of the DHT node creates a content object with the key and its own nodehandle. After that, it inserts the previous content in a message with the following fields: the identifier of the origin node, the identifier of the destination node (which corresponds to the key of the message), the message type (for a SUPPLY_NETWORK message corresponds to type 4) and the previously created content. Once the message has been created, the *supply_network()* method calls the *route()* function with the previously created message. The *supply_network()* method also uses the *continuations* functionality offered by FreePastry.

The previous message is routed to the node responsible for the message's key. Then the FreePastry software calls the *deliver()* implementation of our application. This method extract the type of the message, and if it is a SUPPLY_NETWORK message it extracts the key of the EPC, and after that it extracts from the database all the URLs associated to the key of a specific EPC. It is necessary to take into account that the URLs are obtained in a time order. That is, the first extracted URL corresponds with the first EPCIS which registered information about the corresponding EPC. Then, the application follows the

5.8. Supply network discovery service

```
IF URL[].length == 1 THEN
    ASSIGN_function(URL)
ELSE
    FOR each component of URL[]
        IF URL is the last
            URL.poll(RECEIVE)
        ELSEIF URL is the first
            URL.poll(SHIP)
            ASSIGN_function(URL)
        ELSE
            URL.poll(SHIP)
            URL.poll(RECEIVE)
        ENDIF
    ENDFOR
ENDIF

FUNCTION ASSIGN_function(URL)
    COMPONENTS[] = URL.poll(ASSIGN with components)
    IF COMPONENTS[].length > 0
        FOR each COMPONENT
            supply_network(COMPONENT.\#EPC)
        ENDFOR
    ELSE
        URL.poll(ASSIGN without components)
    ENDIF
RETURN
```

Table 5.5: Algorithm of the *supply_network()* method.

reconstruction algorithm presented in Table 5.5.

As an example, we present the operations performed by the DHT application in order to reconstruct the supply network associated to the previously presented product. In this process, we assume that the DHT node responsible for storing the URLs of the EPCIS with information about the EPC #x is the DHTx node (e.g., DHT1 has the URLs with information about the EPC #1). Therefore, the call to the *supply_network()* method with EPC #1 as a parameter, gets to DHT1. This node has two URLs associated to EPC #1, [URLB, URLA]. The application takes the last URL (URLA) and it sends a query to the query interface of the corresponding EPCIS asking for a RECEIVE event. After that, it receives the necessary information to construct an *Object* array with the same columns represented in Table 5.2. That is, it constructs [#1; RECEIVE; A; 10/02/09; NULL; NULL]. Then, the application takes the first URL (URLB), it sends a query asking for a SHIP event, with the received information it construct the *Object* array [#1; SHIP; B, 08/02/09; A; NULL] and it adds this array to the previously constructed *Object* array. Immediately, the application sends a new query to the same URL asking for an ASSIGN event with components, with the received information it construct the *Object* array [#1; ASSIGN; B; 01/02/09; NULL; #2, #5], and it adds this array to the previously array. Finally, the application calls the *supply_network()* method taking EPC #5 as a parameter. When it receives all the information related to this EPC, it will call the *supply_network()* method taking EPC #2 as a parameter. The operation mode of these two *supply_network()* calls is equivalent to the presented one and, as a result, both of them return an *Object* bi-dimensional array, with all the information associated to the corresponding sub-supply networks associated to EPC #2 and EPC #5. All this information is added to the previously constructed *Object* bi-dimensional array (with the information about EPC #1) and it is returned to the *network()* web service method. The information contained in this *Object* bi-dimensional array corresponds to the information presented in Table 5.6. Figure 5.12 represents the interaction among the different DHT notes involved in the reconstruction of this supply network.

In addition, every DHT node which receives a SUPPLY_NETWORK query stores

EPC	EVENT	COMPANY	DATE	TO	CONTAINS
#1	ASSIGN	B	01/02/09	—	#2,#5
#1	SHIP	B	08/02/09	A	—
#1	RECEIVE	A	10/02/09	—	—
#2	ASSIGN	E	10/01/09	—	#3,#4
#2	SHIP	E	20/01/09	F	—
#2	RECEIVE	F	22/01/09	—	—
#2	SHIP	F	29/01/09	B	—
#2	RECEIVE	B	31/01/09	—	—
#5	ASSIGN	G	13/12/08	—	—
#5	SHIP	G	20/12/08	B	—
#5	RECEIVE	B	23/12/08	—	—
#3	ASSIGN	D	10/12/08	—	—
#3	SHIP	D	11/12/08	E	—
#3	RECEIVE	E	12/12/08	—	—
#4	ASSIGN	C	08/12/08	—	—
#4	SHIP	C	20/12/08	E	—
#4	RECEIVE	E	21/12/08	—	—

Table 5.6: Information to reconstruct the supply network associated to EPC #1.

temporarily in a new database all the deduced information (contained in the *Object* bi-dimensional array), and it associates all this information to the corresponding EPC. By this way, subsequent calls to the *supply_network()* method will be resolved in a shorter period of time. Finally, the access control service is also used by this mechanism. The PEP located in the DHT node (responsible for storing the URLs associated to the EPC included in the *supply_network()* method) calls the *getDecision()* method of the PDP situated in the publish/query interface in order to get a read authorization decision. If it is allowed, the DHT application gets the necessary information to reconstruct the supply network. Otherwise, the DHT application returns an error message. This is possible because we have added a new field to the SUPPLY_NETWORK message: the identity of the query application which calls the *network()* web service method. Our access control mechanism implements one policy for the read operation implemented in the *supply_network()* method of the DHT application. The default behavior is to deny the access to the information except for those clients defined by the companies of the consortium as partners.

5.8.3 Supply network presentation

The last step in the supply network reconstruction process is the graphic representation of the received data, so that the information can be easily understood and interpreted by the final user. This functionality has been added to the previously presented query application. The application implements a new web service client which interacts with the *network()* method of the publish/query web service interface. With the received information, the

5.8. Supply network discovery service

Figure 5.12: Interaction among the DHT nodes involved in the reconstruction of this supply network.

program represents the supply network of a product, showing the product creation and its movement through the companies which make up the network. This graphic representation will also offer time information about the events.

Figure 5.13 shows the application results corresponding to the example presented in Section 5.8.1. The graphic representation is made up of boxes, which represent the companies that compose the supply chain. Different values are associated to each box: the company name (that is drawn in the box), a list of products or items that are created in that company (the EPC of these products are drawn on the top of the box, in brackets) and some connectors that join this box with others. Input connectors are associated to the reception of a product, while output connectors are associated to the sending of a product. As can be seen in the figure, if a user places the mouse over a connector, the sending and receiving dates are shown. In addition, if a user clicks over any box, a new window is opened. This window is a zoom of the box, that shows all the available information about events related to that company (see Figure 5.14).

The application procedure is described in the following flowchart (see Figure 5.15). After receiving the information of all the events associated to the product supply chain, they will be individually analyzed by the application. Because each event is associated to a company, an initial check is made to see if that company is already represented. If not, a new box is created. Then, depending on the event type (ASSIGN or SHIP) a new EPC tag is added to the list of created products or a new connector is created. Besides indicating the EPC of the product, the source and the destination, each connector is associated with a sending date and a receiving date. This last value is obtained from the appropriate RECEIVE event. Although due to space limits it is not detailed, the application makes the required actions (depending on the dates and event types) to draw the boxes and connectors in the right position. In addition, the application offers users the possibility of moving the boxes around the screen, by simply selecting and moving the mouse. Of course, after

Figure 5.13: Graphical representation of the supply network associated to product with EPC #1.

Figure 5.14: Zoom view of company E.

moving a box, the associated connectors will be adapted to the new box situation.

5.9 Conclusions

A distributed hash table (DHT) is a reliable, scalable, wide-area data storage system that frees programmers from many of the complications of building a distributed system. DHTs store blocks of data on hundreds or thousands of machines connected to the Internet, replicate the data for reliability, and quickly locate data despite running over high-latency, wide-area links. This kind of applications are implemented over the key-based routing (KBR) interface offered by a structured P2P network.

In this chapter, it has been proposed a mechanism for automatically obtaining the supply network associated to a specific product using the EPCglobal Network, and it has been implemented in a DHT system. There are other mechanisms with the same objective, but the main difference with the proposal set out in this chapter is that in our proposal the

5.9. Conclusions

Figure 5.15: Application flowchart.

client does not have to perform any operations, that is, it queries the supply network associated to a specific product and the results are obtained directly.

This mechanism has been integrated within the DS component of the EPCglobal Network, which is currently being developed. Recently, there have been several proposals for the implementation of the DS. In most of them the DS has been implemented as a centralized database. As an example, in the BRIDGE Project it has been developed as a centralized database based on LDAP. In this work, we have developed a DS prototype based on DHT. DHTs have several advantages over centralized approaches, however they also have an important drawback: they do not implement any access control mechanism. Therefore, in our DHT-based DS prototype an access control service has also been implemented.

Conclusions

In this chapter the main outcomes of this Thesis are summarized, as well as the possible issues that may be addressed in future works.

6.1 Conclusions

Chapter 1 has provided an overview of the basic principles of peer-to-peer overlay networks. First of all, the concept of overlay network has been presented and the importance of this kind of networks has been highlighted. After that, peer-to-peer networks have been introduced

Chapter 2 has presented unstructured P2P networks. These networks do not impose any structure on the overlay topology. That is, they present a flat topological architecture, where all the nodes belong to the same level. This characteristic makes this kind of networks specially suitable for the implementation of source-rewriting mechanisms, traditionally used in IP networks to provide anonymity. One of these mechanisms, called Crowds, is based on the building of a random walk path between the initiator and the destination. One of the drawbacks of this methodology is that the cost associated with the communication increases dramatically as the number of hops also increases. Therefore, limiting the length of the paths is a key aspect in this scenario. In this work, an effective mechanism to reduce the variance associated with the length of the random walk paths is proposed. In addition, an intuitive but rigorous analytical model to calculate the degree of anonymity in networks using the proposed mechanism is also derived. This model has been validated by means of simulation.

Chapter 3 has presented superpeer-based P2P networks. In this case, the network architecture present a hierarchical topological architecture. Clients are connected to a superpeer, and it is connected to a set of neighbor superpeers. This architecture was used in [Nafaa et al., 2008] to implement a P2P-based VOD service distribution architecture in a broadband operator context. In this architecture, the content is stored in the set-top boxes (STBs), which act as peers, and the sessions are provided by means of multi-source streaming. In this chapter, the previous work has been extended. First of all, the system has been evaluated in a broadband operator scenario, and it has been deduced that the existing mechanisms are not enough to fully take advantage of the network resources. The

second part of the chapter has extended the previous VOD service distribution architecture to the Internet context.

Chapter 4 has presented structured P2P networks. These networks were developed to provide deterministic bounds on the data discovery. They offer a routing service to send a message, with an associated key, to a single node, responsible for that key. For this reason, this service is called Key-Based Routing (KBR). In general, decentralized P2P architectures require some kind of network maintenance, since centralized servers have been removed. Self-organization can be a good way to solve these control issues. In this chapter, it has been presented a self-organizing mechanism for superpeer-based P2P applications, which makes use of the reliable routing functionality associated to structured P2P networks. The proposed mechanism has been implemented in two superpeer-based P2P applications. The first one is a two-level P2P application, where all the superpeers are members of an ALM group. The second one is a hierarchical P2P application. This application is similar to the previous one, but in this case the superpeers have direct communication among them using a hierarchical architecture of superpeers.

Chapter 5 has presented DHT systems. A DHT is a reliable, scalable, wide-area data storage system that frees programmers from many of the complications of building a distributed system. DHTs store blocks of data on hundreds or thousands of machines connected to the Internet, replicate the data for reliability, and quickly locate data despite running over high-latency, wide-area links. This kind of applications are implemented over the KBR interface offered by a structured P2P network. In this chapter, it has been proposed a mechanism for automatically obtaining the supply network associated to a specific product using the EPCglobal Network, and it has been implemented in a DHT system. There are other mechanisms with the same objective, but the main difference with the proposal set out in this chapter is that in our solution the client does not have to perform any operations, that is, it queries the supply network associated to a specific product and the results are obtained directly. This mechanism has been integrated within the DS component of the EPCglobal Network, which is currently being developed. Recently, there have been several proposals for the implementation of the DS. In most of them the DS has been implemented as a centralized database. In this work, it has been developed a DS prototype based on DHT. DHTs have several advantages over centralized approaches, however they also have an important drawback: they do not implement any access control mechanism. Therefore, in our DHT-based DS prototype an access control service has also been implemented.

6.2 Future works

The vast majority of current Internet usage consists of content being disseminated from a source to a number of users, ranging from distributing conventional multimedia data to sharing user generated data over the web such as text, image, and video data (e.g., Facebook, Twitter, and YouTube). Growing demands led service providers like Google, Yahoo, and Microsoft to invest in large *data centers* with hundreds of thousands machines distributed across different geographic regions. Along with the expansion of data centers, backbone network providers have been increasing network capacity by deploying a large

6.2. Future works

number of *high-speed routers* and *fiber transmission systems*. In addition, for more efficient content delivery, large service providers also build their own *private networks* that interconnect their data centers.

As a result, nowadays, the Internet includes a web of tens of millions of networking devices ranging from routers to home gateways, thus consuming considerable energy overall. In addition, the energy efficiency of current networking devices is reported to be very poor. Therefore, as user demand grows, the overall traffic increases and energy consumption will likewise increase unless the efficiency of the network improves proportionally.

Recent research suggest using end-users' set-top boxes (STBs), such as IP-enabled video recorders, digital receivers, or DSL modems, to build collaborative content delivery systems. Two special features make them attractive: manageability of resources, such as disc space and bandwidth, and their availability, since it is often assumed that STBs are always online. However, networking devices have a wide spectrum of energy consumption for packet forwarding, and surprisingly, it rapidly goes up as packets travel toward end users. For instance, when delivering the same amount of data, home gateways and desktop PCs via DSL links could consume 100 and 1000 times more energy than core routers, respectively. One solution to this problem could be the design and implementation of intelligent *standby policies* for STBs.

On the other hand, noting that routers are far more energy efficient than servers and access devices, a radical alternative is to introduce *content routers* into the network as in content-centric networking. It is a new network architecture for content dissemination that replaces conventional host-to-host conversations with named data oriented communications. The basic operation of a content router is very similar to that of an IP router, but the key departure is that it supports name-based routing and content caching throughout the network. This way, content-centric networking obviates the need of deploying pre-planned, application specific mechanisms which require sophisticated network devices for mapping named content to hosts.

Bibliography

- [Alima et al., 2003] Alima, L. O., El-Ansary, S., Brand, P., and Haridi, S. (2003). Dks (n, k, f): A family of low communication, scalable and fault-tolerant infrastructures for p2p applications. In *Proceedings of the 3rd International Symposium on Cluster Computing and the Grid, CCGRID'03*, pages 344–350.
- [Androutsellis-Theotokis and Spinellis, 2004] Androutsellis-Theotokis, S. and Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4):335–371.
- [Armenio et al., 2009] Armenio, F., Barthel, H., Dietrich, P., Duker, J., Floerkemeier, C., Garrett, J., Harrison, M., Hogan, B., Mitsugi, J., Preishuber-Pfluegl, J., Ryaboy, O., Sarma, S., Suen, K., Traub, K., and Williams, J. (2009). Epcglobal architecture framework. Available online at: http://www.epcglobalinc.org/standards/architecture/architecture_1_3-framework-20090319.pdf.
- [Augustin et al., 2006] Augustin, B., Cuvellier, X., Orgogozo, B., Viger, F., Friedman, T., Latapy, M., Magnien, C., and Teixeira, R. (2006). Avoiding traceroute anomalies with paris traceroute. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, IMC'06*, pages 153–158.
- [Balakrishnan et al., 2003] Balakrishnan, H., Kaashoek, M. F., Karger, D., Morris, R., and Stoica, I. (2003). Looking up data in p2p systems. *Communications of the ACM*, 46(2):43–48.
- [Baumgart et al., 2007] Baumgart, I., Heep, B., and Krause, S. (2007). Oversim: A flexible overlay network simulation framework. In *Proceedings of 10th IEEE Global Internet Symposium, GI'07*, pages 79–84.
- [Bennett and Grothoff, 2003] Bennett, K. and Grothoff, C. (2003). GAP – practical anonymous networking. In *Proceedings of Privacy Enhancing Technologies Workshop, PET'03*, pages 141–160.
- [Bi and Lin, 2009] Bi, H. H. and Lin, D. K. J. (2009). Rfid-enabled discovery of supply networks. *IEEE Transactions on Engineering Management*, 56(1):129–141.

- [Borissov, 2005] Borissov, N. (2005). *Anonymous routing in structured peer-to-peer overlays*. PhD thesis, University of California at Berkeley, Berkeley, CA, USA.
- [Boufkhad et al., 2008] Boufkhad, Y., Mathieu, F., de Montgolfier, F., Perino, D., and Viennot, L. (2008). Achievable catalog size in peer-to-peer video-on-demand systems. In *Proceedings of the 7th International Conference on Peer-to-peer Systems*, IPTPS'08, pages 1–6.
- [Brook et al., 1988] Brook, A., Kendrick, D., and Meeraus, A. (1988). Gams, a user's guide. *ACM SIGNUM Newsletter*, 23(3-4):10–11.
- [Buford, 2008] Buford, J. (2008). Management of peer-to-peer overlays. *International Journal of Internet Protocol Technology*, 3(1):2–12.
- [Buford and Yu, 2010] Buford, J. and Yu, H. (2010). Peer-to-peer networking and applications: Synopsis and research directions. In Shen, X., Yu, H., Buford, J., and Akon, M., editors, *Handbook of Peer-to-Peer Networking*, pages 3–45. Springer.
- [Buford et al., 2008] Buford, J., Yu, H., and Lua, E. K. (2008). *P2P Networking and Applications*. Morgan Kaufmann Publishers Inc.
- [Chaum, 1981] Chaum, D. (1981). Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2):84–90.
- [Chaum, 1988] Chaum, D. (1988). The dining cryptographers problem: unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75.
- [Chawathe et al., 2003] Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., and Shenker, S. (2003). Making gnutella-like p2p systems scalable. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM'03, pages 407–418.
- [Chen et al., 2009a] Chen, Y., Han, B., Leblet, J., Straub, G., and Simon, G. (2009a). Network-friendly box-powered video delivery system. In *Proceedings of the 21st International Teletraffic Congress*, ITC'09, pages 1–8.
- [Chen et al., 2009b] Chen, Y., Leblet, J., and Simon, G. (2009b). On reducing the inter-as traffic of box-powered cdn. In *Proceedings of the 18th International Conference on Computer Communications and Networks*, ICCCN'09, pages 1–6.
- [Chen et al., 2009c] Chen, Y.-F., Huang, Y., Jana, R., Jiang, H., Rabinovich, M., Rahe, J., Wei, B., and Xiao, Z. (2009c). Towards capacity and profit optimization of video-on-demand services in a peer-assisted iptv platform. *Multimedia Systems*, 15(1):19–32.
- [Cheng et al., 2007] Cheng, B., Jin, H., and Liao, X. (2007). Supporting vcr functions in p2p vod services using ring-assisted overlays. In *Proceedings of the IEEE International Conference on Communications 2007*, ICC'07, pages 1698–1703.

BIBLIOGRAPHY

- [Chi et al., 2007] Chi, H., Zhang, Q., Jia, J., and Shen, X. (2007). Efficient search and scheduling in p2p-based media-on-demand streaming service. *IEEE Journal on Selected Areas in Communications*, 25(1):119–130.
- [Clark et al., 2005] Clark, D., Lehr, W., Bauer, S., Faratin, P., Sami, R., and Wroclawski, J. (2005). The growth of internet overlay networks: Implications for architecture, industry structure and policy. In *Proceedings of the 33rd Research Conference on Communication, Information and Internet Policy*, TPRC’05.
- [Clarke et al., 2000] Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W. (2000). Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of ICSI Workshop on Design Issues in Anonymity and Unobservability*.
- [Comaniciu et al., 1999] Comaniciu, D., Meer, P., Xu, K., and Tyler, D. (1999). Retrieval performance improvement through low rank corrections. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, CBAIVL’99, pages 50–54.
- [Dabek et al., 2003] Dabek, F., Zhao, B., Druschel, P., Kubiatowicz, J., and Stoica, I. (2003). Towards a common api for structured peer-to-peer overlays. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems*, IPTPS’03, pages 33–44.
- [Dhara et al., 2010] Dhara, K., Guo, Y., Kolberg, M., and Wu, X. (2010). Overview of structured peer-to-peer overlay algorithms. In Shen, X., Yu, H., Buford, J., and Akon, M., editors, *Handbook of Peer-to-Peer Networking*, pages 223–256. Springer.
- [Diaz et al., 2002] Diaz, C., Seys, S., Claessens, J., and Preneel, B. (2002). Towards measuring anonymity. In *Proceedings of Privacy Enhancing Technologies Workshop*, PET’02, pages 54–68.
- [Dierks and Rescorla, 2008] Dierks, T. and Rescorla, E. (2008). RFC 5246: The transport layer security (tls) protocol. Available online at: <http://tools.ietf.org/rfc/rfc5246.txt>.
- [Dingledine et al., 2004] Dingledine, R., Mathewson, N., and Syverson, P. (2004). Tor: the second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium*, SSYM’04, pages 21–21.
- [Drud, 1985] Drud, A. (1985). Conopt: A grg code for large sparse dynamic nonlinear optimization problems. *Mathematical Programming*, 31(2):153–191.
- [Druschel and Rowstron, 2001] Druschel, P. and Rowstron, A. (2001). PAST: A persistent and anonymous store. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, HotOS’01, pages 197–212.
- [Eastlake and Jones, 2001] Eastlake, D. and Jones, P. (2001). RFC 3174: Secure hash algorithm 1 (sha1). Available online at: <http://tools.ietf.org/rfc/rfc3174.txt>.
- [Främling et al., 2006] Främling, K., Ala-Risku, T., Kärkkäinen, M., and Holmström, J. (2006). Agent-based model for managing composite product information. *Computers in Industry*, 57(1):72–81.

- [Främling et al., 2007] Främling, K., Harrison, M., Brusey, J., and Petrow, J. (2007). Requirements on unique identifiers for managing product lifecycle information: comparison of alternative approaches. *International Journal of Computer Integrated Manufacturing*, 20(7):715–726.
- [Freedman and Morris, 2002] Freedman, M. J. and Morris, R. (2002). Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS’02*, pages 193–206.
- [Ganesan et al., 2007] Ganesan, P., Matloub, S., Miranz, T., Rossmann, A., and Sze, R. (2007). U.s. patent number 7,191,215 (filed on march 9th 2005): Method and system for providing instantaneous media-on-demand services by transmitting contents in pieces from client machines.
- [Guo et al., 2003] Guo, Y., Suh, K., Kurose, J., and Towsley, D. (2003). P2Cast: peer-to-peer patching scheme for VoD service. In *Proceedings of the 12th International Conference on World Wide Web*, pages 301–309.
- [Gupta et al., 2003] Gupta, A., Liskov, B., and Rodrigues, R. (2003). One hop lookups for peer-to-peer overlays. In *Proceedings of the 9th Conference on Hot Topics in Operating Systems, HotOS’03*, pages 2–2.
- [Harchol-Balter et al., 1999] Harchol-Balter, M., Leighton, T., and Lewin, D. (1999). Resource discovery in distributed networks. In *Proceedings of the Eighteenth Annual ACM symposium on Principles of Distributed Computing, PODC’99*, pages 229–237.
- [He et al., 2008] He, Y., Lee, I., and Guan, L. (2008). Distributed throughput maximization in hybrid-forwarding p2p vod applications. In *Proceedings of the 2008 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, pages 2165–2168.
- [Housley et al., 2002] Housley, R., Polk, W., Ford, W., and Polo, D. (2002). RFC 3280: X.509 public key infrastructure. Available online at: <http://tools.ietf.org/rfc/rfc3280.txt>.
- [Hua et al., 1998] Hua, K., Cai, Y., and Sheu, S. (1998). Patching: A multicast technique for true video-on-demand services. In *Proceedings of the 12th ACM International Conference on Multimedia*, pages 191–200.
- [Huntington, 2005] Huntington, D. (2005). U.s. patent number 6,970,937 (filed on jun 15th, 2001): User-relayed data broadcasting.
- [Jin and Chan, 2010] Jin, X. and Chan, S.-H. G. (2010). Unstructured peer-to-peer network architectures. In Shen, X., Yu, H., Buford, J., and Akon, M., editors, *Handbook of Peer-to-Peer Networking*, pages 117–142. Springer.
- [Kaashoek and Karger, 2003] Kaashoek, M. F. and Karger, D. R. (2003). Koorde: A simple degree-optimal distributed hash table. In *Second International Workshop on Peer-to-Peer Systems, IPTPS’03*, pages 98–107.

BIBLIOGRAPHY

- [Kailath, 1967] Kailath, T. (1967). The divergence and bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technology*, 15(1):52–60.
- [Klemm et al., 2004a] Klemm, A., Lindemann, C., Vernon, M. K., and Waldhorst, O. P. (2004a). Characterizing the query behavior in peer-to-peer file sharing systems. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC’04, pages 55–67.
- [Klemm et al., 2004b] Klemm, A., Lindemann, C., and Waldhorst, O. P. (2004b). Relating query popularity and file replication in the gnutella peer-to-peer networks. In *Proc. 12th GI/ITG Conf. on Measuring, Modelling and Evaluation of Computer and Communication Systems*.
- [Lamoriniere et al., 2009] Lamoriniere, C., Nafaa, A., and Murphy, L. (2009). Dynamic switching between adaptive fec protocols for reliable multi-source streaming. In *Proceedings of the 28th IEEE Conference on Global Telecommunications*, GLOBE-COM’09, pages 5388–5393.
- [Laoutaris et al., 2008] Laoutaris, N., Rodriguez, P., and Massoulie, L. (2008). Echos: edge capacity hosting overlays of nano data centers. *ACM SIGCOMM Computer Communication Review*, 38(1):51–54.
- [Leong et al., 2006] Leong, B., Liskov, B., and Demaine, E. D. (2006). Epichord: Parallelizing the chord lookup algorithm with reactive routing state management. *Computer Communications*, 29(9):1243–1259.
- [Levine and Shields, 2002] Levine, B. N. and Shields, C. (2002). Hordes - a multicast based protocol for anonymity. *Journal of Computer Security*, 10(3):213–240.
- [Li and Chandra, 2007] Li, X. and Chandra, C. (2007). Efficient knowledge integration to support a complex supply network management. *International Journal of Manufacturing Technology and Management*, 10(1):1–18.
- [Liang et al., 2006] Liang, J., Kumar, R., and Ross, K. W. (2006). The fasttrack overlay: A measurement study. *Computer Networks*, 50(6):842–858.
- [Liao and Jin, 2007] Liao, X. and Jin, H. (2007). Octopus: A hybrid scheduling strategy for p2p vod services. In *Proceedings of the Sixth International Conference on Grid and Cooperative Computing*, GCC’07, pages 26–33.
- [Löser et al., 2007] Löser, A., Staab, S., and Tempich, C. (2007). Semantic social overlay networks. *IEEE Journal on Selected Areas in Communications*, 25(1):5–14.
- [Lu et al., 2005] Lu, T., Fang, B., Sun, Y., and Cheng, X. (2005). Performance analysis of wongoo system. In *Proceedings of Fifth International Conference on Computer and Information Technology*, CIT’05, pages 716–722.
- [Luby, 2002] Luby, M. (2002). Lt codes. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS’02, page 271.

- [Magoni and Hoerd, 2005] Magoni, D. and Hoerd, M. (2005). Internet core topology mapping and analysis. *Computer Communications*, 28(5):494–506.
- [Mahalanobis, 1936] Mahalanobis, P. (1936). On the generalised distance in statistics. *Proceedings of the National Institute of Sciences of India*, 2(1):49–55.
- [Malkhi et al., 2002] Malkhi, D., Naor, M., and Ratajczak, D. (2002). Viceroy: a scalable and dynamic emulation of the butterfly. In *Proceedings of the Twenty-First Annual Symposium on Principles of Distributed Computing*, PODC’02, pages 183–192.
- [Manzanares-Lopez et al., 2010a] Manzanares-Lopez, P., Malgosa-Sanahuja, J., Muñoz-Gea, J. P., and Sanchez-Aarnoutse, J. C. (2010a). Multicast services over structured p2p networks. In Shen, X., Yu, H., Buford, J., and Akon, M., editors, *Handbook of Peer-to-Peer Networking*, pages 875–896. Springer.
- [Manzanares-Lopez et al., 2010b] Manzanares-Lopez, P., Muñoz-Gea, J. P., Malgosa-Sanahuja, J., and Sanchez-Aarnoutse, J. C. (2010b). Anonymity in p2p systems. In Shen, X., Yu, H., Buford, J., and Akon, M., editors, *Handbook of Peer-to-Peer Networking*, pages 785–812. Springer.
- [Manzanares-Lopez et al., 2011] Manzanares-Lopez, P., Muñoz-Gea, J. P., Malgosa-Sanahuja, J., and Sanchez-Aarnoutse, J. C. (2011). An efficient distributed discovery service for epcglobal network in nested package scenarios. *Journal of Network and Computer Applications*, 34(3):480–496.
- [Maymounkov and Mazières, 2002] Maymounkov, P. and Mazières, D. (2002). Kademlia: A peer-to-peer information system based on the xor metric. In *First International Workshop on Peer-to-Peer Systems*, IPTPS’02, pages 53–65.
- [Mealling and Daniel, 2000] Mealling, M. and Daniel, R. (2000). RFC 2915: The naming authority pointer (naptr) dns resource record. Available online at: <http://tools.ietf.org/rfc/rfc2915.txt>.
- [Mislove et al., 2004] Mislove, A., Oberoi, G., Post, A., Reis, C., Druschel, P., and Wallach, D. S. (2004). Ap3: cooperative, decentralized anonymous communication. In *Proceedings of the 11th ACM SIGOPS European Workshop*, EW’04, page 30.
- [Mockapetris, 1987] Mockapetris, P. (1987). RFC 1035: Domain name system. Available online at: <http://tools.ietf.org/rfc/rfc1035.txt>.
- [Möller et al., 2003] Möller, U., Cottrell, L., Palfrader, P., and Sassaman, L. (2003). Mix-master Protocol — Version 2. IETF Internet Draft.
- [Morselli et al., 2005] Morselli, R., Bhattacharjee, B., Srinivasan, A., and Marsh, M. A. (2005). Efficient lookup on unstructured topologies. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Principles of Distributed Computing*, PODC’05, pages 77–86.

BIBLIOGRAPHY

- [Muñoz-Gea et al., 2008a] Muñoz-Gea, J. P., Malgosa-Sanahuja, J., Manzanares-Lopez, P., and Sanchez-Aarnoutse, J. C. (2008a). Structured management of an unstructured network: a reliable and self-organized p2p solution. In *Proceedings of the 7th IASTED International Conference on Communication Systems and Networks*, CSN'08.
- [Muñoz-Gea et al., 2010a] Muñoz-Gea, J. P., Malgosa-Sanahuja, J., Manzanares-Lopez, P., and Sanchez-Aarnoutse, J. C. (2010a). Implementation of traceability using a distributed rfid-based mechanism. *Computers in Industry*, 61(5):480–496.
- [Muñoz-Gea et al., 2010b] Muñoz-Gea, J. P., Malgosa-Sanahuja, J., Manzanares-Lopez, P., and Sanchez-Aarnoutse, J. C. (2010b). Providing vod streaming using p2p networks. In Shen, X., Yu, H., Buford, J., and Akon, M., editors, *Handbook of Peer-to-Peer Networking*, pages 1025–1042. Springer.
- [Muñoz-Gea et al., 2007a] Muñoz-Gea, J. P., Malgosa-Sanahuja, J., Manzanares-Lopez, P., Sanchez-Aarnoutse, J. C., and Garcia-Haro, J. (2007a). An efficient and reliable anonymous solution for a self-organized p2p network. In *Proceedings of the 7th IEEE International Workshop on IP Operations and Management - IPOM'07*, volume 4786 of *Lecture Notes in Computer Science*, pages 25–36. Springer Berlin / Heidelberg.
- [Muñoz-Gea et al., 2007b] Muñoz-Gea, J. P., Malgosa-Sanahuja, J., Manzanares-Lopez, P., Sanchez-Aarnoutse, J. C., and Garcia-Haro, J. (2007b). A self-organized p2p network for an efficient and secure content location & download. In *Revised Selected Papers of the 4th International Workshops on Databases, Information Systems, and Peer-to-Peer Computing - DBISP2P'06*, volume 4125 of *Lecture Notes in Computer Science*, pages 368–375. Springer Berlin / Heidelberg.
- [Muñoz-Gea et al., 2008b] Muñoz-Gea, J. P., Malgosa-Sanahuja, J., Manzanares-Lopez, P., Sanchez-Aarnoutse, J. C., and Garcia-Haro, J. (2008b). Behavior characterization and performance evaluation of a hybrid p2p file sharing system. In *Proceedings of the 3rd International Conference on Software and Data Technologies*, ICSOFT'08, pages 95–102.
- [Muñoz-Gea et al., 2008c] Muñoz-Gea, J. P., Malgosa-Sanahuja, J., Manzanares-Lopez, P., Sanchez-Aarnoutse, J. C., and Garcia-Haro, J. (2008c). A disconnection-aware mechanism to provide anonymity in two-level p2p systems. In *Revised Selected Papers of the 2nd International Conference on Software and Data Technologies - ICSOFT'07*, volume 22 of *Communications in Computer and Information Science*, pages 228–239. Springer Berlin / Heidelberg.
- [Muñoz-Gea et al., 2008d] Muñoz-Gea, J. P., Malgosa-Sanahuja, J., Manzanares-Lopez, P., Sanchez-Aarnoutse, J. C., and Garcia-Haro, J. (2008d). A low-variance random-walk procedure to provide anonymity in overlay networks. In *Proceeding of the 13th European Symposium on Research in Computer Security: Computer Security - ESORICS'08*, volume 5283 of *Lecture Notes in Computer Science*, pages 238–250. Springer Berlin / Heidelberg.

- [Muñoz-Gea et al., 2009a] Muñoz-Gea, J. P., Malgosa-Sanahuja, J., Manzanares-Lopez, P., Sanchez-Aarnoutse, J. C., and Garcia-Haro, J. (2009a). Proposal, design and evaluation of a mechanism to limit the length of anonymous overlay network paths. *Computers & Security*, 28(8):772–784.
- [Muñoz-Gea et al., 2008e] Muñoz-Gea, J. P., Malgosa-Sanahuja, J., Manzanares-Lopez, P., Sanchez-Aarnoutse, J. C., and Guirado-Puerta, A. M. (2008e). A hybrid topology architecture for p2p file sharing systems. In *Revised Selected Papers of the 1st International Conference on Software and Data Technologies - ICSOFT'06*, volume 10 of *Communications in Computer and Information Science*, pages 220–229. Springer Berlin / Heidelberg.
- [Muñoz-Gea et al., 2009b] Muñoz-Gea, J. P., Malgosa-Sanahuja, J., Manzanares-Lopez, P., Sanchez-Aarnoutse, J. C., and Martinez-Rojo, A. M. (2009b). Simulation of a p2p application using oversim. In *Proceedings of the First International Conference on Advances in Future Internet*, AFIN'09, pages 53–60.
- [Muñoz-Gea et al., 2010c] Muñoz-Gea, J. P., Malgosa-Sanahuja, J., Piñero-Escuer, P. J., and Garcia-Haro, J. (2010c). Providing vod services in community networks using p2p technology. In Antonopoulos, N., Exarchakos, G., Li, M., and Liotta, A., editors, *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications*, pages 1032–1052. IGI Global.
- [Nafaa et al., 2008] Nafaa, A., Murphy, S., and Murphy, L. (2008). Analysis of a large-scale vod architecture for broadband operators: A p2p-based solution. *IEEE Communications Magazine*, 46(12):47–55.
- [Nafaa et al., 2009] Nafaa, A., Murphy, S., and Murphy, L. (2009). Irish patents office, application num. s2009/0455: A system and method for multi-source streaming of media content.
- [Naicken et al., 2007] Naicken, S., Livingston, B., Basu, A., Rodhetbhai, S., Wakeman, I., and Chalmers, D. (2007). The state of peer-to-peer simulators and simulations. *ACM SIGCOMM Computer Communication Review*, 37(2):95–98.
- [Nwana, 1996] Nwana, H. S. (1996). Software agents: An overview. *Knowledge Engineering Review*, 11(3):1–40.
- [Pfitzmann and Hansen, 2010] Pfitzmann, A. and Hansen, M. (2010). A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf. v0.34.
- [Phillips et al., 2006] Phillips, W., Johnsen, T., Caldwell, N., and Lewis, M. A. (2006). Investigating innovation in complex health care supply networks. *Health Services Management Research*, 19(3):197–206.

BIBLIOGRAPHY

- [Plaxton et al., 1997] Plaxton, C. G., Rajaraman, R., and Richa, A. W. (1997). Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the Ninth Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA'97, pages 311–320.
- [Poulin et al., 2006] Poulin, M., Montreuil, B., and Martel, A. (2006). Implications of personalization offers on demand and supply networks design: A case from the golf club industry. *European Journal of Operational Research*, 169(3):996–1009.
- [Pouwelse et al., 2008] Pouwelse, J. A., Garbacki, P., Wang, J., Bakker, A., Yang, J., Iosup, A., Epema, D. H. J., Reinders, M., van Steen, M. R., and Sips, H. J. (2008). Tribler: a social-based peer-to-peer system: Research articles. *Concurrency and Computation: Practice & Experience*, 20(2):127–138.
- [Rahman et al., 2005] Rahman, M., Bhattacharya, P., and Desai, B. (2005). Statistical similarity measures in image retrieval systems with categorization & block based partition. In *Proceedings of IEEE International Workshop on Imaging Systems and Techniques*, pages 91–97.
- [Ratnasamy et al., 2001] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. (2001). A scalable content-addressable network. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM'01, pages 161–172.
- [Raymond, 2001] Raymond, J.-F. (2001). Traffic analysis: protocols, attacks, design issues, and open problems. In *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, pages 10–29.
- [Reiter and Rubin, 1998] Reiter, M. and Rubin, A. (1998). Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92.
- [Rowstron and Druschel, 2001] Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350.
- [Sahinidis and Tawarmalani, 2010] Sahinidis, N. V. and Tawarmalani, M. (2010). *BARON 9.0.4: Global Optimization of Mixed-Integer Nonlinear Programs*, User's Manual.
- [Schollmeier, 2001] Schollmeier, R. (2001). A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Proceedings of the First International Conference on Peer-to-Peer Computing*, P2P'01.
- [Sherwood et al., 2002] Sherwood, R., Bhattacharjee, B., and Srinivasan, A. (2002). P5: A protocol for scalable anonymous communication. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 58–70.

- [Sirer et al., 2004] Sirer, E. G., Goel, S., Robson, M., and Engin, D. (2004). Eluding carnivores: file sharing with strong anonymity. In *Proceedings of the 11th ACM SIGOPS European Workshop*, EW'04, page 19.
- [Stoica et al., 2003] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., and Balakrishnan, H. (2003). Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32.
- [Stutzbach and Rejaie, 2006] Stutzbach, D. and Rejaie, R. (2006). Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC'06, pages 189–202.
- [Stutzbach et al., 2007] Stutzbach, D., Zhao, S., and Rejaie, R. (2007). Characterizing files in the modern gnutella network. *Multimedia Systems*, 13(1):35–50.
- [Suh et al., 2007] Suh, K., Diot, C., Kurose, J., Massoulié, L., Neumann, C., Towsley, D. F., and Varvello, M. (2007). Push-to-peer video-on-demand system: Design and evaluation. *IEEE Journal on Selected Areas in Communications*, 25(9):1706–1716.
- [Syverson et al., 1997] Syverson, P. F., Goldschlag, D. M., and Reed, M. G. (1997). Anonymous connections and onion routing. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, SP'97, pages 44–54.
- [Twiss and Ferguson, 2007] Twiss, A. D. and Ferguson, D. S. (2007). U.s. patent application number 20080040420 (filed on nov 8th 2006): Content distribution network.
- [Vlavianos et al., 2006] Vlavianos, A., Iliofotou, M., and Faloutsos, M. (2006). Bitos: Enhancing bittorrent for supporting streaming applications. In *Proceedings of the 25th IEEE International Conference on Computer Communications*, INFOCOM'06, pages 1–6.
- [Wang and Liu, 2008] Wang, D. and Liu, J. (2008). A dynamic skip list-based overlay for on-demand media streaming with vcr interactions. *IEEE Transactions on Parallel Distributed Systems*, 19(4):503–514.
- [Wareham et al., 2005] Wareham, J., Mathiassen, L., Rai, A., Straub, D., and Klein, R. (2005). The business value of digital supply networks: A program of research on the impacts of globalization. *Journal of International Management*, 11(2):201–227.
- [Weise, 2001] Weise, J. (2001). Public key infrastructure overview. Available online at: <http://www.sun.com/blueprints/0801/publickey.pdf>.
- [Wong et al., 2005] Wong, B., Slivkins, A., and Sirer, E. G. (2005). Meridian: a lightweight network location service without virtual coordinates. *ACM SIGCOMM Computer Communication Review*, 35(4):85–96.
- [Wright et al., 2004] Wright, M. K., Adler, M., Levine, B. N., and Shields, C. (2004). The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Transactions on Information and System Security*, 7(4):489–522.

BIBLIOGRAPHY

- [Wright et al., 2008] Wright, M. K., Adler, M., Levine, B. N., and Shields, C. (2008). Passive-logging attacks against anonymous communications systems. *ACM Transactions on Information and System Security*, 11(2):1–34.
- [Xiong and Liu, 2004] Xiong, L. and Liu, L. (2004). Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):873–857.
- [Yang and Garcia-Molina, 2003] Yang, B. and Garcia-Molina, H. (2003). Designing a super-peer network. In *Proceedings of the 19th International Conference on Data Engineering*, ICDE’03.
- [Yiu et al., 2007] Yiu, W.-P. K., Jin, X., and Chan, S.-H. G. (2007). Vmesh: Distributed segment storage for peer-to-peer interactive video streaming. *IEEE Journal on Selected Areas in Communications*, 25(9):1717–1731.
- [Young, 2008] Young, M. (2008). Extensible supply-chain discovery service (esds) concepts. Available online at: <http://tools.ietf.org/id/draft-young-esds-concepts-04.txt>.
- [Zeilenga, 2006] Zeilenga, K. (2006). RFC 4510: Lightweight directory access protocol (ldap). Available online at: <http://tools.ietf.org/rfc/rfc4510.txt>.
- [Zhao et al., 2004] Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., and Kubiawicz, J. D. (2004). Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53.

This Ph.D. Thesis has been supported by the Fundación Séneca, Agencia de Ciencia y Tecnología de la Región de Murcia, with the FPI pre-doctoral fellowship 05140/FPI/06 (from February 2007 to May 2007) and by the Spanish Ministerio de Ciencia e Innovación with the FPU pre-doctoral fellowship AP2006-01567 (from May 2007 to November 2008). It has also been supported by project TEC2010-21405-C02-02/TCM (CALM), funded by the Spanish Ministerio de Ciencia e Innovación and also by the framework of “Programa de Ayudas a Grupos de Excelencia de la Región de Murcia”, funded by Fundación Séneca (Plan Regional de Ciencia y Tecnología 2007/2010).